INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Efficient Signature Verification and Key Revocation using Identity Based Cryptography

eingereicht von

Tobias Guggemos

am 14. Januar 2020

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Efficient Signature Verification and Key Revocation using Identity Based Cryptography

eingereicht von

Tobias Guggemos
am 14. Januar 2020

Erster Gutachter: Prof. Dr. Dieter Kranzlmüller
Zweiter Gutachter: Prof. Dr. Georg Sigl
Tag der mündlichen Prüfung: 24. April 2020

The revelation that the graph appears to climb so smoothly, even though the primes themselves are so unpredictable, is one of the most miraculous in mathematics and represents one of the high points in the story of the primes. On the back page of his book of logarithms, Gauss recorded the discovery of his formula for the number of primes up to N in terms of the logarithm function. Yet despite the importance of the discovery, Gauss told no one what he had found. The most the world heard of his revelation were the cryptic words, 'You have no idea how much poetry there is in a table of logarithms.'

MARCUS DU SAUTOY
(The Music of the Primes, 2003)[1]

[1] The attentive reader may observe several prime numbers carefully and intentionally placed in this book.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Guggemos, Tobias

....

Name, Vorname

München, 14. Januar 2020	*Tobias Guggemos*
....	....
(Ort, Datum)	(Unterschrift Doktorand/in)

Acknowledgments

“Unlike a Master’s Thesis, a Doctoral Thesis is a marathon, not a sprint”

V. DANCIU (December 2015)

Similar to running a marathon, I run through several phases during this PhD and I was only able to cross the finish line with the help of various people who supported me along the way.

As the one giving the starting signal, supporting me and my research over the whole distance and accompanying my finish with discussions, advice and Ice Cream, I truly thank Prof. Kranzlmüller. I feel very proud to be part of the chair at LMU and the MNM-Team with its joyful and fruitful atmosphere and particularly like to thank:

Michael for supervising my Master’s Thesis, encouraging me towards this step and always offering a helpful advice and a cold beer, even in your retirement. Nils, Vitalian and Karl for bringing me up to speed after my sabbatical and introducing me to many aspects of research and teaching which I still apply and appreciate a lot. Also for organizing some memorable team events, especially the legendary Glögg drinking!

Jan for being a perfect office-mate, employing the *embedded security research group* together with Nils and myself, which has since been the roof for over 30 student projects. Matthias, Roger and Tobi F. who started their marathons with me, Cuong, Max, Pascal and Amir who joined later but also Markus and Tobi W. who did the same journey at LRZ. All of you were always happy to proofread, help out with any problems and open for the weirdest discussions – of which only some escalated too much. I enjoyed every after-work party, every swim in the Eisbach and our Doctoral Retreat a lot, and I’m very happy that you became such good friends along the way.

Annette for all the help as our IT-admin but particularly for the brazing-lessons and Miki for keeping the administrative disturbance away. Prof. Reiser and David for being always open for a discussion and David in particular for pointing me to Identity Based Cryptography.

A big thanks goes to Sophia, for assisting me formulating the first rough ideas for this thesis and spending hours and nights with me in the office discussing various ideas, projects and papers. But even more for the joyful weekly rides to Kirchheim, the daily lunch-strolls, nightly tinkering of crypto calendars, . . . just for being such a good friend.

I’d also like to thank the core team of the QuaSiModO project, which allowed me to concentrate my research on cryptography during the last half of the journey and will always have a special status as my first successful research proposal: Alex and Stefan for establishing such a joyful working atmosphere and Tobi H. for being the one implementing some of my ideas. Daniel, for giving advice regarding the security proofs and having the patience proofreading them. Prof. Sigl, for committing as my second evaluator and providing helpful guidance just before the finish.

Also thanks to all the proofreaders and helpers outside university, especially: Corinna for having the patience reading the whole thesis, Michael W. for providing most helpful feedback on some important English phrases and Gert for helping with the design of the book-cover. I need to mention Lindsey Stirling, whose music was inspiring and recreational during writing.

I also thanks my parents and family for their unconditional help and support since I decided to start studying.

Last, but most important, I thank Kerstin for being my best friend, greatest supporter and companion during the last 13 1/2 years. Thanks for reminding me to sometimes leave the office and being a perfect partner in life and traveling around the world! Without your support, in particular during the last „sprint” of writing this thesis, crossing the finish line would have been much harder.

Tobias Guggemos (August, 2020)

Abstract

Cryptography deals with the development and evaluation of procedures for securing digital information. It is essential whenever multiple entities want to communicate safely. One task of cryptography concerns digital signatures and the verification of a signer's legitimacy requires trustworthy authentication and authorization. This is achieved by deploying cryptographic keys. When dynamic membership behavior and identity theft come into play, revocation of keys has to be addressed. Additionally, in use cases with limited networking, computational, or storage resources, efficiency is a key requirement for any solution.

In this work we present a solution for signature verification and key revocation in constrained environments, e.g., in the Internet of Things (IoT). Where other mechanisms generate expensive overheads, we achieve revocation through a single multicast message without significant computational or storage overhead. Exploiting Identity Based Cryptography (IBC) complements the approach with efficient creation and verification of signatures.

Our solution offers a framework for transforming a suitable signature scheme to a so-called Key Updatable Signature Scheme (KUSS) in three steps. Each step defines mathematical conditions for transformation and precise security notions. Thereby, the framework allows a novel combination of efficient Identity Based Signature (IBS) schemes with revocation mechanisms originally designed for confidentiality in group communications.

Practical applicability of our framework is demonstrated by transforming four well-established IBS schemes based on Elliptic Curve Cryptography (ECC). The security of the resulting group Identity Based Signature (gIBS) schemes is carefully analyzed with techniques of *Provable Security*.

We design and implement a testbed for evaluating these kind of cryptographic schemes on different computing- and networking hardware, typical for constrained environments. Measurements on this testbed provide evidence that the transformations are practicable and efficient. The revocation complexity in turn is significantly reduced compared to existing solutions. Some of our new schemes even outperform the signing process of the widely used Elliptic Curve Digital Signature Algorithm (ECDSA).

The presented transformations allow future application on schemes beyond IBS or ECC. This includes use cases dealing with *Post-Quantum Cryptography*, where the revocation efficiency is similarly relevant. Our work provides the basis for such solutions currently under investigation.

Kurzfassung

Die Kryptographie ist ein Instrument der Informationssicherheit und beschäftigt sich mit der Entwicklung und Evaluierung von Algorithmen zur Sicherung digitaler Werte. Sie ist für die sichere Kommunikation zwischen mehreren Entitäten unerlässlich. Ein Bestandteil sind digitale Signaturen, für deren Erstellung man kryptographische Schlüssel benötigt. Bei der Verifikation muss zusätzlich die Authentizität und die Autorisierung des Unterzeichners gewährleistet werden. Dafür müssen Schlüssel vertrauensvoll verteilt und verwaltet werden. Wenn sie in Kommunikationssystemen mit häufig wechselnden Teilnehmern zum Einsatz kommen, müssen die Schlüssel auch widerruflich sein. In Anwendungsfällen mit eingeschränkter Netz-, Rechen- und Speicherkapazität ist die *Effizienz* ein wichtiges Kriterium.

Diese Arbeit liefert ein Rahmenwerk, mit dem Schlüssel effizient widerrufen und Signaturen effizient verifiziert werden können. Dabei fokussieren wir uns auf Szenarien aus dem Bereich des Internets der Dinge (IoT, Internet of Things). Im Gegensatz zu anderen Lösungen ermöglicht unser Ansatz den Widerruf von Schlüsseln mit einer einzelnen Nachricht innerhalb einer Kommunikationsgruppe. Dabei fällt nur geringer zusätzlicher Rechen- oder Speicheraufwand an. Ferner vervollständigt die Verwendung von Identitätsbasierter Kryptographie (IBC, Identity Based Cryptography) unsere Lösung mit effizienter Erstellung und Verifikation der Signaturen.

Hierfür liefert die Arbeit eine dreistufige mathematische Transformation von geeigneten Signaturverfahren zu sogenannten Key Updatable Signature Schemes (KUSS). Neben einer präzisen Definition der Sicherheitsziele werden für jeden Schritt mathematische Vorbedingungen zur Transformation festgelegt. Dies ermöglicht die innovative Kombination von Identitätsbasierten Signaturen (IBS, Identity Based Signature) mit effizienten und sicheren Mechanismen zum Schlüsselaustausch, die ursprünglich für vertrauliche Gruppenkommunikation entwickelt wurden. Wir zeigen die erfolgreiche Anwendung der Transformationen auf vier etablierten IBS-Verfahren. Die ausschließliche Verwendung von Verfahren auf Basis der Elliptic Curve Cryptography (ECC) erlaubt es, den geringen Kapazitäten der Zielgeräte gerecht zu werden. Eine Analyse aller vier sogenannten group Identity Based Signature (gIBS) Verfahren mit Techniken aus dem Forschungsgebiet der *Beweisbaren Sicherheit* zeigt, dass die zuvor definierten Sicherheitsziele erreicht werden.

Zur praktischen Evaluierung unserer und ähnlicher kryptographischer Verfahren wird in dieser Arbeit eine Testumgebung entwickelt und mit IoT-typischen Rechen- und Netzmodulen bestückt. Hierdurch zeigt sich sowohl die praktische Anwendbarkeit der Transformationen als auch eine deutliche Reduktion der Komplexität gegenüber anderen Lösungsansätzen. Einige der von uns vorgeschlagenen Verfahren unterbieten gar die Laufzeiten des meistgenutzten Elliptic Curve Digital Signature Algorithm (ECDSA) bei der Erstellung der Signaturen.

Die Systematik der Lösung erlaubt prinzipiell auch die Transformation von Verfahren jenseits von IBS und ECC. Dadurch können auch Anwendungsfälle aus dem Bereich der *Post-Quanten-Kryptographie* von unseren Ergebnissen profitieren. Die vorliegende Arbeit liefert die nötigen Grundlagen für solche Erweiterungen, die aktuell diskutiert und entwickelt werden.

Contents

1 Introduction

Secure communication is a crucial prerequisite to achieve security and privacy. In this regard, the International Organization for Standardization (ISO) offers a definition for secure communication as the combination of five properties [71]: The concept that an uninvolved party cannot read or alter secure data is referred to as **(I)** *Confidentiality* and **(II)** *Integrity*, respectively. The receivers ability to verify the message's origin is called **(III)** *Authenticity*, which is necessary to validate the origins **(IV)** *Authorization* for a specific resource or service and to prevent the sender from denying having sent the information, which is referred as **(V)** *Non-Repudiation.*

Technically, these properties are implemented as a combination of networking protocols and cryptographic algorithms.

If the communication partners share a common secret (referred as key), the cryptographic mechanism is called *symmetric*. It is *asymmetric*, if there is a mathematical link between a so-called private and public key, by which one is used for en- and decryption, respectively. Such links are typically found in computational expensive mathematical problems such as prime factorization, which is why symmetric cryptography is supposedly more efficient. While scalable confidentiality and integrity can be achieved with symmetric and asymmetric techniques, this is not true for the other properties. Whenever cryptography is applied in public networks, the management of keys is a major challenge for networking protocols.

Considering the following example: Two parties would like to communicate confidentially over a public network based on a symmetric encryption algorithm such as Rijndael's algorithm specified in the Advanced Encryption Standard (AES) [40]. As a first step, they need to exchange a shared secret to encrypt with the same key. Instead of personally exchanging the key, they use a cryptographic key exchange protocol, such as the Diffie-Hellman Key Exchange [34]. However, this is not yet secure, as someone could disturb or intercept the connection, acting as a so-called Man-in-the-Middle (MITM), extracting the shared secret and harming confidentiality of future messages. The integrity of the message is necessary to prevent disturbance (e.g., by using Secure Hash Algorithm (SHA) [32]), while authenticating the exchange (e.g., by RSA [147]) is necessary to detect MITM-attacks. Still, this is not secure, as the public asymmetric keys used for authentication needs to be trusted by both parties. Instead of exchanging their public keys privately, they utilize a trust network such as a Public Key Infrastructure (PKI), which maps public keys to (physical or virtual) identities [77]. By trusting a Certificate Authority (CA), a receiver can ensure the authenticity of all user's managed by the CA.

The example shows the difficulties of managing the different keys, for as little as implementing **confidential** communication of two parties. Similar techniques can be used for multiple properties and the task of security protocols is exactly that. Once a secure channel is established with such a key exchange, many of the properties can be achieved with efficient symmetric cryptography (e.g., Keyed-Hash Message Authentication Code (HMAC) for authenticity and integrity). However, the communication of more than two parties still requires asymmetric cryptography to provide authorization based on sender authentication. This yields two challenges: First, asymmetric cryptography is computationally expensive and therefore delays every message. Second, as shown in the example above, the use of asymmetric keys requires a trust network that maps identities to public keys. A receiver needs to query this network for every received message, delaying its processing.

While these challenges are solved in traditional large scale applications, such as E-Mail, Web-Browsing or Banking, new network technologies pose additional challenges. Wireless Sensor Network (WSN), Smart Cars, Smart Homes, Smart Cities, Mobile Ad-Hoc Networks (MANETs) or Autonomous Driving – often generalized as the Internet of Things (IoT) – are use cases where authorization can change and devices may leave or join the system frequently. Such dynamic behavior requires the trust management of the public keys to be efficient and reliable for every message. Additionally, some use cases require careful management of limited technical resources, e.g., power and energy supply, but also memory, computing, or networking capabilities. Computing and memory restrictions can be managed on the device itself. In contrast, networking as a system wide resource needs to cope with device specific limitation of power and energy while staying compatible to other devices in the network. Especially wireless communication is expensive in terms of energy why compression of networking protocols such as IPv6 is a common practice and applied to security protocols, such as IPsec [122] or TLS [145]. Similarly, network overhead is reduced by replacing cryptography based on prime factorization or the Discrete Logarithm Problem (DLP) with algorithms based on Elliptic Curve Discrete Logarithm Problem (ECDLP). Elliptic Curve Cryptography (ECC) offers the same security level with around eighty percent smaller keys, ciphertext and signatures. Other cryptographic techniques, such as *Identity Based Cryptography (IBC)* or *Hash Based Signatures*, offer even lower networking overhead, but built on different trust models and architectural assumptions.

Especially Identity Based Signature (IBS) emerges as an interesting solution for authentication in such use cases, allowing efficient signature verification with explicit validation of a signer's authorization. However, especially the revocation of such signing keys is not tackled appropriately in literature. With Certificate Revocation Lists (CRLs) the research field of Certificate Lifecycle Managment (CLM) offers practical solutions for public key revocation in traditional use cases, but they fail in the combination of constrained resources and dynamic environments. On the other hand, revocation of symmetric keys is a common problem of group key management, with multicast communication as a typical application. This is of special interest if a device is to be expelled but holds the secret key, e.g., used for confidentiality. There are techniques for revocation of symmetric keys as efficient as distributing one single message within the communication group. However, such efficiency is not yet found for the revocation of asymmetric keys.

1.1 Research Question

Revocation of cryptographic keys in large-scale systems (e.g., E-Mail, Web) is solved by well-balanced trust relationships between different players, such as browsers, CAs, operating systems or hardware manufacturers. Hereby established protocols and systems are widely deployed, but their computational, storage or network overhead can be destructive for certain setups. Regardless of its technical implementation, it requires some sort of trust relationship between the participants. These techniques can only be used to a certain extent for constrained scenarios, as they imply an unacceptable amount of management, network and storage overhead paired with computational complexity. This is especially true when it comes to management of signing keys and their revocation in particular.

We aim on systematically closing this gap by defining a system that allows efficient and secure revocation of signing keys. This is achieved by examining the following research question:

> *How can efficient revocation of cryptographic signing keys be achieved in systems with constrained resources and frequent changes of member's authorization?*

The answer is provided as an efficient solution for revocation of IBS keys by examining the following questions throughout the thesis:

RQ 1: Which use cases fail using state-of-the-art mechanisms for secure communication or their respective optimizations?

RQ 2: What are the requirements to meet *efficiency* in such use cases?

RQ 3: How is key distribution and revocation achieved for symmetric and asymmetric keys?

RQ 4: How can key distribution and revocation be applied in constrained systems?

RQ 5: Which signature schemes are usable in constrained systems, can they benefit from IBC and how do they fit in such architectures?

RQ 6: How can IBS keys be revoked and how can the revocation be achieved with state-of-the-art key distribution systems?

With RQ 1, we first elaborate how different use cases deal with the management of signing keys. Showing the existence of use cases that prefer proprietary solutions is concerning, as those are often less understood and potentially less secure than standardized mechanisms. That alone demands the definition of requirements and RQ 2. Finding out why state-of-the-art is not used in the scenarios, leads to RQ 3, which examines related work. Based on this, RQ 4 studies techniques for optimizing and integrating key management architectures in the target environment. RQ 5 does the same with cryptographic solutions for authentication. As IBC offers properties which fulfill some of the requirements established in RQ 2, a focus is put on its integration in the desired architecture. Within RQ 6, these findings are used for development of an efficient solution for signing key revocation based on standardized protocols.

1.2 Methodology

The methodology of this thesis is use case driven by studying scenarios out of three different communication models: WSN, MANET and Device-to-Device Communication (D2D). The research project *SecureWSN* [151] with the examination of Smart Homes is a prominent example for WSN and copes with high constraints in terms of energy supply and computing/networking capabilities. MANETs are represented by military field communication, which requires very strict security properties while adding dynamic changes of the network topology and strict security to the list of requirements. Autonomous driving, where particularly the dynamic is significantly high represents D2Ds.

The scenarios' constraints are systematically analyzed by the terminology provided by the Internet Engineering Task Force (IETF) standard body in form of RFC 7228 [135] and its potential successor RFC 7228bis [21]. However, classifying the use cases' needs regarding security and communication models require the combination of several standard literature. The security aspect is tackled in form of the five security properties and are presented in form of a more detailed terminology. Typical communication and trust models will be explained and a third terminology is established. The thereof extracted properties are used for a fine grained classification of the aforementioned use cases.

Having such a classification, state of the art and related work is found not suitable for the particular case of revocation of keys used for sender authentication in a group of constrained communication participants. Some of the examined work covers particular aspects such as minimization of computing or networking, while keeping the level of security high. Other revocation mechanisms lack efficiency or only deal with symmetric keys.

With the findings at hand, a cryptographic solution for efficient revocation of signing keys is developed for IBS. It features the inclusion of a symmetric element, which can be revoked efficiently by Logical Key Hierarchy (LKH) [120], Centralized Authorized Key Extension (CAKE) [59] or any other mechanism suitable for the use cases. While LKH is a standard mechanism for symmetric key revocation in form of a binary tree including symmetric keys, CAKE additionally achieves efficient distribution of the keys and was developed within the scope of this thesis. IBS as well as CAKE are integrated in a group key management architecture and a mathematical transformation for their combination is developed. The transformation is exemplarily studied on four IBS schemes, all based on ECC to allow an efficient solution for all use cases. An experimental implementation of those new schemes is provided on a test bed that is particularly designed to study the issue of sender authentication in a group of constrained nodes. It allows the validation of theoretical performance studies and practicable comparison with other mechanisms. The test bed is open source and expandable to allow future research on other aspects of security and cryptography in similar settings.

1.3 Contribution

This thesis examines and identifies use cases where traditional solutions for the revocation of cryptographic material found in the field of CLM are not feasible. It overcomes arising issues for revoking asymmetric keys by a novel combination of low-overhead cryptographic signature schemes, network-inexpensive key management protocols and efficient re-keying mechanisms. We provide a cryptographic solution for revocation by updating signing and verification keys with a single message in the communication group, giving it the name Key Updatable Signature Scheme (KUSS). In contrast to similar constructions, neither the complexity for signature verification nor the signature's size is increased.

The novel contribution of this thesis is a cryptographic mechanism for signing key revocation and its embedding in a group key management system. A systematic transformation for different signature schemes on selected primitives together with transformation requirements for other primitives is provided. Four IBS schemes are transformed, implemented and integrated into a group key management protocol. With the integration being straight-forward for IBS, the resulting schemes are called group Identity Based Signature (gIBS) schemes. The security of the resulting schemes is formally examined with well-established methodologies from the research area of *Provable Security*. The solution is optimized for but not limited to scenarios featuring highly dynamic membership behavior and/or technical constrained networking and computing capabilities. An experimental testbed is designed with the specific architectural assumptions of the use cases in mind. It is then assembled from heterogeneous and resource constrained devices and used for practical evaluation of the developed mechanisms. The design is generalized to allow future examination of other protocols and systems with a similar network topology.

I.) Publications directly associated with the dissertation:

The following publications contributed evaluations regarding the applicability of cryptographic primitives or security protocols used throughout this thesis.

- Tobias Guggemos and Dieter Kranzlmüller. "gIBS – group Identity-Based Signatures: efficiently verifiable IBS key-revocation with a single multicast message". In: *The IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2020)*. Ed. by International Association for Cryptographic Research. (under review). 2020

  Summary: This paper presents parts of the transformation and the security analysis, which are both extended in this dissertation. It also presents measurement results of two of

the four transformations presented in Chapter 5.
The implementation was improved in this dissertation and measurements were extended with the additional schemes.

- Tobias Guggemos, Klement Streit, Marcus Knüpfer, Nils gentschen Felde, and Peter Hillmann. “No Cookies, just CAKE: CRT based Key Hierarchy for Efficient Key Management in Dynamic Groups”. In: *13th International Conference for Internet Technology and Secured Transactions (ICITST-2018).* dec, Cambridge, UK, 2018. DOI: `10.2053/ICITST.WorldCIS.WCST.WCICSS.2018.0002`

  Summary: This publication presents the integration of CAKE – which is Chinese Remainder Theorem (CRT) based cryptographic re-keying scheme – into a Group Key Management Protocol (GKMP), namely G-IKEv2.

  Own contribution: Part of this dissertation is the integration of the previously presented mechanism of CAKE and the newly developed address scheme in G-IKEv2 as well as the theoretical evaluation.

  Other Contributors: M. Knüpfer as a member of the German federal forces contributed the scenario and requirements which is re-furbished in this dissertation. P. Hillmann contributed the description of the cryptographic concept and N. gentschen Felde contributed an address scheme for the used key hierarchy. K. Streit contributed parts of the implementation and the practical evaluation.

- Nils gentschen Felde, Sophia Grundner-Culemann, and Tobias Guggemos. “Authentication in dynamic groups using identity-based signatures”. In: *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* Piscataway, NJ: IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-6876-4. DOI: `10.1109/WiMOB.2018.8589148`

  Summary: This publication presents the initial idea of using IBS schemes in group communication for sender authentication and introduced the mathematically verifiable revocation of the keys by re-calculating all keys in the system. It also presented a taxonomy for choosing IBS schemes for constrained scenarios.

  Own contribution: Part of this dissertation is the introduction of a *Re-Key* phase to the selected schemes. Additionally, the implementation and evaluation of the chosen schemes as well as the integration of IBS into a group key management architecture and the distribution of the keys with a GKMP were contributed.

  Other Contributors: N. gentschen Felde contributed the description of the scenario and its requirements for reliable access management. S. Grundner-Culemann contributed the taxonomy for IBS schemes, selected the schemes which are re-used for this thesis and validated the re-key mechanism.

- Tobias Guggemos. “Dynamic Key Distribution for Secure Group Communications in Constrained Environments”. In: *Doctoral Consortium: Doctoral Consortium on e-Business and Telecommunications.* Vol. 2018. SECRYPT. jul, Porto, Portugal, 2018

  Summary: This paper presents the problem space of the dissertation and was presented during the PhD symposium together with an invited poster.

- Nils gentschen Felde, Tobias Guggemos, Tobias Heider, and Dieter Kranzlmüller. “Secure Group Key Distribution in Constrained Environments with IKEv2”. In: *2017 IEEE*

Conference on Dependable and Secure Computing. Taipei, Taiwan: IEEE, 2017. DOI: `10.1109/DESEC.2017.8073823`

Summary: This publication evaluates the use of G-IKEv2 as a GKMP for the use in constrained networks and proposes protocol optimizations.

Own contribution: Part of this dissertation are the theoretical evaluation of G-IKEv2 against other solution for the use in constrained environments with the given requirements and the enhancements of the protocol to provide full coverage of the requirements.

Other Contributors: N. gentschen Felde contributed the description of the scenario and its requirements. T. Heider contributed the implementation and the measurements of the protocol on a constrained platform.

- Tobias Guggemos, Nils gentschen Felde, and Dieter Kranzlmüller. "Secure Group Communication in Constrained Networks - A Gap Analysis". In: *The 1st 2017 GLOBAL IoT SUMMIT (GIoTS'17).* Geneva, Switzerland: IEEE, 2017, pp. 1–4. DOI: `10.1109/GIOTS.2017.8016270`

  Summary: This publication presents the early problem analysis of this dissertation, identifying the gaps of security mechanism in constrained group communication. It identified the lack of efficient mechanism for sender authentication. It also presented the initial idea for the test bed, which was developed during this thesis and used for some of the experiments.

  Own contribution: Part of this dissertation is the adoption of the properties to the use case of constrained group communication as well as the evaluation of state of the art mechanisms against these requirements.

  Other Contributors: N. gentschen Felde contributed the problem description and his broad knowledge of the ISO /OSI 27001 standard that was used to develop a terminology for security in constrained group settings.

- Daniel Migault, Tobias Guggemos, Sylvain Killian, Maryline Laurent, Guy Pujolle, and Jean Philippe Wary. "Diet-ESP: IP layer security for IoT". in: *Journal of Computer Security* 25.2 (2017), pp. 173–203. DOI: `10.3233/JCS-16857`

  Summary: This journal paper presents optimizations of the IPsec/ESP protocol for the use in constrained environments by making use of compression mechanism. It shows how pre-established configuration during the key exchange can be used to allow high compression rates while keeping configuration and context exchange as low as possible. In addition, a proof of concept on constrained hardware is presented and energy measurements are provided.

  Own contribution: Part of this dissertation is the focus on the IPsec protocol suite and its usability for constrained environments. This paper is the preliminary work for some of the standardization efforts during this PhD project.

  Other Contributors: D. Migault contributed his knowledge on the IPsec protocol stack. S. Killian contributed an enhancement of a prior implementation of the protocol and the measurements on constrained hardware. Prof. M. Laurant contributed her knowledge on constrained networking technologies that was necessary for deriving the requirements for the protocol enhancements. G. Pujolle and J.P. Wary provided the laboratory in the Orange Labs in Paris as well as their knowledge on cryptography, which allowed analyzing the security of the compressed protocol.

- Daniel Migault, Daniel Palomares, Tobias Guggemos, Aurelien Wally, Maryline Laurent, and Jean Philippe Wary. *Recommendations for IPsec Configuration on Homenet and M2M Devices.* Cancun, Mexico, 2015. DOI: `10.1145/2815317.2815323`

  Summary: This paper presents performance measurements and an evaluation of IPsec for machine-to-machine and homenet scenarios in comparison to the TLS protocol. It also compares different protocol and encryption modes in regards to networking and computational overhead.

  Own contribution: Part of this dissertation is the focus on the IPsec protocol suite and its usability for (in this case only slightly) constraint environments. It also paved the way for this PhD project as it clearly showed the necessity for cryptographic and protocol enhancements in such environments.

  Other Contributors: D. Migault contributed his knowledge on the IPsec protocol stack and the initial idea of using IPsec in such environments. D. Palomares provided his experience with different implementations of the IPsec protocol stack that were used for the measurements. Prof. M. Laurant contributed her knowledge on constrained networking technologies. G. Pujolle and J.P. Wary provided the knowledge on the laboratory in the Orange Labs in Paris where the measurements were performed and the description of the considered use cases.

II.) Standardization Effort during this dissertation:

During this dissertation, different optimization for the IPsec protocol suite [122] have been contributed to the IETF standard body of which the following are currently actively discussed in the respective working groups:

- Daniel Migault, Tobias Guggemos, and Yoav Nir. *Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP).* RFC 8750. Mar. 2020. DOI: `10.17487/RFC8750`. URL: `https://rfc-editor.org/rfc/rfc8750.txt`

  Summary: This document describes how the network overhead of IPsec ESP [123] can be reduced by using a so-called implicit initialization vector for certain symmetric ciphers.

- Daniel Migault and Tobias Guggemos. *Minimal ESP.* Internet-Draft draft-ietf-lwig-minimal-esp-00. Work in Progress. Internet Engineering Task Force, Apr. 2019. 13 pp. URL: `https://datatracker.ietf.org/doc/html/draft-ietf-lwig-minimal-esp-00`

  Summary: This document describes a minimal feature set of an IPsec ESP [123] implementation, while none of the requirements of the original standard are violated. Hence, the implementation can be implemented on constrained devices while staying compatible with feature rich implementations.

- Daniel Migault, Tobias Guggemos, Carsten Bormann, and David Schinazi. *ESP Header Compression and Diet-ESP.* Internet-Draft draft-mglt-ipsecme-diet-esp-07. Work in Progress. Internet Engineering Task Force, Mar. 2019. 47 pp. URL: `https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-diet-esp-07`

  Summary: This document describes how the network overhead of IPsec ESP [123] can be reduced by up to 90% by using compression techniques negotiated during the key exchange with the IKEv2 [136] protocol. In contrast to Minimal-ESP, the resulting network packet is not standard conform anymore, why de-compression is necessary before the actual ESP processing takes place.

III.) Outside the scope of this dissertation:

The following publications did not directly contribute to the thesis at hand, but were created as part of ongoing, neither security nor cryptography related research projects.

- Roger Kowalewski, Tobias Fuchs, Karl Furlinger, and Tobias Guggemos. "Utilizing Heterogeneous Memory Hierarchies in the PGAS Model". In: *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2018, pp. 353–357. ISBN: 978-1-5386-4975-6. DOI: 10.1109/PDP2018.2018.00063

- Tobias Guggemos, Vitalian Danciu, and Annette Kostelezky. "Protokollgestützte Selbstbeschreibung in Zugangsnetzen". In: *11. DFN-Forum Kommunikationstechnologien*. may. Günzburg, Germany, 2018

- Vitalian Danciu, Tobias Guggemos, and Dieter Kranzlmüller. "Schichtung virtueller Maschinen zu Labor– und Lehrinfrastruktur". In: *9. DFN-Forum Kommunikationstechnologien*. Rostock, Germany, 2016, pp. 11–20

1.4 Structure of the Thesis

The structure follows the methodology of Section 1.2 and is depicted in Figure 1.1. First, the three terminologies for *Constraints*, *Security*, and *Communication and Trust* are deduced from standards and literature. Next, the three use cases of *SecureWSN*, *Military Field Communication*, and *Autonomous Driving* are examined and classified accordingly.

Related work and state of the art solutions for different aspects are found in form of security protocols and cryptography. Cryptographic work is systematically split into *Confidentiality*, *Integrity*, and *Authenticity*. Mechanisms for key agreement are also presented, as they form the basis for many standard networking technologies from various bodies such as *IETF* and *IEEE*. It turns out, that especially ECC is of interest whenever constraints in form of computing or network capabilities appear. Protocol optimizations found by compression are presented as well.

Proper management of public keys is required for achieving the properties of *Authorization* and *Non-Repudiation* in a large scale. Applicable work exists in from of standardized certificate formats but also in cryptography. Even though some of them deal with revocation, none satisfies regarding efficient revocation of signing keys, which is inevitable for *Authorization* in dynamic groups.

The solution is presented in Chapter 4, which first presents a common system architecture for the cryptographic mechanism developed later in the thesis. It allows management of group membership, keys, and membership authorization and defines a trust model, which is applicable for the use cases. Second, the chapter defines cryptographic transformations of IBS schemes as the main contribution of this thesis. By re-using the efficient revocation of symmetric keys, the transformation enhances signature schemes by including a second symmetric signing key to the scheme. Such an integration is the first transformation, called Two Key Signature Scheme (2KSS). It is further optimized by allowing the symmetric element being efficiently updated to change group membership. This second transformation is called Updatable Two Key Signature Scheme (U2KSS). The last step allows further efficiency, by integrating the symmetric element in the asymmetric key and, thus, reducing the size of the keys. It allows the asymmetric key itself to be updated and the scheme being transformed to a Key Updatable Signature Scheme (KUSS). The previously defined system architecture is improved to integrate such a cryptographic mechanism while staying compatible with the use cases.

In Chapter 5, the transformation is exemplarily implemented on IBS schemes, as they naturally allow the transformation with the advantage of featuring small size signatures. The security of the resulting gIBS schemes is proved under the assumptions provided by the previously defined transformations. Chapter 6 presents the test bed, which consists of different microcontrollers mixed with different network technologies found in the use cases. It includes the implementations of a GKMP, the re-keying mechanisms LKH and CAKE together with all gIBS and their underlying IBS schemes. Those implementations are used for evaluating gIBS' efficiency in Chapter 7 including a complexity analysis as well as practical performance measurements by using the hardware in the testbed. A summary of the earlier presented protocol and re-keying performances is provided as well.

The thesis provides the basis for a variety of future work, especially regarding future developments of KUSS, which turns out to be a novel approach in cryptography and network security. Other, potentially quantum-resistant primitives (e.g., lattices and isogenies) are candidates for further investigations of such transforms. The proposed framework for proving the security is rather conservative and future work may improve the constructions.

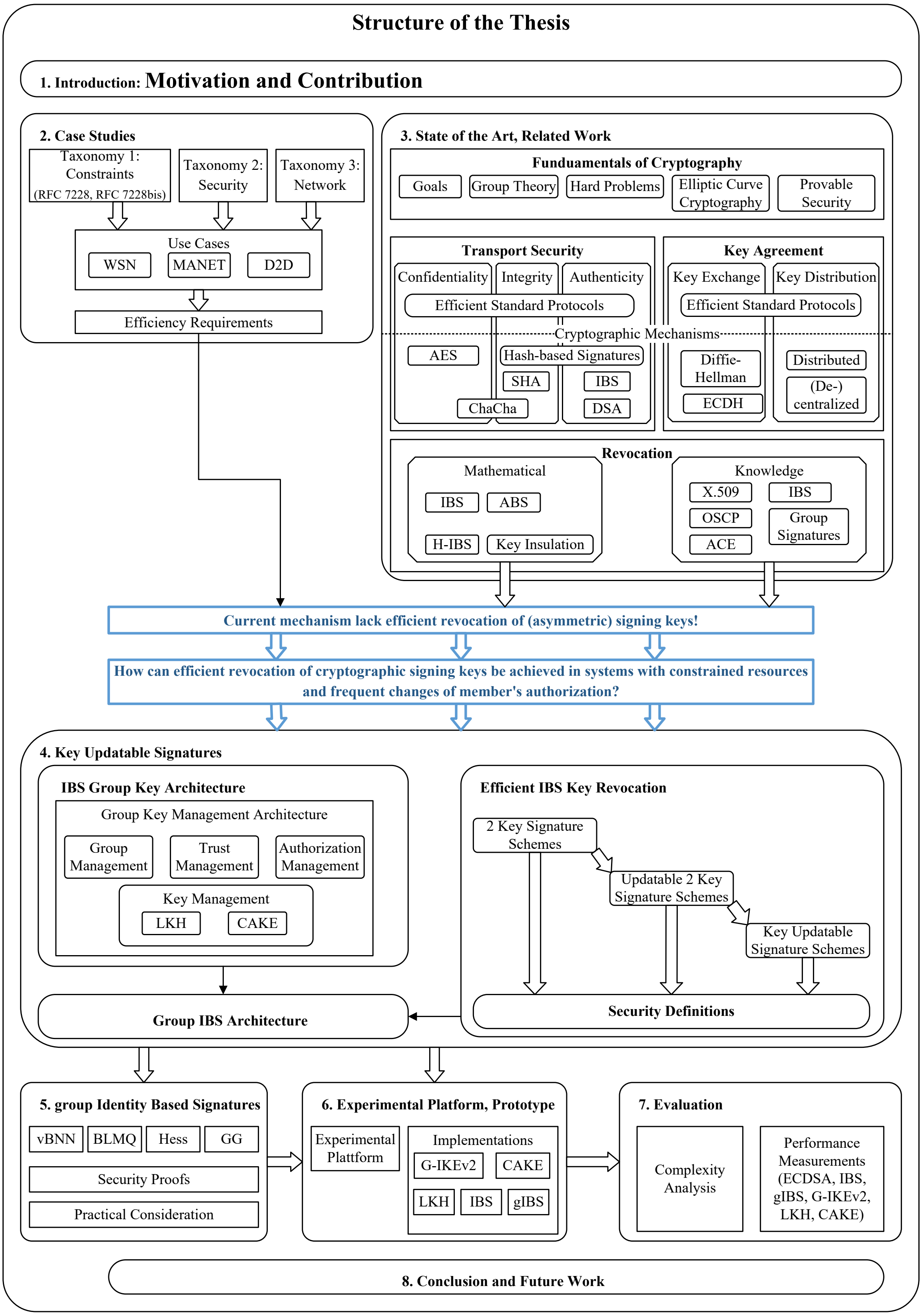

Structure of the Thesis
1. Introduction: Motivation and Contribution
2. Case Studies
Taxonomy 1: Constraints (RFC 7228, RFC 7228bis)
Taxonomy 2: Security
Taxonomy 3: Network
Use Cases
WSN
MANET
D2D
Efficiency Requirements
3. State of the Art, Related Work
Funduamentals of Cryptography
Goals
Group Theory
Hard Problems
Elliptic Curve Cryptography
Provable Security
Transport Security
Confidentiality
Integrity
Authenticity
Efficient Standard Protocols
Key Agreement
Key Exchange
Key Distribution
Efficient Standard Protocols
Cryptographic Mechanisms
AES
Hash-based Signatures
SHA
IBS
ChaCha
DSA
Diffie-Hellman
ECDH
Distributed
(De-) centralized
Revocation
Mathematical
IBS
ABS
H-IBS
Key Insulation
Knowledge
X.509
IBS
OSCP
Group Signatures
ACE
Current mechanism lack efficient revocation of (asymmetric) signing keys!
How can efficient revocation of cryptographic signing keys be achieved in systems with constrained resources and frequent changes of member's authorization?
4. Key Updatable Signatures
IBS Group Key Architecture
Group Key Management Architecture
Group Management
Trust Management
Authorization Management
Key Management
LKH
CAKE
Group IBS Architecture
Efficient IBS Key Revocation
2 Key Signature Schemes
Updatable 2 Key Signature Schemes
Key Updatable Signature Schemes
Security Definitions
5. group Identity Based Signatures
vBNN
BLMQ
Hess
GG
Security Proofs
Practical Consideration
6. Experimental Platform, Prototype
Experimental Plattform
Implementations
G-IKEv2
CAKE
LKH
IBS
gIBS
7. Evaluation
Complexity Analysis
Performance Measurements (ECDSA, IBS, gIBS, G-IKEv2, LKH, CAKE)
8. Conclusion and Future Work

Figure 1.1: Structure and methodology of the thesis.

2 Case Studies

Traditional forms of communication found in the Internet build upon a well-established balance of trusted channels and networks. With the emerge of the so-called Internet of Things (IoT), this assumption slowly changed within the last 10-15 years. The term IoT groups hundreds of use cases and dozens of communication paradigms, which at some point communicate over an untrustworthy network, like the Internet [63]. This heterogeneity led to an ever-growing community of researchers, industry and standardization bodies forming classification of use cases. When talking about communication security, major difficulties are found in the various constraints but also in the dynamic membership behavior leading to different topological setups. This chapter systematically analyzing three use cases from the broad field of IoT, giving a sense for the term *efficiency* in such networks and establishes requirements for signing key revocation in this regard.

Discussing a large number of use cases is not practicable and would exceed the scope of efficient authentication and key revocation in this dissertation. Hence, a high-level classification for use cases is presented first and used to select three use cases covering a wide range of requirements in Section 2.1. This classification is than refined in Section 2.2 by deducing terminologies for constraints, topological differences and security requirements out of standard literature. A detailed description of each of the use cases presented in Section 2.3 and classified with respect to the terminologies. Section 2.4 summarizes the findings and defines the term *efficiency* for the remaining of this thesis and formulates three high-level requirements in that regard.

2.1 Selection of Use Cases

The selection is driven by a coarse-grained classification for constraints, topological differences and security depicted in Figure 2.1. It shows the following three categories as a star graph with the respective dimensions of *Constraints*, *Security* and *Network Topology*:

Constraints describe the capabilities of the most constrained devices of a given scenario as *low*, *medium*, and *high*. We distinguish device local constraints like computation/storage or power/energy and system-wide constraints like networking or system-wide energy.

Low At least one device local aspect (computation/storage or energy) is constrained.

Medium There are either two device local constraints or one system-wide constraint.

High There is at least one device local and one system-wide constraint.

Security describes the required security parameters for communication in the system to achieve **I.)** Confidentiality, **II.)** Integrity, **III.)** Authenticity, **IV.)** Authorization and **V.)** Non-Repudiationas given by [71].

The property with the highest weight is used for classification.

Low The scenario requires communication security against industrial level attackers [152],

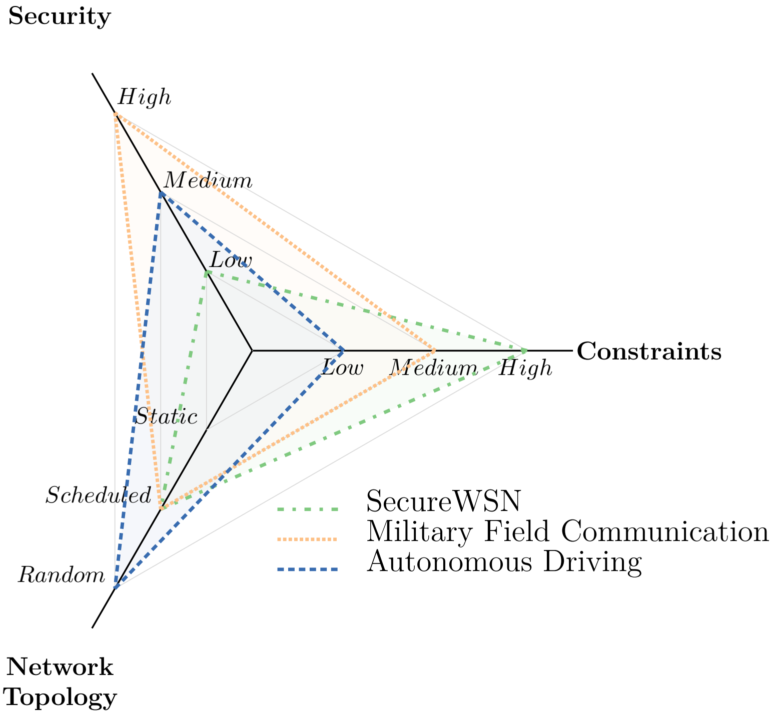


Figure 2.1: Simplified classification of the chosen use cases for further analysis.

achieved with standard security protocols, static key infrastructures and symmetric cryptography during communication.

Medium The scenario requires communication security against industrial level attackers [152], with standard security protocols, managed key infrastructures and asymmetric cryptography during communication.

High The scenario requires communication security against state level attackers [152].

Network Topology describes the topology of the setup. It shows architectural assumptions and pictures dynamic changes.

Static Once the system is set up, neither the participants nor the communication architecture changes.

Scheduled participants and communication architecture may change regularly or in certain pre-defined intervals.

Randomized participants and communication architecture may change anytime during the life-time of the system.

We pick three use cases from **1.)** Wireless Sensor Network (WSN), **2.)** Mobile Ad-Hoc Network (MANET) and **3.)** Device-to-Device Communication (D2D) of which the simplified classification is pictured in Figure 2.1 as follows:

1.) SecureWSN is a use case of WSNs and pictures a Smart Home [151]. It shows high device-local constraints and highly optimized networking protocols are used to cope with the constrained energy resources. Hence, it pictures *high* constraints. The topology can be completely *static*, however, *scheduled* changes are possible as well. For security, optimized standard protocols are used, with a static keying infrastructure and is therefore classified as *low*.

2.) Military field communication is a use case for Vehicular and Mobile Ad Hoc Networks (VANET, MANET). Naturally, it requires security against state level attackers, which is the *highest* level of security in this dissertation. The topology is typically *scheduled*, as communication groups are formed for certain purposes, however, the nature of Ad Hoc networks may also picture *randomized* topologies.

3.) Autonomous Driving is a use case for D2D. It is an example of *randomized* topologies, as cars frequently communicate with other devices they may not know. There may be *low* constraints in regards of computation or energy for certain devices in the car, but not on a system-wide level. The security is classified as *medium*, as it requires managed key infrastructures and potentially asymmetric cryptography for communication with unknown devices.

Hence, for each classification one use case with the highest requirements is found and used for detailed analysis.

2.2 Terminologies for Classification of Use Cases

The use cases' requirements are derived with the focus on cryptographic mechanisms. Choosing the best cryptography for a use case depends on the available resources, the required security properties and the network topology. Hence, three terminologies are developed, two of them – *Constrained Networks* and *Security Parameters* – deduced from standardization. The third one describes the communication and trust paradigms and combines standards and standard literature.

2.2.1 Terminology for Constraints

Classification of constrained networks is part of ongoing research and standardization. The efforts resulted in the description of RFC 7228 [135] and are currently proposed for revision in RFC 7228bis [21]. With device and power classes, RFC 7228 focuses on the terminology of nodes within a network, while its successor adds networking as a third main source of constraint. Additionally, it revises them to offer more accurate grading within the classes:

Device classes range from C0 to C19, although only the classes C0-C4 define considerable constraints. C10 and C13 are inspired by middleboxes and small single-chip computers respectively. Although low-power, they are typically able to host the kernel of legacy operating systems, such as Linux, which is not necessarily true for the lower classes. Classes >C15 are for smartphones, Desktop PCs and servers with no significant memory constraints and therefore out of scope of this work. The draft stays silent about computing capabilities and majorly re-uses the classification ARM®'s processor families Cortex-M.

Class	C0	C1	C2	C3	C4	C10	C13	C15
RAM	$\ll$ 10 KiB	10 KiB	50 KiB	100 KiB	< 1 MiB	< 8 MiB	< 1 GiB	< 4 GiB
ROM	$\ll$ 100 KiB	100 KiB	250 KiB	1 MiB	< 2 MiB	∞	∞	∞

Network classes mainly focuses on the Maximum Transfer Unit (MTU), but also on the bit rate of the physical layer, which are both typical for but not limited to wireless communication. While the bit rate is interesting for time-critical scenarios, the MTU is the major limitation as it comes with the necessity of fragmentation. This affects both, the energy to send the frames over the physical network but also in terms of code size and RAM requirements.

Hence, packet compression turns out to be the weapon of choice [124]. In unconstrained environments, fragmentation is usually tried to be avoided to preserve interoperability.

Class	S0	S1	S2	S3
MTU	$3 - 12$ Byte	$12 - 127$ Byte	$128 - 1279$ Byte	≥ 1280 Byte

Power/Energy classes differentiate between the maximum average power available during the lifetime (in Watt) and the total electricity available until the device runs out of energy (in Joule). Both require different strategies to exploit the provided power to the fullest and are divided into three power and four energy categories respectively: Power is described as “usually powered off” (Class P0), “always on low power” (Class P1) and “always on and connected” (Class P9).

Class	P0	P1	P9
Description	usually powered off	always on low power	always on

Available energy is described as “limited for a single event” (Class E0), “limited for a specific period” (Class E1), “limited for the whole lifetime” (Class E2) and “not limited at all” (Class E9).

Class	E0	E1	E2	E9
Limitation	single event	specific period	whole lifetime	not limited

These properties are relevant to describe a system’s inherent service quality. For example, if the system has to provide a certain functionality within certain time-constraints, the time to process information on the device as well as the time to distribute it within the network are important factors. Hence, the classification of *Device* and *Network* are of relevance for this task. In turn, the more powerful computing and networking hardware, the higher are power and energy consumption. Energy/Power limitations may influence computing and network hardware and the goal of a security solution – e.g as the combination of protocol and cryptographic algorithm – is not to (or only negligibly) impact the service quality while not (or only negligibly) changing the chosen hardware.

2.2.2 Terminology for Security

With the properties defined by the standard for information security (ISO / IEC 27001 [71]) and current standardization activities for securing group communication (RFC 8576 [146]), the definition of security introduced in [57] is enhanced as such:[1]

I.) Confidentiality Property that information is not made available or disclosed to unauthorized individuals, entities or processes [71, p. 2.13].

II.) Integrity Property of accuracy and completeness [71].

III.) Authentication Property that an entity is what it claims to be [71].

III.a) Data Source Authentication The corroboration that the source of data received is as claimed [126].

IV.) Authorization An approval that is granted to a system entity to access a system resource [126]. It can be split into:

IV.a) Forward Access Control Whenever a client leaves a system, it must not be able to access a system’s resource (e.g., for a key agreement protocol, the property that

[1] Although important from a system’s perspective, we leave out *Availability* for the context of this thesis. While the system architecture presented in Chapter 4 is able to provide this service, it cannot be tackled by cryptography and is hence not in the focus of this work.

compromises long-term keying material does not compromise session keys that were previously derived from the long-term material [126]).

IV.b) Backward Access Control Whenever a client joins a system, it must not be able to access prior available system resources (such as keys).

V.) Non Repudiation Ability to prove the occurrence of a claimed event or action and its originating entities [71].

V.a) Proof of Origin provides the recipient of data with evidence that proves the origin of the data, and thus protects the recipient against an attempt by the originator to falsely deny sending the data [126].

V.b) Proof of Receipt provides the originator of data with evidence that proves the data was received as addressed, and thus protects the originator against an attempt by the recipient to falsely deny receiving the data [126].

In theory, all of these properties can be achieved by cryptographic means independently from each other. Technically, some of the properties allow achieving another one without extra costs - or even reducing the costs. For example, most useful signature algorithms to prove *authenticity*, sign the cryptographic hash of a message, which in turn also provides *integrity*. Others explicitly depend on each-other, e.g., to *proof the origin* of a message, the property of *data source authenticity* has to be provided as well. Cryptography provides security against certain attacker capabilities, of which a state-level attacker is typically assumed to be the strongest [152]. Quantifying the security of a cryptographic algorithm is a difficult topic, however, the term Bit-Security features broad acceptance in the cryptographic community:

> *The intuition is that 2^n is the cost of running a brute force attack to retrieve an n-bit key, or the inverse success probability of a trivial attack that guesses the key at random. In other words, n bits of security means "as secure as an idealized perfect cryptographic primitive with an n-bit key" [91].*

2.2.3 Terminology for Network Topology

The last terminology to be set up for classification of the scenarios is for the inherent communication and trust model.

Communication We distinguish the communication models, `1:1` , `1:n` , `n:1`, `n:m`, which are found in various standard literature [23, 166, 172].

Trust Topolgy We distinguish *centralized*, *decentralized* and *distributed* trust relationships. This is of major interest when it comes to the management of keys.

Dynamic The level of dynamicity is deduced from the description of *deployment options* in [74]. It distinguishes *fixed*, *scheduled* and *randomized* deployment. Considering that any (un)deployment changes the communication's participants, this definition fits well to describe the dynamic nature of a system.

Nodes The number of nodes in the system.

These properties are necessary to find the right security solution for the given scenario. A solution working well with high-dynamic might not scale with >1000 nodes or not be able to deal with `n:m` communication. Hence, the scenario needs to be described well to find the most efficient solution under the aforementioned conditions.

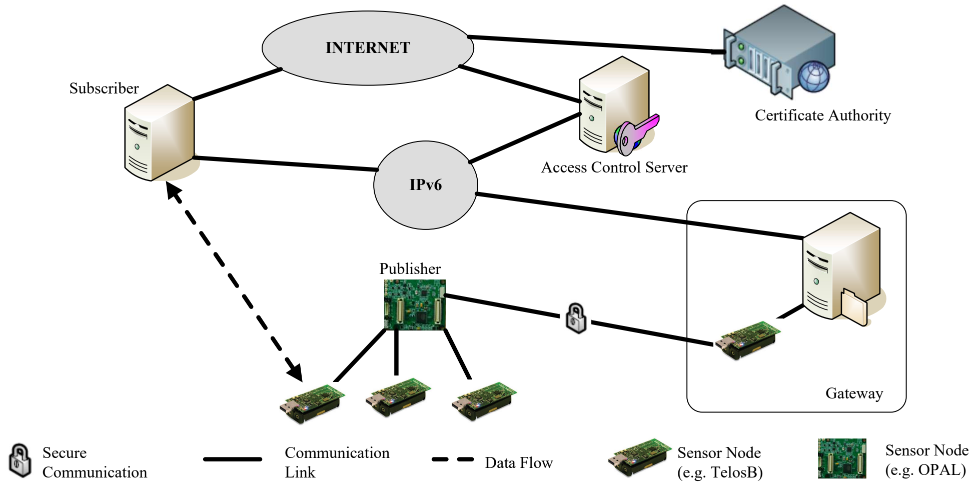


Figure 2.2: Architecture of SecureWSN [151].

2.3 Classification of Use Cases

The following will present the three use cases with a special focus on the classification properties developed above. Use cases are meant to serve as examples to give the reader a sense for the later discussed problem of key revocation. The aim is to show which constraints and communication models apply on the specific use cases and how they affect the presented security properties.

2.3.1 Use Cases 1: Wireless Sensor Network (WSN)

The typical picture of WSNs consists of different sensor nodes that collect individual data and transport it to one sink [74]. The number of devices depend on the use case, which in turn impacts the quality and length of the link to the sink. In many setups, the data-collecting nodes gather under so-called gateways providing the connection to the sink. This gateway may have several tasks, such as **a)** simple forwarding/routing of packets, **b)** data aggregation, **c)** performing protocol (de-)compression and **d)** securing the connection over the Internet.

The research project *SecureWSN* [151] pictures such a scenario as depicted in Figure 2.2. In the center, three constrained nodes are connected to the publisher, which sets up a secure connection to the gateway (see bottom right), which is what is typically understood under the term WSN. The gateway provides *IPv6* connectivity to other services, e.g.,

1. the **subscriber**, which accesses the data from the WSN,
2. an **access controller**, which grants access to the system and its data,
3. a Certificate Authority (CA) for trust management and
4. the **Internet**

The major difference to other use cases in this field is the choice in communication protocols. Choosing IPFIX [134] comes with some advantages for this specific case. However, security properties, trust model and class of constraints would be similar in other scenarios choosing other protocols stacks and hardware. Examples include 6LoWPAN (exemplarily shown in [74, 156]) and are mainly covered by different networking technologies, such as Bluetooth Low Energy (BLE) [158], LoRa [W1] or SigFox [W2].

I.) Classification of Constraints

SecureWSN is a sensor network with most of the constrained nodes running on battery (Class P1 and E2), the *Iris Mote* is typically attached to a power source (Class E) and *Opal Mote* features both (Class E2-E9 and P1-P9). All nodes are communicating with an IEEE 802.15.4 [70] (Class S2) interface. The used protocol TinyIPFIX [142] offers compression of the data, which are necessary to cope with the limited MTU of only 128 Bit. The following hardware is used in the current setting of *SecureWSN*. The module's name is depicted in the rows of the following table, while the columns present the chip's capabilities (RAM, ROM) and the hardware's classifications:

Name	**RAM**	**ROM**	**Device**	**Networking**	**Energy**	**Power**
Iris Mote	8 KB	128 KB	C0	S2	E9	P9
TelosB	10 KB	48 KB	C1	S2	E2-E9	P1
OpenMote	32 KB	512 KB	C2	S2	E2	P1
Opal Mote	52 KB	256 KB	C2	S2	E2-E9	P9

II.) Classification of Security

Security is achieved with a layered concept. The constrained nodes use a pre-shared key to communicate encrypted with the gateway(s), acting as the sink of the WSN. Nodes (and the keys) are programmed with the so-called *CoMaDa* module that also serves as the gateway to the internet which acts as a trust anchor.

The connection is always integer and confidential by using DTLS [130], however, the gateway can read the traffic and no end-to-end security is provided. As a consequence of using DTLS, a CA is required to create and prove certificates. The CA can either be part of the gateway (concrete in *CoMaDa*) or an external service as presented in Figure 2.2. Authenticity is provided through the DTLS handshake while the data source authenticity is guaranteed with the trust in the *CoMaDa* module. The chosen cryptographic algorithms are AES-128 for encryption, SHA-1 for integrity and authenticity during the transport while Elliptic Curve Diffie-Hellmann (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA) are used for the key exchange.

III.) Classification of Network Topology

The natural communication model for WSN is `n:1`, however, SecureWSN also features communication from the subscriber to the sensors, which is unicast (`1:1`). Multicast (`1:n`) to the sensors is not yet implemented.

The concept of the gateway is two-folded (see Figure 2.2). An *Opal Mote* is the publisher of the data including a Trusted Platform Module (TPM) and therefore is able to perform a customized DTLS handshake with the gateway [82]. Second part of the gateway is a conventional computer, attached with an IEEE 802.15.4 networking interface, which prepares the data for the subscribers. The Access Control Server is either part of of *CoMaDa* or an external CA, hence, trust is provided *centralized*.

The number of nodes can range per specific use case, in the case of Smart Home the size of the building would be the main driver. It can only be a few dozens to a few thousands, however, they would typically form smaller groups under a gateway (sometimes called tree-of-trees [74]).

2.3.2 Use Cases 2: Mobile Ad-Hoc Network (MANET)

Military field communication is an example for MANET with the major application of sharing sensing information among mobile devices [W3]. Sensors would gather information about people, devices and the environment of the soldier wearing the sensor(s). Additionally, networks are set up ad-hoc for mission planning, such that a single sensor does not have the burden of handling the mission on their own. This may be extended by multiple devices entering an area of interest, each of which following its own mission, but requiring collaborative sensor data to cope new or unanticipated requirements.

I.) Classification of Constraints

Communication is typically wireless, ranging from short range, such as IEEE 802.15.4 (Class S1) or BLE (Class S1-S2) to long range networks such as LoRa [165, 178] (Class S0-S2). The nodes are mobile and therefore limited in energy supply for the time in the field (Class E1-E2) which is the main driver of power reduction (Class P1). This in turn limits the available computing resources, but as the hardware is not low-cost as in civil use cases, RAM and ROM are typically limited but not heavily constraint. Exemplary devices are Single Chip Computer such as Raspberry Pi (Class C13) or Smartphones (Class C15) [165].

II.) Classification of Security

Naturally, security is a key challenge in military, as a breach may cause serious risks for the person wearing the sensors. The shared information needs to stay *confidential* and *integer.* There are other use cases such urban sensing for firefighting, police activities, etc. with similar requirements and technology choices. *Data Source Authentication* and *Proof-of-Origin* are especially important and in turn require efficient *Authorization.*

III.) Classification of Network Topology

The nature of ad-hoc networks is a distributed network topology [28], same is true for the trust topology. Keeping the hierarchical order in military in mind, the trust chain can also be hierarchical (*decentralized*). As any device needs to communicate with any other device within the ad-hoc network, the communication is distributed. Devices may enter or leave the communication group frequently, hence, the dynamic *randomized.* Communication and especially routing is difficult in ad-hoc networks [74, 166]. Recently, with 300 stable connected nodes, a record has been achieved for MANETs [W3].

2.3.3 Use Cases 3: Device-to-Device Communication (D2D)

The term D2D pictures a wide range of use cases, one of them being autonomous driving. Such cars use sensors to monitor their surroundings but also detect dangers. Tesla®'s Model S - which is the most simple version - comes with dozens of such sensors [W4] for capturing the surrounds, not including those monitoring the car itself. This makes the cars themselves a sensor network.

However, this alone is insufficient for enabling autonomous driving, especially once it comes into scale with millions of such cars. They need to be connected to communicate with services providing data (e.g., traffic), for receiving control information or updates. The communication enlarges when communicating with other cars on the street, which could be only informational but also for safety reasons (e.g., emergency brakes) [3, 48, 64].

I.) Classification of Constraints

In contrast to military communication where the connection is set up ad-hoc (Class S1), cars would additionally access an available network structure, such as LTE [30] (Class S2-S3). Although some of the sensors in a car would be highly computational constrained, they would all gather under a more powerful control unit within the car that serves a the gateway to the network (Class C10-C15). Also power and energy play only negligible roles. Hence, constraints are mostly in terms of networking and mainly driven by the communication time if certain real-time requirements need to be met.

II.) Classification of Security

However, the security requirements are rather strict. *Data Source Authentication* and *Proof-of-Origin* are especially important, which is also highlighted in [64]. In turn, this requires putting attention on *Authorization*. That is specifically true, when considering the number of devices in such scenarios, which can be easily $\gg 1,000$ dynamically forming communication groups while requiring security. Another interesting security property in this scenario is the one of *Proof-of-Receipt*, which might be necessary when it comes to legal liability. Furthermore, as these use cases are highly user-centric, the security property *Proof-of-Origin* comes with privacy issues. However, privacy is left out of the scope of this dissertation.

III.) Classification of Network Topology

The combination of a fixed network such as LTE and ad-hoc communication is typical for D2D and received excessive studies within the last years [3, 48, 64]. The communication profile is `n:m` and the topology is (de-)centralized. Additionally, there are attempts on enabling multicast over D2D [36, 154], allowing dynamic formation of communication groups for a certain interest. One could imagine groups for cars on a certain highway or close to a specific access point or city.

2.4 Summary and Findings

This chapter presents challenges in the area of constrained networks and discusses security properties to be met in such environments. Table 2.1 presents an overview of the use cases' properties elaborated throughout the previous section. The rows detail the discussed properties *Constraints*, *Security* and *Network Topology* for the three fields of application *WSN*, *MANET* and *D2D*, which are depicted in the columns. For each property, we present typically found classifications. In case of multiple use cases, ranges are depicted (e.g., RAM in D2D ranges from class C4 to C13). With SecureWSN, a specific example was given revealing the challenges when deploying cryptography and security protocols in constrained networks. The additional use cases of military communication and autonomous driving are given as examples, where other restrictions apply but the security properties are still challenging.

Although the challenges presented are not new, the analysis indicates the complexity of solving them with state-of-the-art mechanism. Hence, the section also presented a methodology on how to answer RQ 1:

> RQ 1: Which use cases fail using state-of-the-art mechanisms for secure communication or their respective optimizations?

The analysis indicates that the management of trust and the thereby inherent management of keys is a challenge to face. Use case 1 and 3 allow centralized or decentralized trust management, which can be used to distribute and revoke keys with the latter being required for access control. However, even the case of military communication allows the initial setup of trust to be (de-)centralized, as long as eventually distributed management is supported.

The next chapter examines this knowledge and presents state of the art mechanism for exchanging and distributing cryptographic keys. For validation of the examined mechanism, we define the properties for efficiency and hence answering RQ 2:

> RQ 2: What are the requirements to meet *efficiency* in such use cases?

Table 2.1: Overview over derived requirements in the different use cases.

	WSN	MANET	D2D
Constraints			
RAM	C1-C2	C2-C4	C4-C13
ROM	C0-C2	C1-C3	C4-C13
Netwok	S2	S0-S2	S2-S3
Power	P1	P0-P1	P1
Energy	E2	E1-E3	E3
Security			
Confidentiality	✓	✓	✓
Integrity	✓	✓	✓
Authentication	✓	✓	✓
Data Source Authentication	✓	✓	✓
Authorization	✓	✓	✓
Proof of Origin	✗	✓	✓
Proof of Receipt	✗	✗	✓
Topological			
Communication	1:1; n:1	n:m	1:n; n:m
Trust	centralized	decentralized, distributed	decentralized
Dynamic	scheduled	randomized	randomized
Number of Nodes	100-1000	50-300 [W3]	$\gg$ 1000

We will focus on mechanisms being designed for or applicable in the use cases presented in this section, while potentially tackling the challenge of key revocation. Such mechanisms must not reduce the security, while claiming efficiency. With the analysis above an efficient revocation mechanism is considered as one that fulfills one or more of the following efficiency requirements (ER):

ER 1: The networking overhead during communication including the size of the signed message shall not (or only negligibly) increase.

ER 2: The networking overhead for revocation shall be minimal. In the best case this means a maximum of one additional frame on the link layer.

ER 3: The performed cryptographic operation shall not (or only negligibly) increase.

In the remainder of this work, these requirements will be used to pick relevant state-of-the-art in the next Chapter. They serve as the main driver for the design decisions in Chapter 4 and the chosen Identity Based Signature (IBS) schemes in Chapter 5 and 6. Further, the evaluation in Chapter 7 will derive complexity notations based on these requirements.

3 State of the Art and Related Work

Chapter 2 shows a need for efficient security solutions regarding different aspects of secure communication and defines the properties of an *efficient* solution. This work aims at improving signing key revocation, supplementing the other efforts being made during the last decades. Secure communication over a public network requires several mechanisms to work together. This is the task of security protocols, combining transport security, key distribution and trust management with different cryptographic mechanisms. Revocation – which is the focus of this dissertation – is one of the last challenges to be tackled within the lifecycle of a system. That is simply because if the established connection is not secure, revocation is pointless. In that regard, this chapter presents efficient building blocks for this lifecycle found in state-of-the-art network protocols and cryptography.

The necessary fundamentals on cryptography are presented in Section 3.1. This includes a brief introduction to *Provable Security*, which is necessary to analyze the security of the introduced transformations. With that at hand, Elliptic Curve Cryptography (ECC) as a common technique for efficient security solutions in the presented use cases is introduced in Section 3.2. It is also an important building block for the development of Identity Based Signature (IBS) and for the revocation mechanism's efficiency, developed later in this work.

Development of an efficient but meaningful revocation mechanism requires understanding the efforts made for efficient transport security, key agreement/distribution, but also its underlying cryptographic mechanisms. Only recently, RFC 8576 [146] was published and gives an overview of the standardization efforts within the last years, but also states challenges not yet tackled. The aim of Section 3.3 and 3.4 is to explain these efforts. A special focus is put on group key management in Section 3.5, which plays a central role for the efficiency of the later introduced transformation.

Section 3.6 will close this chapter by discussing related work for revocation, being found in standardization and research. They can be split into two main categories, as mechanisms utilizing *knowledge* or *mathematical* approaches.

3.1 Fundamentals of Cryptography

A definition for *Cryptography* can be found in the Concise Oxford Dictionary [163]:

> *The art of writing or solving codes*

Although historically accurate, it does not illustrate the challenges being tackled by cryptography in a digitized world. A modern definition is given in [76]:

> *The scientific study of techniques for securing digital information, transaction and distributed computations.*

While Oxford Dictionary's definition solely pictures the *art* of creating codes – which refers to making texts unreadable by a third party – this definition uses the term *scientific study of techniques for information security.* As presented in Chapter 2, various standards have defined the goals for *information security*. Hence, *studying* cryptography is about developing techniques to achieve such goals for various applications, but also to understand their security. We distinguish cryptographic algorithms for the properties given by [71]:

I.) **encryption** for achieving *confidentiality*

II.) **hashing** for achieving *integrity*

III.) **digital signatures** for achieving *authenticity* and *non-repudiation*

Authorization can be achieved by a combination of those three.

For *integrity*, hashing an information is sufficient, which is not true for *encryption* and *signature* algorithms, which require keys. There are several algorithms on how such keys are used and generally we distinguish:

1. **symmetric** algorithms, where encryption and decryption (respectively signing and verification) is achieved with a single key shared by all participants.
2. **asymmetric** algorithms, where two keys with a mathematical link are used. We distinguish:
   a) **private key** which has to be kept secret by the user and is used for decryption and signing
   b) **public key** which can be shared publicly and used for encryption and verification

This holds a potential risk, as – in theory – the mathematical link between private and public allows the computation of the former from the latter. To overcome this issue, the link is created by mathematical problems, which are assumed difficult to solve even with the largest computers on the planet. Such problems are called *computationally hard* and some of them are found in group theory.

Next, this section outlines the fundamentals of group theory and explains the thereof resulting hard problems for cryptography. With the example of the Diffie-Hellmann protocol the development of a cryptographic algorithm is explained and the relation of different problems is shown. Understanding the security of developed algorithms is the goal of *Provable Security*, which will be introduced as well.

3.1.1 Group Theory

Basic knowledge of group theory is helpful to understand the transformations applied in Chapter 5, which are all based on elliptic curve groups (see Section 3.2). So-called *Finite Abelian Groups* are commonly used in cryptography.

Definition 3.1 (finite abelian group [99])**.** *An abelian group is a set $\mathbb{G}$ together with a binary operation on $\mathbb{G}$ such that the following axioms hold:*

(i) $a * (b * c) = (a * b) * c$ for all $a, b, c \in \mathbb{G}$ (associative law);
(ii) $a * b = b * a$ for all $a; b \in \mathbb{G}$ (commutative law);
(iii) there is an identity (or neutral) element $\mathbb{1} \in \mathbb{G}$ such that $a * \mathbb{1} = a$ for all $a \in \mathbb{G}$;
(iv) for each $a \in \mathbb{G}$, there exists an inverse element $a^{-1} \in \mathbb{G}$ such that $a * a^{-1} = \mathbb{1}$.

The abelian group $\mathbb{G}$ is called a finite abelian group if it has only finitely many elements. The number of elements of the finite abelian group $\mathbb{G}$ is called the order of $\mathbb{G}$.

The special case of cyclic groups is of special interest for cryptography, as they allow so-called *trapdoor functions* which are useful for asymmetric cryptography:

Definition 3.2 (cyclic finite abelian group [99])**.** *A finite abelian group $\mathbb{G}$ is called cyclic if there exists an element $g \in \mathbb{G}$ such that every element of $\mathbb{G}$ is a power of g. The element g is called a generator of the finite cyclic group $\mathbb{G}$. We also say that $\mathbb{G}$ is the cyclic group generated by g and we write $\mathbb{G} = \langle g \rangle$.*

One example of a cyclic finite group which is important for cryptography is the *modular multiplicative group* [60, 181]. It is defined as $(\mathbb{Z}_p^*, \cdot)$, $p \in \mathbb{N}$, with $\mathbb{Z}_p^* = \{1, 2, \cdots p-1\}$ and a generator $g \in \mathbb{Z}_p^*$ such that $g^x = 1 \mod p$, with $x = p - 1$. Given a generator $g \in \mathbb{Z}_p^*$ and $m \in \mathbb{Z}_p^*$, the smallest number $x \in \mathbb{Z}_p^*$ such that $g^x = m \mod p$ is a discrete logarithm:

Definition 3.3 (Discrete Logarithm [99])**.** *Let $\mathbb{F}_q$ be a finite field and let $g \in \mathbb{F}_q^*$ be a primitive element of $\mathbb{F}_q$. For each $a \in \mathbb{F}_q^*$, the unique integer h with $0 \leq h \leq q-2$ such that $g^h = a$ is called the discrete logarithm (or the index) $ind_g(a)$ of a to the base g.*

3.1.2 Computationally Hard Problems

Relative to today's available classic computers, some mathematical problems are called **computationally hard**. We say that a problem is hard, if there is no known algorithm, solving the associated problem in polynomial time. With the different computer model in the field of *Quantum Computing*, this assumption changes for some problems.

The problem of *prime factorization* is commonly known due to its use in the *RSA* algorithm, which was developed 1974 by Rivest, Shamir and Adelman [147]. It is based on the problem of finding all prime factors of a significantly large integer and is most difficult for the product of two similar sized prime numbers.

With the discrete logarithm in cyclic groups there is the so-called Discrete Logarithm Problem (DLP), which is similarly important for modern communication:

Definition 3.4 (Discrete Logarithm Problem [99])**.** *For a cyclic group $\mathbb{G}$ with a generator g, given g, g^a, with $a, g^a \in \mathbb{Z}$ the problem of computing a is known as the Discrete Logarithm Problem.*

Many cryptographic algorithms build upon the hardness of the DLP, notable for this work are the Digital Signature Algorithm (DSA) [8] and Schnorr-signatures [153].

Similarly, the Diffie-Hellmann protocol [34] is based on the DLP and allows two parties (typically called Alice and Bob) to agree on a shared secret k, which can then be used for symmetric algorithms. It assumes that both use the same generator g of a cyclic finite group $\mathbb{G}$. With the example of the cyclic group $\mathbb{Z}_p$ with p being prime, that works as follows:

Step 1: Alice chooses a random integer $a \overset{r}{\leftarrow} \mathbb{Z}_p$, with $2 \leq a \leq p-2$. Bob chooses a random integer $b \overset{r}{\leftarrow} \mathbb{Z}_p$, with $2 \leq b \leq p-2$.

Step 2: Alice sends $g^a \in \mathbb{Z}_p$ to Bob, while Bob sends $g^b \in \mathbb{Z}_p$ to Alice.

Step 3: The common key $k = g^{ab} \in \mathbb{Z}_p$, which Alice computes as $(g^b)^a \in \mathbb{Z}_p^*$ and Bob computes as $(g^a)^b \in \mathbb{Z}_p$

That a third party (commonly denoted as *Eave*) is not able to compute k given g, g^a, g^b, is known as the Computational Diffie-Hellman Problem (CDH).

Definition 3.5 (Computational Diffie-Hellman Problem [99])**.** *For a cyclic group $\mathbb{G}$ with a generator g, given the tuple g, g^a, g^b, with $a, b \in \mathbb{Z}$ the problem of computing g^{ab} is known as the Computational Diffie-Hellman Problem.*

This allows the definition of another problem, called Decisional Diffie-Hellman Problem (DDH). It can be exploited for the so-called bilinear maps in Section 3.2.2

Definition 3.6 (Decisional Diffie-Hellman Problem [99])**.** *For a cyclic group G^* with generator g, given the tuple g, g^a, g^b, g^z, with $a, b, z \in \mathbb{Z}$, the problem of deciding whether or not $g^z = g^{ab}$ is known as the Decisional Diffie-Hellman Problem.*

3.1.3 Provable Security

One important aspect of *cryptography* is the study of developed mechanisms regarding their security. This is commonly known as cryptanalysis and *Provable Security* is a technique for identifying theoretical issues during the construction of cryptographic algorithms.[1]

Constructing cryptographic schemes and proving their security has many flavors, but builds on the following principles [76]:

Principle 1: Formulation of a rigorous and precise definition of security.

Principle 2: When the security relies on assumptions, they must be precisely stated and should be as minimal as possible.

Principle 3: Constructions should be accompanied with rigorous proof of security with respect to the definitions and assumptions (if any).

I.) Defining a security model

Important for Principle 1 is the clear definition of an attacker model in so-called experiments defining the attacker's abilities. The attacker is the so-called adversary, denoted as $\mathcal{A}$. Security goals of the algorithm need to be stated, which for signature schemes is *existential*, *selective*, or *universal* unforgeability. A successful *existential forgery* is given, if the adversary can produce one valid pair of signature/message without being the legitimate signer. This is meant to be the strongest security goal for a signature scheme. For the proof, $\mathcal{A}$ is allowed to query so-called Oracles (denoted as $\mathcal{O}$) defining her power. Oracles can be described as functions, $\mathcal{A}$ can call during the game to obtain information. Typical attacker capabilities are grouping such Oracles to achieve a certain security level. For signature, that is:

key only: $\mathcal{A}$ receives the verification key (public key).

known-message: $\mathcal{A}$ receives a list of pre-selected message/signature pairs.

adaptive chosen-message: $\mathcal{A}$ adaptively obtains signatures for messages of her choice.

The latter allows $\mathcal{A}$ to obtain information about how the signature may look for other messages and pictures a real-world attacker who silently listens conversations with the goal of forging future (unknown) messages. That defines the strongest tools of an adversary, as she is allowed on receiving adaptively changing information, except the signing key. The goal is to prove, that a signature scheme is secure, even if the adversary is that strong and signature scheme being *existential unforgeable under adaptively chosen-message attacks.*

[1] The aim is a brief outline of the fundamentals rather than a comprehensive introduction. We refer the interested reader to one of the following, which have been a great source of inspiration [29, 75, 76, 181].

II.) Security Assumptions

It is almost impossible to unconditionally prove the security of a scheme [76], why defining assumptions is important according to Principle 2. Thus, some assumptions have become common practice in cryptography, most notable for this thesis is the assumption saying that the DLP is computationally hard. It has been studied for years and is clearly defined, even against the threat of quantum computers. Another one – which typically applies to signatures – is the one assuming the randomness of hash functions, referred as the *random oracle model.* In a nutshell, it assumes that the output of a hash function is perfectly random and therefore unpredictable for the adversary. Typically, schemes would not define the actual hash function to be used, hence, it is exchangeable in case of any weakness.

III.) Reduction Proofs

Satisfying Principle 3, all proofs in this work re-use assumptions from the original schemes and are defined in that regard. Proving them is done with the technique of security reduction, meaning that the original assumptions are not hurt after a schemes transformation. Particularly, we prove by contradiction, which – at a high-level view – is done as follows [60]:

1. We *assume* a mathematical problem to be hard.
2. We prove: if a proposed scheme is insecure, that this hard problem then is easy by the output of the proposed scheme.
3. The *assumption* of the hard problem would then be false, in turn the scheme is secure.

Technically, this is done by playing a game with two adversaries (e.g., $\mathcal{A}, \mathcal{B}$), $\mathcal{A}$ following the experiment of a hard problem (e.g., DLP), $\mathcal{B}$ following the experiment of the proposed scheme. $\mathcal{A}$ can call $\mathcal{B}$, who wins her game, and $\mathcal{A}$ uses $\mathcal{B}$'s output to win her own experiment in polynomial time. With the example of the Diffie-Hellman protocol presented in Section 3.1, the conclusion could be:

> *If $\mathcal{B}$ is able to win her experiment and solve CDH, $\mathcal{A}$ can solve the DLP in polynomial time.*

With the so-called *Game Hopping Lemma* [157] it is possible to reduce the security to another scheme, which in turn reduces to a hard problem. As the transformations in Chapter 5 are based schemes with rigorous proofs, this technique will be used (where possible).

3.2 Elliptic Curve Cryptography

Elliptic curves have been introduced for the use in cryptography in 1985 [80, 97]. Even in applications like Browsing moving from the most famous algorithm *RSA* [147] to algorithms based on ECC is found common practice and many modern protocols were extended for using these algorithms. For constrained scenarios, it is considered more less inevitable to cope with the restrictions. This is mostly because they allow smaller key sizes featuring smaller memory footprint and better operation performances for the same security as RSA [143].

The transformations in Chapter 5, require familiarity with the calculation rules on elliptic curves. Additionally, two of the transformed schemes are based on so-called *Pairings* (also called bilinear maps) with special rules, which are outlined as well. It is further important to understand the underlying computational problems, making them a valuable solution for cryptographic purposes.

3.2.1 Group Definitions on Elliptic Curves

An elliptic curve can be defined over different fields F such as $\mathbb{R}, \mathbb{C}$ or $\mathbb{F}_q$ (q is prime) with the equation [181]:

$$y^2 = x^3 + ax + b \tag{3.1}$$

Let $\mathbb{O}$ be the point "at infinity" and $a, b \in F$ with $4a^3 + 27b^2 \neq 0$, then the elliptic curve E is given by [181]:

$$E = \{(x, y) \in F \times F \mid y^2 = x^3 + ax + b \wedge 4a^3 + 27b^2 \neq 0\} \cup \{\mathbb{O}\} \tag{3.2}$$

Elliptic curves are interesting for cryptography, as they naturally carry a (commutative) group structure as given by Definition 3.1 and 3.2. To define a cyclic group over the elliptic curve, we require an associative group law with a corresponding inverse and a neutral (identity) element. Figure 3.1 plots the elliptic curve E for $y^2 = x^3 - x + 1$ over $\mathbb{R}$ three times for illustration of the following rules [181]:

neutral element (identity element) is the point "at infinity", denoted as $\mathbb{O}$ in Figure 3.1.a.

Inverse Elliptic curves are symmetric about the x-axis. Therefore the inverse of point $P = (p_x, p_y)$ is well-defined as $-P = (p_x, -p_y)$, which Figure 3.1.a illustrates as the point's mirror image reflecting at the x-axis. $-\mathbb{O}$ is defined as $-\mathbb{O} = \mathbb{O}$.

Group Operation For the group operation $+$, consider a line through P and Q shown in Figure 3.1.b and its intersection with E as the point S. We define $+$ as the inverse of this intersection, such that $P + Q = -S =: R$. The special cases are:

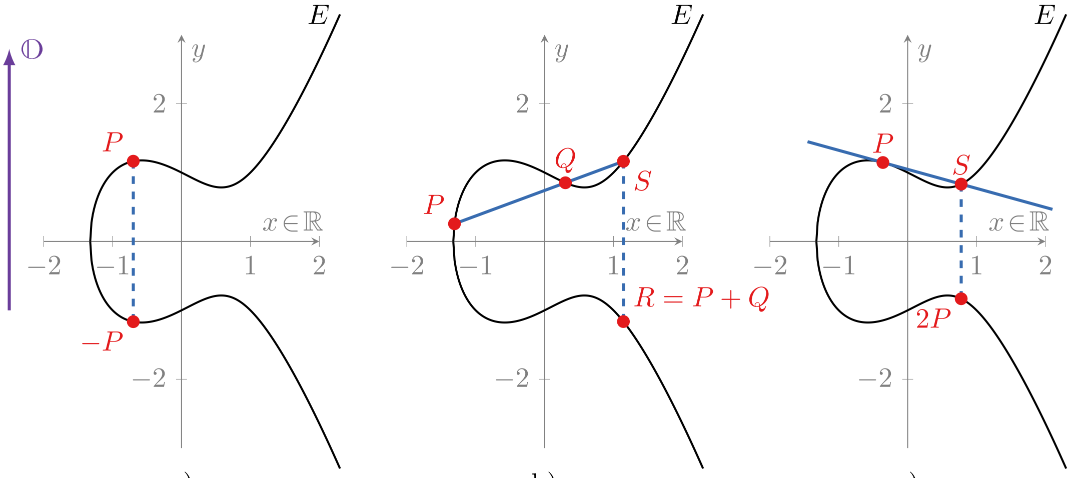
1. if $Q = P$: Take the tangent line at P and its intersection with E as in Figure 3.1.c.
2. if $Q = \mathbb{O}$: Take the vertical line through P, that is $P + \mathbb{O} = -(-P) = P$.
3. if $Q = -P$: Take the vertical line through P and $-P$, that is $P + (-P) = -\mathbb{O} = \mathbb{O}$.

Additionally, $k\,P$ with $k \in \mathbb{Z}$ is adding P exactly k times, particularly $0\,P = \mathbb{O}$ and $-k\,P = -(k\,P)$ for $k > 0$.

The assumption is that given $k\,P \neq \mathbb{O}$, it is computationally hard to determine k which is referred to as the Elliptic Curve Discrete Logarithm Problem (ECDLP) [181]:

Definition 3.7 (Elliptic Curve Discrete Logarithm Problem [181])**.** *Given the generator P of an elliptic curve group of size m and $A = aP$ ($a \in \mathbb{Z}_m$), the task of computing a from A and P is known as the Elliptic Curve Discrete Logarithm Problem. This problem is assumed to be computationally hard.*

For cryptography, the typically used representation of an elliptic curve is the one defined by the *short Weierstrass equation* over finite fields $\mathbb{F}_q$, with p being prime:



a)
Neutral Element $\mathbb{O}$
Inverse element $-P$

b)
Addition $R = P + Q$

c)
Doubling $P + P$
"Tangent rule"

Figure 3.1: Illustration of group operations on elliptic curves with the example elliptic curve E for $y^2 = x^3 - x + 1$ over $\mathbb{R}$ (inspired by [W5]).

Definition 3.8 (Elliptic curve in Weierstrass form [181])**.** *Let $\mathbb{F}_q$ be a finite field of prime characteristic; $q \in \mathbb{Z}$, $q > 3$ and $a, b \in \mathbb{Z}$. Then the elliptic curve over $\mathbb{F}_q$ consists of all points $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$, that fulfill:*

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

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

Let $\mathbb{O}$ be the point "at infinity", then the elliptic curve is given by:

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

3.2.2 Pairings on Elliptic Curves

Pairings are special functions on cyclic groups and are also called bilinear mappings. For the context of this work, it is only of relevance that such functions exist and that their computational rules are defined as follows:

Definition 3.9 (Pairings [9])**.** *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be groups of the same prime order. A pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ has the following properties:*

1. *Bilinearity: $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \in \mathbb{Z} : e(a\,P, b\,Q) = e(P, Q)^{ab}$*
2. *Non-degeneracy: $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2 : e(P, Q) = \mathbb{1} \Leftrightarrow P = \mathbb{O} \vee Q = \mathbb{O}$*
3. *Computability: $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$, there is an efficient algorithm to compute $e(P, Q)$.*

Pairings allow the construction of groups where the CDH is hard, but the DDH is easy (see Definition 3.5 and 3.6). Such groups are called *Gap Diffie-Hellman Groups.* Consider the following example for elliptic curve groups:

Let $a, b, z \in \mathbb{Z}$, $P \in E(\mathbb{F}_q)$, and $e(P, P) \in \mathbb{Z}_p^*$ be a pairing. Assume $P, a\,P, b\,P, z\,P$, and $s = e(z\,P, P)$ (e.g., as a signature) are given, but a, b, z are unknown. In $E(\mathbb{F}_q)$, the CDH, that is computing $ab\,P$, is hard, while deciding whether or not $ab\,P \overset{?}{=} z\,P$ (DDH), becomes easy with pairings. We compute $s' = e(a\,P, b\,P)$, which is equal to s if and only if $ab\,P = z\,P$:

$$s' = e(a\,P, b\,P) = e(P, P)^{ab} = e(P, P)^{z} = e(z\,P, P) = s \tag{3.3}$$

Efficient pairings exist (famous are the ones of *Weil* and *Tate* [29]) and are used for short signature schemes and Identity Based Cryptography (IBC). However, they are widely accepted to be computationally expensive. The literature calculates with $10-21$ elliptic curve operations for one pairing operation [9, 52].

3.3 Efficient Cryptographic Mechanisms

Confidentiality, *integrity*, *authenticity* can be achieved in cryptographic means. Efficient cryptographic algorithms for each of them are outlined in the following, closing with efficient key agreement mechanisms based on the Diffie-Hellman key exchange.

3.3.1 Efficient Confidentiality Solutions

Although encryption can be achieved by using asymmetric cryptography, symmetric algorithms are much more efficient in both, computational complexity and key sizes. This fact is shown in Table 3.1, where the key sizes for DLP/RSA (central column) and ECDLP (right) are compared to those of symmetric keys (left column). The most common algorithm for encryption is found in Advanced Encryption Standard (AES), which offers three of the four presented key-sizes of *128, 192* and *256* as a security parameter and the table presents the respective asymmetric key sizes. As *1024* Bit RSA keys can still be used in some settings, Table 3.1 shows its comparison with an *80* Bit symmetric key. Additionally, AES comes with different operation modes of block (e.g., CBC, CCM) and streaming ciphers (e.g., CTR, GCM). As block ciphers may require padding, the latter are more efficient regarding network overhead and the resulting ciphertext has the same size as the plaintext. ChaCha [15, 144] is another family of stream cipher that became popular during the last few years, mainly because it offers efficient computation in software [150]. ChaCha, as well as AES-CCM and AES-GCM are so-called Authenticated Encryption with Associated Data (AEAD) algorithm, which offers authentication with an included Keyed-Hash Message Authentication Code (HMAC).

The major efficiency advantage of AES over other symmetric ciphers comes with its broad acceptance. As it is relatively easy to be implemented in hardware, many modern chips have a native instruction set for AES. During the standardization of IEEE 802.15.4 [70] it was included in the specification for link layer security. This is why such radio chips often provide an hardware implementation and even the most constrained sensors may have access to hardware accelerated encryption.

3.3.2 Efficient Integrity Solutions

A message's integrity is secured by cryptographic hash functions. The most prominent ones are those standardized under the name Secure Hash Algorithm (SHA). While SHA-1 is considered insecure and was only recently practically broken [162], its successors SHA-2 [32] and SHA-3 [39] are widely accepted.

3.3.3 Efficient Authentication Solutions

Communication of two trusted participants can be authentic with the use of symmetric cryptography and is therefore a commonly used technique in security protocols. However, this is not

Table 3.1: Comparison of key sizes (in Bits) for symmetric algorithms with asymmetric based on DLP, ECDLP [160].

Symmetric Key	DLP/RSA	ECDLP
80	1,024	160
128	3,072	256
192	8,192	384
256	15,360	512

possible over a public network such as the Internet, why authentication based on asymmetric cryptography is required as well.

I.) Authenticity by symmetric cryptography

One technology being widely used is called HMAC [118]. As the name suggests, cryptographic hash function are used and extended with a key to provide authenticity. Obviously, this also provides integrity why in most cases – if a shared secret exists – HMAC will be used to provide integrity as well.

The commonly used algorithms are SHA-1 and SHA-2, which seems odd as SHA-1 is meant to be insecure. However, in the combination with a strong key, there is no insecurity in constructing an HMAC with SHA-1. The security level is actually as strong as the block size of the used hash algorithm, which in the case of SHA-1 is 160 Bit.

Due to the potential hardware acceleration, the AES-CBC-MAC became of interest for the use on constrained hardware. As the original version is insecure for messages of variable length (see e.g., [76, Chapter 4.5]). The variant called AES-XCBC-MAC [18] offers better security. It is used in the AEAD algorithm AES-CCM, but can be used also used for authentication only. Another variant taking advantage of an AES native instruction set is the Galois MAC, which is used in the mode AES-GCM [38].

II.) Authenticity by asymmetric cryptography

Digital signature based on a asymmetric signing algorithm allow *authenticity.* The by far most well-known algorithm is RSA [8, 147], but it requires large keys of at least 2048 Bit for a decent level of security. Another well-known algorithm is DSA [8], which – in contrast to prime factorization in RSA – is based on the DLP and therefore applicable for elliptic curves. Consequently, constrained environments were among the first utilizing the Elliptic Curve Digital Signature Algorithm (ECDSA) [8] in a large scale. It only requires keys of 254-382 Bit for the same level of security (see Table 3.1), depending on the used curve.

Another signature algorithm is the one introduced by Schnorr [153]. It is very similar to DSA, can be used with ECC and offers an efficient way for generating an IBS algorithm.

Hash based signatures can provide *Authenticity* but do not require asymmetric cryptography while the security solely depends on the used hash function.[2] They are typically so-called *one-time* or *few-time* signature schemes. One example is Tesla [102] and its spin-offs, offering asymmetry by delayed authentication but only relying on the security of hash functions. It is therefore quite efficient and was even optimized for the use in Wireless Sensor Networks (WSNs) [104]. However, the delayed authentication results in a serious DoS vulnerabilities and was enhanced thereby [103, 113, 114].

[2] With the security only relying on hash functions, hash based signature are also considered a serious candidate for Post Quantum Cryptography, but this remains out of scope of this work.

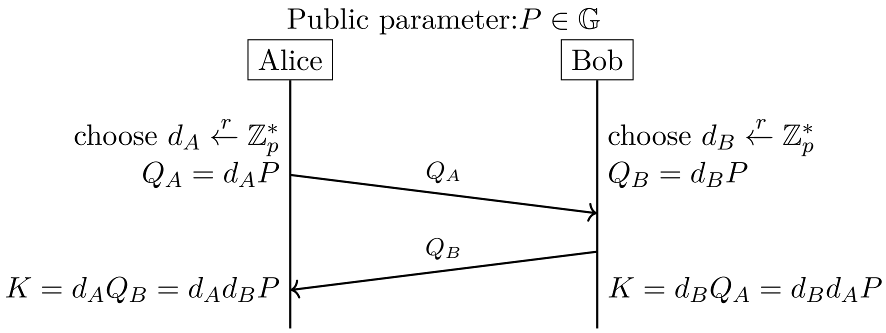

Public parameter:$P \in \mathbb{G}$
Alice
Bob
choose $d_A \overset{r}{\leftarrow} \mathbb{Z}_p^*$
$Q_A = d_A P$
choose $d_B \overset{r}{\leftarrow} \mathbb{Z}_p^*$
$Q_B = d_B P$
Q_A
Q_B
$K = d_A Q_B = d_A d_B P$
$K = d_B Q_A = d_B d_A P$

Figure 3.2: Elliptic Curve Diffie-Hellmann (ECDH).

3.3.4 Key Agreement

Agreeing on a shared secret is a necessity for algorithms providing *confidentiality* and *authenticity* based on symmetric keys. In principle, there are two distinguished ways to achieve this goal, called Key Encaspulation Mechanism (KEM) and Key EXchange (KEX).

A KEM uses an asymmetric encryption algorithm and takes its communication partners public key to encrypt and send a random key. By using specific formats on how to send the key, this method is secure but has a major drawback: The key being used is chosen by only one participant and the other has no impact on the security of the used key.

Even though KEMs were widely used (e.g., in TLS 1.2 [127]), the drawbacks resulted in a move to so called KEX in protocols like TLS 1.3 [145] or IKEv2 [136]. A famous representative is thereby found in the Diffie-Hellman protocol. The initial idea was based on the CDH problem (see Definition 3.5), which can be adopted to ECC and allows the same reduction of public parameters sizes as shown for ECDSA in Table 3.1. Hence, the Elliptic Curve Diffie-Hellmann (ECDH) presented in Figure 3.2 is an interesting solution for efficient key agreement and works as follows:

Step 0: Alice (left) and Bob (right) agree on a public generator P of the elliptic curve group $\mathbb{G} \setminus \mathbb{O}$ and the cyclic group $\mathbb{Z}_p^*$ with p being prime.

Step 1: Alice chooses a random integer $d_A \overset{r}{\leftarrow} \mathbb{Z}_p^*$.
Bob chooses a random integer $d_B \overset{r}{\leftarrow} \mathbb{Z}_p^*$.

Step 2: Alice sends $Q_A = d_A\, P \in \mathbb{G}$ to Bob, while Bob sends $Q_B = d_B\, P \in \mathbb{G}$ to Alice.

Step 3: The common key $K = d_A d_B\, P \in \mathbb{G}$, which Alice computes as $d_A(Q_B) \in \mathbb{G}$ and Bob computes as $d_B(Q_A) \in \mathbb{G}$

3.4 Efficient Security Protocols

Efficiently securing communication is achieved by a combination of asymmetric and symmetric cryptography. With asymmetric algorithms being less efficient, the goal is to use them only for setting up a channel secured by symmetric algorithms [22]:

1. Perform a key exchange (e.g., Diffie-Hellmann) to agree on a shared symmetric key

2. Authenticate the key exchange (e.g., with DSA) to prevent Man-in-the-Middle (MITM)

3. Use the symmetric key for *confidentiality* (e.g., with AES) and *authenticity* (e.g., with HMAC) during the communication.

There are numerous protocols following this principle, most of them split the communication into two phases: *Authenticated Key Agreement* (1 and 2) and *Secure Transport* (3). Some suites like IP security protocol (IPsec) separate with two protocols, while others like Transport Layer Security (TLS) separate within a single protocol format. There have been various efforts in optimizing some of these protocols for the use in constrained systems, but also development in new protocols. One common form of optimization is the use of compression, which mainly deals with the network restrictions, such as the Maximum Transfer Unit (MTU). Another one is minimization of the protocol or defining new profiles, while keeping it standard conform.

The following two sections briefly outline efforts for optimization of standard protocols found in usual Internet traffic and protocols explicitly designed for the considered use cases.

3.4.1 Protocol Optimization and Compression

There have been several proposals to optimize existing protocols. Among others, [65] describe the security challenges of Internet of Things (IoT) with special focus on IP based communication, while providing a functional comparison of TLS/DTLS with IPsec/HIP. This leads to different architectures and solutions for IP and transport layer security [49, 82, 111, 112, 173]. In turn, this encouraged the major standard body for IP protocols – the Internet Engineering Task Force (IETF) – to publish minimization guidelines and optimizations for the protocols TLS, DTLS [140], HIP [98], IKEv2 [139] and ESP [92, 95].[3] Internet Key Exchange (IKEv2) is thereby the preferred key exchange protocol for the IPsec suite and provides an authenticated Diffie-Hellman key exchange, while Encapsulated Security Payload (ESP) provides transport security by using these keys. Similarly, the Host Identity Protocol (HIP) [138] provides the same properties, while using explicit identifiers which allows better efficiency in some contexts. Both are not IPsec specific and can be used e.g., to provide key exchange for other layers such as IEEE 802.15.4. With most application protocols in the considered scenarios being based on UDP, Datagram TLS (DTLS) (which is the UDP based variant of TLS) became the most prominent solution.

Pure optimization is of interest to stay compatible with the standard. However, protocols are typically designed for common usage, which is why there is a natural limit for standard conform optimization. Compression is a common approach to overcome these limitations and has been used for different purposes [125, 129]. In that sense, there were different approaches to adopt compression to security protocols, again, mostly for IPsec [94, 108] and TLS/DTLS [109, 110]. Although there are efforts in standardization, at the time of writing they are still mostly academic.

3.4.2 Constrained Security Protocols

In contrast to the protocols presented before, development of security protocols especially designed for constrained scenarios is a popular alternative. While [64, 78] offer an overview over the academic approaches, the IETF hosts two working groups developing standardized mechanisms, namely CBOR Object Signing and Encryption (COSE)[4] and ACE[5]. COSE aim on offering a security mechanisms for the application layer, in particular for the CBOR data format, which is an extension of JSON [141]. They currently focus on data formats for signing and encryption, while including various standard such as X.509 [128], but it does not deal with key exchange.

[3] Please note, that [92, 98] are not published standard documents and could be subject of change or removal at any time [115]. However, they are adopted as so-called *working group* documents and are therefore expected to survive.

[4] CBOR Object Signing and Encryption (cose): `https://datatracker.ietf.org/wg/cose/`

[5] Authentication and Authorization for Constrained Environments (ace): `https://datatracker.ietf.org/wg/ace/`

The focus of ACE, however, is broader and aims on developing an applicable solution for authentication and authorization. In particular, this includes mechanisms for authenticated key exchange, architectural assumptions for trust management and the adoption and development of data formats such as X.509, OAuth [132].

3.5 Efficient Group Key Management

Two facts make group key distribution interesting for the use cases. First, there is the requirement for group communication, e.g., in form of IP multicast which might be preferred over unicast [137, 159]. Second, managing trust for authentication and authorization requires some form of key distribution, why the architectural assumptions are very similar.

Sharing a symmetric key among a group that can be used for efficient transport security inherits some challenges not found for two-party protocols discussed in the previous section. Consequently, there are specialized protocols as well as cryptographic mechanisms. They can be split into *centralized*, *decentralized* and *distributed* approaches [107]. All of them have the same goal, that for secure communication over public networks the group members agree on a group secret, often called Group Transport Encryption Key (GTEK).

In dynamic groups where members join and leave the group frequently, this secret has to provide the following security features [121]:

Forward Access Control Whenever a group member leaves the group or is expelled, the member in question must afterwards not be able to access a valid group key.

Backward Access Control Whenever a new group member joins a group, the member in question must not be able to access a formerly valid group key.

Key Independence Having access to one key must not yield the possibility to deduce other keys or any valuable information about them.

A group key management architecture as presented in Section 3.5.1 aims on providing those properties as a *centralized* or *decentralized* approach. The cryptographic mechanisms presented in Section 3.5.2 allow optimizations and *distributed* approaches.

3.5.1 Group Key Management Architecture

A group key management architecture is typically centralized and the most obvious way of managing group keys as it leaves the complexity and trust to a single system. Figure 3.3 shows the two roles of a Group Controller Key Server (GCKS) (left) and a Group Member (GM) (right) and three communication channels, namely *private*, *secure group* and *public* channel (depicted bottom up). The figure also depicts the necessary modules (*Controller* for authorization, *Credentials* for authentication) and databases (Group Security Association (GSA) for cryptographic material and Group Security Policies (GSP)) of the two roles according to RFC 2093 [116] and RFC 2094 [117]. A GCKS manages multiple GMs and validates their credentials and authorization upon joining or leaving the communication group.

The private channel is secured with the Key Encryption Key (KEK) (an individually shared secret between a GM and the GCKS). Forward and backward access control can be achieved through a so-called Group Key Encryption Key (GKEK) – which is depicted as part of the GCKS' and GM's GSA in Figure 3.3 – is used to distribute cryptographic material among the group members. The *private channel* is used to re-new and distribute the GKEK to exclude specific GMs from the group. The *secure group channel* is protected with the GKEK, which in turn allows distribution of the GTEK for secure communication among the GMs. Both keys are called the GSA and the GSP defines how they are used. However, forward access control

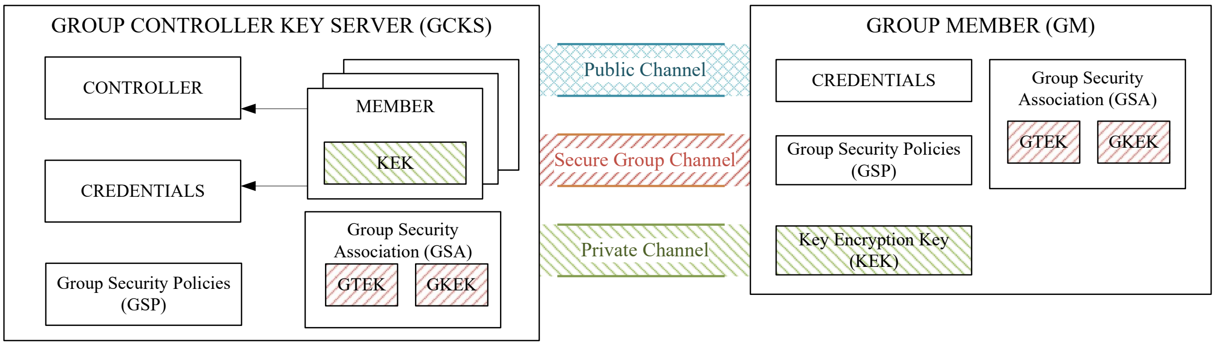

GROUP CONTROLLER KEY SERVER (GCKS)
CONTROLLER
MEMBER
KEK
CREDENTIALS
Group Security Association (GSA)
GTEK
GKEK
Group Security Policies (GSP)
Public Channel
Secure Group Channel
Private Channel
GROUP MEMBER (GM)
CREDENTIALS
Group Security Association (GSA)
GTEK
GKEK
Group Security Policies (GSP)
Key Encryption Key (KEK)

Figure 3.3: Group key management architecture as in RFC 2093 [116] and RFC 2094 [117].

needs attention as expelled members have access to relevant cryptographic material prior to their exclusion.

Hence, the concept of centralization inherits some natural difficulties, such as weak scalability, especially when only one server manages a geographically distributed network. In general, the larger a group gets, the more complex becomes its management and the resulting operations. However, it fits nicely in the use cases' architecture and will be re-used as an underlying architecture for the developed revocation mechanism in Chapter 4.

I.) Decentralized

In contrast, decentralized techniques share the management of the keys between several instances [25]. Thereby, the generation and distribution of group keys is realized by cooperative instances, which are typical hierarchically ordered. In addition, distributed key agreement procedures delegate the key generation process to not only an individual group member, but to a group of members.

Decentralized key distribution still sticks to the concept of a central server, but splits the group in administrative domains for both management operations and key exchanges. Additionally, it distributes the workload over more devices and thus eases the key calculations on the client side. On the other hand, the decentralized concept requires strong trust relationships, which is a showstopper for many use cases where administrative domains can change frequently and devices are potentially physically accessible by arbitrary persons.

II.) Protocols

Most of the commonly used protocols are designed on top of centralized systems. The establishment of the KEK between GM and GCKS follows a two-party key exchange protocol, such as IKEv2 or DTLS/TLS. Management of the group channel is included atop the protocol.

There have been different proposals for an efficient combination of protocols fitting the described use cases. Most of them base on DTLS [79, 106, 168] or IKEv2 [53, 59, 148]. In standardization, the *Internet Security Association and Key Management Protocol* (RFC 2408 [119]) and *Group Domain of Interpretation* (RFC 6407 [131]) were the first such instantiations. With IKEv2 as an efficient revision with stronger security properties and better performance, G-IKEv2 [174] is currently proposed for group communications. *Credentials* and *Controller* can be outsourced to an external service (e. g. X.509 for *credentials* and LDAP for *controller*). New standardization follows this principle and provide group key management atop new frameworks like ACE [101, 167, 169, 170].

3.5.2 Group Key Distribution

Decentralized structures often imply a hierarchical structure that has to be maintained. The distributed schemes involve several group members that use their resources to generate the key material [179]. Both can be achieved in cryptographic means.

Different techniques for distributed key agreement were surveyed in [107], e.g., ring based, hierarchical or even unstructured. The *Group-Diffie-Hellman* Key Exchange [161] is a very obvious extension of its corresponding two-party key exchange. There are also structures using more than one key and the general group is divided into subgroups while an elected master of each subgroup is responsible for communication and the subgroup's key generation. This minimizes the re-keying overhead but in contrast, requires additional cryptographic conversions of messages between subgroups.

Decentralized techniques can be used for optimization of centralized systems as well. The most promising one are secret broadcasting and hierarchical structures. Within the broadcast technique, the (de-)central instance transmits a single message containing keys to all members [2, 27, 179]. Hierarchical structure can also be used centralized to optimize the key distribution and especially revocation to cope with *forward secrecy*. A famous technique is Logical Key Hierarchy (LKH) [86, 149], which is standardized for the use in the previously presented management architectures [120]. Both approaches will be presented as an option to allow efficient revocation of signing keys in Section 4.3.1.

Recent development in the area of mobile messaging encouraged standardization of a *message layer security* protocol[6], which combines distributed with hierarchical techniques. However, they are not applicable for all considered use cases.

3.6 Related Work on Signing Key Revocation

Revocation of signing keys is established and there has been various work in both, standardization and academia. However, especially for very constrained and/or dynamic scenarios, practicable solutions are rare. The work can be split into approaches utilizing **1.)** *mathematical revocation* and **2.)** *knowledge based revocation*, a separation which will be re-used for evaluation in Chapter 7. We focus on standard solutions which were adopted to work in constrained environments (similar to the previously presented security protocols) but leave out work which has never been considered for such use cases. Additionally, we examine some academic solutions which could be adopted.

3.6.1 Mathematical Revocation

In the context of this thesis we consider *mathematical revocation* a mechanism, which allows a signature being mathematically invalid if the signers key was revoked. Theoretically this does not involve extra computation or networking overhead for signer and receiver. All complexity lies in the revocation mechanism that, simplified, allows mathematically and provably "stealing" the signer's key. Sticking to the analogy, this is not only difficult to prove but also requires a very specific trust relationship between the "thief" and the verifier.

We discuss four techniques allowing such revocation, namely

I.) Identity Based Signature (IBS)

II.) Hierarchical Identity Based Signature (H-IBS)

III.) Attribute Based Signatures (ABS)

IV.) Key Insulation

[6] `https://datatracker.ietf.org/wg/mls/`

I.) Identity Based Signature (IBS)

IBS defines a different method for key generation, which changes the trust model compared to e.g., X.509. In typical asymmetric cryptosystems such as RSA/ECDSA, the user's key material is chosen at random and some external infrastructure (like X.509) is used to prove ownership. In 1984, Adi Shamir [155] proposed an asymmetric key system in which the public key is specifically chosen to be mathematically linked to some information about its owner. Typically, some identifying data approved by a trusted authority are used for this. A Trusted Third Party (TTP) is required to compute the so-called User Secret Key (*usk*) with a Master Secret Key (*msk*). As usual, signatures are created with the *usk* while encryption is performed with the public key. Similarly, given the Master Public Key (*mpk*), a signature, and the stated signer's identity, any recipient can verify the signature. This makes additional infrastructure for managing individual public keys superfluous. Thus, IBS offers very low network overhead during communication and a lightweight management infrastructure at the expense of a very strict trust relationship between client and TTP.

They additionally can be build with different cryptographic primitives [22]. The example given by Shamir is based on prime factorization, offering well-studied security at the cost of large key sizes [155]. The vast majority of Identity-based schemes rely on ECC, either utilizing Schnorr-signatures or Pairings [7, 9, 12, 24, 164, 177]. IBS schemes using lattice-based cryptography are also known [37, 175].

Based on the changed trust model, IBS offers a naive approach for mathematical invalidation of signatures. The explicit trust relation to the TTP and the mathematical dependency of the keys allow the TTP to re-issue keys to all systems participants, but the revoked user. Although commonly known it was only recently proven to work in constrained environments featuring group communication [52]. While the idea is very efficient during the actual communication and applicable for the considered scenarios, the revocation mechanism is highly inefficient. It requires the TTP to calculate a new *usk* for each user and send it confidentially via unicast.

II.) Hierarchical Identity Based Signature (H-IBS)

H-IBS offers an hierarchy which – theoretically – can be exploited in a similar way as in the hierarchical key distributions schemes, briefly discussed in Section 3.5. It was first proposed in [51] and allows hierarchical structures based on different roles in the system. The system offers flexible depths of the hierarchy that could be used for example in sensor networks where every node is attached to a router that could act as a so-called *lower-level*-TTP. Logical hierarchies would be possible, but are not studied yet and come at the cost of signatures and *usk*s increasing in size with every new level in the hierarchy. A solution with constant signature sizes is shown in [4], at the cost of larger public keys and lacking flexibility. Moreover, the scheme does not allow renewal of a sub-tree's keying material, which is why the same revocation issues as in centralized IBS schemes occur. Efficient verification is only possible if the communication partners share the same parent entity, reducing the system's flexibility. The more dynamic a system is, the more this *1-to-n-rekeying-effect* (meaning that the renewal of one key triggers n re-keying messages) worsens.

III.) Attribute Based Signatures (ABS)

ABS can be seen as a sibling of IBS that adds attributes to the identifier. In the context of Cloud Computing, Attribute Based Encryption (ABE) received broad acceptance during the last years, as it offers fine grained access control on encrypted data while keeping the master secret key confidential. It was also studied in the context of constrained systems [176].

ABS could allow additional features in comparison to IBS – for example group authentication (n:1), where the recipient is not part of the group. However, the literature for ABS is relatively thin as there are not too many advantages over IBS and one has to deal with management of the attributes in addition to managing identities. Additionally, ABS seems inappropriate for the considered use cases as it often builds upon pairings which are computationally expensive. Exemplarily, we highlight [33], requiring 30 pairings for verification.

IV.) Key Insulation

Key Insulation allows updating keys with respect to a certain period, which is part of the signature [35, 100] . As the defined update of the so-called session keys is similar to extracting a key for IBS it can be seen as mechanism to build IBS schemes [35]. Hence, it offers a somewhat similar idea to what will be shown in the next chapter. However, in the case of key insulation, the updates are dependent on the initially generated key, while they can be independent for the later discussed transformations.

3.6.2 Knowledge Based Revocation

When talking about *knowledge based revocation* we refer to solutions where the verifier of a signature uses external sources for validation, which are other than cryptographic.

There are three different approaches:

- **I.) Public Key Storage** (which is whitelisting)
- **II.) Revocation Lists** (which is blacklisting)
- **III.) Online verification**

I.) Public Key Storage

A public key storage pictures the idea of only accepting signatures of private keys, of which the public keys are known in advance. When the setup is rather static, this is a simplistic and efficient solution, as no – or only simplistic – external trust relation is necessary. For validation of a signature, the verifier simply uses the stored public key of the given identity. If the identity is not known or the verification fails the signature is rejected.

Even though such a solution has obvious problems in large-scale or dynamic systems, there are many examples for use cases in constrained networks utilizing this mechanism. One example which was already presented is the one of *SecureWSN*, where the sensor's (or sensor network's) public keys are pre-installed on the sink receiving the data. However, once the communication is not unidirectional, the number of keys having to be stored on each device grows linear. Coping with these drawbacks, the idea of using a group key manager as a public key storage for WSNs was introduced in [168]. The ACE working group at the IETF is working on a standard for secure group communication featuring this idea [101].

II.) Revocation Lists

Revocation lists are a naive approach for blacklisting keys, not allowed to create a signature. Almost every Public Key Infrastructure (PKI) comes with a certificate format that offers extensions for revocation. The straight-forward way is to maintain a list of revoked public keys or – to gain efficiency – their cryptographic hashes. Two problems arise: First, if every verifier maintains her own list, she needs to be updated/distributed regularly. Second, the verifier needs to trust the issuer of these lists. We examine three solutions, namely the X.509, an IBS system with revocation lists and so-called group signatures.

X.509 certificates are specified in RFC 5280 [128] and base on a very simple work-flow. A user creates his private/public key pair and sends the public key in a so-called *certificate signing request* to a Certificate Authority (CA). The CA signs the public key with her private key and returns the certificate to the user, which she can than append to her signatures to prove ownership. This allows building a *chain of trust*, which is often hierarchically structured. It is the bases for most security protocols, most notably for the context of the this dissertation are TLS/DTLS and IPsec.

X.509 certificates have been shown to work for constrained use cases [44, 69] and offer revocation. Revocation is achieved in form of a so-called Certificate Revocation Lists (CRLs), which are issued, maintained and signed by a CA. The list contains the serial number of the certificate, issued by the CA. For validation, the verifier checks if the signer's certificate is on the list of the issuing CA (or if the CA is on the list of another CA).

Although the mechanism is standardized and used in many cases, it has two disadvantages in the considered environments. First, the certificate of the signer needs to be attached to every message, which can be as large as > 220 Byte [61, 69], Even the compression presented in [44] still requires around 150 Byte of networking overhead, while excluding revocation from the extension. The second is the already mentioned problem of updates.

Identity Based Signatures are only barely considered in the context of constrained settings while coping revocation. An IBS-scheme suitable for authentication in broadcast networks (specifically, aircraft surveillance) and featuring on- and offline signatures was proposed in [177]. However, the considered environment is rather static, which is why key revocation is outside the focus of this approach. Static groups has been discussed as well for broadcast authentication in WSNs [24]. However, it utilizes a revocation mechanism based on identity revocation lists distributed to the communication's participants. While there is no need for network expensive certificates, dealing with revocation list faces the same problems as X.509.

Group Signatures feature the idea of individuals being able to sign in behalf of a group while, their anonymity is provided. This shows a natural conflict between the security property of *Non-Repudiation* and the right for *Anonymity.* After first presented in [26], there have been different proposals for group signature [13, 20, 85, 180]. While only static groups were considered at the beginning, solutions dealing with dynamic groups are available as well [13]. Therefore, the issue of revocation arises and was tackled with different ideas.

Similar to revocation lists, a proof of authority can be attached to the signature [13, 20]. There are proposals for eliminating the issue maintaining these lists while allowing the revocation by public knowledge of the private key [85] . Hence, upon key-disclosure the original owner of the key can upload it to a public database, which makes all signatures invalid. However, due to the main focus on anonymity, such schemes are hardly efficient and often require linearly growing public keys or signatures. Although optimizing these schemes for the scenarios in question would be an interesting approach, the current complexity makes them hardly usable in constrained environments.

III.) Online Verification

The specification of X.509 allows the definition of a CRLs validity. Whenever it expires, the clients are meant to download a new list from the CA. A revoked key/certificate is therefore valid until a new CRL get published by the CA and downloaded by the client. The Online Certificate Status Protocol (OCSP) [133] deals with this issue, by offering a service that allows to validate the status of a public key. Due to the lack of a push mechanism, a client would have to contact the service upon arrival of every message to stay updated, which comes with expensive networking overhead. A similar mechanism could be imagined for the protocols developed for group key distribution, however, the same issue applies.

3.7 Summary and Findings

The chapter describes cryptographic background being necessary to follow the definition and application of the proposed transformation in the next chapters. Mathematical backgrounds for ECC and bilinear maps (pairings) as well as the fundamentals of provable security are presented. In addition, commonly used cryptographic and protocol mechanisms for achieving **(I)** *Confidentiality*, **(II)** *Integrity*, **(III)** *Authenticity*, **(IV)** *Authorization* and **(V)** *Non-Repudiation* in constrained use cases are discussed.

With this background at hand, the chapter gives an answer for the research questions:

> RQ 3: How is distribution and revocation achieved for symmetric and asymmetric keys?
> RQ 4: How can key distribution and revocation be applied in constrained systems?

Especially the centralized group key management architecture is adopted and optimized to the considered constraints. It can be achieved with well-understood combinations of cryptography and protocols, such as DTLS or IPsec.

However, revocations of asymmetric keys is only barely represented in practice, and the solution are either inefficient (e.g., X.509) or do not scale (e.g., public key storage). Even though there are cryptographic mechanisms being studied for the last 20 years, they are only partially applied to cope with the efficiency requirements established in Chapter 2. A promising solution regarding networking overhead and computational effort is found in IBS but the revocation mechanisms are not optimized.

Therefore, the following chapter develops a cryptographic transformation that allows efficient revocation of IBS keys by an arbitrary symmetric key revocation mechanism. The goal is a system, which allows all aspects efficiency being optimized and thereby fulfilling the requirements ER 1, ER 2 and ER 3.

4 Key Updatable Signatures

Ideally, signature schemes enable the verifier to check the signer's authorization. This makes revocation a necessary feature in certificate management systems, as authorization may change over time. However, all revocation mechanisms discussed in Chapter 3 come with computational or network overhead during the actual communication, which contradicts the requirements established in Chapter 2:

ER 1 The networking overhead during communication including the size of the signed message shall not (or only negligibly) increase.

ER 2 The networking overhead for revocation shall be minimal.

ER 3 The performed cryptographic operation shall not (or only negligibly) increase.

Identity Based Signature (IBS) already fulfills the two requirements ER 1 and ER 3. They allow mathematical but inefficient revocation by re-distributing all user's private keys, which contradicts ER 2. Hence, this chapter introduces a transformation framework to extend IBS with an efficient revocation mechanism without losing generality to other signature schemes.

4.1 Methodology

The use of IBS in constrained environments as presented in Chapter 2 and the efficient revocation of IBS keys is achieved in four steps:

1. Allowing the use of IBS in the scenarios in question by integrating IBS keys in an architecture of a Group Key Management Protocol (GKMP) as presented in Section 3.5. This allows the re-use of its mechanisms for access control and identity management, which are beneficial for any Public Key Infrastructure (PKI) and for IBS in particular.

2. Presentation and integration of two efficient re-keying mechanisms for symmetric keys into the same architecture. We use the ones called Logical Key Hierarchy (LKH), which focuses on revocation, and Centralized Authorized Key Extension (CAKE) that additionally deals with issuing keys.

3. Combining both ideas and revoke GKMP managed IBS keys with LKH or CAKE by: Transforming an IBS scheme in such a way, that the inclusion of an efficiently updatable group shared key (gsk) is possible. The transformation is achieved in three consecutive steps.
    a) Signature verification based on the senders' knowledge of a gsk
    b) Updating the gsk with tokens, which are distributed by using e.g., LKH or CAKE.
    c) Combining the IBS keys and the gsk in such a way, that a signature's validation is based only on the knowledge of the public key.

4. Integrating the gsk and its updates to the architecture to manage the necessary keys.

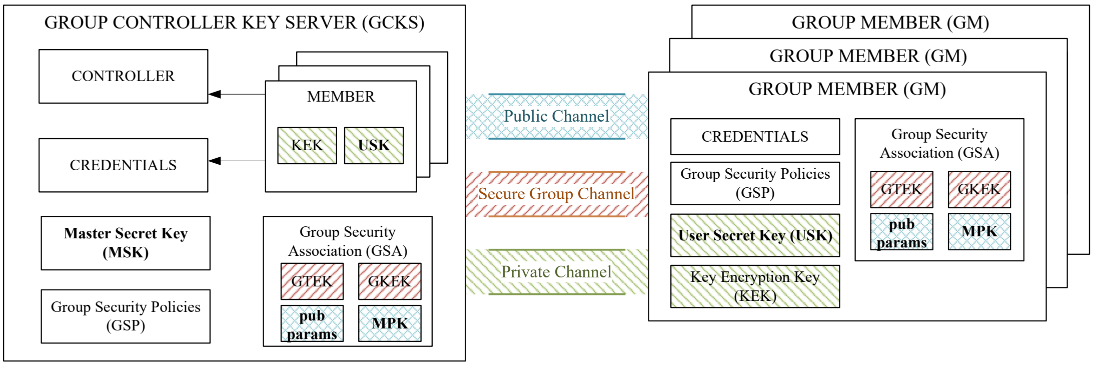


Figure 4.1: IBS key management within a group key management architecture described in RFC 2093 [116] and RFC 2094 [117]. Bold elements are IBS specific.

4.2 IBS Group Key Architecture

IBS schemes, as in Definition 4.1, require a Trusted Third Party (TTP) which chooses the public parameters for the system, verifies a user's identity and authorizes her by generating a signing key for the given identity. In that sense, it is very similar to the architecture for GKMP as defined by RFC 2093 [116] and RFC 2094 [117] (see Section 3.5.1).

Figure 4.1 shows the integration of secret keys as well as the public parameters of IBS in such an architecture. It shows the Group Controller Key Server (GCKS) on the left, the Group Member (GM) on the right and three communication channels in-between with different security properties. The GCKS takes the role of the TTP and uses external services for identification and access control. In RFC 2094 [117] they are called *Controller* and *Credentials*. The so-called Group Security Association (GSA) is extended by the public parameters of the IBS system and distributed to the group members over the public channel. Protocols implementing this standard (e.g., Group Internet Key Exchange (G-IKEv2)[174]) typically define this channel to be authenticated with the credentials of the GCKS. Similarly, the *private channel* – which is established by authenticated key exchange between the GM and GCKS and is used to distribute encryption keys in multicast groups – is used for distributing the usk. The private channel is secured with the Key Encryption Key (KEK).

Definition 4.1 (Identity Based Signature [47])**.** *An IBS scheme $\mathcal{I}$ for message space $\mathtt{M}$ consists of a set of polynomial-time algorithms $\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V}$:*

Setup $\mathcal{G}$: Given the security parameter λ, a Master Secret Key (msk) is chosen and the corresponding Master Public Key (mpk) is calculated, that is $(msk, mpk) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$. Additionally, other relevant public parameters, e. g. group generators, hash and pairing functions and elliptic curve parameters are defined.

Extract $\mathcal{E}$: Given a new member's id and the msk the User Secret Key (usk) is generated, that is $usk \overset{r}{\leftarrow} \mathcal{E}_{msk}(id)$.

Sign $\mathcal{S}$: A user signs a message with their usk, that is $\sigma \overset{r}{\leftarrow} \mathcal{S}_{usk}(m)$

Verify $\mathcal{V}$: Given the stated sender's id, a signature σ and a message m a recipient verifies / falsifies the signature, that is $\{0, 1\} \leftarrow \mathcal{V}_{mpk,id}(m, \sigma)$.

4.2.1 Role and Communication Model

With Definition 4.1, the roles in an IBS system are depicted in Figure 4.2. Besides the essential roles of *Signer* (left) and *Verifier* (right), the IBS specific role of the TTP is twofold, as it needs to setup the system and extract keys as well as granting access to the system. Thus, its roles are split into a Key Generation Center (KGC) (center) and Access Controller (second from right). Additionally, any identity has to be trusted in an IBS system, hence an identity manager (second of left) is used to generate identities upon request. Figure 4.2 shows these five roles and their communication as follows:

KGC: chooses the systems security parameter, defines the public parameters, and generates the required cryptographic material for the signer. Upon request for a usk it validates the identity and the authorization with the Identity- and Access-Manager respectively. It also publishes the mpk for the verifier.

Signer: requests an identity id from the Identity Manager and a signing key usk from the KGC. It then signs messages m with the usk returning a signature σ which can then be sent to some verifier.

Identity Manager: manages the available identities in the system and returns an identity to the Signer upon request. Upon request from the KGC, it validates a given identity.

Access Controller: upon request from the KGC, it validates a given identity's permit to access the system.

Verifier: uses the public parameters of the system mpk and verifies a signature.

4.2.2 IBS Key Revocation

For mathematical revocation of the IBS keys – in particular the usk and mpk – the server calculates a new msk and derives new $usk's$ and mpk [52]. The new $usk's$ are send to the each remaining member using their private channels and the new mpk is published using the authenticated public channel. Revoked members do not receive new keys and any signature created with the old keys will be mathematically invalid when verified with the new mpk.

4.3 Updating a Group Shared Key with LKH and CAKE

Before presenting the inclusion of a gsk into IBS schemes, we first examine its distribution and revocation. The integration of IBS to a GKMP motivates a mechanism which can be efficiently used within a GKMP as well. Such are found in form of centralized re-keying schemes, which use a Group Key Encryption Key (GKEK) for distributing a Group Transport Encryption Key (GTEK). Both keys are depicted in the GSAs of GCKS and GM in Figure 4.1 and establish the so-called *Secure Group Channel*. The question arises, how such keys can be efficiently updated, especially when an excluded member is in possession of the keys.

With LKH, this section first presents a widely used mechanism based on a symmetric encryption with a key tree and shows its integration to the IBS architecture. Improving the idea of a key hierarchy in combination with a cryptographic mechanism for compressing key material, is presented with CAKE.

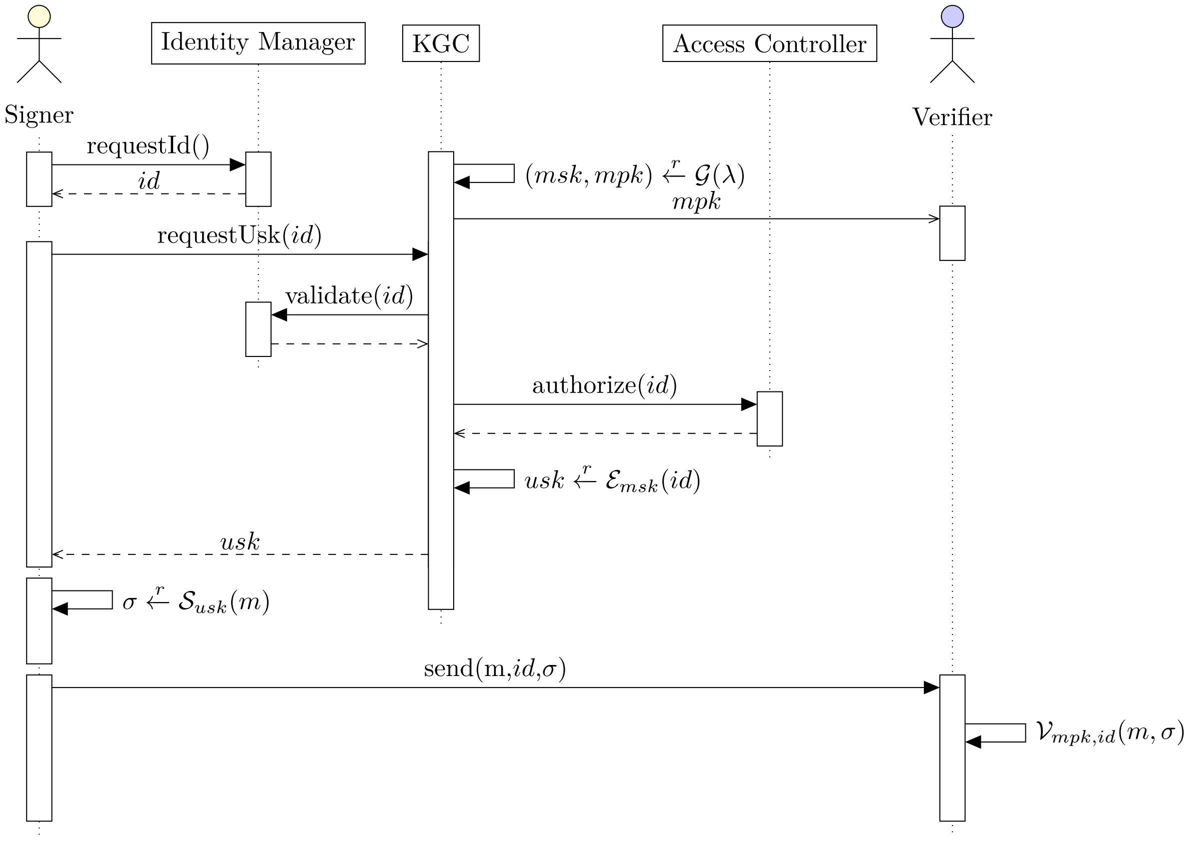

Identity Manager
KGC
Access Controller
Signer
Verifier
requestId()
id
$(msk, mpk) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$
mpk
requestUsk(id)
validate(id)
authorize(id)
$usk \overset{r}{\leftarrow} \mathcal{E}_{msk}(id)$
usk
$\sigma \overset{r}{\leftarrow} \mathcal{S}_{usk}(m)$
send(m,id,σ)
$\mathcal{V}_{mpk,id}(m, \sigma)$

Figure 4.2: Communication model for IBS.

4.3.1 Logical Key Hierarchy

The keying mechanism called LKH defines a hierarchy of trees managed by the key server. To setup the system (or group), the server creates a tree of random, symmetric keys. They do not mathematically depend on each other, the tree is used as a logical management structure. The tree's leaves are keys exclusively shared by client and server, while the root is a key shared by all participants. Keys on the path from the root to the leave are distributed to the corresponding clients.

Figure 4.3 shows how a GM can be excluded with a single message by LKH using a binary tree. The left tree's root (denoted as *1*) is the GKEK, while the leaves (denoted as *A*-*H*) are the GMs KEK. The GCKS knows all keys in the tree, the GMs know all keys on their corresponding path to the root key. In the example of Figure 4.3, the left part shows the tree before member D is excluded and in possession of $\{D, 5, 2, 1\}$. When D is excluded from the group, the GCKS updates all keys on D's path to the root; in right part of Figure 4.3 these new keys $\{5', 2', 1'\}$ are hatched. Then it encrypts them with the lowest child keys unknown to the excluded member D. The following symmetric encryption take place:

- encrypt 5' with C
- encrypt 2' with 5'
- encrypt 2' with 4
- encrypt 1' with 2'
- encrypt 1' with 3

The five cipher texts can be sent in one single message to the remaining members *A-C; E-H* and D will not be able to successfully decrypt the new shared secret *1'* or either of the new keys *2'* or *5'*.

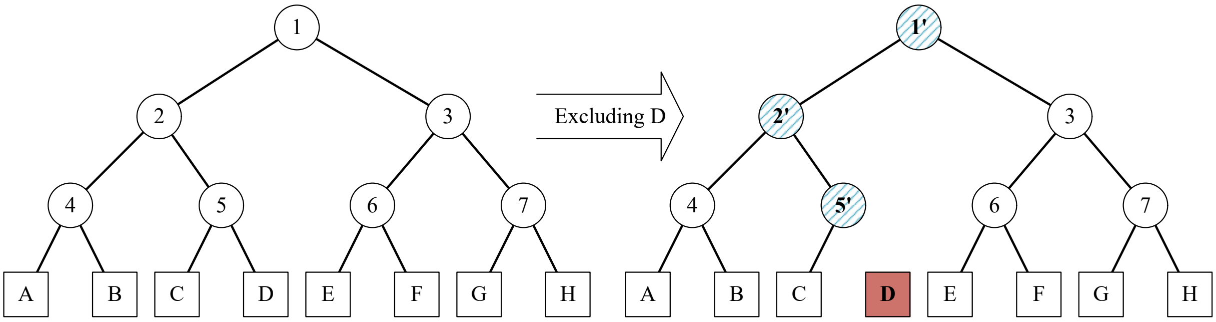


Figure 4.3: Initial LKH tree before (left) and after (right) D (in red) is excluded. The replaced keys in the tree (hatched) are encrypted with their child nodes.

One advantage of LKH is its exclusive dependency on symmetric cryptography for security. Concrete encryption algorithms are exchangeable with the only disadvantage of potentially larger keys in the tree. The order a of the tree is not strictly defined, but for storage and networking optimization a binary tree (i.e. $a = 2$) is recommended [107, 120]. Thus, a group with N participants requires a key storage for $log_a(N)$ keys for clients and $(a^{log_a(N)} - 1)/(a - 1)$ keys for the server.

The second advantage of LKH is its integration in popular GKMPs, e.g., G-IKEv2 [174]. First, the client initiates a Diffie-Hellman exchange with the server, the so-called KEK, that is later used as the trees' leave. The GKEK is also already defined by the architecture (see Figure 4.1), hence, only additional payloads for the tree management are required. During the authentication step of G-IKEv2, the client downloads its set of keys in a *LKH_DOWNLOAD_ARRAY* with corresponding indexes of each key in the tree. Later, all keys can be updated with a *LKH_UPDATE_ARRAY* in multicast or individually with a *LKH_INBAND_REKEY* message.

4.3.2 Centralized Authorized Key Extension

Using LKH allows efficient exclusion of one GM, however, it does not deal with bulk action, e.g., when numerous clients join or leave the system simultaneously. In [59], a protocol was presented which allows such action while keeping the efficiency of LKH. It extends the idea of hierarchy of keys with a mechanism called Secure Lock (SL), which was first presented in [27].

The SL allows to efficiently encrypt individual user messages over a broadcast medium using a single message. It uses the Chinese Remainder Theorem (CRT) which is defined as follows (see Theorem 1.2.9 [99]):

Definition 4.2 (Chinese Remainder Theorem [99])**.** *If $m_1, \ldots, m_k \in \mathbb{N}$ with $k \geq 2$ are pairwise coprime moduli and $r_1, \ldots, r_k \in \mathbb{Z}$ are arbitrary, then there exists a unique $a \in \mathbb{Z}_m$ with $m = m_1 \cdots m_k$ such that:*

$$a \equiv r_j \mod m_j; \text{ for } 1 \leq j \leq k \tag{4.1}$$

For key distribution, this SL (*Lock*) utilizes two secrets by each GM's of the system, a symmetric KEK and a prime number p_i, both known by the server. A new GKEK is encrypted with each participant's KEK, such that the GMs message is

$$m_i \equiv GKEK_{new} * KEK_i \tag{4.2}$$

Then, a congruence system is built:

$$\begin{aligned} P = \prod_{i=1}^{k} p_i; \quad L_i = \frac{P}{p_i}; \quad Y_i = L_i^{-1} \mod p_i \\ Lock\_X = \sum_{i=1}^{k} \Big(m_i \cdot L_i \cdot Y_i \Big) \mod P \end{aligned} \tag{4.3}$$

where i is the index of the GM and k is the number of participants in the system. This $Lock\_X$ is broadcasted in a single message, and a GM obtains the new GKEK:

$$\begin{aligned} m_i = Lock\_X \mod p_i \\ GKEK = m_i / KEK_i \end{aligned} \tag{4.4}$$

Even though the new GKEK can be distributed efficiently with that technique, the message becomes very large. It depends on the number of clients and the size of the primes p_i chosen by the server. In particular, if p_i is chosen at random:

Proposition 4.1. *Let k be the number of participants and b be the size of p_i in bits, than the maximum size of $Lock\_X$ is*

$$|Lock\_X| = k \cdot b \tag{4.5}$$

and the average size of $Lock\_X$ is

$$\overline{Lock\_X} = k \cdot (b - 1) \tag{4.6}$$

With large number of participants, this quickly becomes a large message that might be difficult to handle. Thus, CAKE uses a key hierarchy, similar to LKH to decrease the message size. In contrast to LKH, each node in CAKE represents a key pair (p_i, k_i), where i is the position in the tree, p_i the nodes prime and k_i the node's symmetric key. With x being the order of the tree, this reduces the average size to

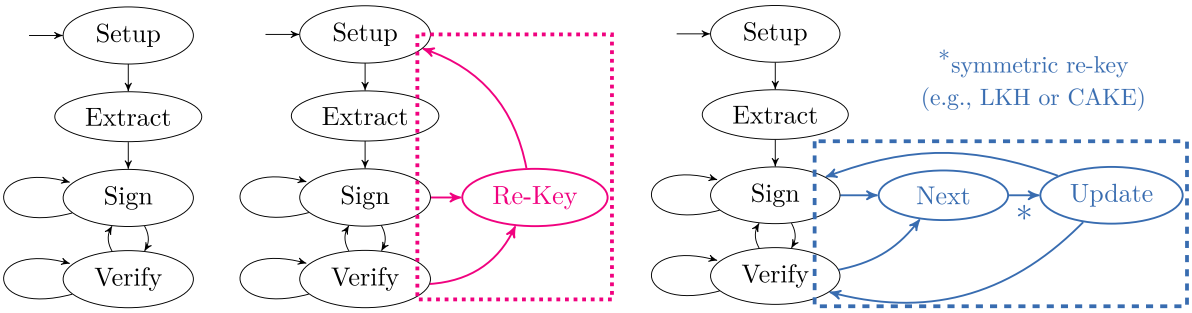
$$\overline{Lock\_X} = log_x(k) \cdot (b - 1) \tag{4.7}$$

In [68], the hierarchy was presented as a ternary tree to improve the efficiency for systems with < 100 participants and extended with an addressing scheme for flexible and efficient tree operations in [59]. However, this can be easily adopted to a certain use case. During the group's lifecycle, the following events are secured as follows:

member revocation: Similar to LKH, single member is revoked by re-calculating the keys on the expelled members path to the root. The SL is built by all key-pairs, the expelled member does not possess. In contrast to LKH, each client has to perform only a *modulo* operation to decrypt the SL.

mutlipe member revocation: Excluding multiple members simultaneously is equivalent to revoking a single member. The SL is built with other key pairs none of the excluded member is not in possession of.

mutlipe member join when multiple members join the group simultaneously, one SL can be calculated for all of them and send via a single multicast message. Neither LKH nor G-IKEv2 specify an action for this event.



(a) No Re-Key (b) Inefficient Re-Key [52] (c) Efficient Re-Key

Figure 4.4: State machines of the different re-keying approaches for IBS.

4.4 Efficiently Updating IBS Keys

This section presents a cryptographic transformation that allows the revocation of a user's secret key by updating all other users' keys with a common symmetric update token. The similarity of IBS to a group key management system encourages updating the update token by an efficient key distribution mechanism. LKH or CAKE presented in the previous section are such examples.

That such a symmetric element can be used for key revocation utilizes the following characteristics of symmetric and asymmetric authentication:

- Membership of a group can be proven to other members by proving knowledge of a group-wide shared secret. Changing this secret means excluding any party that does not receive an update. In [57], this property was presented as *group authenticity*.

- Any signature scheme that involves unknown parties requires a trust anchor such as the KGC in IBS or the Certificate Authority (CA) in certificate based schemes. Its users can therefore be considered a group under that TTP. User key revocation equals the exclusion from this group.

- Changing the trust anchor's key-pair implicitly revokes all underlying user keys. In IBS, this makes all signatures mathematically invalid as the *mpk* is no longer available. In certificate based schemes, such a revocation interrupts the trust-chain for all users under the revoked CA and invalidates them. This mechanism is sometimes referred as "folklore IBS" [12].

Concept for IBS Key Revocation

The idea for revoking IBS keys is presented in Figure 4.4, showing the possible re-key mechanisms of IBS keys in form of state machines. Figure 4.4a presents the four algorithms of IBS as in Definition 4.1 as states, showing that after IBS is set up and keys are extracted the system stays in the states *Sign* and *Verify*. Re-Keying or revocation is not part of the definition and other mechanisms such as revocation lists are used. This changes with the idea sketched in Section 4.2.2 and [52], presented in Figure 4.4b. After key extraction, the system can move to the state *Re-Key*, which in turn triggers a re-setup of the IBS system and *Extract* keys for all clients. As mentioned earlier, this is inefficient but allows mathematical validation of revoked keys. Figure 4.4c sketches how this inefficiency is overcome by the transformations discussed in the following. Two new states, *Next* and *Update* are introduced, which allow efficiently changing the master key material with an update token. The latter is generated in *Next* and distributed by using efficient mechanisms – such as the earlier discussed LKH and CAKE – and used to efficiently *Update* the keys of all clients. This removes the need to re-setup the IBS system and the computational complexity for the KGC is reduced from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$.

Technically, the proposed mechanism adds a symmetric key to the asymmetric master key material of the KGC. It serves as gsk and access key to the group and its update will change the KGCs key and, thus, revokes the user's keys. How and why it can be used to revoke a user key through a single group message without harming security (and how this can be achieved for IBS schemes) will be shown in three transformation steps:

1. **Two Key Signature Scheme (2KSS):** Integration of a shared secret in IBS in such a way that individual and group authentication can be achieved with a single signature.

2. **Updatable Two Key Signature Scheme (U2KSS):** Updating the shared secret with an update token.

3. **Key Updatable Signature Scheme (KUSS):** Updating hybrid representations of the keys.

Using such transformations allow fine-grained updating of existing signature schemes, which has two advantages: First, existing signature schemes, which are well-analyzed regarding their performance and security, can be re-used. Second, as the security model of the schemes is clearly defined, we can use the *Game Hopping Lemma* (see Section 3.1.3) to analyze the security of the resulting schemes. The goal is to transform the schemes in such a way, that the original security properties stay in tact, for IBS that is the existential unforgeability under adaptively chosen-message-and-identity attacks (EUF-CMA) [12, 41].

General applicability of the concept

Each of the following transformation lists the requirements to a signature scheme to perform the transformation. For the sake of readability the following is specifically adapted to IBS schemes as they are of major interest. How signatures schemes on specific primitives can be converted to a IBS scheme is shown in [12]. Hence, this simplification is possible without loss of generality. The major difference of IBS schemes to other signature schemes is that the process for key generation $\mathcal{K}$ is split:

Key Generation $\mathcal{K}$: the probabilistic algorithm outputs a key pair (sk, pk) for a security parameter λ, that is, $(sk, pk) \xleftarrow{\mathcal{T}} \mathcal{K}(\lambda)$. In IBS schemes, this algorithm is split into $\mathcal{G}, \mathcal{E}$ returning $sk = usk$ and $pk = (mpk, id)$ as in Definition 4.1.

Methodology for Transformation

In the remaining of this section, the three transformation steps (2KSS, U2KSS, and KUSS) are each presented as follows:

I.) **Description** of the idea and the goal of the transformation.

II.) **Definition** of the resulting scheme by its algorithms (as for IBS in Definition 4.1).

III.) Stating mathematical **transformability conditions**, where necessary.

IV.) Definition of a **security notion**, including oracles and experiments, which are later used for proving the security.

4.4.1 Two Key Signature Scheme (2KSS)

I.) Description

2KSS is the transformation of an IBS scheme to include a symmetric key in the signing and verification process, while preserving the size and security of the signature. The signing process is changed in a way that the symmetric key can be seen as an extension to the asymmetric key.

II.) Definition

With 2KSS the IBS algorithms for signing (see Definition 4.1) is extended to include a *gsk* (which is referred to as $\mathbf{g}$), which can then be used to prove group access during verification.

Definition 4.3 (Two Key Signature Scheme). *A 2KSS for message space* $\mathtt{M}$ *consists of a set of polynomial-time algorithms* $\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V}$ *defined by IBS scheme* $\mathcal{I}$*, which are modified from the underlying signature scheme as follows:*

Setup $\mathcal{G}$**:** In addition to the key pair (msk, mpk), the probabilistic algorithm outputs a third key $\mathbf{g}$ for a security parameter λ, that is, $(msk, mpk, \mathbf{g}) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$.

Extract $\mathcal{E}$**:** outputs a user secret key *usk* for a given identity *id*, that is, $usk \overset{r}{\leftarrow} \mathcal{E}_{msk}(id)$.

Sign $\mathcal{S}$**:** takes $\mathbf{g}$ as an additional input and outputs a signature for message m, that is $\sigma \overset{r}{\leftarrow} \mathcal{S}_{usk,\mathbf{g}}(m)$.

Verify $\mathcal{V}$**:** On input of message m and (mpk, id), this algorithm takes $\mathbf{g}$ as an additional input and returns an answer $\mathcal{V}_{mpk,id,\mathbf{g}}(m, \sigma)$ whether or not σ is a valid signature of m.

III.) Mathematical conditions for the transformation

This approach is suitable for all signature schemes that fulfill the condition:

Transformability Condition 4.1. *It is possible to choose* $\mathbf{g}$ *in such a way that it can be included in the mathematical process of signing with usk as it happens in* $\mathcal{S}_{usk}(m)$*. This way, the signing algorithm still produces only one signature.*

IV.) Security Notion

For achieving EUF-CMA, adding a shared secret to a signature scheme must not harm security relative to the underlying signature scheme. This property is later referred as *Token Security*, which is inspired by [84]. The used oracles for the following experiments are a generalized and adopted from the ones used in [9, 12, 24, 47, 67].

Signing $\mathcal{O}_{\mathcal{S}}(id, m)$**:** On input of a message $m \in \mathcal{M}$, return $\sigma \overset{r}{\leftarrow} \mathcal{S}_{usk,\mathbf{g}}(m)$.

Extracting $\mathcal{O}_{\mathcal{E}}(id)$**:** On input of an identity id, return $(usk, \mathbf{g}) \overset{r}{\leftarrow} \mathcal{E}_{msk}(id)$.

Random oracle: All hash functions are modeled as random oracles.

EUF-CMA for IBS under the 2KSS transformation as in Definition 4.4 adapts Definition 1 in the paper by Galindo and Garcia [47], which shows EUF-CMA for IBS.

Definition 4.4 (EUF–2KSS–CMA). *A 2KSS Identity Based Signature scheme $\mathcal{I} = (\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$ is said to be secure against existentially unforgeable under adaptively chosen-message-and-identity attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the probability that $P(\mathbf{EUF\text{-}2KSS\text{-}CMA}_{\mathcal{I}}(\mathcal{A}) = 1)$ in the experiment defined below is a negligible function of λ. During this experiment, $\mathcal{A}$ has access to an extract and signing oracle, denoted as $\mathcal{O}_{\mathcal{E}}$ and $\mathcal{O}_{\mathcal{S}}$.*

```
$\mathbf{EUF\text{-}2KSS\text{-}CMA}_{\mathcal{I}}(\mathcal{A})$ :
$(msk, mpk, \mathbf{g}) \leftarrow \mathcal{G}(\lambda)$
$(id^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{E}}(\cdot), \mathcal{O}_{\mathcal{S}}(\cdot,\cdot)}(mpk, \mathbf{g})$
return $\mathcal{V}(mpk, \mathbf{g}, \sigma^*, m^*, id^*)$
```

Trivial wins where the adversary $\mathcal{A}$ queries id^* from $\mathcal{O}_{\mathcal{E}}(\cdot)$ or σ^* from $\mathcal{O}_{\mathcal{S}}(\cdot, \cdot)$ are excluded. Also, the same id is not allowed to be queried twice to $\mathcal{O}_{\mathcal{E}}(\cdot)$.

A random 2KSS system for a security parameter λ is set up. The adversary $\mathcal{A}$ is allowed to call the oracles for extracting secret keys and signing messages as often as she wishes. She wins the game, if she is able to find a signature (id^*, σ^*) for a given message m^* and has not received one of them during a call of the oracles. The scheme is considered secure if her chances of winning the game are better than a negligible function of the security parameter λ.

4.4.2 Updatable Two Key Signature Scheme (U2KSS)

I.) Description

In 2KSS, membership changes require to send a new *gsk* to all group members. This yields a potential attack, as an attacker could intercept and use it to illegitimately authenticate themselves. Due to this threat, minimizing the amount of cryptographic material sent over a network is common practice in communication security. As a remedy, the *gsk* is only distributed to members upon joining the system and later updated by an *update token* Δ. This is the transformation from 2KSS to U2KSS.

The left part of Figure 4.5 shows client E joining the system (pictured as a cloud with the four members *A,B,C,D*). An update token Δ is generated by a so-called Key Update Center (KUC). It is send to the KGC which in turn updates and sends $\mathbf{g}$ to the new client E together with its private *usk* (denoted as $\{usk, \mathbf{g}\}$). Previous members (*A,B,C,D* in Figure 4.5) receive only the update token Δ. The right part shows E being excluded from the group by distributing a new Δ only among the remaining members and the KGC.

II.) Definition

We say, that $\mathbf{g}$ evolves with so-called *epochs*; signatures are created with respect to a specific *epoch* e. By invoking an algorithm *Next* ($\mathcal{N}$), the communication group moves from epoch e to epoch $e + 1$. $\mathcal{N}$ generates the update token Δ_{e+1} for the new epoch. The update token is maintained and distributed by an entity with a new role: the *KUC*. By invoking an algorithm *Update* $\mathcal{U}$, the shared secret is updated with the token. All roles using the *gsk* – namely *KGC*, *Signer* and *Verifier* – may use this algorithm. A 2KSS with this modification is a U2KSS and defined by:

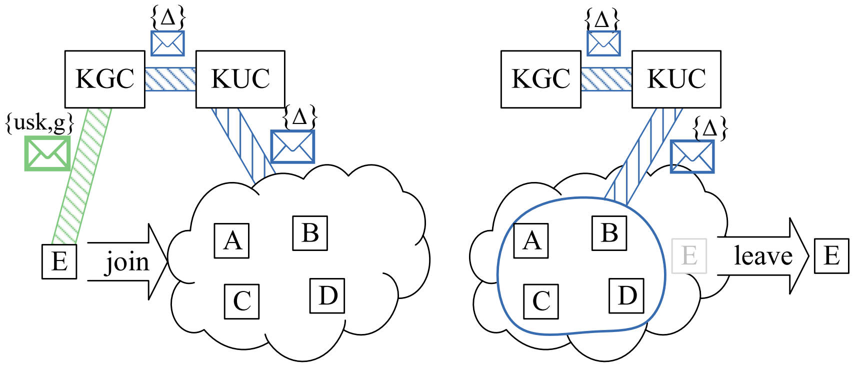


Figure 4.5: Distributing the group shared key $\mathbf{g}$ and updating it with the update token Δ.

Definition 4.5 (Updatable Two Key Signature Scheme)**.** *A U2KSS for message space* M *consists of a set of polynomial-time algorithms* $(\mathcal{G},\mathcal{E},\mathcal{N},\mathcal{U},\mathcal{S},\mathcal{V})$*. It is the extension of an IBS scheme* $\mathcal{I}$ *by the algorithms Next* ($\mathcal{N}$)*, which generates a new update token, and Update* ($\mathcal{U}$)*, which updates the shared secret using the update token. More specifically:*

Setup $\mathcal{G}$**:** is a probabilistic algorithm that outputs a key tuple $(msk, mpk, \mathbf{g}_0)$ for a security parameter λ where $\mathbf{g}_0$ is the *gsk* in epoch $e = 0$. That is, $(msk, mpk, \mathbf{g}_0) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$.

Extract $\mathcal{E}$**:** outputs a user secret key *usk* for a given identity *id*, that is, $usk \overset{r}{\leftarrow} \mathcal{E}_{msk}(id)$.

Next $\mathcal{N}$**:** is a probabilistic algorithm which outputs an update token Δ_{e+1} for a security parameter λ, that is, $\Delta_{e+1} \overset{r}{\leftarrow} \mathcal{N}(\lambda)$.

Update $\mathcal{U}$**:** takes $\mathbf{g}_e$ of epoch e and Δ_{e+1} of the following epoch $e+1$ as input and outputs an updated $\mathbf{g}$, such that $\mathbf{g}_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(\mathbf{g}_e)$.

Sign $\mathcal{S}$**:** takes $(usk, \mathbf{g}_e)$ as input and outputs the signature σ_e with respect to epoch e for message m. That is $\sigma_e \overset{r}{\leftarrow} \mathcal{S}_{usk,\mathbf{g}_e}(m)$.

Verify $\mathcal{V}$**:** On input of message m and $(mpk, id, \mathbf{g}_e)$, return an answer $\mathcal{V}_{mpk,id,\mathbf{g}_e}(m, \sigma_e)$ whether or not σ_e is a valid signature of m with respect to epoch e.

IV.) Security Notion

With the U2KSS transformation, Forward and Post-Compromise Security are achieved in addition to EUF-CMA in 2KSS, which does not provide this property.

Forward Security An adversary compromising some *usk* and $\mathbf{g}_e$ in some epoch e^* does not gain any advantage in forging a signature for previous epochs $e < e^*$. That means that all signatures from epochs *before* the compromise retain their integrity [14, 72].

Post-Compromise Security An adversary compromising *usk* and $\mathbf{g}_e$ in some epoch e^* does not gain any advantage in forging signatures from epochs $e > e^*$ after that compromise. That means that all signatures from epochs *after* the compromise retain integrity [79].

Adapting the experiment in Definition 4.4 and covering the new properties requires additional oracles to be accessible for an adversary. The oracles for *Next* and *Update* are more or less direct calls of the functions in Definition 4.5:

$\mathcal{O}_{\mathcal{N}}$: When triggered, this oracle returns a new update token $\Delta_{e+1} \overset{r}{\leftarrow} \mathcal{N}(\lambda)$.

$\mathcal{O}_{\mathcal{U}}(\mathbf{g}_e, \Delta_{e+1})$: When triggered, this oracles returns $\mathbf{g}_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(\mathbf{g}_e)$.

Definition 4.6 (EUF–U2KSS–CMA). *A U2KSS identity-based signature scheme $\mathcal{I} = (\mathcal{G}, \mathcal{E}, \mathcal{N}, \mathcal{U}, \mathcal{S}, \mathcal{V})$ is said to be secure against existentially unforgeable under adaptively chosen-message-and-identity attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the probability that $P(\mathbf{EUF\text{-}U2KSS\text{-}CMA}_{\mathcal{I}}(\mathcal{A}) = 1)$ in the experiment defined below is a negligible function of λ. During this experiment, $\mathcal{A}$ has access to the oracles defined above, namely $\mathcal{O}_{\mathcal{E}}$, $\mathcal{O}_{\mathcal{N}}$, $\mathcal{O}_{\mathcal{U}}$ and $\mathcal{O}_{\mathcal{S}}$.*

```
$\mathbf{EUF\text{-}U2KSS\text{-}CMA}_{\mathcal{I}}(\mathcal{A})$ :
$(msk, mpk, \mathbf{g}) \leftarrow \mathcal{G}(\lambda)$
$(id^*, m^*, \sigma^*, e^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{E}}(\cdot), \mathcal{O}_{\mathcal{N}}, \mathcal{O}_{\mathcal{U}}(\cdot,\cdot), \mathcal{O}_{\mathcal{S}}(\cdot,\cdot)}(mpk, \mathbf{g})$
return $\mathcal{V}(mpk, \mathbf{g}, \sigma^*, m^*, id^*, e^*)$
```

Trivial wins where the adversary $\mathcal{A}$ queries id^* from $\mathcal{O}_{\mathcal{E}}(\cdot)$ together with Δ_{e^*} from $\mathcal{O}_{\mathcal{N}}$ or σ^* from $\mathcal{O}_{\mathcal{S}}(\cdot,\cdot)$ are excluded. Also, the same *id* is not allowed to be queried twice to $\mathcal{O}_{\mathcal{E}}(\cdot)$.

A random U2KSS system for a security parameter λ is set up. The adversary $\mathcal{A}$ is allowed to call the oracles for extracting secret keys and signing messages as often as she wishes. Further, she may oracle update tokens and updates for corresponding keys. As in Definition 4.4, she wins the game, if she is able to find a signature (id^*, σ^*) for a given message m^*. With the introduction of forward and post-compromise security, she may now oracle the *usk* for id^*, as long as she does not oracle the update token for e^*. The scheme is considered secure, if her chances of winning the game are worse than a negligible function of the security parameter λ.

4.4.3 Key Updatable Signature Scheme (KUSS)

I.) Description

Although with U2KSS revocation is possible with a single message, explicit knowledge of the *gsk* is still necessary during signing and verification. As it must be kept secret within the group of participants, $\mathbf{g}$ cannot be included in the signature. With KUSS, this section presents a transformation, improving the efficiency of signing and verification by hybridization of the asymmetric and symmetric keys. In the best case, the transformation makes the inclusion of $\mathbf{g}$ in signing and verification completely unnecessary. If that is impossible, the transformation allows the removal of $\mathbf{g}$ from signing *or* verification.

The 2KSS transformation of some IBS schemes (such as the one by Galindo and Garcia (GG) [47]), can happen by randomly choosing $\mathbf{g} \overset{r}{\leftarrow} \mathbb{Z}_p^*$ (as it is true for the *usk*) and changing the following computation in the signature algorithm (x is a random element of $\mathbb{Z}_p^*$, h is a message hash):

GG [47]: $\sigma = x + h \cdot usk$

Possible 2KSS transform: $\sigma = x + h \cdot usk \cdot \mathbf{g}$

The verification algorithm changes analogously. Here it shall only be pointed out that mpk exclusively appears when multiplied with $\mathbf{g}$. In such a case, $usk{\cdot}\mathbf{g}$ and $mpk{\cdot}\mathbf{g}$ can be pre-computed and represented as hybrid keys $h^{\smile}usk$ and $h^{\smile}mpk$ which replace usk and mpk, respectively. The *Extract*-algorithm can be altered to produce a hybrid user key, leaving the effort of computing it to the usually more resourceful TTP. The $h^{\smile}mpk$ can be handled identically.

II.) Definition

With the combination of the shared secret introduced by the 2KSS transform and the update mechanism by U2KSS, the creation of a signature scheme whose signing and verification keys are updatable by update tokens is possible. The result is a scheme which allows all keys $k \in \{mpk, usk, \mathbf{g}\}$ to be updatable by an update token Δ. This requires the algorithms defined in the previous three steps to be modified so that they reflect the application of Δ. The KGC holds the master key pair (msk, mpk) and the shared secret $\mathbf{g}$. The update tokens are managed by KUC as introduced in Definition 4.5.

The keys k evolve with *epochs* and signatures are created with respect to a specific epoch e. When moving from epoch e to $e+1$, the KUC invokes *Next* $(\mathcal{N})$ to generate the update token Δ_{e+1} for the new epoch. The client and the KGC invoke *Update* $(\mathcal{U})$ algorithm to update the key material k_e with Δ_{e+1} and output k_{e+1}.

Definition 4.7 (KUSS)**.** *A Key Updatable Signature Scheme for message space* $\mathtt{M}$ *consists of a set of polynomial-time algorithms* $(\mathcal{G}, \mathcal{E}, \mathcal{N}, \mathcal{U}, \mathcal{S}, \mathcal{V})$*:*

Setup $\mathcal{G}$: On input of a security parameter λ, it outputs a master key pair $(msk, mpk_0, \mathbf{g}_0) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$.

Extract $\mathcal{E}$: is an either probabilistic or deterministic algorithm run by the KGC. On input of a secret key msk, a shared secret $\mathbf{g}_e$ of epoch e and a user id, it outputs a user secret key usk_e for epoch e. That is $usk_e \overset{r}{\leftarrow} \mathcal{E}_{msk,\mathbf{g}_e}(id)$.

Next $\mathcal{N}$: is a probabilistic algorithm run by the KUC. On input of epoch e it outputs an update token Δ_{e+1} for epoch $e+1$. That is $\Delta_{e+1} \overset{r}{\leftarrow} \mathcal{N}(\lambda)$.

Update $\mathcal{U}$: is a deterministic algorithm run by the Client and KGC which updates $k_e = (mpk_e, usk_e, \mathbf{g}_e)$. On input of k_e for epoch e and an update token Δ_{e+1} for epoch $e+1$ it outputs k_{e+1}. That is $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$.

Sign $\mathcal{S}$: On input of a message m and a secret key usk_e for epoch e it outputs a signature $\sigma_e \overset{r}{\leftarrow} \mathcal{S}_{usk_e}(m)$.

Verify $\mathcal{V}$: On input of a message m, a user's id, a signature σ_e and a public key mpk_e for epoch e, return an answer $\mathcal{V}_{mpk_e,id}(m, \sigma_e)$ whether or not σ_e is a valid signature of m with respect to epoch e.

III.) Mathematical Conditions

The hybridization for signing is suitable for all signature schemes that fulfill the condition:

Transformability Condition 4.2. *usk and $\mathbf{g}$ can be part of a binary operation during the signing process, such that the pre-computed value of this operation can be used to sign different messages in the same manner in which the usk was used in $\mathcal{S}_{usk}(m)$.*

The hybridization for verification is suitable for all signature schemes that fulfill:

Transformability Condition 4.3. $\mathbf{g}$ *can be part of a binary operation during the signing process, such that the pre-computed value h˘mpk can be used to verify different messages in the same manner in which the mpk was used in* $\mathcal{V}_{mpk}(m, \sigma)$.

IV.) Security Notion

The definition of KUSS allows a slightly tighter security model, as the adversary can be allowed to oracle every usk_e outside the challenge epoch, which is not possible with U2KSS:

$\mathcal{O}_{\mathcal{E}}(k_e, \Delta_{e+1})$: When triggered, this oracles returns $usk_e \overset{r}{\leftarrow} \mathcal{E}_{msk,\mathbf{g}_e}(id)$.

$\mathcal{O}_{\mathcal{U}}(k_e, \Delta_{e+1})$: When triggered, this oracles returns $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$.

The experiment for proving EUF-CMA-security is almost identical to Definition 4.6, without the shared secret:

Definition 4.8 (EUF–KUSS–CMA). *A KUSS IBS-scheme* $\mathcal{I} = (\mathcal{G}, \mathcal{E}, \mathcal{N}, \mathcal{U}, \mathcal{S}, \mathcal{V})$ *is said to be secure against existentially unforgeable under adaptively chosen-message-and-identity attacks if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *the probability* $P(\mathbf{EUF\text{–}KUSS\text{–}CMA}_{\mathcal{I}}(\mathcal{A}) = 1)$ *in the following experiment is a negligible function of* λ. *During the experiment,* $\mathcal{A}$ *has access to the oracles* $\mathcal{O}_{\mathcal{E}}, \mathcal{O}_{\mathcal{N}}, \mathcal{O}_{\mathcal{U}}$ *and* $\mathcal{O}_{\mathcal{S}}$.

```
$\mathbf{EUF\text{–}KUSS\text{–}CMA}_{\mathcal{I}}(\mathcal{A})$ :
$(msk, mpk, \mathbf{g}) \leftarrow \mathcal{G}(\lambda)$
$(id^*, m^*, \sigma^*, e^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{E}}(\cdot), \mathcal{O}_{\mathcal{N}}, \mathcal{O}_{\mathcal{U}}(\cdot,\cdot), \mathcal{O}_{\mathcal{S}}(\cdot,\cdot)}(mpk)$
return $\mathcal{V}(mpk, \sigma^*, m^*, id^*, e^*)$
```

Trivial wins where the adversary $\mathcal{A}$ queries id^* from $\mathcal{O}_{\mathcal{E}}(\cdot)$ together with Δ_{e^*} from $\mathcal{O}_{\mathcal{N}}$ or σ^* from $\mathcal{O}_{\mathcal{S}}(\cdot,\cdot)$ are excluded. Also, the same id is not allowed to be queried twice to $\mathcal{O}_{\mathcal{E}}(\cdot)$.

4.5 Group IBS Architecture

Figure 4.6 shows the adaptation of the IBS architecture to include the update mechanism established in the previous section. The *gsk* ($\mathbf{g}$) and the update token (Δ) are added to the GSA, as well as a key distribution mechanism such as LKH or CAKE (symbolized with binary tree at the GCKS and a key path at the GM). The GCKS takes the role of the KGC and KUC, which sends the *usk* and *gsk* to new members through the established *private channel.* This *group secure channel* is used to distribute Δ to the group's participants before a join or after an exit occurs.

The different phases of KUSS are mapped to the lifecycle of a communication group as follows:

Setup The GCKS sets up a communication group, calls $(msk, mpk_0, \mathbf{g}_0) \overset{r}{\leftarrow} \mathcal{G}(\lambda)$ and stores the values in the GSA.

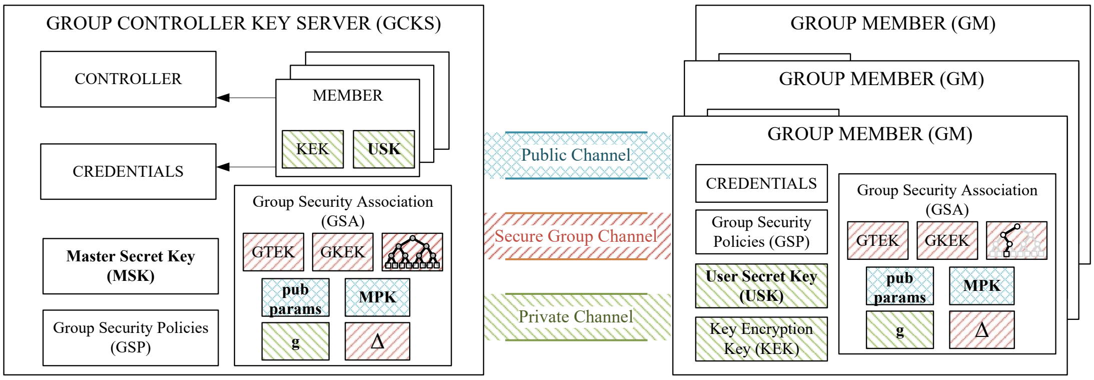


Figure 4.6: IBS key management within a group key management architecture described in RFC 2093 [116] and RFC 2094 [117]. Bold elements are IBS specific, **g** is introduced with 2KSS and Δ re-keying in U2KSS and KUSS. The *Secure Group Channel* managed with LKH or CAKE.

Join A client with id wishes to join the communication group. It establishes a private channel with the GCKS with a GKMP (e.g., G-IKEv2).

1. The GCKS calls $\Delta_{e+1} \xleftarrow{r} \mathcal{N}(\lambda)$ to move to the next epoch and sends a re-key message including Δ_{e+1} to all clients of epoch e through the secure group channel.
2. All clients of epoch e call $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$ and store the new usk_{e+1} in their Security Association and $(mpk_{e+1}, \mathbf{g}_{e+1})$ in their GSA.
3. The GCKS calls $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$ and stores the new $(mpk_{e+1}, \mathbf{g}_{e+1})$ in its GSA.
4. The GCKS calls $usk_{e+1} \xleftarrow{r} \mathcal{E}_{\mathbf{g}_{e+1}}(id)$ and sends $(usk_{e+1}, \mathbf{g}_{e+1}, mpk_{e+1})$ to the joining client.

Leave A client with id is excluded from the communication group.

1. The GCKS calls $\Delta_{e+1} \xleftarrow{r} \mathcal{N}(\lambda)$ to move to the next epoch and sends a re-key message including Δ_{e+1} to all clients of epoch $e+1$ through the secure group channel. Excluding id from this channel can happen with a *CKS*, such as LKH.
2. All clients of epoch $e+1$ call $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$ and store the new usk_{e+1} in their Security Association and $(mpk_{e+1}, \mathbf{g}_{e+1})$ in their GSA.
3. The GCKS calls $k_{e+1} \leftarrow \mathcal{U}_{\Delta_{e+1}}(k_e)$ and stores the new $(mpk_{e+1}, \mathbf{g}_{e+1})$ in its GSA.

Communication All clients of epoch e call $\sigma_e \xleftarrow{r} \mathcal{S}_{usk_e}(m)$ and $\mathcal{V}_{mpk_e,id}(m, \sigma_e)$ to sign and verify messages for epoch e.

4.6 Summary and Findings

The theoretical concepts developed in this chapter are the corner stone of a practicable revocation mechanism as the overall goal of this work. With IBS being an efficient signature mechanism in constrained systems, its keys are integrated in a centralized group key architecture. Hence, the following research question is answered:

> RQ 5 Which signature schemes are usable in constrained systems, can they benefit from Identity Based Cryptography (IBC) and how do they fit in such architectures?

This architecture at hand, a systematic transformation that allows efficient updating of IBS key is developed. Three transformations allow fine-grained stating of mathematical requirement and security properties for each step and will be applied on existing IBS schemes in the next chapter. Thus, we also partially answer research question RQ 6:

> RQ 6: How can IBS keys be revoked and how can the revocation be achieved with state-of-the-art key distribution systems?

The transformation is defined as three consecutive steps, the most efficient transformation being a fully hybrid KUSS scheme, where the *gsk* can be completely eliminated outside the KGC. We show the distribution of update tokens with LKH and CAKE as part of a centralized architecture with a single message. However, the transformations are not restricted to that: As long as all communication participants can agree on an update token, even distributed mechanisms are possible. Hence, the mechanism is able to cope with the specific architectural settings of military communication (see Case 2 in Chapter 2).

5 group Identity Based Signatures

Chapter 4 shows the integration of Identity Based Signature (IBS) in a group key architecture, which fits the architectural assumptions of the use cases. The previous chapter additionally introduces a novel transformation of signature schemes meeting certain mathematical conditions during signature creation and verification. IBS schemes can be efficiently constructed with Elliptic Curve Cryptography (ECC) while allowing straight-forward application of the transformation. Hence, this chapter applies the transformation on existing signature schemes, particularly four based on ECC. We chose them according to the terminology presented in [52].

During the remaining of this work, the following names will be used for distinguishing the schemes:

GG: The one presented by Galindo and Garcia in [47]

vBNN: The one presented by Cao et. al. in [24], which they called *vBNN*.

Hess: The one presented by F. Hess in [67].

BLMQ: The one presented by Barreto, Limbert, McCullagh and Quisquater in [9].

Due to their foundation on ECC, they all at least fulfill Transformability Condition 4.1 as a necessity to apply any of the transformations. Additionally, all of them fulfill Transformability Condition 4.2, while only Hess meets Transformability Condition 4.3. All four are systematically transformed as described in Chapter 4 while preserving correctness and security. Aspects regarding practicability and performance estimations are presented at the end of the chapter.

5.1 Methodology for Transforming existing IBS schemes

ECC-based IBS schemes can be constructed efficiently with so-called pairings (see Section 3.2.2) or with concatenated Schnorr-signatures [153]. The former are the most efficient IBS construction in terms of networking overhead. The use of pairings allows the size of the signature being as small as a single group element of the elliptic curve (in addition to the hash of the message). However, the calculation of pairings is more expensive than the basic elliptic curve operation (e.g., [47] concluded the costs for one pairing being about 21 operations in the finite field of the elliptic curve), why slightly increasing the signature size with a Schnorr-signature allows significant reduction of computation time.

The application of the transformations abides by the following blueprint, to cope with ER 1 - ER 3 defined in Chapter 2:

I.) Show that signing and verification algorithm can be extended by a group-known symmetric key (transforming to Two Key Signature Scheme (2KSS)):

   a) not changing the fundamental operations during signing and verification (e.g., hash functions, operations in cyclic groups). This ensures that the efficient implicit revocation provided by IBS stays intact (ER 2);

  b) introducing only a minimum amount of additional operations (ER 3);
  c) keeping the size of the signature (ER 1);
  d) ensuring the correctness of the verification;

II.) Extending the scheme to allow updates of the shared secret (transforming to Updatable Two Key Signature Scheme (U2KSS)).

III.) Integrating shared secret in the signing and verification keys (transforming to Key Updatable Signature Scheme (KUSS)), while:

  a) not changing the fundamental operations during signing and verification;
  b) introducing only a minimum amount of additional operations (ER 3);
  c) keeping the size of the signature (ER 1);
  d) ensuring the correctness of the verification;

The decision for a certain IBS scheme is use case specific, hence, Section 5.2 first transforms the two Schnorr-based schemes before Section 5.3 applies the transformations to the two Pairing based schemes. Each scheme's three transformation steps are summarized in Table 5.1- 5.4. The rows present the six algorithms *Setup*, *Extract*, *Next*, *Update* (for Trusted Third Party (TTP) and client), *Sign* and *Verify* of the schemes according to Definition 4.3 - 4.7 presented as columns. For readability, the tables gray out calculations which do not change compared to the previous transformation step. All four transformed schemes are analyzed regarding their security implication and practical considerations in Section 5.4 and Section 5.5 respectively.

5.1.1 Preliminaries

One straightforward inclusion of the shared secret to the signing algorithm of all considered schemes is adding it as an additional input to the hash function. This would, however, change the nature of the hash function to a Keyed-Hash Message Authentication Code (HMAC), which require different security considerations. Additionally, this would prevent the KUSS transformation.

5.1.2 Notions

The following notation is used to distinguish elements of elliptic curves and integers:

	Elliptic Curve ($\in \mathbb{G}$)	Integers($\in \mathbb{Z}_p$)
Element of ...	Upper Case	lower case
Hashing algorithm mapping on ...	$H = H_x(y)$	$h = h_x(y)$
User Secret Key element of ...	Usk	usk
Master Public Key element of ...	Mpk	mpk
Random element of ...	$R \xleftarrow{r} \mathbb{G}$	$r \xleftarrow{r} \mathbb{Z}_p^*$

5.2 Transformation of Schemes based on Schnorr Signatures

Signing with RSA is achieved by encrypting the hash of the message with the private key of the signer. Verification is in turn decrypts the signature with the public key. These en- and decryption steps are the computationally most expensive parts of the algorithm. In 1991, C. P. Schnorr introduced a signature scheme, which - unlike e.g., RSA - allows pre-computation for the signer [153]. The modular exponentiation (ME) is applied to a random value (which can be done prior arrival of a message to be signed), while the actual signature is computed with the cheaper modular multiplication (MM) and a hash. The mechanism can be applied to ECC, by simply exchanging the ME with a elliptic curve group exponentiation (GE).

The private key k is chosen at random, $k \xleftarrow{r} \mathbb{Z}_p^*$, the public key is $K = k\,P$. The signature for message m is $\sigma = (s, h)$ calculated as follows:

$$\begin{aligned} x &\xleftarrow{r} \mathbb{Z}_p^* \\ X &= x\,P \\ h &= h_1(X, m) \\ s &= x + k \cdot h \\ \sigma &= (s, h) \end{aligned} \tag{5.1}$$

Notably, X is pre-computable and the signature is accepted if and only if:

$$\begin{aligned} \tilde{X} &= s\,P - h\,K \\ s &\overset{?}{=} h_1(\tilde{X}, m) \end{aligned} \tag{5.2}$$

The idea to construct an IBS scheme with Schnorr-signatures is to concatenate two of them. The Key Generation Center (KGC) generates a Schnorr-signature with its private key (the Master Secret Key (*msk*)) for the user's identity. This signature together with its verification value is the user's private key (the User Secret Key (*usk*)). The verification value can be seen as the public key of the user. Together with the Master Public Key (*mpk*) it can be used to validate another Schnorr-signature for the message and allows authorization of the signer. More details on the generation will be shown during the following transformation of two of such IBS schemes.

5.2.1 Scheme 1: GG

The scheme by Galindo and Garcia [47] (denoted as GG) was presented in 2009 and is almost identical to the first Schnorr-like IBS proposed in [11]. Thus, the following can be used as a blueprint to transform the latter as well. The resulting transformations are presented in Table 5.1, the following will discuss the decisions for each transformation step.

In GG, $usk = (u, R)$, with $u \in \mathbb{Z}_p^*$ being a Schnorr-signature created by the KGC and $R \in \mathbb{G}$. The signature of a message m is then calculated as another Schnorr-signature with u as the secret being kept by the signer and $x \in \mathbb{Z}_p^*$ being the randomized input to the signature. In particular, the signer randomly chooses $x \xleftarrow{r} \mathbb{Z}_p^*$ and calculates $X = x\,P$. It then generates a hash $h = h_2(id, m, X)$ and calculates $s = x + h \cdot u$, with $s \in \mathbb{Z}_p^*$. The signature is $\sigma = (s, R, X)$.

The verifier validates the signature by calculating the hashes of the two Schnorr-signatures, such that $c = h_1(R, id)$ and $d = h_2(id, m, X)$. This allows the verification with the *mpk* ($Mpk \in \mathbb{G}$) of the system by and accepting the signature if and only if

$$s\,P = X + d(R + c\,Mpk) \tag{5.3}$$

I.) 2KSS transformation

GG's signature is $\sigma = (s, R, X)$, but only s, X are part of the signing algorithm while R is part of the user's *usk*. Hence, there are two options for including the group shared key (*gsk*) in the signing algorithm, without changing the hash function, being in s or X.

Option 1: Inclusion of g in X would change its generation to $X = (x \cdot \mathbf{g})\,P$. However, this updated X is included in the hash h which is not part of σ. Unless the output of the hash function is predictable (and thus insecure), the verifier has no way to include **g** in the verification function.

Table 5.1: Transformations for GG.

	2KSS-GG	U2KSS-GG	KUSS-GG
public params	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$
Setup	$msk \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $\mathbf{g} \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $Mpk = mskP$	$msk \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $\mathbf{g}_0 \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $Mpk = mskP$	$msk \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $\mathbf{g}_0 \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $Mpk_0 = (msk \cdot \mathbf{g}_0)P$
Extract	$r \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $R = r\,P$ $u = r + msk \cdot h_1(R, id)$ $usk = (u, R)$	$r \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $R = r\,P$ $u = r + msk \cdot h_1(R, id)$ $usk = (u, R)$	$r \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $R = r\,P$ $u_e = (r + msk \cdot h_1(R, id)) \cdot \mathbf{g}_e$ $usk = (u_e, R)$
Next		$\Delta_{e+1} \overset{r}{\leftarrow} \mathbb{Z}_p^*$	$\Delta_{e+1} \overset{r}{\leftarrow} \mathbb{Z}_p^*$
Update (TTP)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Update (Client)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $u_{e+1} = u_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Sign	$x \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, X)$ $s = (x + h \cdot u) \cdot \mathbf{g}$ $\sigma = (s, R, X)$	$x \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, X)$ $s_e = (x + h \cdot u) \cdot \mathbf{g}_e$ $\sigma_e = (s_e, R, X)$	$x \overset{r}{\leftarrow} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, X)$ $s_e = x + h \cdot u_e$ $\sigma_e = (s_e, R, X)$
Verify	$c = h_1(R, id)$ $d = h_2(id, m, X)$ $sP \overset{?}{=} \mathbf{g}\,(X + d(\,R + c\,Mpk))$	$c = h_1(R, id)$ $d = h_2(id, m, X)$ $s_eP \overset{?}{=} \mathbf{g}_e\,(X + d(\,R + c\,Mpk))$	$c = h_1(R, id)$ $d = h_2(id, m, X)$ $s_eP \overset{?}{=} X + d(\mathbf{g}_eR + c\,Mpk_e)$

^{p} prime number ^{P} generator of an ell. curve group ^{m} message $^{\{0,1\}^*}$ arbitrary length binary string $^{\sigma}$ signature $^{\mathbb{Z}_p^*}$ $\mathbb{Z}_p$ without identity ($\mathbb{1}$) element $^{h_i()}$ hash function in $\mathbb{Z}_p^*$

Option 2: Inclusion of g in s is left as the only applicable. Embedding $\mathbf{g}$ in the Schnorr-signature, changing its calculation to $s = (x + h \cdot u) \cdot \mathbf{g}$, which changes the verification condition as follows:

$$\frac{s}{\mathbf{g}} P = X + d(R + c\,Mpk) \Leftrightarrow s\,P = \mathbf{g}(X + d(R + c\,Mpk)) \tag{5.4}$$

Proof. The signature is correctly verified as $s\,P = \mathbf{g}(\,X + d(R + c\,Mpk))$:

$$\begin{aligned} sP &= (x + h \cdot u) \cdot \mathbf{g}\,P & = \\ &= (x \cdot \mathbf{g})P + (h \cdot u \cdot \mathbf{g})P & = \\ &= \mathbf{g}\,X + d \cdot \mathbf{g}(r + msk \cdot h_1(R, id))P & = \\ &= \mathbf{g}\,X + d \cdot \mathbf{g}(r + msk \cdot c)P & = \\ &= \mathbf{g}\,X + d((r \cdot \mathbf{g})P + (c \cdot msk \cdot \mathbf{g})P) & = \\ &= \mathbf{g}(X + d(R + c \cdot Mpk)) \end{aligned} \tag{5.5}$$

□

II.) U2KSS transformation

The transformation to U2KSS is straightforward by introducing the required algorithms *Next* and *Update*. *Next* generates an update token $\Delta_{e+1} \in \mathbb{Z}_p^*$ for epoch $e + 1$, which is used by the peers to update $\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ The inclusion of the (updatable) shared secret $\mathbf{g}_e$ to the signing and verification algorithm does not change compared to the 2KSS transformation. A signature is correct with respect to the epoch e in which it was created in. Please refer to Table 5.1 for details.

III.) KUSS transformation

Updating the signing and verification keys in GG, requires changing the generation of *usk* and the *Mpk*. The update token Δ generated by the *Next* algorithm is the same as in U2KSS (see Table 5.1, *Next* column); it remains to examine the other algorithms of client and KGC. As the *usk* in GG is a Schnorr-signature of the signers identity, the inclusion of the shared secret to the signing and verification keys (usk, Mpk) is similar to its integration to the signature in 2KSS-GG. However, this implicitly changes the signing algorithm, why verification also requires attention. First, the updatable shared secret $\mathbf{g}_e$ is used in the *extract* algorithm to produce a *usk* with respect to epoch e. As the *usk* consists of two variables (u, R) it can be either included in both or one of them.

Option 1: Inclusion of g in R would require to change *Extract* as follows:

$$\begin{aligned} r &\xleftarrow{r} \mathbb{Z}_p^*; \\ R_e &= (r \cdot \mathbf{g}_e)\,P \\ u_e &= r + msk \cdot h_1(R_e, id) \\ usk &= (u_e, R_e) \end{aligned} \tag{5.6}$$

This yields the same disadvantage as including $\mathbf{g}$ in X for the signature creation in 2KSS-GG, as R_e is the input of the hash function used to generate the Schnorr-signature for the signers identity. The inclusion of $\mathbf{g}_e$ to R already affects u as the second part of the *usk* and to successfully update the *usk* the client needs to apply Δ_{e+1} to (R_e, u_e). While it is certainly possible to update R_e (e.g., $R_{e+1} = \Delta_{e+1}\,R_e$) and create the hash $h_1(R_{e+1}, id)$, updating u_e is impossible without the knowledge of r and msk, which are meant to be kept secret from the user.

Option 2: Inclusion of $\mathbf{g}$ in u would change *Extract* as follows:

$$\begin{aligned} u_e =& (r + msk \cdot h_1(R, id)) \cdot \mathbf{g}_e \\ usk =& (u_e, R) \end{aligned} \tag{5.7}$$

This allows a straightforward update of u by the client, e.g., $u_{e+1} = u_e \cdot \Delta_{e+1}$ and the signature creation is implicitly changed to:

$$s_e = x + h \cdot u \cdot \mathbf{g}_e \tag{5.8}$$

For verifying the knowledge of $\mathbf{g}_e$, the verifier validates s_e without the knowledge of x and u_e, but with the *Mpk* in combination with (R, X, id). As in the original scheme, the verifier calculates $s_e\, P$, which now includes $\mathbf{g}$ and requires the verification condition to be examined. By re-writing $s_e\, P$ to

$$\begin{aligned} s_e\, P = (x + h \cdot u \cdot \mathbf{g}_e)P & \quad = \\ = X + h \cdot u \cdot \mathbf{g}_e P & \quad = \\ = X + h \cdot \Big(r + msk \cdot h_1(R, id)\Big) \cdot \mathbf{g}_e P & \quad = \\ = X + h \cdot \Big(\mathbf{g}_e \cdot r + \mathbf{g}_e \cdot msk \cdot h_1(R, id)\Big)P & \quad = \\ = X + h \cdot \Big(\mathbf{g}_e R + h_1(R, id) \cdot \mathbf{g}_e\, Mpk\Big) & \end{aligned} \tag{5.9}$$

as $d = h_2(id, M, X) = h$ and $d = h_1(R, id)$, the verifier can successfully verify the signers knowledge of $\mathbf{g}_e$ with respect to epoch e, if and only if the verifier knows $\mathbf{g}_e$. Thus, GG is not suitable for a hybrid KUSS verification as defined with Transformability Condition 4.3, unless the signature is extended to:

$$\sigma_e = (s_e, R, R_e, X), \text{with } R_e = \mathbf{g}_e\, R \tag{5.10}$$

Updating the mpk: As shown before, the pre-calculation of $\mathbf{g}_e\, R$ is impossible, as R is not updatable, why $\mathbf{g}_e$ is required during verification and needs to be calculated in the client's *Update* algorithm. However, this is not true for the *Mpk*. Upon setup, the KGC can include $\mathbf{g}$ in the calculation of the *Mpk*, e.g., $Mpk_0 = msk \cdot \mathbf{g}_0\, P$. Moving Mpk_e to the next epoch is straightforward with $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$, which can be done by all clients during the *Update* algorithm. This allows more efficient verification, as $\mathbf{g}_e\, Mpk$ does not need to be calculated for every message and the KUSS-GG verification condition is thus:

$$s_e P = X + d(\mathbf{g}_e R + c\, Mpk_e) \tag{5.11}$$

The correctness of the signature is shown with:

Proof.

$$\begin{aligned} s_e P = (x + h \cdot u_e)P & \quad = \\ = xP + (h \cdot u_e)P & \quad = \\ = X + d \cdot \mathbf{g}_e(r + msk \cdot h_1(R, id))P & \quad = \\ = X + d \cdot \mathbf{g}_e(r + msk \cdot c)P & \quad = \\ = X + d((r \cdot \mathbf{g}_e)P + (c \cdot msk \cdot \mathbf{g}_e)P) & \quad = \\ = X + d(\mathbf{g}_e R + c \cdot Mpk_e) & \end{aligned} \tag{5.12}$$

□

5.2.2 Scheme 2: vBNN

The IBS scheme called vBNN [24] also uses Schnorr-signatures and is a variant of BNN that was introduced in [11]. Its main motivation is the reduction of the signature size to achieve better performance in Wireless Sensor Networks (WSNs).

The first difference between vBNN and GG is the creation of the hash h during signing (and therefore also during verification). In GG, this is $h = h_2(id, m, X)$, while vBNN calculates $h = h_2(id, m, R, X)$. GG defines the signature σ as a triple of $\sigma = (s, R, X)$, which requires one element of $\mathbb{Z}_p^*$ and two elements of $\mathbb{G}$ to be sent on the wire. As elements of $\mathbb{G}$ typically require more bits, vBNN optimizes this by sending the triple $\sigma = (s, R, h)$. The verification algorithms changes to:

$$\begin{aligned} c =& h_1(R, id) \\ \widetilde{X} =& s\,P - h(R + c\,Mpk) \\ h \overset{?}{=}& h_2(id, M, R, \widetilde{X}) \end{aligned} \tag{5.13}$$

The signature is valid, if the two hashes are equal, which is the case if $\widetilde{X}$ is equal to the randomly generated $X = x\,P$.

The transformation of vBNN is almost identical to the transformation of GG and the resulting transformation are presented in Table 5.2. The following highlights the decisions which differ to GG.

I.) 2KSS transformation

As in GG, the transformation to 2KSS-vBNN simply extends the signature's calculation to $s = (x + h \cdot u) \cdot \mathbf{g}$. The verification is changed analogously, such that the calculation $\widetilde{X}$ verifies the knowledge of $\mathbf{g}$:

$$\widetilde{X} = \frac{s}{\mathbf{g}} P - h(R + c\,Mpk) \tag{5.14}$$

II.) U2KSS transformation

As in GG, the transformation to U2KSS is straightforward by introducing the requested algorithm *Next* and *Update*. The inclusion of the (updatable) shared secret $\mathbf{g}_e$ to the signing and verification algorithm does not change compared to the 2KSS transformation, why the correctness stays intact with respect to the epoch e, the signature was created.

III.) KUSS transformation

The KUSS-vBNN transformation follows the argumentation for KUSS-GG. The shared secret is included in the $usk = (u, R)$, more specifically in u that can be updated by the client by calculating:

$$u_{e+1} = u_e \cdot \Delta_{e+1} \tag{5.15}$$

The Mpk_e can be updated using the same operation as in GG, but as R is not updatable, $\mathbf{g}_e$ is necessary for verification.

As in GG, $\mathbf{g}_e$ is not required during signing but for verification, which changes compared to Equation 5.13 as follows:

$$\begin{aligned} c =& h_1(R, id) \\ \widetilde{X} =& s\,P - h(\mathbf{g}_e\,R + c\,Mpk_e) \\ h \overset{?}{=}& h_2(id, M, R, \widetilde{X}) \end{aligned} \tag{5.16}$$

Table 5.2: Transformations for vBNN.

	2KSS-vBNN	U2KSS-vBNN	KUSS-vBNN
public params	$h_{1,2} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$	$h_{1,2} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$	$h_{1,2} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$
Setup	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g} \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = msk P$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = msk P$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk_0 = (msk \cdot \mathbf{g}_0) P$
Extract	$r \xleftarrow{r} \mathbb{Z}_p^*$ $R = r\,P$ $u = r + msk \cdot h_1(R, id)$ $usk = (u, R)$	$r \xleftarrow{r} \mathbb{Z}_p^*$ $R = r\,P$ $u = r + msk \cdot h_1(R, id)$ $usk = (u, R)$	$r \xleftarrow{r} \mathbb{Z}_p^*$ $R = r\,P$ $u_e = (r + msk \cdot h_1(R, id)) \cdot \mathbf{g}_e$ $usk = (u_e, R)$
Next		$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$	$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$
Update (TTP)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Update (Client)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $u_{e+1} = u_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Sign	$x \xleftarrow{r} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, R, X)$ $s = (x + h \cdot u) \cdot \mathbf{g}$ $\sigma = (s, R, h)$	$x \xleftarrow{r} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, R, X)$ $s_e = (x + h \cdot u) \cdot \mathbf{g}_e$ $\sigma_e = (s_e, R, h)$	$x \xleftarrow{r} \mathbb{Z}_p^*$ $X = xP$ $h = h_2(id, m, R, X)$ $s_e = x + h \cdot u_e$ $\sigma_e = (s_e, R, h)$
Verify	$c = h_1(R, id)$ $\widetilde{X} = \frac{s}{\mathbf{g}} P - h(R + c\, Mpk)$ $h \stackrel{?}{=} h_2(id, m, R, \widetilde{X})$	$c = h_1(R, id)$ $\widetilde{X} = \frac{s_e}{\mathbf{g}_e} P - h(R + c\, Mpk)$ $h \stackrel{?}{=} h_2(id, m, R, \widetilde{X})$	$c = h_1(R, id)$ $\widetilde{X} = s_e P - h(\mathbf{g}_e R + c\, Mpk_e)$ $h \stackrel{?}{=} h_2(id, m, R, \widetilde{X})$

^{p} prime number ^{P} generator of an ell. curve group ^{m} message
$^{\{0,1\}^*}$ arbitrary length binary string $^{\sigma}$ signature $^{\mathbb{Z}_p^*}$ $\mathbb{Z}_p$ without identity ($\mathbb{1}$) element
$^{h_i()}$ hash function in $\mathbb{Z}_p^*$

Proof. The signature is correctly verified if $\widetilde{X} = X$

$$\begin{aligned}\widetilde{X} &= s_eP - h(\mathbf{g}_eR + c\,Mpk_e) &&= \\ &= (x + h \cdot u_e)P - h((r \cdot \mathbf{g}_e)P + (c \cdot msk \cdot \mathbf{g}_e)P) &&= \\ &= xP + (h \cdot u_e)P - h(r \cdot \mathbf{g}_e + (c \cdot msk \cdot \mathbf{g}_e)P &&= \\ &= xP + (h \cdot u_e)P - h(r + h_1(R, id) \cdot msk) \cdot \mathbf{g}_e\, P &&= \\ &= X + (h \cdot u_e)P - (h \cdot u_e)P &&= \\ &= X\end{aligned} \tag{5.17}$$

□

Thus, vBNN is not suitable for a hybrid KUSS verification as defined with Transformability Condition 4.3, unless the signature is extended to:

$$\sigma_e = (s_e, R, R_e, X), \text{with } R_e = \mathbf{g}_e\, R \tag{5.18}$$

5.3 Transformation of Schemes based on Pairings

Pairings were introduced to ECC in [73] and have since been used for various cryptographic applications. One of such is the use for identity-based cryptography and in particular for IBS. The following will examine two prominent examples of pairing based IBS schemes, namely by Hess [67] and the one donated as BLMQ [9] . BLMQ is slightly more efficient, as it requires no expensive pairing operation for signing and only one pairing operation for verification, while Hess requires one for signing and two for verification. However, Hess is the only scheme that is found to be capable of fully hybrid KUSS, where *usk* as well as *mpk* can be changed in such a way that the knowledge of the shared secret is unnecessary during signing and verification.

5.3.1 Scheme 3: Hess

The IBS scheme developed by Florian Hess [67] was among the first using bilinear maps on elliptic curves for identity-based cryptography. While the creation of *msk* and *Mpk* does not differ from the IBS schemes based on Schnorr-signatures, the other steps are different. The resulting transformations are presented in Table 5.3, the following discusses the concepts of Hess' algorithm and the decisions for each transformation step.

In Hess, the *usk* is a point on the elliptic curve, created by hashing the user's *id* into $\mathbb{G}$[1]. Creating a signature requires two random elements, $x \xleftarrow{r} \mathbb{Z}_p^*$ and $Q \xleftarrow{r} \mathbb{G}$, and a pairing with the generator P. The result $r = e(Q, P)^x$ is included in the signature $\sigma = (h, S)$, with $h = h_2(m, r)$ and $S \in \mathbb{G}$ such that:

$$S = h\,Usk + x\,Q \quad (\text{with } Usk = msk\,H_1(id)) \tag{5.19}$$

The verification condition is based on the fact, that another pairing $e(S, P)$ implicitly includes $r = e(Q, P)^x$:

$$\begin{aligned}e(S, P) &= e\Big(h \cdot mskH_1(id) + x\,Q, P\Big) &&= \\ &= e\Big(h \cdot msk\,H_1(id)\Big) \cdot e(xQ, P) &&= \\ &= e(H_1(id), P)^{h \cdot msk} \cdot e(Q, P)^x &&= \\ &= e(H_1(id), P)^{h \cdot msk} \cdot r\end{aligned} \tag{5.20}$$

[1] This involves some restriction in the choice of $\mathbb{G}$, refer to [46] for details

Table 5.3: Transformations for Hess.

	2KSS-Hess	U2KSS-Hess	KUSS-Hess
public params	$H_1 : \{0,1\}^* \to \mathbb{G}^*$ $h_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$	$H_1 : \{0,1\}^* \to \mathbb{G}^*$ $h_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$	$H_1 : \{0,1\}^* \to \mathbb{G}^*$ $h_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$
Setup	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g} \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = mskP$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = mskP$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk_0 = (msk \cdot \mathbf{g}_0)P$
Extract	$Usk = mskH_1(id)$	$Usk = mskH_1(id)$	$Usk_e = (msk \cdot \mathbf{g}_e)H_1(id)$
Next		$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$	$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$
Update (TTP)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Update (Client)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$Usk_{e+1} = \Delta_{e+1} Usk_e$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Sign	$x \xleftarrow{r} \mathbb{Z}_p^*, Q \xleftarrow{r} \mathbb{G}^*$ $r = e(Q,P)^x$ $h = h_2(m,r)$ $S = (\mathbf{g} \cdot h)\, Usk + xQ$ $\sigma = (h,S)$	$x \xleftarrow{r} \mathbb{Z}_p^*, Q \xleftarrow{r} \mathbb{G}^*$ $r = e(Q,P)^x$ $h = h_2(m,r)$ $S_e = (\mathbf{g}_e \cdot h)\, Usk + xQ$ $\sigma_e = (h,S_e)$	$x \xleftarrow{r} \mathbb{Z}_p^*, Q \xleftarrow{r} \mathbb{G}^*$ $r = e(Q,P)^x$ $h = h_2(m,r)$ $S_e = h\, Usk_e + xQ$ $\sigma_e = (h,S_e)$
Verify	$\widetilde{r} = e(S,P)$ $\cdot e(H_1(id), -Mpk)^{\frac{h}{\mathbf{g}}}$ $h \overset{?}{=} h_2(m,\widetilde{r})$	$\widetilde{r} = e(S,P)$ $\cdot e(H_1(id), -Mpk)^{\frac{h}{\mathbf{g}_e}}$ $h \overset{?}{=} h_2(m,\widetilde{r})$	$\widetilde{r} = e(S,P)$ $\cdot e(H_1(id), -Mpk_e)^h$ $h \overset{?}{=} h_2(m,\widetilde{r})$

^{p} prime number ^{P} generator of an ell. curve group ^{m} message $^\sigma$ signature $^{\{0,1\}^*}$ arbitrary length binary string $^{h_i()}$ hash function in $\mathbb{Z}_p^*$ $^{H_i()}$ hash function in $\mathbb{G}$ $^{\mathbb{Z}_p^*}$ $\mathbb{Z}_p$ without identity ($\mathbb{1}$) element $^{\mathbb{G}}$ cyclic group generated by P $^{\mathbb{G}^*}$ $\mathbb{G}$ without identity element $^{e(P,Q)}$ bilinear pairing

As r was kept secret by the signer, it can only be known by him and, thus, validates the signature. For extracting r from $e(S, P)$, the verifier needs to calculate another pairing, which eliminates $e(H_1(id), P)^{h \cdot msk}$. This can be achieved with the knowledge of $Mpk = msk\, P$, as

$$e(H_1(id), Mpk) = e(H_1(id), P)^{msk} \tag{5.21}$$

The hash h is part of the signature and

$$e(H_1(id), -Mpk)^{h} = e(H_1(id), P)^{-h \cdot msk} \tag{5.22}$$

With Equation 5.20 multiplying $e(S, P)$ and $e(H_1(id), -Mpk)^h$ allows extraction of r from S. That forms the validation condition:

$$\begin{aligned} \tilde{r} =& e(S, P) \cdot e(H_1(id), -Mpk)^{h} & = \\ =& e(H_1(id), P)^{h \cdot msk} \cdot e(Q, P)^{x} \cdot e(H_1(id), P)^{-h \cdot msk} & = \\ =& e(Q, P)^{x} & \\ \Rightarrow h \overset{?}{=}& h_2(M, \tilde{r}) & \end{aligned} \tag{5.23}$$

I.) 2KSS transformation

There are three options for including gsk in Hess' signing and verification, namely in S itself or in $h\, Usk$ or $x\, Q$ as parts of S.

Option 1: including $\mathbf{g}$ in S The straightforward option of signing with $\mathbf{g}$ is to change the creation of S to

$$S = \mathbf{g}(h\, Usk + x\, Q) = (\mathbf{g} \cdot h)\, Usk + (\mathbf{g} \cdot x)\, Q \tag{5.24}$$

and change the creation of $\tilde{r}$ during verification to:

$$\begin{aligned} \tilde{r} =& e(S, P)^{\mathbf{g}^{-1}} \cdot e(H_1(id), -Mpk)^{h} & = \\ =& e\Big((h \cdot msk \cdot \mathbf{g})\, H_1(id) + (x \cdot \mathbf{g})\, Q, P\Big)^{\mathbf{g}^{-1}} \cdot e(H_1(id), -Mpk)^{h} & = \\ =& e(H_1(id), P)^{\frac{h \cdot msk \cdot \mathbf{g}}{\mathbf{g}}} \cdot e(Q, P)^{\frac{x \cdot \mathbf{g}}{\mathbf{g}}} \cdot e(H_1(id), -Mpk)^{h} & = \\ =& e(Q, P)^{x} \Rightarrow r & \end{aligned} \tag{5.25}$$

Allowing the successful extraction of $e(Q, P)^x$ as described above with the costs of two additional multiplications in $\mathbb{Z}_p^*$ for signing and one additional exponentiation in $\mathbb{Z}_p^*$ for verification.

Option 2: including $\mathbf{g}$ in $x\, Q$ The modification to

$$S = h\, Usk + (x \cdot \mathbf{g})\, Q \tag{5.26}$$

changes the creation of $\tilde{r}$ during verification to

$$\begin{aligned} \tilde{r} &= e(S, P)^{-\mathbf{g}} \cdot e(H_1(id), -Mpk)^{-\mathbf{g} \cdot h} \\ \Rightarrow h &\overset{?}{=} h_2(M, \tilde{r}) \end{aligned} \tag{5.27}$$

which - with one additional multiplication in $\mathbb{Z}_p^*$ - is slightly more efficient for signing but - with one exponentiation and multiplication in $\mathbb{Z}_p^*$ - less efficient for verification.

Option 3: including g in $h\,Usk$ The inclusion in $h\,Usk$ changes signing to

$$S = (h \cdot \mathbf{g})\,Usk + x\,Q \tag{5.28}$$

and the creation of $\widetilde{r}$ during verification to

$$\begin{aligned} \widetilde{r} &= e(S, P) \cdot e(H_1(id), -Mpk)^{\frac{h}{\mathbf{g}}} \\ \Rightarrow h &\overset{?}{=} h_2(M, \widetilde{r}) \end{aligned} \tag{5.29}$$

hence, only requiring one additional multiplication in $\mathbb{Z}_p^*$ for signing and verification. Additionally, this allows straightforward transformation to KUSS.

II.) U2KSS transformation

The transformation to U2KSS is straightforward by introducing the required algorithm *Next* and *Update*. The inclusion of the (updatable) shared secret $\mathbf{g}_e$ to the signing and verification algorithm does not change compared to the 2KSS transformation, why the correctness stays intact with respect to the epoch e, in which the signature was created. Please refer to Table 5.3 for details.

III.) KUSS transformation

The options for 2KSS-Hess discussed above, lead to the integration of the shared secret exclusively where *Usk* and *Mpk* appear during signing and verifying. Thus, the transformation to KUSS is the simple integration in both keys and evolving them during the *Update* process. In addition, KUSS-Hess does only require the KGC to keep the shared secret for *extracting* Usk_e with respect to epoch e. It is therefore capable of a fully hybrid KUSS, as defined by Transformability Condition 4.2 and 4.3. Please refer to Table 5.3 for details.

With the transformation options discussed in 2KSS-Hess, proving the correctness of a KUSS-Hess signature is thus:

Proof. The signature is correctly verified if $\widetilde{r} = e(Q, P)^x$:

$$\begin{aligned} \widetilde{r} &= e(S_e, P) \cdot e(H_1(id), -Mpk_e)^h && = \\ &= e(h\,Usk_e + xQ, P) \cdot e(H_1(id), -Mpk_e)^h && = \\ &= e((h \cdot msk \cdot \mathbf{g}_e)H_1(id) + xQ, P) \cdot e(H_1(id), -(msk \cdot \mathbf{g}_e)P)^h && = \\ &= e(H_1(id), P)^{h \cdot msk \cdot \mathbf{g}_e} \cdot e(Q, P)^x \cdot e(H_1(id), P)^{-h \cdot msk \cdot \mathbf{g}_e} && = \\ &= e(Q, P)^x \end{aligned} \tag{5.30}$$

□

5.3.2 Scheme 4: BLMQ

BLMQ as originally introduced in [9] also utilizes pairings but claims better efficiency than e.g., Hess. Additionally, it overcomes the restriction for hashing in $\mathbb{G}$ by generating the *Usk* differently:

$$Usk = \frac{1}{msk + h_1(id)} P \tag{5.31}$$

BLMQ also allows one of the pairing operation – $e(P, Q)$ – for signing and verification to be pre-calculated as the generators $Q \in \mathbb{G}_2, P \in \mathbb{G}_1$ are fixed during *Setup*. The hash of the signature

$\sigma = (h, S)$ is similar to Hess, being $h = h_2(M, r)$, where $r = e(P, Q)^x$ with $x \xleftarrow{r} \mathbb{Z}_p^*$ The second part of the signature $S \in \mathbb{G}_1$, is

$$S = (x + h)\, Usk \tag{5.32}$$

Similar to the verification condition in Hess, the verifier in BLMQ needs to compute $\widetilde{r}$ to compare the hash of the signature with a self-generated one

$$h \overset{?}{=} h_2(m, \tilde{r}) \tag{5.33}$$

Thus, q^x needs to be computed from the signature S, by performing the pairing $e(S, h_1(id)\, Q + Mpk)$ which can be re-written:[2]:

$$\begin{aligned} &e(S, h_1(id)\, Q + Mpk) &= \\ =&e\Big(\frac{x+h}{msk + h_1(id)}\, P, h_1(id)\, Q\Big) \cdot e\Big(\frac{x+h}{msk + h_1(id)}\, P, msk\, Q\Big) &= \\ =&e(P, Q)^{x+h} &= \\ =&q^{x+h} & \end{aligned} \tag{5.34}$$

Multiplying the result of this pairing with q^{-h} returns $\tilde{r} = q^x = r$, allowing successful verification. The resulting transformations are presented in Table 5.4, the following will discuss the decisions for each transformation step.

I.) 2KSS transformation

The *gsk* can be included in signing and verification, either multiplicative or additive.

Option 1 (multiplicative): The straightforward inclusion of the shared secret to the signature is its multiplication with the original S, such that

$$S = \mathbf{g} \cdot (x + h)\, Usk \tag{5.35}$$

which allows efficient signing and verification by only one additional multiplication in $\mathbb{Z}_p^*$ in each step (see KUSS-BLMQ for more details). Additionally, this includes a potential security issue for U2KSS-BLMQ, which will be discussed in Section 5.5.2.

Option 2 (additive): However, even greater efficiency is achieved by modifying the signature creation to

$$S = (\mathbf{g} + x + h)\, Usk \tag{5.36}$$

allowing modification of the verification, such that

$$\tilde{r} = e(S, h_1(id)\, Q + Mpk) \cdot e(Q, P)^{-h-\mathbf{g}} \tag{5.37}$$

II.) U2KSS transformation

The transformation to U2KSS is straightforward by introducing the required algorithm *Next* and *Update*. The inclusion of the (updatable) shared secret $\mathbf{g}_e$ to the signing and verification algorithm does not change compared to the 2KSS transformation, why the correctness stays intact with respect to the epoch e, the signature was created in. Please refer to Table 5.4 for details.

[2] As the reformation is almost identical to Hess, the interested reader is referred to Equation 5.43

Table 5.4: Transformations for BLMQ.

	2KSS-BLMQ	U2KSS-BLMQ	KUSS-BLMQ
public params	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$ $Q \leftarrow \mathbb{G}_2$ $P = \psi(Q) \in \mathbb{G}_1$ $q = e(P,Q)$	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$ $Q \leftarrow \mathbb{G}_2$ $P = \psi(Q) \in \mathbb{G}_1$ $q = e(P,Q)$	$h_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$ $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{Z}_p^*$ $Q \leftarrow \mathbb{G}_2$ $P = \psi(Q) \in \mathbb{G}_1$ $q = e(P,Q)$
Setup	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g} \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = mskQ$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk = mskQ$	$msk \xleftarrow{r} \mathbb{Z}_p^*$ $\mathbf{g}_0 \xleftarrow{r} \mathbb{Z}_p^*$ $Mpk_0 = (msk \cdot \mathbf{g}_0)Q$
Extract	$Usk = \frac{1}{msk + h_1(id)} P$	$Usk = \frac{1}{msk + h_1(id)} P$	$Usk_e = \frac{1}{(msk + h_1(id)) \cdot \mathbf{g}_e} P$
Next		$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$	$\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$
Update (TTP)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Update (Client)		$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$	$\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ $Usk_{e+1} = (1/\Delta_{e+1}) Usk_e$ $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$
Sign	$x \xleftarrow{r} \mathbb{Z}_p^*$ $r = q^x$ $h = h_2(m, r)$ $S = (\mathbf{g} + x + h)\, Usk$ $\sigma = (h, S)$	$x \xleftarrow{r} \mathbb{Z}_p^*$ $r = q^x$ $h = h_2(m, r)$ $S_e = (\mathbf{g}_e + x + h)\, Usk$ $\sigma_e = (h, S_e)$	$x \xleftarrow{r} \mathbb{Z}_p^*$ $r = q^x$ $h = h_2(m, r)$ $S_e = (x + h)\, Usk_e$ $\sigma_e = (h, S_e)$
Verify	$\tilde{r} = e(S, h_1(id)\, Q + Mpk)$ $\cdot q^{-h-\mathbf{g}}$ $h \overset{?}{=} h_2(m, \tilde{r})$	$\tilde{r} = e(S_e, h_1(id)\, Q + Mpk)$ $\cdot q^{-h-\mathbf{g}_e}$ $h \overset{?}{=} h_2(m, \tilde{r})$	$\tilde{r} = e(S_e, h_1(id) \cdot \mathbf{g}_e Q + Mpk_e)$ $\cdot q^{-h}$ $h \overset{?}{=} h_2(m, \tilde{r})$

^{p} prime number ^{Q,P} generator of an ell. curve group ^{m} message ${}^{\sigma}$ signature
${}^{\{0,1\}^*}$ arbitrary length binary string ${}^{\mathbb{G}_x}$ cyclic group generated by P, Q
${}^{\mathbb{Z}_p^*}$ $\mathbb{Z}_p$ without identity ($\mathbb{1}$) element ${}^{h_i()}$ hash function in $\mathbb{Z}_p^*$ ${}^{e(P,Q)}$ bilinear pairing
${}^{\psi}$ isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(Q) = P$

III.) KUSS transformation

As a result of the more efficient generation of the *Usk* in BLMQ, the integration of the shared secret is more difficult than in Hess. Again, there are the two options for additive and multiplicative inclusion.

Option 1 (additive): As for 2KSS-BLMQ, adding the shared secret such as

$$Usk = \frac{1}{msk + h_1(id) + \mathbf{g}} P \tag{5.38}$$

would be straightforward and efficient, however, as the *msk* is unknown by the clients, updating the *Usk* would become impossible.

Option 2 (multiplicative): Hence, multiplication is the only way of integrating the shared secret to *Usk* and *Mpk* respectively. *Extract* changes to:

$$Usk = \frac{1}{(msk + h_1(id))\mathbf{g}_e} P \tag{5.39}$$

which implicitly changes the signature creation to

$$S = \frac{x + h}{(msk + h_1(id))\mathbf{g}_e} P \tag{5.40}$$

During validation of the signature, $\mathbf{g}$ is required as a multiplicative element to even out the signature. As the interested reader may have noticed, this requires $\mathbf{g}$ to appear in combination with $msk + h_1(id)$ during the creation of $\tilde{r}$. The appearance with *msk* is pre-calculated as part of the *Mpk*, while the appearance with $h_1(id)$ needs to be explicit

$$\tilde{r} = e(S, (h_1(id) \cdot \mathbf{g}_e)\, Q + Mpk_e) \cdot e(Q, P)^{-h}; \text{with} Mpk_e = (msk \cdot \mathbf{g}_e)\, Q \tag{5.41}$$

Updating the *Usk* needs to account for division, why is as:

$$Usk_{e+1} = ({}^{1}/_{\Delta_{e+1}}) Usk_e \tag{5.42}$$

Updating the mpk: The update process for Mpk_e and $\mathbf{g}_e$ is as in any other scheme, refer to Table 5.4 for more details. With these modifications, the correctness of a KUSS-BLMQ signature is proven in Equation 5.43.

Proof. The signature is correctly verified if $\widetilde{r} = q^x$:

$$\begin{aligned}
\widetilde{r} &= e(S_e, (h_1(id) \cdot \mathbf{g}_e)Q + Mpk_e) \cdot q^{-h} && = \\
&= e(S_e, (h_1(id) \cdot \mathbf{g}_e)Q) \cdot e(S_e, Mpk_e) \cdot q^{-h} && = \\
&= e\Big((x + h)Usk_e, (h_1(id) \cdot \mathbf{g}_e)Q\Big) \cdot e\Big((x + h)\, Usk_e, Mpk_e\Big) \cdot q^{-h} && = \\
&= e\Big(\frac{x + h}{(msk + h_1(id)) \cdot \mathbf{g}_e} P, (h_1(id) \cdot \mathbf{g}_e)Q\Big) \cdot \\
&\quad \cdot e\Big(\frac{x + h}{(msk + h_1(id)) \cdot \mathbf{g}_e} P, (msk \cdot \mathbf{g}_e)Q\Big) \cdot q^{-h} && = \\
&= q^{\frac{(x+h) \cdot h_1(id) \cdot \mathbf{g}_e}{(msk + h_1(id)) \cdot \mathbf{g}_e}} \cdot q^{\frac{(x+h) \cdot msk \cdot \mathbf{g}_e}{(msk + h_1(id)) \cdot \mathbf{g}_e}} \cdot q^{-h} && = \\
&= q^{\frac{(x+h) \cdot h_1(id) \cdot \mathbf{g}_e + (x+h) \cdot msk \cdot \mathbf{g}_e}{(msk + h_1(id)) \cdot \mathbf{g}_e} - h} && = \\
&= q^{x + h - h} = q^x
\end{aligned} \tag{5.43}$$

□

5.4 Security Analysis

The aim of this section is to formally prove the security of the previously transformed schemes with techniques from provable security [60]. An IBS scheme is meant to be most secure if it meets existentially unforgeable under adaptively chosen-message-and-identity attacks (EUF-CMA). The property EUF-CMA is defined for all transformations in Section 4.4.1-4.4.3.

The analysis is carried out as follows:

1. Proving *Token Security* for the 2KSS and U2KSS transforms, stating that the security of the IBS scheme is still EUF-CMA if a *gsk* is added to signing and verification process. We show, that a forged 2KSS signature can be converted to a signature of the underlying IBS scheme. Using the *Game Hopping Lemma* (refer to Section 3.1.3), we hereby prove by *Contradiction* that the underlying scheme would be insecure if 2KSS were insecure.

2. Proving *Token Security* for the KUSS transforms, stating that the security of the IBS scheme is still EUF-CMA, if a *gsk* included in the signing and verification keys and therefore in the signature. The principle is similar to the previous step, however, the reduction of Schnorr based schemes is not as straightforward as for 2KSS and U2KSS. In fact, the additional *Latin Square Property* needs to be examined for them.

3. Showing the signature's *Forward Security* by examining the *Latin Square Property.*

4. Showing the signature's *Post-Compromise Security*, which can be reduced to the One-More Discrete Logarithm Problem (1m-DLP) for all schemes by using the *Latin Square Property.*

5.4.1 Preliminaries for the Analysis

Most of the following proofs can be done via reduction to the security of the original schemes, which define EUF-CMA with some variant of the following experiment, which is equivalent to Definition 1 in the paper by Galindo and Garcia [47]:

Definition 5.1 (EUF-IBS-CMA [47])**.** *An identity-based signature scheme* $\Sigma = (\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$ *is said to be secure against existential forgery under adaptive chosen message and identity attacks if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, the probability of the experiment* $P(\mathbf{EUF\breve{}IBS\breve{}CMA}_\Sigma(\mathcal{A}) = 1)$ *defined below is a negligible function of* λ*. During this experiment,* $\mathcal{A}$ *has access to two of the oracles defined above, namely* $\mathcal{O}_\mathcal{E}$ *and* $\mathcal{O}_\mathcal{S}$*.*

```
$\mathbf{EUF\breve{}IBS\breve{}CMA}_\Sigma(\mathcal{A})$ :
$(msk, mpk) \leftarrow \mathcal{G}(\lambda)$
$(id^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{E}(\cdot), \mathcal{O}_\mathcal{S}(\cdot,\cdot)}(mpk)$
return $\mathcal{V}(mpk, \sigma^*, m^*, id^*)$
```

Trivial wins where the adversary $\mathcal{A}$ queries id^* from $\mathcal{O}_\mathcal{E}(\cdot)$ or σ^* from $\mathcal{O}_\mathcal{S}(\cdot,\cdot)$ are excluded. Also, the same *id* is not allowed to be queried twice to $\mathcal{O}_\mathcal{E}(\cdot)$.

Where direct reduction is not possible, we reduce a scheme's security to one of the following:

I.) **Latin square property**, which shows the preservation of randomness for group operations in $\mathbb{Z}_p^*$.

Definition 5.2 (One-More Discrete Logarithm Problem [12, 81])**.** *With K_{cg} being a cyclic group generator, the experiment for the 1m-DLP is given as follows [12]:*

```
$\mathbf{EXP}^{\textit{1m-dlp}}_{K_{cg},A}(k)$ :
    $(\langle\mathbb{G}\rangle, q, g) \overset{r}{\leftarrow} K_{cg}(1^k)$
    $i \leftarrow 0; n \leftarrow 0$
    $(y_1 \cdots, y_m) \overset{r}{\leftarrow} A\Big(1^k, \langle\mathbb{G}\rangle, q, g : \texttt{Chall}, \texttt{DLog}\Big)$
    If $m = 1$ and $n < m$ and $g^{y_i}$ for all $i \in \{1, \cdots m\}$
    Then return $1$ else return $0$

Oracle $\texttt{Chall}$ :
    $i \leftarrow i + 1; Y_i \overset{r}{\leftarrow} \mathbb{G}$
    Return $Y_i$
Oracle $\texttt{DLog}(Y)$ :
    $n \leftarrow n + 1; y \leftarrow d\log_{\mathbb{G},g}(Y)$
    Return $y$
```

The solver is supplied with a challenge oracle that produces a random group element $Y_i \in \mathbb{G}$ when queried and a discrete log oracle. After t queries to the challenge oracle (where t is chosen by the solver) and at most $t - 1$ queries to the discrete log oracle, the solver must find the discrete logs of all t elements Y_i [81].

II.) One-More Discrete Logarithm Problem, which is a version of the Discrete Logarithm Problem (DLP) which is often used to prove IBS.

Both are briefly explained in the following.

I.) Preserving Randomness in $\mathbb{Z}_p^*$

Most modification to the schemes in the previous sections break down to additional additions or multiplications in $\mathbb{Z}_p^*$. Whenever this is required to stay random, the Latin square property is used to prove security:

Lemma 5.1 (Latin square property)**.** *Let $(G, \circ)$ be a group, $a \in G$, $|G| = k$.*
Then $\forall (b, c) \in \leftarrow G \times G : P(c = a \circ b) = \frac{1}{k}$.

Since the probability of obtaining $c \in G$ by randomly drawing an element from G is also $\frac{1}{k}$, this means that operating on two randomly drawn group elements returns a random group element. Specifically, for the case of $(G, \circ) = (\mathbb{Z}_p^*, \cdot)$:

Lemma 5.2 (Randomness preservation in $(\mathbb{Z}_p^*, \cdot)$)**.** *Randomly selecting $c \in \mathbb{Z}_p^*$ yields a certain element of G with the same probability as multiplying a given element $a \in \mathbb{Z}_p^*$ by a random $b \in \mathbb{Z}_p^*$.*

Proof. Let $p \in \mathbb{Z}$ be prime. $(\mathbb{Z}_p^*, \cdot)$ is a group with $p - 1$ elements. Let $a, b \in \mathbb{Z}_p^*$, where b is randomly selected.
$\forall x \in \mathbb{Z}_p^* : P(x = x' | x' \overset{r}{\leftarrow} \mathbb{Z}_p^*) = \frac{1}{p-1} = P(x = a \cdot b)$. □

II.) One-More Discrete Logarithm Problem

All transformed schemes rely on the DLP and the 1m-DLP as in Definition 5.2 is used for reduction of the transformations if direct reductions are impossible.

As in Definition 3.4, the DLP is the challenge on computing a given $Y = g^a$, with $Y \in \mathbb{G}$ and $g, a \in \mathcal{Z}$. According to [81], the definition of 1m-DLP and its prove allow efficient reduction to the DLP.

5.4.2 Proving 2KSS and U2KSS Token Security

Token security aims on showing that the introduction of the *gsk* does not harm the security of the underlying signature scheme. As all U2KSS transformation are constructed identical to 2KSS, they allow the same proofs.

Idea of the proofs: Let X denote any of the schemes in {Hess, BLMQ, GG, vBNN}. Suppose there exists an adversary $\mathcal{A}$ who can win the EUF-2KSS-CMA-game, i.e. forge a valid 2KSS-X-signature σ on message m^* and identity id^*, with non-negligible probability ε within time t without being in possession of id's corresponding private key. To prove Theorem 5.2-5.4 it suffices to show that $\mathcal{A}$'s existence would imply the existence of an adversary $\mathcal{B}$ who can win the EUF-IBS-CMA-game for X with non-negligible probability within time $t' = f(t)$ (where f is a polynomial function of t).

I.) Schnorr-Like Schemes: vBNN, GG

With the general idea of the proof, proving EUF-2KSS-CMA-security for GG and vBNN is all about finding a valid configuration of the EUF-2KSS-CMA-game that outputs a signature, useful for the EUF-IBS-CMA-game. As the signature creation in both schemes is $s = (x + h + u) \cdot \mathbf{g}$, the idea of both proofs is the same by providing $\mathcal{B}$ with a signature $\tilde{s} = \mathbf{g}^{-1} \cdot s$.

Theorem 5.1 (GG's EUF-2KSS-CMA-security)**.** *If GG is existentially unforgeable under adaptively chosen-message-and-identity attacks then GG's 2KSS transform is existentially unforgeable under adaptively chosen-message-and-identity attacks.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can win the EUF-2KSS-CMA-game for 2KSS-GG with non-negligible probability ε within time t, let $\mathcal{B}$ be an adversary playing the EUF-IBS-CMA-game, and let λ be fixed, $(msk, Mpk) \leftarrow \mathcal{G}(\lambda)$. Then $\mathcal{B}$ can call $\mathcal{A}(Mpk, \mathbf{g})$ (with $\mathbf{g} \overset{r}{\leftarrow} \mathbb{Z}_p^*$) to obtain a tuple (id^*, m^*, σ), where $\sigma = (s, R, X)$ such that

$$sP = \mathbf{g}(\,X + d(R + c\,Mpk)) \tag{5.44}$$

$\mathcal{B}$ obtains this with probability ε within time t. $\mathcal{B}$ outputs $(id^*, m^*, \widetilde{\sigma})$ where $\widetilde{\sigma} = (\widetilde{s}, R, X)$ and $\widetilde{s} = (\mathbf{g}^{-1} \cdot s)$ to the EUF-IBS-CMA game. The corresponding verification algorithm $\mathcal{V}(Mpk, \widetilde{\sigma}, m^*, id^*)$ will output 1 as

$$\widetilde{s}P = \mathbf{g}^{-1}sP = \mathbf{g}^{-1}\Big(\mathbf{g}(X + d(R + c\,Mpk))\Big) = X + d(R + c\,Mpk) \tag{5.45}$$

Therefore, $\widetilde{\sigma}$ is a valid GG-signature that $\mathcal{B}$ forged within time $t' \leq t + t_m$ with non-negligible probability ε, where t_m is the time to compute a multiplication in $\mathbb{Z}_p^*$. This contradicts Theorem 3 in [47], which states that the GG-scheme is EUF-CMA-secure. □

Theorem 5.2 (vBNN's EUF-2KSS-CMA-security)**.** *If vBNN is existentially unforgeable under adaptively chosen-message-and-identity attacks then vBNN's 2KSS transform is existentially unforgeable under adaptively chosen-message-and-identity attacks.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can win the EUF-2KSS-CMA-game for 2KSS-vBNN with non-negligible probability ε within time t, Let $\mathcal{B}$ be an adversary playing the EUF-IBS-CMA-game, and let λ be fixed, $(msk, Mpk) \leftarrow \mathcal{G}(\lambda)$. Then $\mathcal{B}$ can call $\mathcal{A}(Mpk, \mathbf{g})$ with $\mathbf{g} \overset{r}{\leftarrow} \mathbb{Z}_p^*$. $\mathcal{A}$ outputs a tuple (id^*, m^*, σ), where $\sigma = (s, R, h)$ such that

$$h = h_2\Big(id^*, m^*, R, X)\Big) \tag{5.46}$$

$\mathcal{B}$ calculates $\tilde{h}$ as:

$$\begin{aligned} c =& h_1(R, id^*) \\ \widetilde{X} =& (\mathbf{g}^{-1} \cdot s)\, P - h(R + c\, Mpk) \\ \tilde{h} =& h_2(id^*, m^*, R, \widetilde{X}) \end{aligned} \tag{5.47}$$

$\mathcal{B}$ outputs $(id^*, m^*, \widetilde{\sigma})$ where $\widetilde{\sigma} = (\widetilde{s}, R, \tilde{h})$ and $\widetilde{s} = (\mathbf{g}^{-1} \cdot s)$ to the EUF-IBS-CMA game. The verification algorithm $\mathcal{V}(Mpk, \widetilde{\sigma}, m^*, id^*)$ will output 1 as

$$\begin{aligned} \tilde{h} &= h_2(id^*, m^*, R, \widetilde{X}) & = \\ &= h_2(id^*, m^*, R, \widetilde{s}P + d(R + c\, Mpk) & = \\ &= h_2(id^*, m^*, R, (\mathbf{g}^{-1} \cdot s)P + d(R + c\, Mpk) & = \\ &= h_2(id^*, m^*, R, (\mathbf{g}^{-1} \cdot (x + h + u) \cdot \mathbf{g})P + d(R + c\, Mpk) & = \\ &= h_2(id^*, m^*, R, (x + h + u)P + d(R + c\, Mpk) & = \\ &= h_2(id^*, m^*, R, X) & \end{aligned} \tag{5.48}$$

Therefore, $\widetilde{\sigma}$ is a valid vBNN-signature that $\mathcal{B}$ forged within time $t' \leq t + t_m$ with non-negligible probability ε, where t_m is the time to compute a multiplication in $\mathbb{Z}_p^*$. This contradicts Theorem 1 in [24], which states that the vBNN-scheme is EUF-CMA-secure. □

II.) Pairing based Schemes: Hess and BLMQ

In the two schemes based on pairing, the hash produced while $\mathcal{A}$ plays the EUF-2KSS-CMA-game, can be manipulated such that $\mathcal{B}$ playing the EUF-IBS-CMA-game will win with non-negligible probability.

Theorem 5.3 (Hess' EUF-2KSS-CMA-security). *If Hess is existentially unforgeable under adaptively chosen-message-and-identity attacks then Hess's 2KSS transform is existentially unforgeable under adaptively chosen-message-and-identity attacks.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who wins the EUF-2KSS-CMA-game with non-negligible probability ε within time t. Let $\mathcal{B}$ be an adversary playing the EUF-IBS-CMA-game, and let λ be fixed, $(msk, Mpk) \leftarrow \mathcal{G}(\lambda)$. Then $\mathcal{B}$ can call $\mathcal{A}(Mpk, \mathbf{g})$ $(\mathbf{g} \xleftarrow{r} \mathbb{Z}_p^*)$ to obtain a tuple (id^*, m^*, σ), where $\sigma = (h, S)$ such that

$$h = h_2(m^*, e(S, P) \cdot e(H_1(id^*), -\mathbf{g}\, Mpk)^h) \tag{5.49}$$

$\mathcal{B}$ obtains this with probability ε within time t. $\mathcal{B}$ outputs $(id^*, m^*, \widetilde{\sigma})$ where $\widetilde{\sigma} = (\tilde{h}, S)$ and $\widetilde{h} = h \cdot \mathbf{g}$. The corresponding verification algorithm $\mathcal{V}(Mpk, \widetilde{\sigma}, m^*, id^*)$ which is

$$\begin{aligned} \widetilde{r} =& e(S, P) \cdot e(H_1(id^*), -Mpk)^h \\ h \overset{?}{=}& h_2(m^*, \widetilde{r}) \end{aligned} \tag{5.50}$$

returns 1 as

$$\begin{aligned} \tilde{r} =& e(S, P) \cdot e(H_1(id^*), -Mpk)^{\tilde{h}} & = \\ =& e(S, P) \cdot e(H_1(id^*), -Mpk)^{h \cdot \mathbf{g}} & = \\ =& e(H_1(id^*), P)^{h \cdot msk \cdot \mathbf{g}} \cdot e(Q, P)^x \cdot e(H_1(id^*), P)^{-msk \cdot h \cdot \mathbf{g}} & = \\ =& e(Q, P)^x & \end{aligned} \tag{5.51}$$

Therefore, $\widetilde{\sigma}$ is a valid forged Hess-signature that $\mathcal{B}$ forged within time $t' \leq t + t_m$ with non-negligible probability ε, where t_m is the time to compute a multiplication in $\mathbb{Z}_p^*$. This contradicts Theorem 3 in [67], which states that the Hess-scheme is EUF-CMA-secure. □

Theorem 5.4 (BLMQ's EUF-2KSS-CMA-security)**.** *If BLMQ is existentially unforgeable under adaptively chosen-message-and-identity attacks then BLMQ's 2KSS transform is existentially unforgeable under adaptively chosen-message-and-identity attacks.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can win the EUF-2KSS-CMA-game with non-negligible probability ε within time t, Let $\mathcal{B}$ be an adversary playing the EUF-IBS-CMA-game, and let λ be fixed, $(msk, Mpk) \leftarrow \mathcal{G}(\lambda)$. Then $\mathcal{B}$ can call $\mathcal{A}(Mpk, \mathbf{g})$ (with $\mathbf{g} \overset{r}{\leftarrow} \mathbb{Z}_p^*$) to obtain a tuple (id, m, σ), where $\sigma = (h, S)$ such that

$$h = h_2(m^*, e(S, h_1(id^*)P + Mpk) \cdot q^{-h}) \tag{5.52}$$

$\mathcal{B}$ obtains this with probability ε within time t. $\mathcal{B}$ outputs $\widetilde{\sigma} = (\tilde{h}, S)$ where $\tilde{h} = h + \mathbf{g}$ to the EUF-IBS-CMA game. Then the verification algorithm $\mathcal{V}(Mpk, \sigma^*, m^*, id^*)$ calculates

$$\widetilde{r} = e(S, h_1(id^*)\,Q + Mpk) \cdot q^{-\tilde{h}} \tag{5.53}$$

and returns 1 for $\tilde{h} \overset{?}{=} h_2(m, r)$, with $r = q^x$ as

$$\begin{aligned} \tilde{h} &= h_2(m^*, e(S, h_1(id^*)P + Mpk) \cdot q^{-(h+g)}) & \overset{\text{Eq. 5.43}}{=} \\ &= h_2(m^*, q^{(x+h+g)-(h+g)}) & = \\ &= h_2(m^*, q^x) \end{aligned} \tag{5.54}$$

Therefore, $\widetilde{\sigma}$ is a valid forged BLMQ-signature that $\mathcal{B}$ forged within time $t' \leq t + t_m$ with probability ε, where t_m is the time to compute a point multiplication in $\mathbb{G}$. This contradicts Theorem 1 in [9], claiming EUF-CMA-security for BLMQ. □

5.4.3 Proving KUSS Token Security

Compared to 2KSS and U2KSS, the KUSS transformation changes the signing and verification algorithms in all schemes, except Hess. Thus, the proof for KUSS-Hess is identical to 2KSS-Hess, while the other three schemes require a different approach for proving the security:

I.) Schnorr-Like Schemes: The Latin square property (see Lemma 5.1) is used to outline the prove for KUSS-vBNN and KUSS-GG.

II.) BLQM: KUSS-BLMQ's token security is proven by reduction.

I.) Schnorr-Like Schemes: vBNN, GG

It was mentioned before that the use of a hash function in vBNN and GG inhibits the inclusion of the *gsk* in R. Therefore, the *gsk* is still required for verification to calculate $\mathbf{g}\,R$ (see Table 5.1 and 5.2). Unfortunately, this fact inhibits direct reduction as used for the 2KSS and U2KSS transformations. A complete proof is necessary, which basically copies the original ones, with the modification of including $\mathbf{g}$ in a single equation. The interested reader is referred to the original proofs in [12, 47], here we only sketch the additional argument:

To build a KUSS-variant for $X \in \{\text{GG,vBNN}\}$ from the respective 2KSS-scheme, only the signature creation and verification are modified. Discussing the verification is omitted, since any computations the adversary could learn from could be done in the 2KSS-variant as well:

Instead of computing

$$\begin{aligned} &x \overset{r}{\leftarrow} \mathbb{Z}_p^* \\ &s = x \cdot \mathbf{g} + h \cdot u \cdot \mathbf{g} \end{aligned} \tag{5.55}$$

in X-2KSS, the signer in X-KUSS computes

$$\begin{aligned} &x \xleftarrow{r} \mathbb{Z}_p^* \\ &s = x + h \cdot u \cdot \mathbf{g} \end{aligned} \tag{5.56}$$

As x is a randomly drawn element of $\mathbb{Z}_p^*$, it can be concluded from the Latin square property that $x \cdot \mathbf{g}$ with $x, g \xleftarrow{r} \mathbb{Z}_p^*$ is equally random as $x \xleftarrow{r} \mathbb{Z}_p^*$ (see Lemma 5.2). The resulting s of Equation 5.56 is as random as s in Equation 5.55. Both IBS scheme's EUF-CMA security depend on this randomness: the proofs of Theorem 3 in [47] and Theorem 6.3 in [12] model the signing oracle $\mathcal{O}_{\mathcal{S}}$ as a random function. Hence, for the context of this work, it is fair to assume that the following conjecture holds:

Conjecture 5.1 (vBNN's and GG's EUF-KUSS-CMA-security)**.** *If vBNN and GG are existentially unforgeable under adaptively chosen-message-and-identity attacks, then vBNN's and GG's KUSS transform are existentially unforgeable under adaptively chosen-message-and-identity attacks.*

II.) BLMQ

Theorem 5.5 (BLMQ's EUF-KUSS-CMA-security)**.** *If BLMQ is existentially unforgeable under adaptively chosen-message-and-identity attacks then BLMQ's KUSS transform is existentially unforgeable under adaptively chosen-message-and-identity attacks.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who wins the EUF-KUSS-CMA-game for KUSS-BLMQ with non-negligible probability ε within time t. Let $\mathcal{B}$ be an adversary playing the EUF-IBS-CMA-game, and let λ be fixed, $(msk, Mpk) \leftarrow \mathcal{G}(\lambda)$. Then $\mathcal{B}$ can call $\mathcal{A}(\widetilde{Mpk}, \mathbf{g})$, with $\mathbf{g} \xleftarrow{r} \mathbb{Z}_p^*$ and $\widetilde{Mpk} = \mathbf{g}Mpk$ to obtain a tuple (id^*, m^*, σ), where $\sigma = (h, S)$ such that

$$h = h_2(m, e(S, h_1(id^*)\mathbf{g}P + \widetilde{Mpk}) \cdot q^{-h}) \tag{5.57}$$

$\mathcal{B}$ outputs $\widetilde{\sigma} = (h, \widetilde{S})$ where $\widetilde{S} = \mathbf{g}S$ to the EUF-IBS-CMA game. Then the verification algorithm $\mathcal{V}(Mpk, \sigma^*, m^*, id^*)$ calculates

$$\widetilde{r} = e(\widetilde{S}, h_1(id^*)\, Q + Mpk) \cdot q^{-\tilde{h}} \tag{5.58}$$

and returns 1 for $\tilde{h} \overset{?}{=} h_2(m^*, r)$, with $r = q^x$

$$\begin{aligned} h &= h_2(m^*, e(\widetilde{S}, h_1(id^*)P + Mpk) \cdot q^{-h}) && = \\ &= h_2(m^*, e(\mathbf{g}S, h_1(id^*)P + Mpk) \cdot q^{-h}) && = \\ &= h_2(m^*, e(\mathbf{g}(x+h)Usk, h_1(id^*)P + Mpk) \cdot q^{-h}) && = \\ &= h_2(m^*, e(\frac{\mathbf{g}(x+h)}{(msk + h_1(id^*)) \cdot \mathbf{g}}P, h_1(id^*)P + Mpk) \cdot q^{-h}) && = \\ &= h_2(m^*, e(\frac{x+h}{(msk + h_1(id^*))\cdot}P, h_1(id^*)P + Mpk) \cdot q^{-h}) && \overset{\text{Eq. 5.43}}{=} \\ &= h_2(m^*, q^{(x+h-h)}) && = \\ &= h_2(m^*, q^x) \end{aligned} \tag{5.59}$$

Therefore, $\widetilde{\sigma}$ is a valid forged BLMQ-signature that $\mathcal{B}$ forged within time $t' \leq t + t_m$ with probability ϵ, where t_m is the time to compute a point multiplication in $\mathbb{G}$. This contradicts Theorem 1 in [9], claiming EUF-CMA-security. □

5.4.4 Forward Security

2KSS does not feature updates and, hence, does not require to proving forward security, while it can be achieved in U2KSS and KUSS if and only if:

1. Entry of a new member triggers a call of *Next* and *Update*, i.e. a move to epoch $e+1$.

2. Upon entry, the new client receives $\mathbf{g}_{e+1}$ but not Δ_{e+1}.

The second condition provides forward security in the case of re-entry: If a device has no knowledge of previously used $\mathbf{g}_i$, there would be no harm in disclosing Δ. However, if it was a member two sessions before, knowledge of Δ_{e+1} would allow the computation of the previous $\mathbf{g}_e$. Consider the following scenario: Let

$$\begin{aligned} &g_0 \xleftarrow{r} \mathbb{Z}_p^*; \Delta_{1,2} \xleftarrow{r} \mathbb{Z}_p^* \\ &g_1 = g_0 \cdot \Delta_1, \\ &g_2 = g_0 \cdot \Delta_1 \cdot \Delta_2 \end{aligned} \tag{5.60}$$

be the shared secrets in rounds 0, 1, and 2, respectively. A client who was a group member in round 0, but not round 1, who re-joins the group in round 2, knows g_0 and g_2, but neither g_1, Δ_1 nor Δ_2. If it learned Δ_2, it could compute $\frac{g_2}{g_0 \cdot \Delta_2} = \Delta_1$ and learn $g_1 = g_0 \cdot \Delta_1$. According to Lemma 5.1, the probability of guessing such Δ_i is $\epsilon = {}^1\!/_{p-1}$. Handling a client's entry by updating $\mathbf{g}$ and not disclosing the current Δ to a new member (even a re-entering one) therefore provides forward security for U2KSS and KUSS.

5.4.5 Post-Compromise Security

Post-Compromise security can be achieved in KUSS if and only if *Next* and *Update* are triggered upon a client leaving the group in epoch e, i.e., who knows $\mathbf{g}_e$. This is efficiently done by drawing Δ_{e+1} and distributing it confidentially among the remaining group members (who can then calculate $\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$). According to Lemma 5.1, the probability of guessing $\mathbf{g}_{e+1} = \mathbf{g}_e \cdot \Delta_{e+1}$ (where $\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$) given $\mathbf{g}_e$ is ${}^1\!/_{p-1}$.

It remains to show that it is not possible to guess Δ_{e+1} from the *Mpk*, with more than negligible probability. The following is based on the assumption, that the 1m-DLP is hard, i.e.

> Let K_{cg} be a cyclic group generator. We say that the One-More Discrete Logarithm Problem associated with K_{cg} is hard if, for any polynomial-time adversary $\mathcal{A}$ the advantage
> $$\mathbf{Adv}^{\text{1m-dlp}}_{K_{cg},A}(k) = \Pr[\mathbf{Exp}^{\text{1m-dlp}}_{K_{cg},A}(k) = 1] \tag{5.61}$$
> is a negligible function of k. [12, Section 6.1]

Theorem 5.6 (KUSS' post-compromise security)**.** *If the One-More Discrete Logarithm Problem is hard then computing Δ_{e+1} from Mpk is hard.*

Proof. Let $\mathcal{A}$ have access to the Oracle $\mathcal{O}_{\mathcal{N}}$, which moves the global state to epoch $e+1$, and to $\mathcal{O}_{\mathcal{U}}(Mpk)$, which returns a $Mpk_{e+1} = \Delta_{e+1}\, Mpk_e$ (where $\Delta_{e+1} \xleftarrow{r} \mathbb{Z}_p^*$). In all schemes, $Mpk = msk\, P$, with $msk \xleftarrow{r} \mathbb{Z}_p^*$ and $P \in \mathbb{G}$ and, thus:

$$\begin{aligned} Mpk_{e+1} &= (msk \cdot \mathbf{g}_e \cdot \Delta_{e+1})\, P &&= \\ &= (msk_e \cdot \Delta_{e+1})\, P &&= \\ &= msk_{e+1}\, P \end{aligned} \tag{5.62}$$

According to Lemma 5.2, given msk_e the probability $P(k \xleftarrow{r} \mathbb{Z}_p^* | msk_{e+1} = msk_e \cdot k) = {}^1\!/_{p-1}$. The output of $\mathcal{O}_{\mathcal{U}}(Mpk)$ fits the definition of *random target points* in the `Chall` Oracle, where $Y \xleftarrow{r} \mathbb{G}$. Hence, the advantage for $\mathcal{A}$ to compute Δ_{e+1} from Mpk is bounded by the 1m-DLP:

$$\mathbf{Adv}^{\text{1m-dlog}}_{K_{cg},A}(k) = \Pr[\mathbf{Exp}^{\text{1m-dlog}}_{K_{cg},A}(k) = 1] \tag{5.63}$$

In turn, the 1m-DLP efficiently reduces to DLP [81]. □

A former member can therefore not correctly guess Δ_{e+1} and calculate a valid usk_{e+1} for the session with better than brute-force chances. This shows that all considered KUSS-schemes provide Post-Compromise Security. The same applies for all presented U2KSS and extracting $\mathbf{g}_e$ from a signature.

5.5 Practical Considerations

Before the next chapter is going to present a experimental setup to practically evaluate the usefulness of group Identity Based Signature (gIBS), this section discusses some aspects regarding practical security and performance.

5.5.1 Pseudo Randomness

In theory, all hash functions are modeled as random oracles and, thus, provide perfect randomness. With the use of the *Latin square property* (see Lemma 5.1), the signature of most schemes relies on this assumption. However, choosing random number in practice typically utilizes so-called Pseudo Random Function (PRF).

Due to the evolution of $\mathbf{g}$, any member of the group could silently listen and store all temporary Δ_{e+1} to gather information about the TTP's PRF. Thus, the PRF used for drawing Δ becomes less secure over time. Therefore, it needs to be re-seeded on a regular basis to preserve its security. With Δ only being drawn by the TTP, its PRF can be re-seeded without any effect on the participating group members. It should also be different from the one used to draw random elements that remain secret such as the msk.

Every well-known PRF defines a maximum number of values to be extracted to keep the security properties intact. When this value is reached, a re-keying of $\mathbf{g}$ is required and the seed of Δ has to be renewed.

5.5.2 Signature Replay

Replay-attacks are a well-known problem in network security. One feature of gIBS is that such attacks can be prevented by regularly re-keying the shared secret without the necessity of another anti-replay-mechanism on the protocol level. However, one of the transformed schemes does not feature this:

Let a client be expelled in epoch $e-1$ and re-join in epoch $e+1$. In KUSS-BLMQ, a signature from epoch e has the form $\sigma_e = (h, S_e)$, with $S_e = (x+h)Usk_e$, which is $S_e = (x+h)\mathbf{g}_e Usk$, h being the hash of the message and $x \xleftarrow{r} \mathbb{Z}_p^*$. As the client does not know $\mathbf{g}_e$, the signature looks perfectly random; however, she knows $S_{e-1} = (x+h) \cdot \mathbf{g}_{e-1} Usk$ and $S_{e+1} = (x+h) \cdot \mathbf{g}_{e+1} Usk$. If and only if she knows $\mathbf{g}_{e+1}$ and $\mathbf{g}_{e-1}$, she can calculate $S^*_{e-1} = \frac{1}{\mathbf{g}_{e-1}} S_{e-1}$ and $S^*_{e+1} = \frac{1}{\mathbf{g}_{e+1}} S_{e+1}$. If $S^*_{e-1} = S^*_{e+1}$, she knows that the signature was generated with the same original Usk (without knowing Usk, notably).

Practically, this allows a replay-attack of messages signed with previously used $\mathbf{g}$'s of other users in the system. Let an attacker know the shared secret $\mathbf{g}_e$ of a recorded signature σ_e in some epoch $e < e^*$ and have access to the current $\mathbf{g}_{e^*}$. She can calculate a valid signature

$\sigma_{e^*} = \frac{1}{\mathbf{g}_e}\sigma_e \cdot \mathbf{g}_{e^*}$. She can neither change the message nor the identity, but can evolve the signature to a new epoch. However, such a replay attack can be easily mitigated with state-of-the-art mechanisms, such as sequence numbers or time stamps as part of the message. Note that such a replay-attack is possible in all original schemes and for their 2KSS- and U2KSS-variants. Thus, the KUSS-modification prevents this attack for all schemes except BLMQ.

Hess, GG and vBNN

In Hess, GG, and vBNN, the $usk := \mathbf{g} \cdot usk$ is integrated in the signature as part of a sum with a random element $x \overset{r}{\leftarrow} \mathbb{Z}_p^*$:

In KUSS-Hess: $\sigma = (h, S)$, with $S = h\, Usk + xQ$ and $Q \overset{r}{\leftarrow} \mathbb{G}^*$.

In KUSS-{GG, vBNN}: $\sigma = (h, s, X)$, with $s = x + h \cdot u$ and $X = x\,P$

This makes it impossible to purposefully alter σ while retaining a valid usage of usk without knowing x. In KUSS-{GG, vBNN}, computing $x = \frac{X}{P}$ (which is hard according to Definition 3.7) would be required. In KUSS-Hess, the pairing used in the verification phase to derive $(Q, x) \leftarrow e(Q, P)^x$ would need to be exploited, which requires solving the DLP.

5.5.3 Performance Estimations

This section presents a theoretical analysis of the costs introduced by the transformation in comparison to the original schemes. The comparison highlights the following operations, ordered by complexity:

GE Group Exponentiation in $\mathbb{G}$, typically donated as $a\,P$.

GO Group Operation in $\mathbb{G}$, e.g., $Q + P \in \mathbb{G}$ or $Q - P \in \mathbb{G}$

RG Randomly drawing an element of $\mathbb{G}$, denoted as $P \overset{r}{\leftarrow} \mathbb{G}$

HG Hashing in $\mathbb{G}$

RZ Random number generation in $\mathbb{Z}_p^*$, denoted as $a \overset{r}{\leftarrow} \mathbb{Z}_p^*$

HZ Hashing in $\mathbb{Z}_p^*$

ME Modular Exponentiation in $\mathbb{Z}_p^*$, e.g., $a^b \in \mathbb{Z}_p^*$

MM Modular Multiplication in $\mathbb{Z}_p^*$, e.g., $a \cdot b \in \mathbb{Z}_p^*$

Some of the discussed schemes are based on pairing, which is an expensive operation in ECC. The comparison assumes that performing a pairing is equivalent to 21 GE as stated in [9] and used for evaluation of different IBS schemes in [52, 54].

The major advantage of U2KSS and KUSS schemes over their originating schemes is the enhanced re-keying that allows efficient revocation. During integration of the shared secret to the selected schemes, special attention was put on keeping the computational overhead as small as possible. Table 5.5 compares the transformations developed throughout this chapter with the originating schemes. The columns depict the originally defined IBS phases *Setup*, *Extract*, *Sign* and *Verify* and additionally the new phases for *Re-Keying* on server and Group Member (GM). The server is a combination of the KGC and Key Update Center (KUC), running the *Next* and *Update* algorithms, as presented in Table 5.1 - 5.4. All four transformed schemes are presented together with their respective transformation to 2KSS, U2KSS and KUSS. The first row of each scheme depicts the costs for the original scheme, while the rows below shows the overhead (+) introduced by the transformations for *Setup*, *Extract*, *Sign* and *Verify*.

Re-Key is the phase with major changes, hence all cells present the absolute costs. The original schemes do not define a *Re-key* mechanism, why it assumes calling *Setup* to generate new master keys and *Extract* for the remaining n GMs [52]. Hence, it is no surprise that this step benefits most of the U2KSS and KUSS transforms, where the complexity is reduced from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ (highlighted in dark green in Table 5.5).

The downside is the introduction of new keys to be generated, stored and updated, as well as the *Re-Key* step on the GM. However, the table shows that none of the modifications should significantly harm the performance of the schemes. Most of them introduce an additional random generation or multiplications in $\mathbb{Z}_p^*$ that are cheap (highlighted in yellow in Table 5.5). As KUSS defines updating the keys – which is an element of $\mathbb{G}$ in all schemes –, the re-key phase on the clients is affected with additional **GE** (highlighted in red), but with slight benefits during the signing process compared to U2KSS (highlighted in light green). During signature verification all schemes except for Hess require some calculations with the shared secret, but no more than a single **MM**. In the best case of Hess, the algorithms of signing and verifying in KUSS show no difference to the originating scheme.

5.6 Summary and Findings

This chapter presented the application of the transformations defined in Chapter 4. Therefore, this chapter delivers an answer for the following research question:

> RQ 6: How can IBS keys be revoked and how can the revocation be achieved in state-of-the-art key distribution systems?

The revocation is shown to work with IBS schemes based on the Elliptic Curve Discrete Logarithm Problem which is widely considered to be efficient for the use cases presented in Chapter 2. It was shown in theory, that none of the transformations will significantly influence the computation for neither signing nor verification. Hence, ER 3 holds in theory, but will be further evaluated in Section 7.4.3. With the detailed security analysis, it is also shown that the optimizations do not harm the security of the schemes.

Table 5.5: Theoretical overhead/enhancement for the different phases.

	Setup	**Extract**	**Re-Key (Server)**	**Re-Key (GM)**	**Sign**	**Verify**
vBNN	$1GE + 1RZ$	$1GE{+}1RZ{+}$ $1HZ + 2MM$	$1GE + 1RZ+$ $n(1GE{+}1RZ{+}1HZ{+}2MM)$	/	$1GE + 1RZ+$ $1HZ + 2MM$	$3GE + 2GO+$ $1HZ + 1MM$
2KSS-vBNN	$+1RZ$	/	$1RZ$	/	$+1MM$	$+1MM$
U2KSS-vBNN	$+1RZ$	/	$1RZ + 1MM$	$1MM$	$+1MM$	$+1MM$
KUSS-vBNN	$+1RZ + 1MM$	$+1MM$	$1GE + 1RZ + 1MM$	$1GE + 2MM$	/	$+1MM$
GG	$1GE + 1RZ$	$1GE{+}1RZ{+}$ $1HZ + 2MM$	$1GE + 1RZ+$ $n(1GE{+}1RZ{+}1HZ{+}2MM)$	/	$1GE + 1RZ+$ $1HZ + 2MM$	$3GE + 2GO+$ $1HZ + 1MM$
2KSS-GG	$+1RZ$	/	$1RZ$	/	$+1MM$	$+1MM$
U2KSS-GG	$+1RZ$	/	$1RZ + 1MM$	$1MM$	$+1MM$	$+1MM$
KUSS-GG	$+1RZ + 1MM$	$+1MM$	$1GE + 1RZ + 1MM$	$1GE + 2MM$	/	$+1MM$
Hess	$1GE + 1RZ$	$1GE + 1HG$	$1GE + 1RZ+$ $n(1GE + 1HG)$	/	$23GE + 1GO+$ $1RG{+}1HZ{+}1MM$	$43GE + 1HG+$ $1HZ{+}1ME{+}1MM$
2KSS-Hess	$+1RZ$	/	$1RZ$	/	$+1MM$	$+1MM$
U2KSS-Hess	$+1RZ$	/	$1RZ + 1MM$	$1MM$	$+1MM$	$+1MM$
KUSS-Hess	$+1RZ + 1MM$	$1MM$	$1GE + 1RZ + 1MM$	$2GE$	/	/
BLMQ	$22GE{+}1RG{+}1RZ$	$1GE + 2MM$	$1GE + 1RG + 1RZ+$ $n(1GE + 2MM)$	/	$1GE + 1RZ+$ $1HZ{+}1ME{+}3MM$	$22GE + 1GO+$ $1HZ{+}1ME{+}1MM$
2KSS-BLMQ	$+1RZ$	/	$1RZ$	/	$+1MM$	$+1MM$
U2KSS-BLMQ	$+1RZ$	/	$1RZ + 1MM$	$1MM$	$+1MM$	$+1MM$
KUSS-BLMQ	$+1RZ + 1MM$	$+1MM$	$1GE + 1RZ + 1MM$	$2GE + 2MM$	/	$+1MM$

6 Testbed and Prototypes

This chapter describes the implementation of a testbed, picturing all case studies presented in Chapter 2 despite their different characteristics. All use cases showed dynamic membership behavior of nodes with different capabilities. Hence, the testbed's architecture allows heterogeneous nodes with different constraints and capabilities to join or leave frequently. The testbed enables the use of the *Group IBS Architecture* presented in Section 4.5 and hence implementing and testing group Identity Based Signature (gIBS) for all use cases in question. This includes implementation of a Group Key Management Protocol (GKMP) and the two revocation mechanisms presented in Section 4.3, which are not available for constrained devices. Thus, this chapter provides a proof-of-concept of the architecture and the transformations of all four Identity Based Signature (IBS) schemes in practice.

6.1 Concept

While many testbeds found in academia and industry focus on the evaluation of hardware components, the purpose of this work is interoperability and scalability of security solutions. The basis of the testbed's design is IP multicast, which is used to picture dynamic communication groups.

Figure 6.1 shows the concept for interoperability between different networking technologies. Devices with different network interface are connected and managed by a *Group Manager* as the central component of the testbed. Networks are depicted at the bottom of the figure and are cable based (e.g., Ethernet, second right) and wireless such as for home networking (e.g., WiFi, left) as well as low and long range networks (e.g., IEEE 802.15.4, second left and LoRA on the right respectively). By using IP multicast, communication groups spanning different network interfaces can be formed (presented by *Group A,B,D* in Figure 6.1) or using only one technology (presented by *Group C* in Figure 6.1). The use of powerful resources such as cloud-services is supported as well, presented in form of the *Group Manager*.

The *Group Manager* can be used as the server component of the *Group IBS Architecture* or any chosen GKMP. Hence, the concept allows not only evaluation of gIBS but different security solutions for similar environments.

6.2 A Testbed for Researching Secure Group Communication

The goal of the testbed is to allow evaluation of communication groups with a variety of microcontrollers and networking technologies. A partially open source testbed provided by the Future Internet Testing Facility (FIT) consortium [1, W6] is found as a good basis for implementing the presented concept [105]. On contrast to the FIT IoT-Lab [W6] that uses customized hardware, this setup consists of commodity hardware. With the focus on multicast, routing between different networking technologies is a major interest.

Figure 6.2 represents this with a central switch connecting all of the following components:

Open Node is the (constrained) device being programmed by the user to run an experiment.

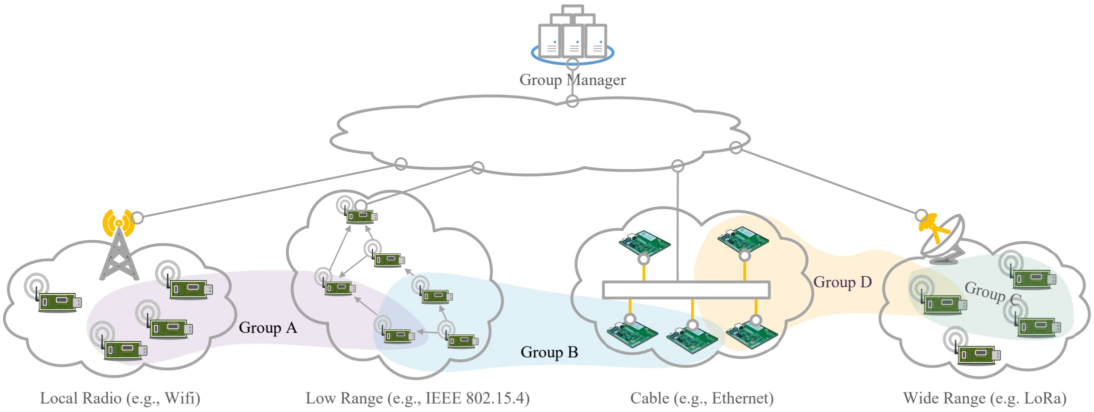


Figure 6.1: High-Level concept of the testbed.

Figure 6.2 depicts them connected to the Control Node via a serial port (UART) and with their respective networking technology.

Control Node is a small single-chip computer which supplies the Open Node with a power source and is used to program it with firmware uploaded by the user. If available, it also forwards the serial port of the Open Node such that the user can interact during the experiment. It is accessible through a REST API by the Site Server. Figure 6.2 shows eight of them arranged around and connected the central switch via Ethernet.

Gateway is one single chip computer for each network technology that serves as a router for the Open Nodes to the outside world via Ethernet. In case of wireless technologies with a star topology (e.g., LoRa, WiFi in Station Mode), it also serves as a wireless access point. Figure 6.2 displays two of them at the right by stating the network technology, e.g. LoRa or ZigBee Gateway and the logo of the respective technology.

Management Node is connected to the Switch and manages the Control Nodes and Gateways by providing software updates and also as the router for interconnection of the different gateways.

Site Server provides access to the system via a web or command line interface, shown in the left of Figure 6.2. It uses the REST API provided by the Control Node to start and stop experiments on the Open Node and serves as the system's router to the Munich Scientific Network (MWN).

The following presents hard- and software used for the different components in the testbed. As it is meant to be open source, the available software and operating systems limit the choice of hardware for computing and networking.

1. Different open source operating systems for microcontrollers are discussed and one is picked for the testbed.
2. The chosen hardware is presented, spanning over different classes of constraints and supported by the operating system. This includes the development boards used for the open nodes, the different networking technologies and the hardware for the Control Nodes and Gateways.
3. The final setup is presented including the composition of microcontrollers with network technologies and their installation in a network rack.

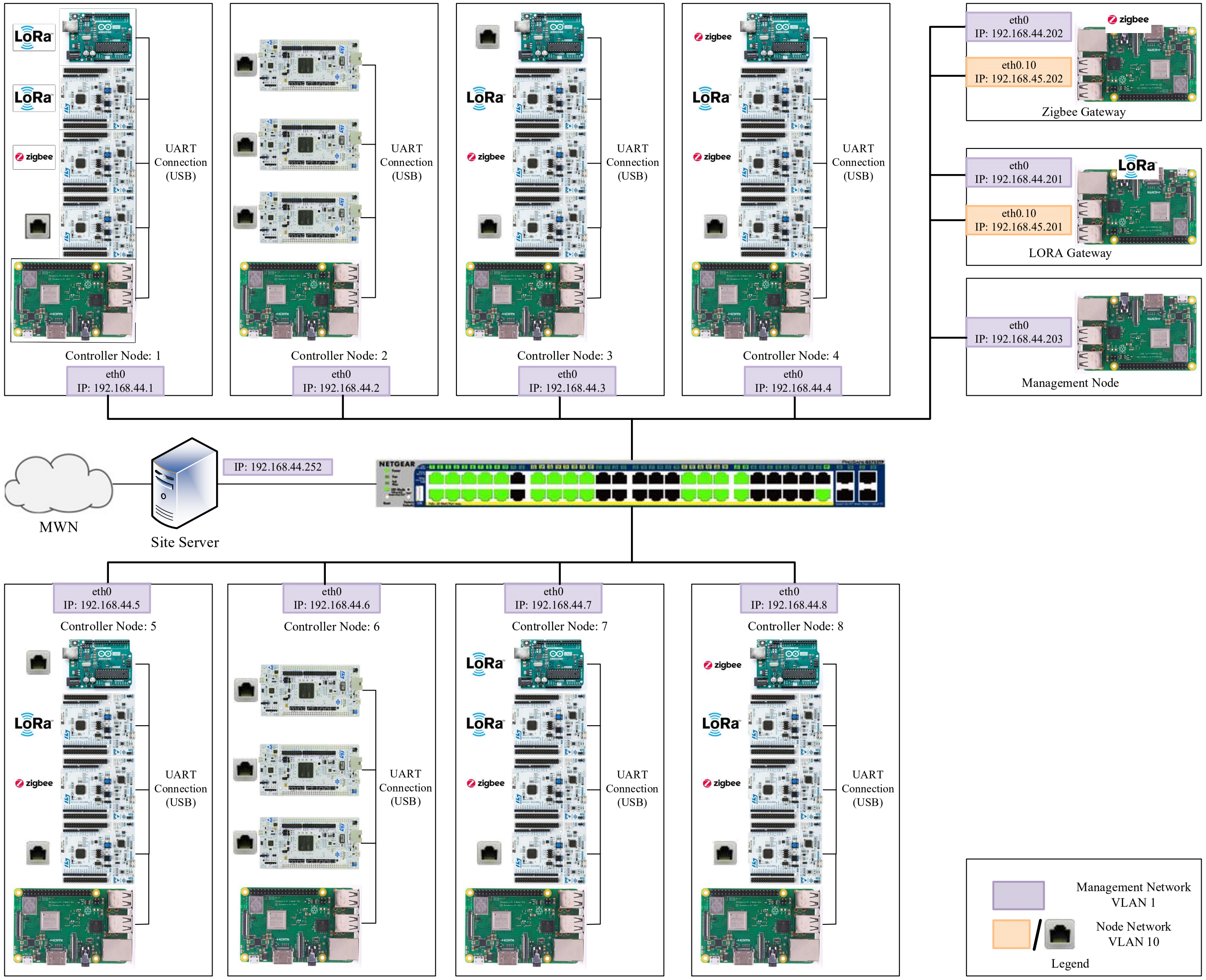


Figure 6.2: Physical setup of the Testlab, supporting IEEE 802.15.4, Ethernet and LoRA networks [105].

Table 6.1: Comparison of different operating systems usable for the Testbed [6].

OS	Min RAM	Min ROM	C Support	C++ Support	Multi-Threading	Modularity	Real-Time
Contiki	<2 KB	<30 KB	•	×	•	•	•
TinyOS	<1 KB	<4 KB	×	×	•	×	×
RIOT OS	~1.5 KB	~5 KB	✓	✓	✓	✓	✓
Linux	~1 MB	~1 MB	✓	✓	✓	•	•

✓ Full Support • Partial Support × No Support

6.2.1 Operating Systems

The major producers of microcontrollers in the considered classes of hardware are ARM®, Arduino® and Texas Instruments® which all offer software that allows programming their own products. Out of a multitude of academic and industrial projects [50, 62], three open source projects reached a considerable large user basis and offering a broad support for different hardware and networking technologies. From top to bottom, Table 6.1 shows a comparison of the three IoT operating systems *Contiki*, *TinyOS* and *RIOT OS* with *Linux*. It compares the required memory for running the operating system, the supported programming languages and functionalities, like multi-threading, modularity and real-time support. Besides allowing Multi-Threading and offering Real-Time capabilities, RIOT OS features the best support for a broad range of hardware of various architectures [5, 6, W7]. At the time of writing, RIOT OS is available for >100 microcontrollers various networking modules featuring different technologies (most notable being IEEE 802.15.4, Bluetooth Low Energy (BLE), LoRA, WiFi and Ethernet). Additionally, RIOT OS supports a wide range of external software modules, including several cryptographic libraries which is of special interest for this work. Hence, while not being restricted to one particular operating system, the hardware in the testbed is specifically selected to be supported by RIOT OS.

6.2.2 Hardware

We distinguish the hardware used in the testbed regarding their use as **I.)** Open Nodes, **II.)** Networking Modules and **III.)** Gateway and Controller Nodes

I.) Open Nodes

As the major producer of microprocessors, most chosen microcontrollers feature ARM® processor which are specialized for different use cases. Picturing a broad range of potential scenarios we focus on the following

ARM Cortex M0 32-bit processor with low power and minimal code footprint.

ARM Cortex M0+ 32-bit processor with focus on energy efficiency.

ARM Cortex M3 32-bit processor for highly deterministic real-time applications.

ARM Cortex M4 32-bit processor for digital signal control markets, such as automotive or motor control.

ARM Cortex M7 32-bit processor featuring high performance, e.g., for automotive or medical applications.

Table 6.2: Chosen development boards in the testbed.

Development Board	**Architecture**	**Clock Speed**	**Flash Memory**	**SRAM**	**On-Board Networking**
Arduino M0+	ARM Cortex-M0+	48 MHz	256 KB	32 KB	-
ST Nucleo-F091RC	ARM Cortex-M0	48 MHz	256 KB	32 KB	-
ST Nucleo-F103RB	ARM Cortex M3	72 MHz	128 KB	20 KB	-
ST Nucleo-F411	ARM Cortex-M4	84 MHz	512 KB	96 KB	-
ST Nucleo-767ZI	ARM Cortex-M7	216 MHz	2 B	512 KB	Ethernet
TI CC3200 Launchpad	ARM Cortex-M4	80 MHz	512 kB	256 KB	Wifi
Raspbery Pi 3 Model B+	ARM Cortex-a53	1.2 GHz	Micro-SD	1 GB	WiFi, BLE, Ethernet

II.) Networking Modules

The majorly used networking technology in the area of IoT is IEEE 802.15.4, however the goal of this testbed is to include a variety of other modules. With RIOT OS' support of IEEE 802.15.4, Ethernet and LoRA, the following networking modules were chosen:

- Semtech SX1272 868/915 MHZ Lora MBED SHIELD
- Texas Instruments SimpleLinkWi-Fi CC3200
- Microchip AT86RF233 Zigbee / 802.15.4[1]
- WIZnet Ethernet-Module W5100
- STM32 Ethernet Module

III.) Gateways and Controller Nodes

For the Gateways and Controller Nodes, the Raspberry Pi version 3 was chosen as it features a broad open source community. It supports the used networking modules for LoRa and IEEE 802.15.4 and comes with on board WiFi, Bluetooth and Ethernet modules, thus, featuring all networking technologies in questions.

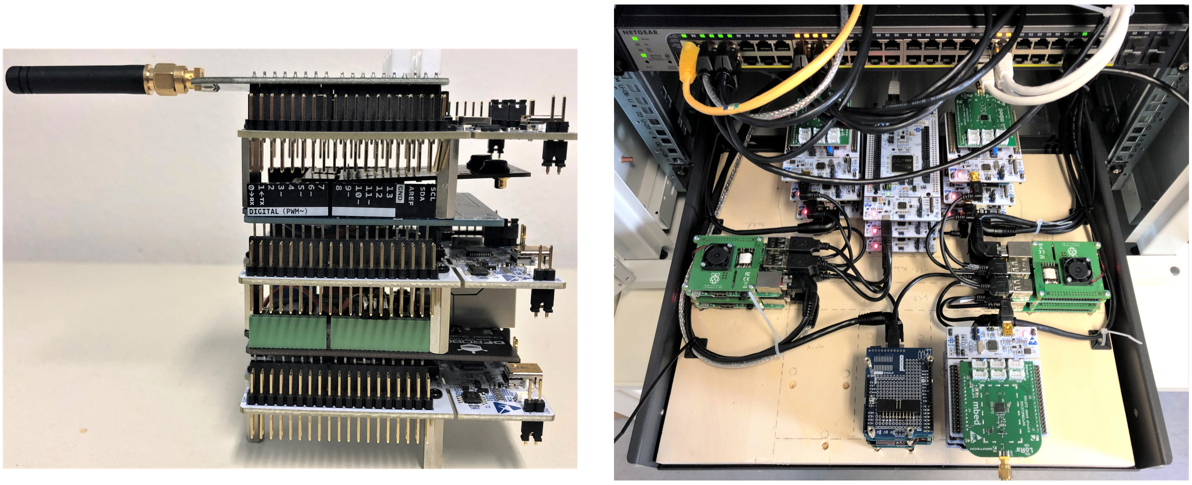
6.2.3 Final Setup and Access

Table 6.2 shows the chosen development boards for the testbed and presents their capabilities regarding CPU, memory and on-board networking. While two of the boards already include a networking module – namely TI CC3200 with WiFi[2] and ST Nucleo-767ZI with Ethernet –, the others are each assembled with one of the networking modules for LoRa, IEEE 802.15.4 and Ethernet, respectively.

Figure 6.3 pictures the physical setup of the testbed. Three open nodes are assembled together as so-called towers, consisting of one type of development board with three different network technologies. Figure 6.3a exemplarily shows three Nucleo F091 RC with an Ethernet, Zigbee and LoRa module, respectively (from the bottom to the top in the picture). Notably, each of these towers is represented twice in the testbed as presented in the topology in Figure 6.2. It shows how five of these towers are assembled in a networking rack together with the switch as shown in the top of Figure 6.3b. The picture also shows how the Raspberry Pi 3 at the left and right are connected to the Open Nodes using USB. As each of the Raspberry Pi 3 has four USB ports, it manages up to four Open Nodes. The Open Nodes are powered through the serial connection with the Raspberry Pis, which in turn are connected to a switch featuring Power

[1] Please note, that Zigbee is a trademark following the IEEE 802.15.4 specification of Layer 2 and are therefore seen as equivalent in the context of this work.

[2] RIOT OS' support for TI CC3200 is currently under development [89, W8].



(a) Three Open Nodes are assembled as a tower.

(b) All Nodes of the Testbed including the switch are assembled in a rack.

Figure 6.3: The installation of the testbed.

over Ethernet. Hence, the whole testbed is almost autarkic, requiring only an external power and network plug.

The Raspberry Pis also serve as the Gateways for LoRa and IEEE 802.15.4, while – as soon as available – their on board WiFi module will be used to also serve as the gateway for the TI CC3200s. The Site Server is a virtual machine being available within the MWN, providing a web-interface and command line interface for flashing and accessing the nodes. It is hence available for a multitude of students and researchers.

6.3 Prototypical Implementation

The system architecture presented in Chapter 4 allows integrating the four gIBS schemes developed in Chapter 5. It is specifically designed to use existing protocols which are meant to be implementable on devices with limited memory and computing capabilities found in the testbed. The implementation of the system architecture requires four steps:

1. **Implementation of a GKMP**: As the testbed uses IP multicast, we chose Group Internet Key Exchange (G-IKEv2) as the GKMP for IP security protocol (IPsec) as the natural choice for securing IP traffic. G-IKEv2 is currently under standardization and, thus, not part of common Internet Key Exchange (IKEv2) implementations. Hence, it is implemented for RIOT OS as a client and integrated to Strongswan [W9] as a server.

2. **Implementation of symmetric re-keying**: In Chapter 4, Logical Key Hierarchy (LKH) and Centralized Authorized Key Extension (CAKE) were presented as efficient re-keying mechanisms for symmetric keys, which are additionally integrable to G-IKEv2. Both mechanisms are integrated into client and server implementations.

3. **Implementation of IBS**: All four IBS schemes which are transformed to Key Updatable Signature Scheme (KUSS) in Chapter 5 are implemented in RIOT OS.

4. **Implementation of gIBS**. The IBS implementations are extended as described in Chapter 5 to support re-keying.

The columns of Table 6.3 show the required memory for RIOT OS plus G-IKEv2, LKH, CAKE and the two Schnorr- and Pairing-IBS schemes respectively. Summing up the first six rows

Table 6.3: Static memory consumption of the G-IKEv2 client in RIOT OS [53], featuring LKH and CAKE [59] with the example of 100 group members as well as IBS schemes based on Schnorr and Pairings [52, 90].

Feature	**G-IKEv2**	**LKH**	**CAKE**	**Schnorr-IBS**	**Pairing-IBS**
RIOT kernel (incl. stack)	2,560 Byte	2,560 Byte	2,560 Byte	2,560 Byte	2,560 Byte
RIOT IPv6 stack	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte
RIOT UDP stack	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte
RIOT net cache	928 Byte	928 Byte	928 Byte	928 Byte	928 Byte
RIOT 6LoWPAN cache	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte	1,024 Byte
RIOT packet buffer	1,280 Byte	1,280 Byte	1,280 Byte	1,280 Byte	1,280 Byte
$\sum$ RIOT	7,840 Byte	7,840 Byte	7,840 Byte	7,840 Byte	7,840 Byte
mini-gmp lib	-	-	640 Byte	-	-
relic-toolkit	-	-	-	5,060 Byte	10,772 Byte
IKE SA	~ 210 Byte				
SAD for 1 group membership	~ 100 Byte	~ 212 Byte	~ 234 Byte		
SPD for 1 group membership	40 Byte				
$\sum$	8,190 Byte	8,052 Byte	8,714 Byte	12,900 Byte	18,612 Byte

results in an overhead of ~8 KB for the operating system – which mostly consists of networking buffers and the kernel – and stays the same for all implementations. The bottom rows are for cryptographic libraries, independent of the used elliptic curve. While G-IKEv2 and LKH do not require additional libraries, CAKE requires an inexpensive (640 Byte) multiple precision library. For IBS a more feature-rich cryptographic library is required and represented as relic [W10]. Pairings require additional features to be compiled with relic, a fact that results in additional ~5 KB of memory compared to Schnorr-IBS schemes. The Security Association (SA) are statically assigned memory buffers, to store the keys.

6.3.1 Group Key Management with G-IKEv2

IKEv2 is the protocol for exchanging keys for the use in IPsec and therefore a natural choice when talking about securing IP multicast. As the testbed pictures group communication in form of multicast, it was chosen to implement G-IKEv2 which is already defined for the use in constrained environments [53, 111, 139]. Additionally, only very few other proposals and even less implementations of GKMPs exist [112, 168]. Within the context of this thesis, a server and client implementation of G-IKEv2 have been developed.

I.) G-IKEv2 client implementation

The client is designed as a minimal subset of the rather complex G-IKEv2 [174] protocol and follows many of the decision made for IKEv2 within RFC 7815 [139]. Any payloads not being mandatory are ignored by the implementation. Additionally, the user chooses a minimal set of supported cryptographic algorithms and installs them on the hardware, making the initial exchanges of G-IKEv2 static. The Group Controller Key Server (GCKS) sends the cryptographic material and metadata to the client in so-called *proposals*. If they are not acceptable for the client, it can simply ignore the responses which in turn implicitly refuses the connection. This allows minimal memory footprint of the application (see Table 6.3) and reduced networking overhead. However, it should be noted that in contrast to IKEv2 where both peers agree on an SA, the client downloads the Group Security Association (GSA) within the `GSA_AUTH` message from the server and has therefore no influence on its content.

As a first step, only the initialization and registration phase of G-IKEv2 have been implemented for RIOT OS [66]. The evaluation presented in [53] allows confidence that the protocol is suitable for the considered scenarios. It uses microcontrollers with ARM M0+ and ARM M4 microprocessors, showing computational overhead for message processing of only few microseconds. Only the procession of the so-called `IKE_SA_INIT` message requires considerable overhead as it includes the computation of the Diffie-Hellman exchange and charges with $190ms$ and $420ms$ on the two tested processors [53]. Notably, similar finding were presented independently in [111].

II.) G-IKEv2 server implementation

The only available server implementation for the evaluation in [53] was part of the Cisco® IOS [W11], which is not designed for the considered scenarios and, hence, producing quite overwhelming networking overhead. Thus, a server implementation based on the open source IKEv2 application *Strongswan* [W9] was developed as well [42]. With this implementation, the server can be any device which is capable of a Linux operating system. In the testbed, these are the gateway nodes with an ARM Cortex-A8 processor.

6.3.2 Key Distribution with LKH and CAKE

The mechanism for distribution and revocation of LKH-keys within G-IKEv2 is proposed for standardization in [174] and allows efficient packet parsing but lacks efficiency in terms of networking overhead. Its description requires elements of the key hierarchy to be sent multiple times, which increases the networking overhead. Hence, an efficient implementation of LKH in RIOT OS is presented together with CAKE in [43, 59]. Compared to G-IKEv2, both implementation show increased storage requirements for the SA, as the key hierarchies need to be represented. CAKE offers minimized networking overhead compared to LKH, but with the costs of more keys in the security association. Additionally, CAKE requires a multi-precision library to solve the *Secure Lock*. Such a library was found with *mini-gmp* and requires additional 640 Byte of memory. Table 6.3 shows that none of the extension have a significant impact on the required memory and the application can be run on any device with a minimum of 8 KB of main memory. This requirement is met by all devices used in the testbed.

A minimal server implementation with the CAKE extension was developed as well. It could be shown that a device featuring an ARM M3 processor with 64 KB of RAM is able to play the role of the GCKS with a group of 14 clients [59].

6.3.3 IBS for Sender Authenticity

Most implementations and applications of IBS are found as integration to protocols [87, 88] or proof-of-concepts in cryptographic libraries. Although useful for evaluating the use of IBS in protocols, it stays difficult to compare schemes outside the boundaries of such protocols. Hence, a test suite based on RIOT OS and the cryptographic library Relic [W10] was developed in [90]. It allows the evaluation of the use of IBS in a multicast environment with different elliptic curves representations and primitives (e.g., Pairings or Schnorr signatures).

An abstraction layer for IBS is defined by five types and five methods which have to be implemented to evaluate an IBS schemes. The following structures are meant to hold the scheme's specific parameters and are serialized to allow communicating them over a network:

`ibs_fixed_params` includes fixed parameters, which are generated during the *Setup* phase of IBS and do not change over time (e.g., the elliptic curve generator P).

`ibs_master_params` include the parameters which are hold by the Trusted Third Party (TTP). Typically, that is the Master Secret Key (*msk*) and Master Public Key (*mpk*).

`ibs_public_master_params` include the parameters which are publicly shared with the participants but can change over time. Typically, that is the *mpk*.

ibs_user_secret includes the parameters of the User Secret Key (*usk*).
ibs_signature includes the parameters of the signature.

The types are used within the following five generic functions which represent the IBS algorithms as in Definition 4.1:

ibs_reset_master_params Takes the fixed parameters and runs the *Setup* algorithm of the TTP. Typically, this means drawing a new *msk* and calculating a new *mpk*.
ibs_extract_public_master_params Extracts the public parameters from the master parameters, which are meant to be distributed among the participants.
ibs_extract_user_secret Takes the fixed and master parameters together with the identity as input, runs the *Extract* algorithm of the TTP and returns `ibs_user_secret`.
ibs_sign_message Takes the fixed parameters, a user secret and a message string as input, runs the *Sign* algorithm and outputs a `ibs_signature`.
ibs_verify_message Takes the fixed and public master parameters as an input to the *Verify* algorithm for a given `ibs_signature` and message.

The major purpose is the evaluation of the IBS performance, which is why a minimized protocol without additional security properties for the communication between client and TTP is developed. As for the implementation of the G-IKEv2 server, the TTP is an ARM Cortex-A8 processor and distributes the keys upon request to clients. The clients can send messages via IEEE 802.15.4, signed with an IBS signature via multicast to the group. The IBS inherent feature of key revocation by re-calculating the *msk* and all *usk*s is also supported. The main difference to the memory requirements in Table 6.3 is the cryptographic functionality provided by relic, requiring 5 KB for Schnorr based and 10 KB for paring based schemes. The application allows measuring the sizes of the cryptographic material as well as performance evaluation of the algorithms for signing, verifying. The results are presented in [52]. All four schemes that are transformed Chapter 5 and four different elliptic curves are supported. An implementation of Elliptic Curve Digital Signature Algorithm (ECDSA), which is provided in the Relic Library, complements the setup.

6.3.4 The gIBS Prototype

The new phases for the KUSS transforms *Next* and *Update* are added to the IBS implementation. To compare gIBS schemes with classical IBS schemes, changes of the implementation are applied to them as well. First, this requires the master parameters to be extended with the group shared key (*gsk*) and – depending on the scheme – the extension of the master public parameters. Additionally, a new type is required which represents the update token:

ibs_rekey_params includes the update token of a gIBS scheme. For classic IBS schemes, this parameter holds all values of the `ibs_user_secret` and `ibs_public_master_params`

Next, the new algorithms for the re-key operations at client and TTP are as in Definition 4.7:

ibs_rekey_ttp runs the algorithms *Next* to draw a new update token and *Update* for the `ibs_master_params`. The update token is returned within `ibs_rekey_params`.
ibs_rekey_client runs the *Update* algorithm for `ibs_user_params` and `ibs_public_master_params`.

With this modification, the successful verification of gIBS signatures can be demonstrated. Revoking keys with the update token is also shown to exclude a sender of a message from the group as its signature are not longer verifiable by the receivers. Additional performance measurements for the *Setup* and *Extract* phases as well as the new phases were added and used for the evaluation in Chapter 7.

6.4 Summary and Finding

With the implementation of a GKMP, two re-keying mechanisms, and IBS, four prototypes were presented forming the building blocks of a system where efficient revocation of signing keys is possible. The design and setup of a testbed out of different resource-constrained devices allows the demonstration of the developed prototypes. An instantiation is shown in Figure 6.4, where the GCKS at the top is one of the Raspberry PIs in the testbed, running Strongswan as a G-IKEv2 server and Relic for the IBS and gIBS implementations. The Group Members (GMs) at the bottom are microcontrollers out of the testbed presented before and the FIT IoT-Lab, e.g., *Arduino M0+* (left), *ST Nucleo-F411* (center) or the *FIT IoT-Lab M3 Open Node* (right). They are all installed with RIOT OS running the G-IKEv2 client and Relic for the IBS and gIBS implementations. The GMs are connected wireless with each-other and the GCKS using IPv6 multicast over IEEE 802.15.4. With the use of an *FIT IoT-Lab M3 Open Node* for the implementation, the testbed's interoperability is demonstrated.

Thus, this section gives an answer for the research questions:

RQ 4 How to apply key distribution and revocation in constrained systems?

RQ 5 Which signature schemes are usable in constrained systems, can they benefit from IBC and how do they fit in such architectures?

All presented techniques can be implemented on constrained devices with a minimum of memory, networking and computing capabilities. However, a minimum amount of memory of around 10-20 KB is found necessary to perform asymmetric cryptography based on elliptic curves, either for Diffie-Hellmann, ECDSA or IBS. Hence, this chapter successfully showed that the transformation of the schemes in Chapter 5 work in practice on constrained nodes. It additionally verified the applicability of the presented architecture in Section 4.5 on a physical setup.

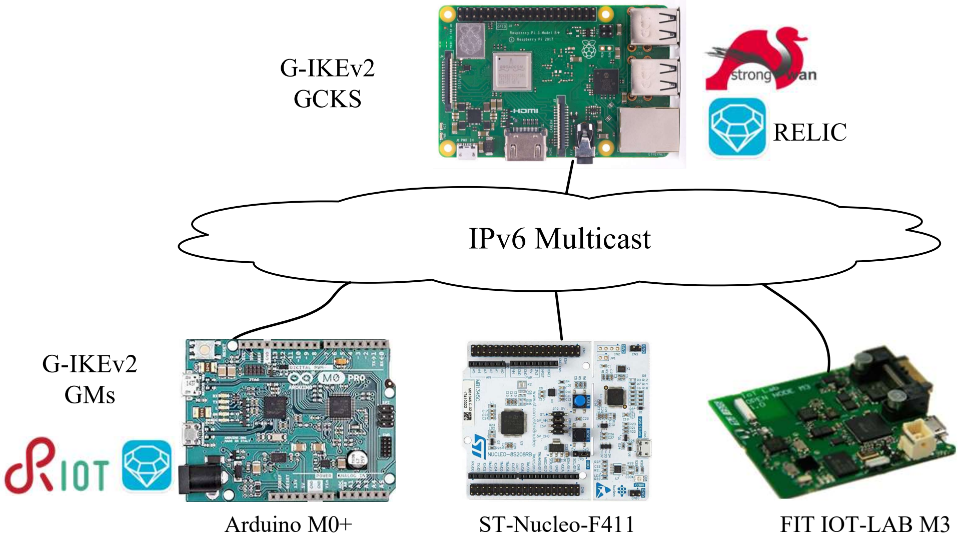


Figure 6.4: Implementation of a gIBS system with G-IKEv2 on the testbed.

7 Evaluation

The introduced transformations for Identity Based Signature (IBS) schemes aim at minimizing the overhead for key revocation, while keeping signing and verification during the actual communication efficient. This chapter evaluates the efficiency and applicability of group Identity Based Signature (gIBS) for the three use cases.

In a Key Updatable Signature Scheme (KUSS), the revocation of signing keys relies on symmetric re-keying mechanisms such as Logical Key Hierarchy (LKH), Centralized Authorized Key Extension (CAKE) and many others [107]. In the literature, their efficiency has been extensively studied, e.g., in [25, 28, 45]. Among others, the use of signature algorithms and certificate structures in constrained environments has been studied in [24, 44, 112].

Hence, this chapter studies how gIBS improves signing key revocation and lays out its computational costs. We therefor compare the complexity of the introduced revocation mechanism with related work and measure the performance of gIBS in comparison to Elliptic Curve Digital Signature Algorithm (ECDSA), as the de-facto-standard for the use cases.

7.1 Methodology

Chapter 6 presents an implementation of the *Group IBS Architecture* and thereby shows that the transformed IBS schemes can be used on hardware with constrained memory. With this proof of concept, this chapter evaluates three aspects of the gIBS schemes:

1. The transformations are evaluated against the efficiency requirements established in Chapter 2. For that purpose, a complexity analysis is carried out and used for comparing the revocation mechanism of four gIBS schemes with other mechanisms found in standards and literature. They outperform all related work by at least one property, while none of the properties requires complexity higher than $\mathcal{O}(\log n)$.

2. The key- and signature sizes of all gIBS schemes are analyzed for different representations of elliptic curves, namely *Edwards Curves* [16], *BLS-12* [19] and *BN* [10].

3. The performance of the IBS and gIBS implementation is analyzed on the testbed. We first measure the computation time for *signing* and *verification* of the four IBS schemes and compare them to ECDSA. An optimization of the two Schnorr-based schemes (vBNN and GG) allows significant improvement of signing performance over ECDSA. That at hand, the performance of all phases of all gIBS schemes is compared to their originating schemes, showing only negligible computational overhead over the originating schemes, while significantly reducing the effort for re-keying.

4. The evaluations of the Group Key Management Protocol (GKMP) and the symmetric re-keying as presented in Chapter 6 are summarized.

We thereby show, that the transformations are applicable for the use cases while solving the issue of expensive revocation mechanism found in related work.

7.2 Complexity Analysis

The analysis of the use cases in Chapter 2 derives three requirements regarding efficiency. Given those, we consider a signature scheme's properties of a) signature size (ER 1), b) signing and c) verification complexity (ER 3) as well as a revocation mechanism's properties for d) computational complexity (ER 3), e) network and f) storage overhead (ER 2). All of them can be discussed in form of complexity classes for a group of n communication partners exchanging m messages, with $m \ggg n$:

$\mathcal{O}_{|\sigma|}$ **Signature Size:** complexity of the signature's size.

$\mathcal{O}_S$ **Signing:** complexity for the signer to calculate the signature.

$\mathcal{O}_V$ **Verification:** complexity for the receiver to verify a signature.

$\mathcal{O}_{N_V}$ **Verification Network Overhead:** complexity of network overhead for verifying m signatures.

$\mathcal{O}_{R_{TTP}}$ **Preparing Revocation:** complexity to prepare revocation information at the Trusted Third Party (TTP).

$\mathcal{O}_{R_{GM}}$ **Processing Revocation:** complexity to process revocation information at the Group Member (GM).

$\mathcal{O}_{N_R}$ **Revocation Network Overhead:** complexity of number of messages count for revocation.

$\mathcal{O}_{|R|}$ **Revocation Size:** complexity of revocation message size.

Chapter 3 presents different revocation mechanisms as related work, each of which allows either knowledge or mathematical based revocation. Both come with a naive, not optimized revocation mechanism and we first describe the complexity of these *generic* approaches. Next, the optimizations of mechanisms in related work and state of the art are examined and their complexity is compared to either the mathematical or knowledge based *generic* approach. We start with knowledge based and continue with mathematical approaches before the IBS algorithms which were transformed to Two Key Signature Scheme (2KSS), Updatable Two Key Signature Scheme (U2KSS) and KUSS are evaluated.

The complexity of each algorithm is summarized in a table such as the following, with each column presenting one of the eight complexity classes:

$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$

The first row presents the respective *generic* approach and the following row(s) the difference of the presented technique to it. We present complexities in absolute values, while equal and additional complexity is presented with ± and +, respectively. The section will close with an overview and comparison of all examined solutions.

7.2.1 Definition of generic Revocation Mechanism

Signature schemes allow the definition of *naive* or *generic* revocation mechanism for validation of the signers legibility to send the message. With a knowledge based mechanism, the verifier needs to perform additional checks on the public key of the signer, while with a mathematical the verification fails instantly.

I.) Knowledge

A very commonly used mechanism in many scenarios is the distribution of certificates/keys on sender and receiver prior their communication. Receivers store the keys of potential senders, use them to verify the signature and update the list to allow implicit revocation. This is useful in small static settings but does not scale in rather large or dynamic environments. The information size to allow revocation is therefore $\mathcal{O}_{|R|}(n)$. The complexity to verify the signature is also $\mathcal{O}_V(n)$, as it requires the verifier to search in its (potentially unsorted) database for the correct key. Upon entry or exposure of a system's participant, all others have to update their list of keys, resulting in n messages and hence $\mathcal{O}_{N_R}(n)$. There is no network overhead during verification, all other properties have complexity $\mathcal{O}(1)$:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Knowledge	1	1	$\log n$	-	1	1	n	n

II.) Mathematical

The idea of mathematical invalidation is an inherent property of IBS (see [52]), but can be achieved with other mechanism as well. In difference to a knowledge based mechanism, the verifier has no overhead during signature validation but during preparation of the revocation information. In some scenarios this is preferred as the preparation can be typically done on a more powerful TTP and quickens the verification. The preparation now requires confidential information for each participant and, hence, $\mathcal{O}_{R_{TTP}}(n)$, but the verification complexity and the size of the revocation information drops to $\mathcal{O}(1)$. All other properties are equal to the knowledge based revocation:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1

7.2.2 Complexity of Knowledge Based Approaches

This section studies knowledge based approaches, found in use cases, standards, and academia. In contrast to the *generic* mechanism for knowledge based validation most of them use revocation lists rather than handling whitelists. However, we begin with two constrained use cases, where the use of whitelists is found to fit:

- **I.)** **SecureWSN and ACE** both use *generic* whitelisting.
- **II.)** **X.509** is the widely adopted and specified in RFC 5280 [128], which defines the distribution and validation of Certificate Revocation Lists (CRLs).
- **III.)** **Online Certificate Status Protocol** defines online validation of certificates as an alternative (or addition) to X.509's CRLs.
- **IV.)** **vBNN-IBS** defines the use of identity revocation lists.
- **V.)** **Group signatures** mathematically include a revocation list in the signature.

I.) SecureWSN and ACE

In SecureWSN [151] and the proposed standards for group key distribution in ACE [101, 169] both use a *generic* approach for validation. The public keys are either pre-installed or can be downloaded from the Group Controller Key Server (GCKS). This "whitelisting" allows short signatures (sometimes called "raw" signatures) as the identity of the signer is sufficient rather than sending the public key over the network.

	$\mathcal{O}_{\vert\sigma\vert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\vert R\vert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
SecureWSN	±	±	±	-	±	±	±	±
ACE	±	±	±	-	±	±	±	±

II.) X.509

X.509 is the description of a format for digital certificates defined by RFC 5280 [128], which allows a Certificate Authority (CA) to define and distribute revocation lists. The signature of the message includes the certificate of the signer, including the issuing CA, hence the signature size $\mathcal{O}_{|\sigma|}$ and its creation complexity $\mathcal{O}_S$ is increased compared to a raw signature by a constant factor. The same overhead appears for the revocation information prepared $\mathcal{O}_{R_{TTP}}$, stored $\mathcal{O}_{|R|}$ and processed $\mathcal{O}_{R_{GM}}$:

	$\mathcal{O}_{\vert\sigma\vert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\vert R\vert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
X.509	+1	+1	±	±	+1	+1	±	+1

III.) Online Certificate Status Protocol

Online Certificate Status Protocol (OCSP) as defined in RFC 6960 [133] allows to validate a signer's certificate even if the CRL defined by X.509 is not yet updated and can be used without any CRL stored by the verifier. Hence, the revocation's networking overhead $\mathcal{O}_{N_R}$ and its processing $\mathcal{O}_{R_{GM}}$ are eliminated. Similiarly, the revocation information $\mathcal{O}_{|R|}$ for a member is now constant. However, the networking overhead for verification of a signature is increased as it is necessary for every message $\mathcal{O}_{N_V}(m)$ sent in the network. In that regard, the verification's complexity $\mathcal{O}_V$ becomes a constant factor:

	$\mathcal{O}_{\vert\sigma\vert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\vert R\vert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
OCSP	±	±	+1	m	±	-	-	1

IV.) vBNN-IBS

In [24] the idea of revocation lists was picked up for the use of IBS in Wireless Sensor Networks (WSNs). The difference to X.509 is the fact, that the revocation lists consists of user's identities rather than a certificates serial number. As IBS allows validation of a signature by only knowing the signer's identity and the Master Public Key there is no computational overhead during verification ($\mathcal{O}_V$) compared to the *generic* mechanism. Still, the revocation list needs to be validated. The complexities are identical to the *generic* mechanism:

	$\mathcal{O}_{\vert\sigma\vert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\vert R\vert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
vBNN-IBS	±	±	±	-	±	±	±	±

V.) Group Signatures

Group signatures allow anonymous signing on behalf of a group, some of them allow revocation. The schemes in [13, 20] deal with list and the same complexities as vBNN-IBS, while the scheme introduced in [85] allows improvements. The verification complexity $\mathcal{O}_V$ is reduced to be constant with the costs of signing being $\mathcal{O}_S(\log n)$. Also the revocation information to be prepared $\mathcal{O}_{R_{TTP}}$, distributed $\mathcal{O}_{N_R}$ and stored $\mathcal{O}_{|R|}$ is reduced to $\log n$:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
[85]	±	$\log n$	1	-	$\log n$	±	$\log n$	$\log n$

Another approach of revocation enabled group signatures is presented in [180] and allows constant time verification $\mathcal{O}_V$ but requires the verifier to keep track over all revoked users for each epoch. Hence, the revocation information $\mathcal{O}_{|R|}$ is quadratic:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Knowledge	1	1	$\log n$	-	1	1	n	n
[180]	±	±	1	-	±	±	±	n^2

7.2.3 Complexity of Mathematical Approaches

IBS as presented in [52] as well as its sibling Attribute Based Signatures (ABS) can be used with the *generic* mathematical approaches and are therefore not presented again. Besides the work at hand, there have been other approaches on using mathematical mechanisms to revoke signing keys in a group.

I.) **Hierarchical Identity Based Signature (H-IBS)** is another sibling of IBS offering a similar approach while their mathematical or organizational structure differs.

II.) **Key insulation** are similarly constructed as the KUSS transforms but optimize other aspects.

I.) Hierarchical Identity Based Signature (H-IBS)

Even though not explicitly described, H-IBS as in [4, 51] allow the same mathematical revocation as IBS, the difference is within the construction of the trusted third party. The signer's keys are not linked to a single TTP but are part of a key hierarchy, which can be logical or physical [54, 171]. The scheme proposed in [4] requires the maximum number levels to be fixed during setup and is therefore inflexible for dynamic use cases. However, in [51] such flexibility is possible but comes at a cost: In any case the verifier needs to know all public keys of the nodes on the path from the signer to the node. Hence, the signature size $\mathcal{O}_{|\sigma|}$ and verification time $\mathcal{O}_V$ are logarithmic, while revoking information $\mathcal{O}_{|R|}$ reduces to a logarithmic function:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1
Inline-H-IBS [51, 171]	$\log n$	±	$\log n$	-	±	±	$\log n$	±

Another modification of [51] is also presented [171], where the verifier knows all public keys in the trees. This allows pre-computation of the verification information and the verification itself stays constant, while the revocation time $\mathcal{O}_{R_{TTP}}$ decreases logarithmic for the TTP . However, it comes with linear size revocation information and procession time at the verifier ($\mathcal{O}_{R_{GM}}(n)$, $\mathcal{O}_{|R|}(n)$).

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1
Pre-Computed-H-IBS [51, 171]	±	±	±	-	$\log n$	n	$\log n$	n

II.) Key Insulation

Key Insulation such as in [35, 100] allows validation of the signer's legibility based on a period (which is similar to the epoch introduced in Section 4.4.2). It allows revocation of a single key based on this token, however, in difference to IBS each key pair has to be pre-computed rather than using a single Master Public Key. Hence, the complexities are mostly equal to the mathematical *generic* approach, only the complexity for revocation $\mathcal{O}_{R_{TTP}}$ is reduced to constant while the size of revocation information $\mathcal{O}_{|R|}$ is linear:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1
[100]	±	±	+1	±	1	±	±	n

7.2.4 Complexity of gIBS

The transformations presented in Chapter 4 and their implementations on IBS schemes presented in Chapter 5 are meant to lower the complexity of revocation, with a special interest on resource constrained and dynamic scenarios. As it allows mathematical validation of the legibility, the overhead/reduction is compared to the mathematical *generic* revocation. With one exception, the transforms of all schemes behave identical in terms of complexity, why they are grouped. The KUSS-Hess transformation offers even lower complexity, why it is presented separately.

I.) 2KSS Transforms

2KSS transforms allow the integration of a shared secret to the signature, which is than used to validate the legibility of the signer. In all transformed schemes, the signature size $\mathcal{O}_{|\sigma|}$ is not changed compared to the originating scheme and the complexity for signing $\mathcal{O}_S$ and verification $\mathcal{O}_V$ is increased by a constant factor. By design, there is no network overhead during verification $\mathcal{O}_{N_V}$. Depending on the used key hierarchy, the preparation and size of the revocation information ($\mathcal{O}_{R_{TTP}}, \mathcal{O}_{|R|}$) is $\log_x n$, where x is the order of the tree and requires a single message ($\mathcal{O}_{N_R}(1)$) to be sent. It should be noted, that the revocation information is presented in form of symmetric keys, which lowers the actual size compared to sending of asymmetric keys as in Section 7.2.1.

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1
2KSS	±	+1	+1	-	$\log n$	±	1	$\log n$

II.) U2KSS Transforms

U2KSS transforms allow updating the shared secret to achieve forward secrecy. In contrast to 2KSS this introduces constant overhead for processing the revocation information $\mathcal{O}_{R_{GM}}$, as the shared secret needs to be updated with the update token. All other properties are equal to 2KSS and are the same for all considered schemes:

	$\mathcal{O}_{\lvert\sigma\rvert}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\lvert R\rvert}$
Math	1	1	1	-	n	1	n	1
U2KSS	±	+1	+1	-	$\log n$	+1	1	$\log n$

III.) KUSS Transforms

The KUSS transforms for vBNN, GG and BLMQ allow updating of the signing key, which removes the overhead during signing prior introduced by 2KSS and U2KSS. Although updating the signing key might be computational more complex than updating the shared secret, its still only a constant factor $\mathcal{O}_{R_{GM}}(1)$. All other properties are equal to U2KSS and are the same for all considered schemes:

	$\mathcal{O}_{\|\sigma\|}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\|R\|}$
Math	1	1	1	-	n	1	n	1
KUSS	±	±	+1	-	$\log n$	+1	1	$\log n$

IV.) KUSS-Hess Transformation

The KUSS transform of Hess' IBS scheme allows updating the public key in such a way, that the shared secret is not necessary for the verifier. Hence, the complexity for verification $\mathcal{O}_V$ is not changed compared to the generic approach, while all other properties stay equal to the other KUSS transforms:

	$\mathcal{O}_{\|\sigma\|}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\|R\|}$
Math	1	1	1	-	n	1	n	1
KUSS-Hess	±	±	±	-	$\log n$	+1	1	$\log n$

7.2.5 Summary

The findings of all revocation mechanisms discussed in the previous sections are summarized in Table 7.1. Most notably, we see in the last four rows, that the three transforms developed in Chapter 4 offer good trade-offs by only requiring logarithmic complexity for two out of the eight properties ($\mathcal{O}_{R_{TTP}}, \mathcal{O}_{N_R}$), while all others are constant.

This overhead is produced during the management steps rather than during the typically more frequent signing and verification. The influence of these phases is the drawback of all other mechanisms, except the group signature scheme in [100] and Pre-Computed H-IBS which are inefficient regarding revocation. While looking very efficient, the necessary online verification of OCSP and inevitable additional network traffic, would be a show-stopper for some of the considered scenarios. Another advantage of gIBS lies in the fact, that the storage and networking overhead for revocation is all based on symmetric cryptography. This allows to reduce the actual network overhead compared to other mechanisms where public keys or cryptographic hashes need to be transported.

The developed gIBS schemes come with an additional advantage over the other revocation mechanisms for signing keys discussed above. As presented in Section 5.5.3, none of the overheads produced by gIBS includes expensive operation. Section 7.4 shows this with practical evaluation of the computational performance on constrained hardware.

7.3 Network and Storage Overhead

Based on the implementation presented in Section 6.3 the network and storage consumption of the transformed schemes is presented. First, we show how the KUSS transforms behave when using different implementations of elliptic curves in comparison to ECDSA. Second, the actual networking overhead when implementing gIBS in Group Internet Key Exchange (G-IKEv2) is elaborated. The actual sizes are measured on the platform developed in Section 6.2 and the IoT-Lab provided by the FIT consortium [1].

Table 7.1: Overview of different solutions regarding complexity.

	$\mathcal{O}_{\|\sigma\|}$	$\mathcal{O}_S$	$\mathcal{O}_V$	$\mathcal{O}_{N_V}$	$\mathcal{O}_{R_{TTP}}$	$\mathcal{O}_{R_{GM}}$	$\mathcal{O}_{N_R}$	$\mathcal{O}_{\|R\|}$
ACE [101, 169]	1	1	$\log n$	-	1	1	n	n
SecureWSN [151]	1	1	$\log n$	-	1	1	n	n
X.509 [128]	1	1	n	-	1	1	n	n
OCSP [133]	1	1	1	m	1	-	-	1
vBNN-IBS [24]	1	1	n	-	1	1	n	n
Group Signature [85]	1	$\log n$	1	-	$\log n$	1	$\log n$	$\log n$
Group Signature [180]	1	1	1	-	1	1	n	n^2
IBS [52]	1	1	1	-	n	1	n	1
Inline-H-IBS	$\log n$	1	$\log n$	-	1	1	$\log n$	n
Pre-Computed-H-IBS	1	1	1	-	$\log n$	n	$\log n$	n
Key-Insulation [100]	1	1	1	-	1	1	n	n
2KSS	1	1	1	-	$\log n$	1	1	$\log n$
U2KSS	1	1	1	-	$\log n$	1	1	$\log n$
KUSS	1	1	1	-	$\log n$	1	1	$\log n$
KUSS-Hess	1	1	1	-	$\log n$	1	1	$\log n$

$\mathcal{O}_{|\sigma|}$ Signature Size $\mathcal{O}_S$ Signing $\mathcal{O}_V$ Verification $\mathcal{O}_{|R|}$ Revocation Size
$\mathcal{O}_{N_V}$ Verification Network Overhead $\mathcal{O}_{R_{TTP}}$ Preparing Revocation at the TTP
$\mathcal{O}_{R_{GM}}$ Processing Revocation at the GM. $\mathcal{O}_{N_R}$ Revocation Network Overhead

7.3.1 Parameter Size for different Elliptic Curves

With the implementation presented in Chapter 6 the size of the data types stored on the clients and send over the wire can be observed. During a systems life-cycle, participants can join or leave the system or send messages. Table 7.2 presents the sizes of the relevant parameters for these phases in Byte by using different elliptic curves. The first two columns present the *user's* private and the group's public data which are sent during joining. Next, *Re-Key* depicts the data that is send to the remaining users whenever a client leaves the system. The last column shows the message's overhead for the signature. The rows show all schemes presented in Chapter 5, together with their respective KUSS transforms. For comparison with other mechanisms, Raw-ECDSA – which is ECDSA without using a certificate structure such as X.509 – is presented in the last row of Table 7.2.

The Schnorr-based IBS schemes have larger private keys than ECDSA, however, the advantage of ECDSA over the pairing based schemes is negligible in most cases except that the parameters for the group increase. Public keys in ECDSA are significantly smaller, especially compared to the pairing based schemes. This is simply explained, as the IBS public keys are already proven “trustworthy” by the TTP, which is not true for Raw-ECDSA public keys where a certificate structure such as X.509 is required.

For comparison, different representations of elliptic curves are measured as well. They are abbreviated as *Ed* (Edwards Curves [16] with 255 bit prime number field), *BLS* (BLS-12 [19] with 381 bit prime number field) and *BN-254* (BN [10] using a 254 bit prime number field) and *BN-382* (BN [10] using a 382 bit prime number field). All of them offer at least 128-bit security, except BN-254 which is only 100-bit security but might be interesting for medium security use cases requiring lower networking overhead. The used cryptographic library (relic [W10]) does not offer pairings on Edwards curves, which is why BLMQ and Hess are not supported with that curve. All integer elements (e.g., User Secret Key (*usk*), group shared key (*gsk*)) are from the prime number field for the specific elliptic curve.

All measurements show the expected behavior. The key and signature sizes correspond with

Table 7.2: Sizes of the different parameters in Byte.

	User (usk)				Group (mpk + gsk)				Rekey				Signature			
	Ed	BLS	BN-254	BN-382	Ed	BLS	BN-254	BN-382	Ed	BLS	BN-254	BN-382	Ed	BLS	BN-254	BN-382
GG	67	99	67	99	34	50	34	50	101	149	101	149	101	149	101	149
KUSS	67	99	67	99	100	99	67	99	33	49	33	49	101	148	101	149
vBNN	67	99	67	99	34	50	34	50	101	149	101	149	101	132	100	132
KUSS	67	99	67	99	100	99	67	99	33	49	33	49	101	132	100	132
BLMQ		50	34	50		98	66	98		148	100	148		83	67	83
KUSS		50	34	50		147	99	147		49	33	49		83	67	83
Hess		50	34	50		98	66	98		147	100	147		83	67	83
KUSS		50	34	50		98	66	98		49	33	49		83	67	83
ECDSA	32	32	32	48	33	49	33	49					99	99	98	130

the curve parameters showing only a small overhead of $2-4$ Byte, which is easily explained with the overhead of the data structures. It is shown that the two pairing based schemes have significantly lower signature sizes. Additionally, they partially prove the claim of the authors of vBNN [24], saying that the signature's size is lower. With the used hash algorithm SHA-256, this is only true for BN-382 and BLS where the presentation of an elliptic curve point is larger than the hashes output of 32 Byte.

7.3.2 Networking Overhead

The transformations are expected to value most whenever re-keying is required, which happens when a client joins or leaves the group. Compared to the originating IBS schemes, the KUSS transforms come with a network overhead during setup of the system, where the gsk is distributed in addition to usk and Master Public Key (mpk). However, they show a significantly reduced overhead during re-keying, where only the update token, which is a single element of $\mathbb{Z}_p^*$, is securely distributed to all clients. For the original schemes, we assume that re-keying requires re-distribution of usk and mpk.

For distributing the key securely, we evaluate embedding the presented IBS schemes together with their respective KUSS transformations within the G-IKEv2 protocol. It offers an authenticated key exchange for setting up the private channel between GM anc GCKS, the secure group channel is implemented with LKH and CAKE respectively. That the protocol fits on the considered class of devices was shown in [42, 43, 53, 59, 66]. Please note, that every other protocol supporting similar efficient symmetric re-keying mechanisms can be used. The following outlines network overhead for distributing the keys in G-IKEv2, during:

I.) **Joining** The networking overhead for the client joining the system. It only affects the GCKS and the joining GM.

II.) **Re-Keying (Join)** The networking overhead for clients which were part of the system, before a new member joined. This is for achieving the property of *Backward Security.*

III.) **Re-Keying (Leave)** The networking overhead for clients, which are part of the system after a member left the group. This is for achieving the property of *Post-Compromise Security.*

I.) Joining

A client who wishes to participate in a system, connects to the TTP (in the terminology of G-IKEv2, this is denoted as GCKS). Thereby, the client - which is the *Initiator* in G-IKEv2 terminology - performs a Diffie-Helmman key exchange with the GCKS; this is `IKE_SA_INIT` exchange in the G-IKEv2 terminology. The `IKE_SA_INIT` includes the public DH-values, cryptographic Nonces and the supported ciphers (e.g., for hash, encryption and signing algorithms). If minimized as in [53], the message as well as the response can be as small as 98 byte. For authentication, the `GSA_AUTH` exchange is required. The initiator sends the message, which most notably includes the *IDg* as the identifier of the group it wishes to join. According to the protocol, the *IDg* can be anything, including IP multicast addresses, Fully-Qualified Domain Name (FQDN), etc. Showcasing the use of vBNN or GG in combination with AES-128 as the algorithm for key encryption with the Group Key Encryption Key (GKEK), the response includes the following cryptographic keys for LKH (left) and CAKE (right) when using the different elliptic curves:

	LKH				CAKE			
	Ed	BLS	BN-254	BN-382	Ed	BLS	BN-254	BN-382
GKEK	16 Byte				16 Byte			
usk	67	99	67	99	67	99	67	99
mpk	100	99	67	99	100	99	67	99
$\sum$	183	214	150	214	183	214	150	214
+ Key Path	$16 \cdot \log_2 n$ Byte				$32 \cdot \log_3 n$ Byte			

The lowest overhead is found with 150 Byte in *BN-254*, which is sufficient if no re-keying is necessary. For re-keying, the table shows the size of the *Key Path* when using LKH or CAKE, which depends on the number of members in the system. Hence, the `GSA_AUTH` response may not fit into one frame if the Maximum Transfer Unit (MTU) is too low, as it would be the case for IEEE 802.15.4 [70] or LoRa [W1]. However, as this is a single message sent to the client on its first join, it should be acceptable for most use cases.

II.) Re-Keying (Join)

Whenever a client joins the system, the system moves to a new epoch in order to retain *Forward Security*. Therefor, the GCKS sends an authenticated `GSA_REKEY` message to all previous GMs; the protocol overhead can be as small as 60 Byte [43, 174]. In addition, the GCKS encrypts the update token as the *Rekey Param* for epoch $e+1$ with the a new GKEK, which is in turn encrypted with the GKEK used in epoch e: Showcasing the use of vBNN or GG in combination with AES-128 as the algorithm for key encryption, this is as follows:

	Ed	BLS	BN-254	BN-382
GKEK	16 Byte			
Rekey Param	33	49	33	49
$\sum$	49	65	49	65

As the `GSA_AUTH` it does not fit in Lora or SigFox frames and needs to be fragmented. However, it fits into IEEE 802.15.4 frames with an MTU of 128 *Byte*.

III.) Re-Keying (Leave)

Whenever a client leaves the system, the system moves to a new epoch in order to retain *Post-Compromise Security*. Therefor, the GCKS sends an authenticated `GSA_REKEY` message to all remaining GMs. With 98 Byte, the protocol overhead during leaving is larger than for joining [43, 59, 174]. It includes the update token (see Table 7.2, column "Rekey") encrypted with a new GKEK for epoch $e+1$. The GKEK is encrypted with the key hierarchy (denoted as *Key Path Update*) as presented in Section 4.3 with the following networking overhead:

	LKH				CAKE			
	Ed	BLS	BN-254	BN-382	Ed	BLS	BN-254	BN-382
GKEK	16 Byte				16 Byte			
Rekey Param	33	49	33	49	33	49	33	49
$\sum$	49	65	49	65	49	65	49	65
Key Path Update	$2 \cdot 16 \cdot (\log_2 n - 1)$ Byte				$16 \cdot (\log_3 n - 1)$ Byte			

The numbers show, that the re-key does not fit in a single frame of IEEE 802.15.4 [70] or LoRa [W1]. However, the actual overhead of the KUSS transforms is the *Rekey* parameter depicted in Table 7.2. Its overhead is even less if the use case already uses a mechanism for efficiently distributing symmetric keys.

7.4 Performance Analysis of gIBS

The computational performance of the different gIBS schemes is evaluated in three steps:

1. We measure the computation time for signing and verifying messages and compare them with ECDSA.
2. We present an optimization in the signing algorithms for the schemes vBNN and GG and compare them with ECDSA as well.
3. We measure all gIBS schemes, compare them with their originating schemes and evaluate their computational overhead and improvements.

This allows the conclusion, that the computational overhead of gIBS is negligible compared to the originating schemes as well as compared to the widely used ECDSA.

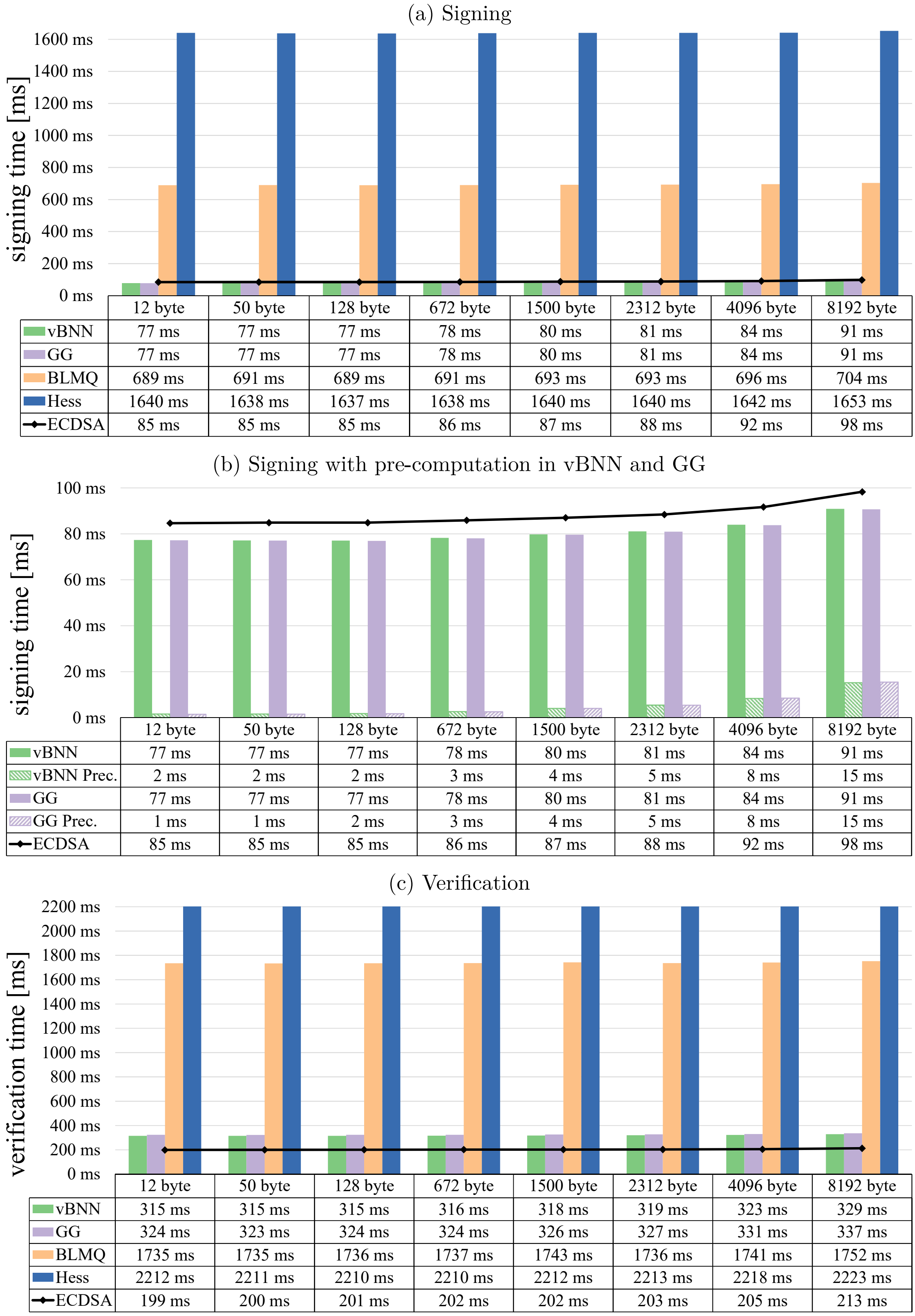
Test Setup

The evaluation uses the hard- and software presented in Chapter 6. The following computational performances are given for an 72 Mhz ARM Cortex M3 microcontroller, running *RIOT OS* [W7]. We measure the different phases of the IBS, ECDSA and gIBS algorithms by using the implementations based on the *Relic*-library [W10] (see Section 6.3). All values are presented as an arithmetic mean of 20 individual runs each run measuring the average time for 100 computations of a certain phase. We measure with RIOT OS' *xtimer*, the overall maximum standard deviation is as low as 2.07 %.

7.4.1 Comparing IBS with ECDSA

With the main purpose of IBS being the reduction of networking overhead compared to certificate based solution, we further evaluate the computational efficiency of the selected schemes. First, we are going to elaborate the computational overhead/reduction of IBS over Raw-ECDSA signatures. As pairings on Edwards curves are not implemented in the used cryptographic library, the measurements for the similar efficient but slightly less secure curve *BN-254* are presented in the following. The results presented are those from measurements performed on an ARM Cortex M3 and aim on comparing the time to sign and verify a message for different payload sizes. The plots in Figure 7.1 present the signing and verification time for the four examined schemes (vBNN, GG, BLMQ and Hess) as bar charts. The x-axis shows the evaluation for different scenarios by choosing typical MTUs, namely those of SigFox [W2] (12 Byte), Lora [W1] (50 Byte), Zigbee [70] (128 Byte), Bluetooth Low Energy [158] (672 Byte), Ethernet (1,500 Byte) and Wifi (2,312 Byte) in addition to some larger cases of 4 and 8 KB. For comparison, ECDSA is added as a line chart in all plots.

Figure 7.1: Time to sign/verify messages with the IBS schemes compared to ECDSA on 72 Mhz ARM Cortex M3 for typical message sizes.



(a) Signing

	12 byte	50 byte	128 byte	672 byte	1500 byte	2312 byte	4096 byte	8192 byte
🟩 vBNN	77 ms	77 ms	77 ms	78 ms	80 ms	81 ms	84 ms	91 ms
🟪 GG	77 ms	77 ms	77 ms	78 ms	80 ms	81 ms	84 ms	91 ms
🟧 BLMQ	689 ms	691 ms	689 ms	691 ms	693 ms	693 ms	696 ms	704 ms
🟦 Hess	1640 ms	1638 ms	1637 ms	1638 ms	1640 ms	1640 ms	1642 ms	1653 ms
◆ ECDSA	85 ms	85 ms	85 ms	86 ms	87 ms	88 ms	92 ms	98 ms

(b) Signing with pre-computation in vBNN and GG

	12 byte	50 byte	128 byte	672 byte	1500 byte	2312 byte	4096 byte	8192 byte
🟩 vBNN	77 ms	77 ms	77 ms	78 ms	80 ms	81 ms	84 ms	91 ms
▨ vBNN Prec.	2 ms	2 ms	2 ms	3 ms	4 ms	5 ms	8 ms	15 ms
🟪 GG	77 ms	77 ms	77 ms	78 ms	80 ms	81 ms	84 ms	91 ms
▨ GG Prec.	1 ms	1 ms	2 ms	3 ms	4 ms	5 ms	8 ms	15 ms
◆ ECDSA	85 ms	85 ms	85 ms	86 ms	87 ms	88 ms	92 ms	98 ms

(c) Verification

	12 byte	50 byte	128 byte	672 byte	1500 byte	2312 byte	4096 byte	8192 byte
🟩 vBNN	315 ms	315 ms	315 ms	316 ms	318 ms	319 ms	323 ms	329 ms
🟪 GG	324 ms	323 ms	324 ms	324 ms	326 ms	327 ms	331 ms	337 ms
🟧 BLMQ	1735 ms	1735 ms	1736 ms	1737 ms	1743 ms	1736 ms	1741 ms	1752 ms
🟦 Hess	2212 ms	2211 ms	2210 ms	2210 ms	2212 ms	2213 ms	2218 ms	2223 ms
◆ ECDSA	199 ms	200 ms	201 ms	202 ms	202 ms	203 ms	205 ms	213 ms

Comparing ECDSAs signing algorithm with the two Schnorr-based IBS schemes, there is almost no difference in terms of operations, namely group exponentiation (GE), modular exponentiation (ME), modular multiplication (MM) and Hashing in $\mathbb{Z}_p^*$. The measurements presented in Figure 7.2a support this assumption, showing that vBNN as well as GG are able to outperform ECDSA by around 10%. Please note, that the implementation for vBNN could be significantly improved compared to the measurements presented in [52]. Even though the selected elliptic curve is meant to be *pairing-friendly* [10], this operation impacts the signing algorithm significantly. BLMQ allows pre-computation of one of the pairings and does not require the generation of a new elliptic curve, but the overhead of around 500 % compared to the Schnorr-based schemes and ECDSA is huge. However, the signing time of around 700 ms may be acceptable for use cases, where networking overhead is the major restricted resource. With a factor of 2.5 compared to BLMQ, the overhead produced by Hess' IBS construction is even larger. Hence, the only algorithm found capable for a full KUSS transformation is only useful in environments, with the necessity of minimum amount of keys being stored on the system's participants.

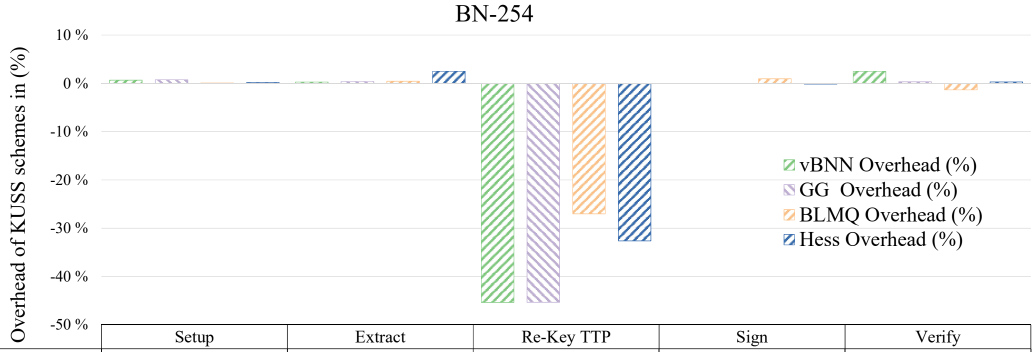
7.4.2 Optimization of IBS' Signing Performance

Both Schnorr-based IBS schemes offer improving the signing time by pre-computation. As exemplary shown by the following equation for signing in vBNN, the GE ($X = x\,P$) – which is the most expensive operation of a Schnorr-signature – is independent of the message m that is going to be signed:

$$\begin{aligned} X =& xP \text{ with: } x \overset{r}{\leftarrow} \mathbb{Z}_p^* \\ h =& h_2(id, m, R, X) \\ s =& (x + h \cdot u) \cdot \mathbf{g} \\ \sigma =& (s, R, h) \end{aligned} \tag{7.1}$$

Hence, it can be pre-calculated and used to speed-up sending a signed message. The original publication of vBNN [24] states this property, which reduces the signing to one hashing and a MM, but does not prove it with measurements. Hence, Figure 7.2b proves it by comparing the implementations featuring this property with the original schemes an ECDSA. It outperforms the other implementations by reducing the signing time to only a couple of ms. The increasing times for different message sizes show that the speed solely depends on the used hashing function (in this case SHA-256). This improvement seems valuable for use cases, where messages need to be send as fast as possible and could be interesting for a number of security critical scenarios (e.g., in health or safety).

Verifying a signature in ECDSA requires two GEs, where the two Schnorr-based IBS schemes require three. The measurements presented in Figure 7.2c prove this intuition, showing that the verification is about 50% slower than in ECDSA. Considering that this includes the validation of the signers validity, the verification time of ca. 300 ms still seems acceptable. The Raw-ECDSA processing presented here does not include this step, which is why at least one additional verification needs to be performed unless the public keys are pre-shared. The measurements allow similar conclusion for the two pairing based schemes as found for signing. In contrast to signing, BLMQ requires a pairing to be performed which significantly impacts the performance by a factor of 2.5 compared to signing. BLMQ's verification time is about 8.5 times slower than ECDSA and about 5.5 times slower than vBNN and GG. Hess' verification performance is even worse than signing and about 11 times slower than ECDSA.



	Setup	Extract	Re-Key TTP	Sign	Verify
vBNN Overhead (%)	0.65 %	0.28 %	-45.39 %	0.00 %	2.48 %
GG Overhead (%)	0.75 %	0.37 %	-45.34 %	0.02 %	0.33 %
BLMQ Overhead (%)	0.07 %	0.44 %	-27.04 %	0.93 %	-1.35 %
Hess Overhead (%)	0.18 %	2.48 %	-32.64 %	-0.17 %	0.29 %

Figure 7.3: Overhead/Reduction of KUSS transforms compared to the originating schemes.

7.4.3 Comparing gIBS with IBS

Especially the performance of the two Schnorr-based schemes leave IBS as an interesting option for use cases with the need for low networking overhead and time-bounded applications. Hence, it remains to show that the transformations to KUSS do not impact the computation. The comparison of the four transformed schemes with their originating ones for BN-254 is presented in Figure 7.3. It presents the five phases, *Setup*, *Extract*, *Re-Key TTP* (which is *Next* and *Update* for the TTP), *Sign* and *Verify* on the x-axis. The bars compare each of the four gIBS schemes with their respective original scheme. Positive values are interpreted as an overhead, while negative values show enhancement caused by gIBS over the classic approach. With the original schemes not defining a re-key mechanism for the clients, its *Update* is not included in the plots.

It experimentally proves the intuition presented in Section 5.5.3, saying that there is only negligible impact on the performance. KUSS transformation of all schemes show no negative consequences for the most frequent phases signing and verification, which are potentially time-critical. The measurements also show, that the re-key algorithm performed by the TTP is significantly reduced. The comparison assumes re-keying in the original schemes to perform setup and extract for one single user, while with KUSS the re-key operation is constant for all users.

None of the originating IBS schemes was designed for the purpose of efficient re-keying. Hence, Table 7.3 shows the measurements for all other phases except *Sign* and *Verify.* The rows present time in *ms* for *Setup*, *Extract* and *Rekey* for the TTP and the client for the four different elliptic curves. The columns show the the original IBS schemes and their respective KUSS transformations. As before, all measurements are based on the constrained M3 node, even though the TTP would be more powerful in most use cases. However, even the steps performed on the TTP (*Setup*, *Extract*, *Rekey TTP*) show acceptable numbers for a node as constrained as the M3. As this step is not present in originating schemes, re-keying on the client is of special interest as a typical task for such nodes. However, this step is not time-critical in most cases, why even the worst performance of 1.5*sec* in case of Hess and BLMQ with *BN-382* seems acceptable. The Schnorr-based schemes (vBNN and GG) achieve this step in 139 to 380 ms, depending on the used elliptic curve.

Table 7.3: Computation time for Setup, Extract and Re-Key phases in the different KUSS compared to the originating schemes (all times in ms).

	Phase	vBNN	KUSS	GG	KUSS	BLMQ	KUSS	Hess	KUSS
BN-254	Setup	126	127	126	127	364	364	363	369
	Extract	126	127	126	127	132	132	172	127
	Rekey TTP	253	138	253	138	496	362	536	138
	Rekey Client	0	139	0	139	0	501	0	139
BN-382	Setup	357	362	361	362	1126	1102	1126	1104
	Extract	358	362	362	362	373	376	503	493
	Rekey TTP	715	384	722	384	1494	1102	1621	1104
	Rekey Client	0	380	0	380	0	1506	0	1489
BLS12	Setup	262	263	262	265	746	733	747	745
	Extract	264	264	265	266	273	272	579	581
	Rekey TTP	526	264	528	266	1015	734	1326	746
	Rekey Client	0	263	0	265	0	977	0	953
Ed	Setup	211	213	208	212				
	Extract	213	213	210	213				
	Rekey TTP	424	212	418	212				
	Rekey Client	0	211	0	211				

7.5 Performance of G-IKEv2, LKH, and CAKE

For gaining maximum benefit of gIBS, we require the GKMP and the re-keying mechanisms to be efficient on constrained hardware. With the GCKS being typically more powerful, the performance of the implementations running at the GMs is of major interest. As described in Section 6.3, we chose G-IKEv2 [174] as a protocol and LKH [120] and CAKE [59] as re-keying mechanisms. All implementations were evaluated in previous work and the findings on an ARM M3 microcontroller are summarized as follows:

G-IKEv2: The performance of the protocol was evaluated in [53, 66]. We measure the performance of the initial key exchange between GM and GCKS:

- `IKE_SA_INIT`: The initial Diffie-Hellman key exchange of G-IKEv2 is an Elliptic Curve Diffie-Hellmann (ECDH) and requires ~ 187 ms on the GM.
- `GSA_AUTH`: For authentication, the implementation uses pre-shared keys and requires as low as ~ 6 ms for this second exchange.

LKH: The performance of LKH was evaluated in [43, 59]. We show measurements of tree depths between 4 and 10 enabling between 16 and 1024 GMs: At first, the performance of LKH is evaluated in [59] without protocol integration. It shows that decrypting the key hierarchy at the GM requires between 0.5 ms for 16 GMs and 1.1 ms for 1024 GMs. Second, G-IKEv2 is improved and LKH is integrated in [43]:

- `GSA_AUTH`: Upon entry, the GM receives and decrypts its personal key path. Processing the packet requires between 4.3 ms (16 GMs) and 5.5 ms (1024 GMs).
- `GSA_REKEY`: The relation of the remaining GM to the leaving GM in the key tree affects packet procession (refer to Section 4.3.1). If they are siblings – which is the worst case – it requires between 3.4 ms (16 GMs) and 14.9 ms (1024 GMs). If they only share the root only one decryption is necessary and requires 2.1 ms.

CAKE: The performance of CAKE was evaluated in [59]. Due to its ternary tree, the number of GMs is not directly comparable with LKH. We use a tree depth of 2 with 9 GMs and 7 with 2187 GMs for comparison. For that cases, CAKE shows a decryption time – which in all cases is only a modulo and XOR operation – between 0.2 ms and 0.7 ms.

7.6 Results

This chapter evaluates the benefits and drawbacks when using the developed transformation and measures their performance on resource constrained devices. The complexity notion developed in Section 7.2 gives a more specific answer on research question

> RQ 2 What are the requirements to meet *efficiency* in such use cases?

The analysis shows that communication and computation complexity can be significantly reduced within the system. The accumulated complexity for revocation of signing keys is $2 \log n$, which is better than any system found in literature. Additionally, in contrast to all solutions proposed for constrained environments (namely ACE, SecureWSN, X.509, vBNN-IBS), the revocation does not impact the most frequent operations of *Signing* and *Verification*. Key-Insulation or Group Signatures offer similar complexity as KUSS, but deal with multiple expensive operations, such as Pairings. The use of IBS as the basis of the transformation, complements the system with minimal networking overhead.

With the implementation of different IBS schemes and their evaluation on a constrained node allows to answer research question

> RQ 5: Which signature schemes are usable in constrained systems, can they benefit from IBC and how do they fit in such architectures?

As shown before in Chapter 2 and 3, ECDSA is the de-facto-standard for authentication in the use cases. The measurements in Section 7.4.1 once again demonstrate why this is valid choice when no certificates are necessary. However, the two Schnorr-based IBS schemes outperform ECDSA in terms of signing times. Additionally, any use case requiring trusted signatures, e.g., by X.509 certificates, can benefit from the implicit sender validation offered by IBS with only small overhead compared to Raw-ECDSA.

By supplementing the evaluation with the implementation of the gIBS schemes and including it in the G-IKEv2 protocol, we can also practically answer the sixth research question:

> RQ 6: How can IBS keys be revoked and how can the revocation be achieved in state-of-the-art key distribution systems?

It was shown, that the transformations work in practice on constrained nodes with a MCU featuring a 70 MHz processor. The evaluation showed no significant performance impact during signing and verification. Further, the operations being necessary on a constrained client whenever another user is revoked are efficient and show acceptable computation times even in the worst configurations.

The integration of gIBS in a GKMP shows another benefit. First, it is shown that protocols and re-key mechanisms are efficiently implementable on constrained hardware. Second, mechanisms such as LKH are already specified for many of such protocols and usable for the use cases in question. Even though they produce overhead that might not fit in a single frame, the integration comes with the benefit of interoperability due to use of standardized solutions. As the developed transforms are not bound to a specific solution, any implemented group key distribution mechanism in the use case can be reused to distribute the update tokens. This makes the overhead of gIBS solely dependent on the used elliptic curve, which is as low as 33 or 49 Byte for the curves discussed in this chapter.

8 Conclusion and Future Work

Secure communication must ensure a number of different characteristics, among them *Confidentiality*, *Integrity* and *Authenticity* [71]. Each of these key characteristics can be achieved by utilizing computationally hard problems, like the *Discrete Logarithm Problem*, *Prime Factorization* and others. The field of research which studies the applicability of such mathematical problems for the use in communication security is called *Cryptography.* A multitude of cryptographic schemes for achieving security during communication have been developed.

Our work specifically deals with *data source authentication*, which validates if "the source of data received is as claimed" [126]. We study this characteristic in dynamic, IP-based communication networks with participants burdened by various constraints, such as computational, storage or network limitations. Three representative and security critical use cases within the areas of Wireless Sensor Networks, Mobile Ad-Hoc Networks and Device-to-Device Communications are chosen and their specific requirements are presented. All of these scenarios require cryptographic authentication mechanisms for setting up secure communication or for verifying message authenticity. The constraints of our target environments paired with the dynamic behavior of the communication participants challenge *authorization* mechanisms. In fact, for meaningful authorization appropriate management of cryptographic material becomes inevitable and includes both, deployment and revocation of signing keys.

Existing solutions lack the necessary efficiency, as they are typically designed for rather static environments with potentially powerful communication partners. We address this situation and extend existing approaches by systematically answering the following research question:

> *How to achieve efficient revocation of cryptographic signing keys in systems with constrained resources and frequent changes of participant's authorization?*

As a first step, we deduce a definition of the term *efficiency* within the context of our work from the three use cases mentioned above. All cases favor symmetric over asymmetric cryptography and Elliptic Curve Cryptography (ECC) over Discrete Logarithm Problem (DLP) or prime factorization. We state that efficiency is achieved whenever computational and networking overhead for the revocation and communication are negligible. Examining and evaluating state-of-the-art mechanisms out of standardization and academia, highlights that revocation of signing keys is only barely studied for our target environments. Some mechanism such as cryptographic group key distribution and Identity Based Cryptography (IBC) are at least found to be partially efficient.

A main novelty of our work is the combination of these two well-established cryptographic techniques and their integration in a key management protocol. Similar to other systems providing authentication (e.g., a Public Key Infrastructure (PKI)), our solution requires a Trusted Third Party (TTP) to manage group memberships. In distinction to distributing revocation lists in other approaches, our approach revokes signing keys mathematically with a single and efficient push message. Consequently, the verification of the signature's correctness explicitly validates the signer's authorization. The newly introduced schemes provide *Forward-* and *Post-Compromise*-Security without significant performance impact for the communication. In fact, some configurations allow to outperform the state-of-the-art mechanism Elliptic Curve Digital Signature Algorithm (ECDSA).

Suchlike efficiency is possible by modifying Identity Based Signature (IBS) schemes to allow mathematically updating their signing and verification keys. The update is a symmetric cryptographic element that can be efficiently exchanged or updated with group key distribution mechanisms. Such mechanisms are available in various flavors and fit nicely in the (de-)centralized trust architectures found in all above mentioned use cases.

Our solution builds on a framework of well-defined mathematical transformation consisting of three consecutive steps: First, an interchangeable symmetric token is introduced in the original signing and verification algorithm. In a second step, the token is allowed to be updated and finally integrated in the key pair(s) of the underlying signature scheme. The result now features key updates, introducing the name Key Updatable Signature Scheme (KUSS). Fine-grained transformability conditions and security models are given for each step, making the modification applicable for schemes beyond IBC.

With this theoretical foundations, the goal of *efficient* signing key revocation is achieved by transforming and implementing four IBS schemes of different flavors of ECC. Using IBS minimizes networking overhead and ECC allows computational efficiency while conveniently meeting the transformability conditions. Systematic evaluation shows that the security of the transformed schemes are not harmed. In fact, all resulting group Identity Based Signature (gIBS) schemes are proven *existentially unforgeable under adaptively chosen-message-and-identity attacks (EUF-CMA)*, being the highest possible security claim. A complexity analysis shows no significant overhead over the originating schemes while outperforming all related work. The computational efficiency of the four gIBS schemes is practically evaluated on a testbed picturing the target environments. It shows that the transformed schemes can be successfully integrated to constrained hardware with a minimum memory and networking overhead. With Logical Key Hierarchy (LKH) and Centralized Authorized Key Extension (CAKE), efficient symmetric keys distribution is provided as the second building block. With that, the networking overhead for revocation is further optimized and the desired single message revocation for IBS is practically shown .

Future Work

Our approach demonstrates substantial benefits of IBS over certificate-based solutions in constrained environments. On a first glance, the IBS inherent strict trust relationship to a TTP seems to limit possible scenarios. However, literature for the latter is manifold and our use case analysis showed that there is no difficulty in imaging such scenarios. Our cryptographic transformations significantly simplify the management of keys and could serve as a door-opener for exploiting IBS in other comparable use cases.

The transformations in turn allow to interrupt this trust relationship. Imagine a scenario, where a trust anchor distributes IBS User Secret Keys (*usk*s) to the communication's participants. With the KUSS transformation, the participant could coordinate updates of their private keys with a distributed key exchange mechanism, such as the Group Diffie-Hellmann key exchange [73]. This allows exclusion of the TTP from the system while still featuring the network efficiency of IBS.

Based on the benefits of IBS, its integration into protocols of the IP family seems to be a valuable and logical next step. In particular, transport security protocols such as Encapsulated Security Payload (ESP), Datagram TLS (DTLS) or the ones discussed in the IETF's COSE working group seem perfect candidates. However, even key exchange protocols like Internet Key Exchange (IKEv2) and Host Identity Protocol (HIP) might benefit from the reduced signature size.

Additionally, it might be desirable to use cryptographic primitives other than ECC. The original scheme introduced by Adi Shamir utilizes prime factorization and features the mathematical operations. Another interesting target are IBS schemes based on Lattices [37, 175] which are currently believed to resist the threat of quantum computers [17]. However, as verification in corresponding schemes is not fault tolerant, its unclear if all developed transformation are possible in their current form. Isogenie-based primitives are another area of cryptography, where IBC is still in a very early stage of research.

Most of the currently proposed schemes in *Post-Quantum Cryptography* are not yet efficient regarding networking and computational overhead. As this field of research is relatively new, efficient revocation is not yet in the spotlight. Therefore, our proposed combination of IBS, KUSS and their integration into standard protocols could form a first step in tackling these challenges.

Erratum: Hess' EUF-2KSS-CMA–security

Theorem 5.3 in Section 5.4.2 on page 73 supposes the EUF-CMA security of Hess' 2KSS transformation under the condition that the original scheme is EUF-CMA secure. The proof for this theorem is incorrect, as the last step of the verification algorithm is missing in the proof.
In Hess, for a message m^*, the signer calculates:

$$h = h_2(m^*, r); \text{ with: } r = e(Q, P)^x \text{ and: } x \xleftarrow{r} \mathbb{Z}_p^*; Q \xleftarrow{r} \mathbb{G} \tag{1}$$

the verification for a signature $\sigma = (h, S)$ returns 1 if

$$h \overset{?}{=} h_2(m^*, \widetilde{r}); \text{ with:} \widetilde{r} = e(S, P) \cdot e(H_1(id^*), -Mpk)^h \tag{2}$$

The proof of Theorem 5.3 states the condition for successful verification as $r \overset{?}{=} \widetilde{r}$, which is true if and only if the sender sends $\tilde{h} = h \cdot \mathbf{g}$ instead of h. However, the verifier also calculates $h_2(m^*, \widetilde{r})$ for comparison with h which is unequal to $\tilde{h} = h \cdot \mathbf{g}$, the verification fails and the adversary $\mathcal{B}$ is hence unable to calculate a valid Hess signature from a 2KSS-Hess signature.

Thus, the Theorem 5.3 has to be restated and re-proven:

Theorem 5.3–A (Hess' EUF-2KSS-CMA security)**.** *In the random oracle model, suppose that an adaptive adversary $\mathcal{A}$ exists which makes at most $n_1 \geq 1$ queries of the identity hash and extraction oracle, at most $n_2 \geq 1$ queries of the message hash and signature oracle, and at most $n_3 \geq 1$ queries of the next and update oracle, which succeeds within time $T_{\mathcal{A}}$ of making an existential forgery of the Hess+$\mathbf{g}$ scheme with probability*

$$\varepsilon_A \geq \frac{a \cdot n_1 \cdot n_3 \cdot n_2^2}{p} \tag{3}$$

for some constant $a \in \mathbb{Z}^{\geq 1}$. Then there exist another probabilistic algorithm $\mathcal{C}$ and a constant $c \in \mathbb{Z}^{\geq 1}$ such that for any given $P, R \in G_1^*$ and $Mpk \in G^*$ with $Mpk = s(P)$ as described in [67], $\mathcal{C}$ computes $s(R)$ in the expected time

$$T_{\mathcal{C}} \leq \frac{c \cdot n_1 \cdot n_2 \cdot T_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}} \tag{4}$$

For certain instances of s, the computation of $s(R)$ means that $\mathcal{C}$ solves the Bilinear Diffie-Hellman-problem with respect to

$$(P, Mpk, R) \tag{5}$$

Proof. We assume familiarity with the proof in [67, Theorem 2] and the corresponding Example 1, which is the concrete instantiation to which we applied the KUSS-transform. In our derivation the function $s(x) := tx$ and $q(x) := e(x, Mpk)$ change to $s_e(x) := t\mathbf{g}_e x$ and $q_e(x) := e(x, \mathbf{g}_e Mpk)$, respectively. The proof is identical in essence, and we only illustrate the main cornerstones and additional arguments.

As in the original proof, there are the *Identity-Hash-*, *Extraction-*, *Message-Hash-*, and *Signature-Oracles*. The *Extraction-* and *Signature-Oracle* change conceptually with $s_e(x)$ and $q_e(x)$, as their output now depends on the epoch the query is made in. However, the output changes by a random value and is hence random. The oracles can therefore be used in the same way as before. We add the ***Next-*** and the ***Update-Oracle***, as follows:

Next: For the e-th *Next*-query, we return Δ_e where $\Delta_e \in \mathbb{Z}_p^*$ and $\Delta = (\Delta_e)_{e=1,2,..}$ constitutes a random tape. We model the transition from one epoch to the next by *Next*-queries: For example "events in the same epoch" means "events between which the adversary did not query *Next*". When *Next* is queried, we also update the function s_{e-1} such that $s_e(P) = \Delta_e \cdot Q_{e-1} =: Q_e$ holds. The function q_e is changed accordingly such that $p(Q_e) = q_e(P)$ holds. Because G is cyclic, $Q_e \in G^*$ is still a generator. The *Next*-oracle publishes q_e and Q_e while s_e is kept secret. After the e-th *Next*-query, we say that we "are in epoch e". Before the first *Next*-query, we are in epoch 0.

Update: The *Update-Oracle* takes as input a key Usk_{id_i} that was extracted or updated in a previous epoch (i.e. at least one *Next*-query has happened since the last *Extract*- or *Update*-queries with respect to this key). Let $\mathbb{K} = \{\Delta_{k_1}, \Delta_{k_2}, ..\}$ be the set of all tokens produced by the *Next*-oracle since the *Extract*- or *Update*-oracle returned Usk_{id_i}. Then the *Update*-oracle outputs $\mathbf{g} \cdot Usk_{id_i}$, where $\mathbf{g} = \prod_{\Delta_k \in \mathbb{K}} \Delta_k$ is a random element of $\mathbb{Z}_p^*$ since all Δ_e are random. This simulation fails if $\mathbf{g} = 1$, which happens with probability ${n_3}/{p}$ during n_3 *Next-Oracles*. As in [67], no identity may be queried twice, not even in different epochs, because the *Hash*- and the *Extraction-Oracle* do not depend on epochs.

Assume the i-th identity hash or extraction query for id_i happens in epoch e. Then we return $s_e(H(id_i)) = \lambda_i Q_e$ if requested. As λ and Δ are random and independent, $s_e(H(id_i))$ is a random element in G. When signing a message for id in epoch e, we compute $y_e \coloneqq q_e(H(id))$ instead of $y \coloneqq q(H(id))$, which is necessary to verify the resulting signature (u_e, r).

Running $\mathcal{A}$ and answering its oracle queries still depends deterministically on λ, δ, γ, the new tape Δ, and the random tape ω with which the adversary $\mathcal{A}$ is provided. Compared to [67, Theorem 2], the probability for a simulation failure increases by ${n_3}/{p}$ and is ${2n_2^2+n_3}/{p}$. Still, running $\mathcal{A}$ results in a signature in time $T_{\mathcal{A}}$ with probability at least $\varepsilon_{\mathcal{A}} - \frac{2n_2^2+n_3}{p} \geq \frac{\varepsilon_{\mathcal{A}}}{2}$ (true for $a = 14$), using at most n_1 identity and extraction queries, n_2 message hash and signature queries and n_3 next and updates queries. The application of the Forking Lemma to $\mathcal{A}$ and the resulting machines $\mathcal{B}$ and $\mathcal{C}$ can be adopted almost verbatim from [67, Theorem 2] up to the final step of dividing the signing equation. There are only two additional notes:

1. The tape Δ remains the same for all runs of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ can be handled exactly like γ with respect to the dependencies of ω, ω' and ω''.
2. The signatures yielded by replays under the same ω-tape given to a machine are valid for the same epoch e, namely the epoch that was reached with the last of the $\leq n_3$ next-queries. Hence, q_e is used in the verification equations for both (u_1, r) and (u_2, r).

We divide the two signing equations $p(u_i) = y_e^{v_i} r^{w_i}$ for $y_i = q_e(\mu_i R)$ and obtain

$$p\left(u_1 - ({w_1}/{w_2})u_2\right) = q_e(R)^{\mu_1 v_1 - ({w_1}/{w_2})\mu_2 v_2} \tag{6}$$

Since q_e is a monomorphism it holds that $\forall \mathbf{g}_e \in \mathbb{Z}_p^*, x \in G : q_e(x) = q_0(x)^{\mathbf{g}_e}$:

$$p\left(u_1 - ({w_1}/{w_2})u_2\right) = q_0(R)^{g_e(\mu_1 v_1 - ({w_1}/{w_2})\mu_2 v_2)} \tag{7}$$

where $g_e(\mu_1 v_1 - ({w_1}/{w_2})\mu_2 v_2) \neq 0$ since $g_e \neq 0$ and $\mu_1 v_1 - ({w_1}/{w_2})\mu_2 v_2 \neq 0$. Hence we get

$$p\Big(g_e\big(\mu_1 v_1 - ({w_1}/{w_2})\mu_2 v_2\big)^{-1}\big(u_1 - ({w_1}/{w_2})u_2\big)\Big) = q_0(R) \tag{8}$$

and we solve $p(x) = q(R) = q_0(R)$ and thus compute $x = s(R)$ in expected time $T_C \leq {c n_1 n_2 T_A}/{\varepsilon_A}$ with $c = 124416$. In our scheme, $s_e(P) = msk\mathbf{g}_e\, P = Mpk_e$ and msk is kept secret. Computing $s_e(R) = msk\mathbf{g}_e\, R$ given P and $s_e(P)$ is computationally hard and doing so in polynomial time amounts to solving the Bilinear Diffie-Hellman problem in $\mathbb{G}$ as defined in [67]. The rest of the proof holds verbatim. □

Bibliography

[1] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. "FIT IoT-LAB: A large scale open experimental IoT testbed". In: *IEEE World Forum on Internet of Things*. Piscataway, NJ: IEEE, 2015, pp. 459–464. ISBN: 978-1-5090-0366-2. DOI: 10.1109/WF-IoT.2015.7389098.

[2] Cory J. Antosh and Barry E. Mullins. "The Scalability of Secure Lock". In: *IEEE International Performance, Computing and Communications Conference, 2008*. Piscataway, NJ: IEEE, 2008, pp. 507–512. ISBN: 978-1-4244-3368-1. DOI: 10.1109/PCCC.2008.4745086.

[3] Arash Asadi, Qing Wang, and Vincenzo Mancuso. "A Survey on Device-to-Device Communication in Cellular Networks". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 1801–1819. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2319555.

[4] Man Ho Au, Joseph K. Liu, Tsz Hon Yuen, and Duncan S. Wong. "Practical Hierarchical Identity Based Encryption and Signature schemes Without Random Oracles". In: *IACR Cryptology ePrint Archive* 2006 (2006), p. 368. URL: https://eprint.iacr.org/2006/368.pdf.

[5] Emmanuel Baccelli, Cenk Gundogan, Oliver Hahm, Peter Kietzmann, Martine S. Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wahlisch. "RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT". In: *IEEE Internet of Things Journal* 5.6 (2018), pp. 4428–4440. DOI: 10.1109/JIOT.2018.2815038.

[6] Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias Wahlisch, and Thomas Schmidt. "RIOT OS: Towards an OS for the Internet of Things". In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Piscataway, NJ: IEEE, 2013, pp. 79–80. ISBN: 978-1-4799-0056-5. DOI: 10.1109/INFCOMW.2013.6970748.

[7] Joonsang Baek, Young-Ji Byon, Eman Hableel, and Mahmoud Al-Qutayri. "An Authentication Framework for Automatic Dependent Surveillance-Broadcast Based on Online/Offline Identity-Based Signature". In: *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. 2013, pp. 358–363. DOI: 10.1109/3PGCIC.2013.61. (Visited on 08/01/2016).

[8] Elaine B. Barker. *Digital Signature Standard (DSS)*. 2013. DOI: 10.6028/NIST.FIPS.186-4.

[9] 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: *Advances in Cryptology- Asiacrypt 2005*. Ed. by 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. Vol. 3788. Lecture Notes in Computer Science Ser. New York: Springer, Jan. 2006, pp. 515–532. ISBN: 978-3-540-30684-9. DOI: 10.1007/11593447_28.

[10] Paulo S. L. M. Barreto and Michael Naehrig. “Pairing-Friendly Elliptic Curves of Prime Order”. In: *Selected areas in cryptography.* Ed. by Bart Preneel and Stafford Tavares. Vol. 3897. Lecture notes in computer science. Berlin: Springer, 2006, pp. 319–331. ISBN: 978-3-540-33108-7. DOI: 10.1007/11693383_22.

[11] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. “Security Proofs for Identity-Based Identification and Signature Schemes”. In: *Advances in Cryptology - EUROCRYPT 2004.* Ed. by Christian Cachin and Jan L. Camenisch. Vol. 3027. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2004, pp. 268–286. ISBN: 978-3-540-21935-4. DOI: 10.1007/978-3-540-24676-3_17.

[12] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. “Security Proofs for Identity-Based Identification and Signature Schemes”. In: *Journal of Cryptology* 22.1 (2009), pp. 1–61. ISSN: 0933-2790. DOI: 10.1007/s00145-008-9028-8.

[13] Mihir Bellare, Haixia Shi, and Chong Zhang. “Foundations of Group Signatures: The Case of Dynamic Groups”. In: Springer, Berlin, Heidelberg, 2005, pp. 136–153. DOI: 10.1007/978-3-540-30574-3_11.

[14] Mihir Bellare and Bennet Yee. “Forward-Security in Private-Key Cryptography”. In: *Topics in Cryptology – CT-RSA 2003.* Ed. by Marc Joye. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1–18. ISBN: 978-3-540-36563-1.

[15] Daniel J. Bernstein. “ChaCha, a variant of Salsa20”. In: (2008). URL: http://cr.yp.to/chacha/chacha-20080120.pdf.

[16] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. “Twisted Edwards Curves”. In: *Progress in cryptology - AFRICACRYPT 2008.* Ed. by Serge Vaudenay. Vol. 5023. Lecture notes in computer science. Berlin: Springer, 2008, pp. 389–405. ISBN: 978-3-540-68159-5. DOI: 10.1007/978-3-540-68164-9_26.

[17] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography.* 1. Aufl. s.l.: Springer-Verlag, 2009. ISBN: 978-3-540-88702-7.

[18] John Black and Phillip Rogaway. “CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions”. In: *Advances in Cryptology - CRYPTO 2000.* Ed. by Mihir Bellare. Vol. 1880. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2000, pp. 197–215. ISBN: 978-3-540-67907-3. DOI: 10.1007/3-540-44598-6_12.

[19] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing”. In: *Advances in Cryptology - ASIACRYPT 2001.* Ed. by Colin Boyd. Vol. 2248. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2001, pp. 514–532. ISBN: 978-3-540-42987-6. DOI: 10.1007/3-540-45682-1_30.

[20] Dan Boneh and Hovav Shacham. “Group signatures with verifier-local revocation”. In: *Proceedings of the 11th ACM conference on Computer and communications security.* Ed. by Vijay Atluri, Birgit Pfitzmann, and Patrick McDaniel. New York, NY: ACM, 2004, p. 168. ISBN: 1581139616. DOI: 10.1145/1030083.1030106.

[21] Carsten Bormann, Mehmet Ersue, Ari Keränen, and Carles Gomez. *Terminology for Constrained-Node Networks.* Internet-Draft draft-bormann-lwig-7228bis-04. Work in Progress. Internet Engineering Task Force, Mar. 2019. 23 pp. URL: https://datatracker.ietf.org/doc/html/draft-bormann-lwig-7228bis-04.

[22] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for authentication and key establishment.* Second edition. Information security and cryptography. Berlin, Germany: Springer, 2020. ISBN: 978-3-662-58146-9.

[23] Oliver Bringmann. *Eingebettete Systeme*. 3rd ed. München: De Gruyter Oldenbourg, 2018. ISBN: 9783110518528.

[24] Xuefei Cao, Weidong Kou, Lanjun Dang, and Bin Zhao. "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks". In: *Computer Communications* 31.4 (2008), pp. 659–667. ISSN: 01403664. DOI: 10.1016/j.comcom.2007.10.017.

[25] Yacine Challal and Hamida Seba. "Group Key Management Protocols: A Novel Taxonomy". In: *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 2.10 (2008), pp. 3620–3633. ISSN: 2010-376X.

[26] David Chaum and Eugène van Heyst. "Group signatures". In: *EUROCRYPT'91 Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques.*

[27] Guang-Huei Chiou and Wen-Tsuen Chen. "Secure broadcasting using the secure lock". In: *IEEE Transactions on Software Engineering* 15.8 (1989), pp. 929–934. ISSN: 0098-5589. DOI: 10.1109/32.31350.

[28] Jan Chudzikiewicz, Tomasz Malinowski, Janusz Furtak, and Zbigniew Zieliński. "The Procedure of Key Distribution in Military IoT Networks". In: *Computer Networks*. Ed. by Piotr Gaj, Micha Sawicki, and Andrzej Kwiecieân. Vol. 1039. Communications in Computer and Information Science. Cham: Springer International Publishing and Imprint: Springer, 2019, pp. 34–47. ISBN: 978-3-030-21951-2. DOI: 10.1007/978-3-030-21952-9_3.

[29] Henri Cohen, Gerhard Frey, and Roberto Avanzi. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete mathematics and its applications. Boca Raton: Chapman & Hall/CRC, 2006. ISBN: 9780367801625.

[30] Erik Dahlman, Johan Sköld, and Stefan Parkvall. *4G LTE/LTE-advanced for mobile broadband*. 2nd ed. Burlington and Amsterdam: Elsevier Science and Academic Press, 2013. ISBN: 9780124199859.

[31] Vitalian Danciu, Tobias Guggemos, and Dieter Kranzlmüller. "Schichtung virtueller Maschinen zu Labor– und Lehrinfrastruktur". In: *9. DFN-Forum Kommunikationstechnologien*. Rostock, Germany, 2016, pp. 11–20.

[32] Quynh H. Dang. *Secure hash standard*. 2012. DOI: 10.6028/NIST.FIPS.180-4.

[33] Pratish Datta, Tatsuaki Okamoto, and Katsuyuki Takashima. "Efficient Attribute-Based Signatures for Unbounded Arithmetic Branching Programs". In: *Public-key cryptography - PKC 2019*. Ed. by Dongdai Lin. Vol. 11442. Lecture notes in computer science. Cham: Springer, 2019, pp. 127–158. ISBN: 978-3-030-17252-7. DOI: 10.1007/978-3-030-17253-4_5.

[34] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638.

[35] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. "Strong Key-Insulated Signature Schemes". In: *Public Key Cryptography - PKC 2003*. Ed. by Yvo G. Desmedt. Vol. 2567. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2002, pp. 130–144. ISBN: 978-3-540-00324-3. DOI: 10.1007/3-540-36288-6_10.

[36] Jinling Du, Wensheng Zhu, Jing Xu, Zhenhong Li, and Haifeng Wang. "A Compressed HARQ Feedback for Device-to-Device Multicast Communications". In: *2012 IEEE Vehicular Technology Conference (VTC Fall 2012)*. Piscataway, NJ: IEEE, 2012, pp. 1–5. ISBN: 978-1-4673-1881-5. DOI: 10.1109/VTCFall.2012.6399309.

[37] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. "Efficient Identity-Based Encryption over NTRU Lattices". In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer Berlin Heidelberg, 2014, pp. 22–41. DOI: 10.1007/978-3-662-45608-8_2.

[38] M. J. Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.* 2007. DOI: 10.6028/NIST.SP.800-38D.

[39] Morris J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* 2015. DOI: 10.6028/NIST.FIPS.202.

[40] Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, and James F. Dray Jr. *FIPS PUB 197, Advanced Encryption Standard (AES).* U.S.Department of Commerce/National Institute of Standards and Technology. 2001. DOI: 10.6028/NIST.FIPS.197.

[41] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* 10. Auflage. De Gruyter Studium. Berlin and Boston: De Gruyter Oldenbourg, 2018. ISBN: 9783110584684.

[42] Wolfgang Engelbrecht. "Group Key Management with Strongswan". Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2018. URL: mnm-team.org/pub/Fopras/enge18/.

[43] Marinus Enzinger. "Efficiently Re-Keying Multicast Groups with LKH in G-IKEv2". Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2019. URL: http://mnm-team.org/pub/Fopras/enzi19.

[44] Filip Forsby, Martin Furuhed, Panos Papadimitratos, and Shahid Raza. "Lightweight X.509 Digital Certificates for the Internet of Things". In: *Interoperability, Safety and Security in IoT.* Ed. by Giancarlo Fortino, Carlos E. Palau, Antonio Guerrieri, Nora Cuppens, Frédéric Cuppens, Hakima Chaouchi, and Alban Gabillon. Vol. 242. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham: Springer International Publishing, 2018, pp. 123–133. ISBN: 978-3-319-93796-0. DOI: 10.1007/978-3-319-93797-7_14.

[45] Janusz Furtak, Zbigniew Zielinski, and Jan Chudzikiewicz. "Security techniques for the WSN link layer within military IoT". In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT).* Piscataway, NJ: IEEE, 2016, pp. 233–238. ISBN: 978-1-5090-4130-5. DOI: 10.1109/WF-IoT.2016.7845508.

[46] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121. ISSN: 0166-218X. DOI: 10.1016/j.dam.2007.12.010.

[47] David Galindo and Flavio D. Garcia. "A Schnorr-Like Lightweight Identity-Based Signature Scheme". In: *Progress in cryptology - AFRICACRYPT 2009.* Ed. by Bart Preneel. Vol. 5580. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2009, pp. 135–148. ISBN: 978-3-642-02383-5. DOI: 10.1007/978-3-642-02384-2_9.

[48] Pimmy Gandotra, Rakesh Kumar Jha, and Sanjeev Jain. "A survey on device-to-device (D2D) communication: Architecture and security issues". In: *Journal of Network and Computer Applications* 78 (2017), pp. 9–29. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2016.11.002.

[49] Oscar Garcia-Morchon, Sye Loong Keoh, Sandeep Kumar, Pedro Moreno-Sanchez, Francisco Vidal-Meca, and Jan Henrik Ziegeldorf. “Securing the IP-based internet of things with HIP and DTLS”. In: *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks - WiSec ’13*. Ed. by Levente Buttyán, Ahmad-Reza Sadeghi, and Marco Gruteser. New York, New York, USA: ACM Press, 2013, p. 119. ISBN: 9781450319980. DOI: `10.1145/2462096.2462117`.

[50] Padmini Gaur and Mohit P. Tahiliani. “Operating Systems for IoT Devices: A Critical Survey”. In: *2015 IEEE Region 10 Symposium*. Ed. by John Vig, Anil K. Roy, Chonggang Wang, and Manik Lal Das. Piscataway, NJ: IEEE, 2015, pp. 33–36. ISBN: 978-1-4799-1782-2. DOI: `10.1109/TENSYMP.2015.17`.

[51] Craig Gentry and Alice Silverberg. “Hierarchical ID-Based Cryptography”. In: *Advances in Cryptology - ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2002, pp. 548–566. ISBN: 978-3-540-00171-3. DOI: `10.1007/3-540-36178-2_34`.

[52] Nils gentschen Felde, Sophia Grundner-Culemann, and Tobias Guggemos. “Authentication in dynamic groups using identity-based signatures”. In: *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Piscataway, NJ: IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-6876-4. DOI: `10.1109/WiMOB.2018.8589148`.

[53] Nils gentschen Felde, Tobias Guggemos, Tobias Heider, and Dieter Kranzlmüller. “Secure Group Key Distribution in Constrained Environments with IKEv2”. In: *2017 IEEE Conference on Dependable and Secure Computing*. Taipei, Taiwan: IEEE, 2017. DOI: `10.1109/DESEC.2017.8073823`.

[54] Sophia Grundner-Culemann. “Identity-based source authentication in constrained networks”. Masterthesis. Munich: Ludwig-Maximilians-Universität, 2017. URL: `http://mnm-team.org/pub/Diplomarbeiten/grun17/`.

[55] Tobias Guggemos. “Dynamic Key Distribution for Secure Group Communications in Constrained Environments”. In: *Doctoral Consortium: Doctoral Consortium on e-Business and Telecommunications*. Vol. 2018. SECRYPT. jul, Porto, Portugal, 2018.

[56] Tobias Guggemos, Vitalian Danciu, and Annette Kostelezky. “Protokollgestützte Selbstbeschreibung in Zugangsnetzen”. In: *11. DFN-Forum Kommunikationstechnologien*. may. Günzburg, Germany, 2018.

[57] Tobias Guggemos, Nils gentschen Felde, and Dieter Kranzlmüller. “Secure Group Communication in Constrained Networks - A Gap Analysis”. In: *The 1st 2017 GLOBAL IoT SUMMIT (GIoTS’17)*. Geneva, Switzerland: IEEE, 2017, pp. 1–4. DOI: `10.1109/GIOTS.2017.8016270`.

[58] Tobias Guggemos and Dieter Kranzlmüller. “gIBS – group Identity-Based Signatures: efficiently verifiable IBS key-revocation with a single multicast message”. In: *The IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2020)*. Ed. by International Association for Cryptographic Research. (under review). 2020.

[59] Tobias Guggemos, Klement Streit, Marcus Knüpfer, Nils gentschen Felde, and Peter Hillmann. “No Cookies, just CAKE: CRT based Key Hierarchy for Efficient Key Management in Dynamic Groups”. In: *13th International Conference for Internet Technology and Secured Transactions (ICITST-2018)*. dec, Cambridge, UK, 2018. DOI: `10.2053/ICITST.WorldCIS.WCST.WCICSS.2018.0002`.

[60] Fuchun Guo, Willy Susilo, and Yi Mu. *Introduction to Security Reduction.* SpringerLink Bücher. Cham: Springer, 2018. ISBN: 9783319930497. DOI: 10.1007/978-3-319-93049-7.

[61] Vipul Gupta, Michael Wurm, Yu Zhu, Matthew Millard, Stephen Fung, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. *Sizzle: a standards-based end-to-end security architecture for the embedded internet.* 2005.

[62] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. “Operating Systems for Low-End Devices in the Internet of Things: A Survey”. In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 720–734. DOI: 10.1109/JIOT.2015.2505901.

[63] David Hanes. *IoT fundamentals: Networking technologies, protocols, and use cases for the Internet of Things.* Indianapolis, Indiana: Cisco Press, 2017. ISBN: 978-1-58714-456-1.

[64] Michael Haus, Muhammad Waqas, Aaron Yi Ding, Yong Li, Sasu Tarkoma, and Jorg Ott. “Security and Privacy in Device-to-Device (D2D) Communication: A Review”. In: *IEEE Communications Surveys & Tutorials* (2017), p. 1. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2649687.

[65] Tobias Heer, Oscar Garcia-Morchon, René Hummen, Sye Loong Keoh, Sandeep S. Kumar, and Klaus Wehrle. “Security Challenges in the IP-based Internet of Things”. In: *Wireless Personal Communications* 61.3 (2011), pp. 527–542. ISSN: 0929-6212. DOI: 10.1007/s11277-011-0385-5.

[66] Tobias Heider. “Minimal G-IKEv2 implementation for RIOT OS”. Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2017. URL: http://mnm-team.org/pub/Fopras/heid17.

[67] Florian Hess. “Efficient Identity Based Signature Schemes Based on Pairings”. In: *Selected Areas in Cryptography.* Ed. by Kaisa Nyberg and Howard Heys. Vol. 2595. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2003, pp. 310–324. ISBN: 978-3-540-00622-0. DOI: 10.1007/3-540-36492-7_20.

[68] Peter Hillmann, Marcus Knüpfer, and Gabi Dreo Rodosek. “CAKE: Hybrides Gruppen-Schlüssel-Management Verfahren”. In: *1617-5468* (2017). ISSN: 1617-5468. URL: http://dl.gi.de/bitstream/20.500.12116/476/1/paper03.pdf.

[69] René Hummen, Jan H. Ziegeldorf, Hossein Shafagh, Shahid Raza, and Klaus Wehrle. *Towards viable certificate-based authentication for the internet of things.* 2013. DOI: 10.1145/2463183.2463193.

[70] *IEEE standard for local and metropolitan area networks: Part 15.4: Low-rate wireless personal area networks (LR-WPANs).* New York: Institute of Electrical and Electronics Engineers, 2011. ISBN: 978-0-7381-6683-4.

[71] ISO. *ISO/IEC 27000 - Information technology - Security techniques - Information security management systems - Overview and vocabulary.* 2016-02-15. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c066435_ISO_IEC_27000_2016(E).zip (visited on 01/03/2017).

[72] Gene Itkis. “Forward security, adaptive cryptography: Time evolution”. In: (2004).

[73] Antoine Joux. “A One Round Protocol for Tripartite Diffie–Hellman”. In: *Algorithmic Number Theory.* Ed. by Wieb Bosma. Vol. 1838. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2000, pp. 385–393. ISBN: 978-3-540-67695-9. DOI: 10.1007/10722028_23.

[74] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks.* Reprint. with corr. Chichester: Wiley, 2007. ISBN: 9780470519233.

[75] Christian Karpfinger and Hubert Kiechle. *Kryptologie: Algebraische Methoden und Algorithmen.* 1. Aufl. Wiesbaden: Vieweg + Teubner, 2010. ISBN: 978-3-8348-0884-4. DOI: `10.1007/978-3-8348-9356-7`.

[76] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman and Hall/CRC, 2014. ISBN: 9781466570276. DOI: `10.1201/b17668`.

[77] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network security: Private communication in a public world.* 2. ed. Prentice Hall series in computer networking and distributed systems. Upper Saddle River, NJ: Prentice Hall PTR, 2002. ISBN: 0130460192.

[78] Minhaj Ahmad Khan and Khaled Salah. “IoT security: Review, blockchain solutions, and open challenges”. In: *Future Generation Computer Systems* 82 (2018), pp. 395–411. ISSN: 0167739X. DOI: `10.1016/j.future.2017.11.022`.

[79] Yongdae Kim, Adrian Perrig, and Gene Tsudik. “Simple and fault-tolerant key agreement for dynamic collaborative groups”. In: *Proceedings of the 7th ACM conference on Computer and communications security.* Ed. by Dimitris Gritzalis, Sushil Jajodia, and Pierangela Samarati. New York, NY: ACM, 2000, pp. 235–244. ISBN: 1581132034. DOI: `10.1145/352600.352638`.

[80] Neal Koblitz. “Elliptic curve cryptosystems”. In: *Mathematics of Computation* 48.177 (1987), p. 203. ISSN: 0025-5718. DOI: `10.1090/S0025-5718-1987-0866109-5`.

[81] Neal Koblitz and Alfred Menezes. “Another look at non-standard discrete log and Diffie-Hellman problems”. In: *Journal of Mathematical Cryptology* 2.4 (2008), p. 268. ISSN: 1862-2976. DOI: `10.1515/JMC.2008.014`.

[82] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. “DTLS based security and two-way authentication for the Internet of Things”. In: *Ad Hoc Networks* (2013). DOI: `10.1016/j.adhoc.2013.05.003`.

[83] Roger Kowalewski, Tobias Fuchs, Karl Furlinger, and Tobias Guggemos. “Utilizing Heterogeneous Memory Hierarchies in the PGAS Model”. In: *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP).* IEEE, 2018, pp. 353–357. ISBN: 978-1-5386-4975-6. DOI: `10.1109/PDP2018.2018.00063`.

[84] Anja Lehmann and Björn Tackmann. “Updatable Encryption with Post-Compromise Security”. In: *Advances in Cryptology - EUROCRYPT 2018.* Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture notes in computer science. Cham: Springer International Publishing, 2018, pp. 685–716. ISBN: 978-3-319-78371-0. DOI: `10.1007/978-3-319-78372-7_22`.

[85] Benoît Libert, Thomas Peters, and Moti Yung. “Scalable Group Signatures with Revocation”. In: Springer, Berlin, Heidelberg, 2012, pp. 609–627. DOI: `10.1007/978-3-642-29011-4_36`.

[86] Zenghui Liu, Yingxu Lai, Xubo Ren, and Shupo Bu. “An Efficient LKH Tree Balancing Algorithm for Group Key Management”. In: *International Conference on Control Engineering and Communication Technology (ICCECT), 2012.* Piscataway, NJ: IEEE, 2012, pp. 1003–1005. ISBN: 978-1-4673-4499-9. DOI: `10.1109/ICCECT.2012.213`.

[87] Tobias Markmann. “Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves”. Masterthesis. Hamburg: Hamburg Univsersity of Applied Sciences, 2015. (Visited on 09/07/2019).

[88] Tobias Markmann, Thomas C. Schmidt, and Matthias Wählisch. “Federated End-to-End Authentication for the Constrained Internet of Things Using IBC and ECC”. In: *ACM SIGCOMM Computer Communication Review* 45.4 (2015), pp. 603–604. ISSN: 0146-4833. DOI: 10.1145/2829988.2790021.

[89] Wladislav Meixner. “Portierung und Entwicklung eines CC3200 Treibers für RIOT-OS”. Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2019. URL: http://www.mnm-team.org/pub/Fopras/meix19/.

[90] Andrian Melnikov. “A Testbed for Evaluating ID-based Authentication in Constrained Networks”. Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2018. URL: http://mnm-team.org/pub/Fopras/meln18.

[91] Daniele Micciancio and Michael Walter. “On the Bit Security of Cryptographic Primitives”. In: *Advances in Cryptology - EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture notes in computer science. Cham: Springer International Publishing, 2018, pp. 3–28. ISBN: 978-3-319-78380-2. DOI: 10.1007/978-3-319-78381-9_1.

[92] Daniel Migault and Tobias Guggemos. *Minimal ESP*. Internet-Draft draft-ietf-lwig-minimal-esp-00. Work in Progress. Internet Engineering Task Force, Apr. 2019. 13 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-lwig-minimal-esp-00.

[93] Daniel Migault, Tobias Guggemos, Carsten Bormann, and David Schinazi. *ESP Header Compression and Diet-ESP*. Internet-Draft draft-mglt-ipsecme-diet-esp-07. Work in Progress. Internet Engineering Task Force, Mar. 2019. 47 pp. URL: https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-diet-esp-07.

[94] Daniel Migault, Tobias Guggemos, Sylvain Killian, Maryline Laurent, Guy Pujolle, and Jean Philippe Wary. “Diet-ESP: IP layer security for IoT”. In: *Journal of Computer Security* 25.2 (2017), pp. 173–203. DOI: 10.3233/JCS-16857.

[95] Daniel Migault, Tobias Guggemos, and Yoav Nir. *Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)*. RFC 8750. Mar. 2020. DOI: 10.17487/RFC8750. URL: https://rfc-editor.org/rfc/rfc8750.txt.

[96] Daniel Migault, Daniel Palomares, Tobias Guggemos, Aurelien Wally, Maryline Laurent, and Jean Philippe Wary. *Recommendations for IPsec Configuration on Homenet and M2M Devices*. Cancun, Mexico, 2015. DOI: 10.1145/2815317.2815323.

[97] Victor S. Miller. “Use of Elliptic Curves in Cryptography”. In: *Advances in cryptology - CRYPTO ’85 ; proceedings*. Ed. by Hugh C. Williams. Vol. 218. Lecture notes in computer science. Berlin: Springer, 1986, pp. 417–426. ISBN: 978-3-540-16463-0. DOI: 10.1007/3-540-39799-X_31.

[98] Robert Moskowitz, Rene Hummen, and Miika Komu. *HIP Diet EXchange (DEX)*. Internet-Draft draft-ietf-hip-dex-08. Work in Progress. Internet Engineering Task Force, June 2019. 50 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-hip-dex-08.

[99] Harald Niederreiter and Arne Winterhof. *Applied Number Theory*. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-22320-9. DOI: 10.1007/978-3-319-22321-6.

[100] Go Ohtake, Goichiro Hanaoka, and Kazuto Ogawa. “An Efficient Strong Key-Insulated Signature Scheme and Its Application”. In: *Public key infrastructure*. Ed. by Stig F. Mjølsnes, Sjouke Mauw, and Sokratis K. Katsikas. Vol. 5057. Lecture notes in computer science. Berlin: Springer, 2008, pp. 150–165. ISBN: 978-3-540-69484-7. DOI: 10.1007/978-3-540-69485-4_11.

[101] Francesca Palombini and Marco Tiloca. *Key Provisioning for Group Communication using ACE*. Internet-Draft draft-ietf-ace-key-groupcomm-02. Work in Progress. Internet Engineering Task Force, July 2019. 36 pp. URL: `https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-02`.

[102] A. Perrig, R. Canetti, J. D. Tygar, and Dawn Song. "Efficient authentication and signing of multicast streams over lossy channels". In: *Proceedings*. Los Alamitos, California: IEEE Computer Society, 2000, pp. 56–73. ISBN: 0-7695-0665-8. DOI: `10.1109/SECPRI.2000.848446`.

[103] Adrian Perrig. "The BiBa one-time signature and broadcast authentication protocol". In: *Proceedings of the 8th ACM conference on Computer and Communications Security*. Ed. by Mike Reiter. New York, NY: ACM, 2001, p. 28. ISBN: 1581133855. DOI: `10.1145/501983.501988`.

[104] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. "SPINS: security protocols for sensor networks". In: *Wireless Networks* 8.5 (2002), pp. 521–534. ISSN: 1022-0038. DOI: `10.1023/A:1016598314198`.

[105] Curt Polack. "An IoT Test Bed with Heterogeneous Embedded Platforms and Network Technologies". Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2019. URL: `http://mnm-team.org/pub/Fopras/pola19`.

[106] Pawani Porambage, An Braeken, Corinna Schmitt, Andrei Gurtov, Mika Ylianttila, and Burkhard Stiller. "Group Key Establishment for Enabling Secure Multicast Communication in Wireless Sensor Networks Deployed for IoT Applications". In: *IEEE Access* 3 (2015), pp. 1503–1511. DOI: `10.1109/ACCESS.2015.2474705`.

[107] Sandro Rafaeli and David Hutchison. "A survey of key management for secure group communication". In: *ACM Computing Surveys* 35.3 (2003), pp. 309–329. ISSN: 03600300. DOI: `10.1145/937503.937506`.

[108] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig. "Securing communication in 6LoWPAN with compressed IPsec". In: *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. 2011, pp. 1–8. ISBN: 978-1-4577-0512-0. DOI: `10.1109/DCOSS.2011.5982177`.

[109] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. "Lithe: Lightweight Secure CoAP for the Internet of Things". In: *Sensors Journal, IEEE* 13.10 (2013), pp. 3711–3720. ISSN: 1530-437X. DOI: `10.1109/JSEN.2013.2277656`.

[110] S. Raza, D. Trabalza, and T. Voigt. "6LoWPAN Compressed DTLS for CoAP". In: *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*. 2012, pp. 287–289. ISBN: 978-1-4673-1693-4. DOI: `10.1109/DCOSS.2012.55`.

[111] Shahid Raza and Runar Mar Magnusson. "TinyIKE: Lightweight IKEv2 for Internet of Things". In: *IEEE Internet of Things Journal* 6.1 (2019), pp. 856–866. DOI: `10.1109/JIOT.2018.2862942`.

[112] Shahid Raza, Ludwig Seitz, Denis Sitenkov, and Goran Selander. "S3K: Scalable Security With Symmetric Keys—DTLS Key Establishment for the Internet of Things". In: *IEEE Transactions on Automation Science and Engineering* 13.3 (2016), pp. 1270–1280. ISSN: 1545-5955. DOI: `10.1109/TASE.2015.2511301`.

[113] Mike Reiter, ed. *Proceedings of the 8th ACM conference on Computer and Communications Security*. New York, NY: ACM, 2001. ISBN: 1581133855. DOI: `10.1145/501983`.

[114] Leonid Reyzin and Natan Reyzin. “Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying”. In: *Information Security and Privacy.* Ed. by Lynn Batten and Jennifer Seberry. Vol. 2384. Lecture notes in computer science. Berlin and Heidelberg: Springer, 2002, pp. 144–153. ISBN: 978-3-540-43861-8. DOI: 10.1007/3-540-45450-0_11.

[115] S. Bradner. *The Internet Standards Process – Revision 3.* RFC 2026 (Best Current Practice). RFC. Updated by RFCs 3667, 3668, 3932, 3978, 3979, 5378, 5657, 5742, 6410, 7100, 7127, 7475, 8179. Fremont, CA, USA: RFC Editor, Oct. 1996. DOI: 10.17487/RFC2026. URL: https://www.rfc-editor.org/rfc/rfc2026.txt.

[116] H. Harney and C. Muckenhirn. *Group Key Management Protocol (GKMP) Specification.* RFC 2093 (Experimental). RFC. Fremont, CA, USA: RFC Editor, July 1997. DOI: 10.17487/RFC2093. URL: https://www.rfc-editor.org/rfc/rfc2093.txt.

[117] H. Harney and C. Muckenhirn. *Group Key Management Protocol (GKMP) Architecture.* RFC 2094 (Experimental). RFC. Fremont, CA, USA: RFC Editor, July 1997. DOI: 10.17487/RFC2094. URL: https://www.rfc-editor.org/rfc/rfc2094.txt.

[118] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* RFC 2104 (Informational). RFC. Updated by RFC 6151. Fremont, CA, USA: RFC Editor, Feb. 1997. DOI: 10.17487/RFC2104. URL: https://www.rfc-editor.org/rfc/rfc2104.txt.

[119] D. Maughan, M. Schertler, M. Schneider, and J. Turner. *Internet Security Association and Key Management Protocol (ISAKMP).* RFC 2408 (Proposed Standard). RFC. Obsoleted by RFC 4306. Fremont, CA, USA: RFC Editor, Nov. 1998. DOI: 10.17487/RFC2408. URL: https://www.rfc-editor.org/rfc/rfc2408.txt.

[120] D. Wallner, E. Harder, and R. Agee. *Key Management for Multicast: Issues and Architectures.* RFC 2627 (Informational). RFC. Fremont, CA, USA: RFC Editor, June 1999. DOI: 10.17487/RFC2627. URL: https://www.rfc-editor.org/rfc/rfc2627.txt.

[121] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm. *Multicast Security (MSEC) Group Key Management Architecture.* RFC 4046 (Informational). RFC. Fremont, CA, USA: RFC Editor, Apr. 2005. DOI: 10.17487/RFC4046. URL: https://www.rfc-editor.org/rfc/rfc4046.txt.

[122] S. Kent and K. Seo. *Security Architecture for the Internet Protocol.* RFC 4301 (Proposed Standard). RFC. Updated by RFCs 6040, 7619. Fremont, CA, USA: RFC Editor, Dec. 2005. DOI: 10.17487/RFC4301. URL: https://www.rfc-editor.org/rfc/rfc4301.txt.

[123] S. Kent. *IP Encapsulating Security Payload (ESP).* RFC 4303 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Dec. 2005. DOI: 10.17487/RFC4303. URL: https://www.rfc-editor.org/rfc/rfc4303.txt.

[124] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals.* RFC 4919 (Informational). RFC. Fremont, CA, USA: RFC Editor, Aug. 2007. DOI: 10.17487/RFC4919. URL: https://www.rfc-editor.org/rfc/rfc4919.txt.

[125] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* RFC 4944 (Proposed Standard). RFC. Updated by RFCs 6282, 6775, 8025, 8066. Fremont, CA, USA: RFC Editor, Sept. 2007. DOI: 10.17487/RFC4944. URL: https://www.rfc-editor.org/rfc/rfc4944.txt.

[126] R. Shirey. *Internet Security Glossary, Version 2.* RFC 4949 (Informational). RFC. Fremont, CA, USA: RFC Editor, Aug. 2007. DOI: 10.17487/RFC4949. URL: https://www.rfc-editor.org/rfc/rfc4949.txt.

[127] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246 (Proposed Standard). RFC. Obsoleted by RFC 8446, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919, 8447. Fremont, CA, USA: RFC Editor, Aug. 2008. DOI: 10.17487/RFC5246. URL: https://www.rfc-editor.org/rfc/rfc5246.txt.

[128] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* RFC 5280 (Proposed Standard). RFC. Updated by RFCs 6818, 8398, 8399. Fremont, CA, USA: RFC Editor, May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/rfc/rfc5280.txt.

[129] K. Sandlund, G. Pelletier, and L-E. Jonsson. *The RObust Header Compression (ROHC) Framework.* RFC 5795 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Mar. 2010. DOI: 10.17487/RFC5795. URL: https://www.rfc-editor.org/rfc/rfc5795.txt.

[130] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2.* RFC 6347 (Proposed Standard). RFC. Updated by RFCs 7507, 7905. Fremont, CA, USA: RFC Editor, Jan. 2012. DOI: 10.17487/RFC6347. URL: https://www.rfc-editor.org/rfc/rfc6347.txt.

[131] B. Weis, S. Rowles, and T. Hardjono. *The Group Domain of Interpretation.* RFC 6407 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Oct. 2011. DOI: 10.17487/RFC6407. URL: https://www.rfc-editor.org/rfc/rfc6407.txt.

[132] D. Hardt (Ed.) *The OAuth 2.0 Authorization Framework.* RFC 6749 (Proposed Standard). RFC. Updated by RFC 8252. Fremont, CA, USA: RFC Editor, Oct. 2012. DOI: 10.17487/RFC6749. URL: https://www.rfc-editor.org/rfc/rfc6749.txt.

[133] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP.* RFC 6960 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, June 2013. DOI: 10.17487/RFC6960. URL: https://www.rfc-editor.org/rfc/rfc6960.txt.

[134] B. Claise (Ed.), B. Trammell (Ed.), and P. Aitken. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information.* RFC 7011 (Internet Standard). RFC. Fremont, CA, USA: RFC Editor, Sept. 2013. DOI: 10.17487/RFC7011. URL: https://www.rfc-editor.org/rfc/rfc7011.txt.

[135] C. Bormann, M. Ersue, and A. Keranen. *Terminology for Constrained-Node Networks.* RFC 7228 (Informational). RFC. Fremont, CA, USA: RFC Editor, May 2014. DOI: 10.17487/RFC7228. URL: https://www.rfc-editor.org/rfc/rfc7228.txt.

[136] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Internet Key Exchange Protocol Version 2 (IKEv2).* RFC 7296 (Internet Standard). RFC. Updated by RFCs 7427, 7670, 8247. Fremont, CA, USA: RFC Editor, Oct. 2014. DOI: 10.17487/RFC7296. URL: https://www.rfc-editor.org/rfc/rfc7296.txt.

[137] A. Rahman (Ed.) and E. Dijk (Ed.) *Group Communication for the Constrained Application Protocol (CoAP).* RFC 7390 (Experimental). RFC. Fremont, CA, USA: RFC Editor, Oct. 2014. DOI: 10.17487/RFC7390. URL: https://www.rfc-editor.org/rfc/rfc7390.txt.

[138] R. Moskowitz (Ed.), T. Heer, P. Jokela, and T. Henderson. *Host Identity Protocol Version 2 (HIPv2).* RFC 7401 (Proposed Standard). RFC. Updated by RFC 8002. Fremont, CA, USA: RFC Editor, Apr. 2015. DOI: 10.17487/RFC7401. URL: https://www.rfc-editor.org/rfc/rfc7401.txt.

[139] T. Kivinen. *Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation.* RFC 7815 (Informational). RFC. Fremont, CA, USA: RFC Editor, Mar. 2016. DOI: 10.17487/RFC7815. URL: https://www.rfc-editor.org/rfc/rfc7815.txt.

[140] H. Tschofenig (Ed.) and T. Fossati. *Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things.* RFC 7925 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, July 2016. DOI: 10.17487/RFC7925. URL: https://www.rfc-editor.org/rfc/rfc7925.txt.

[141] T. Bray (Ed.) *The JavaScript Object Notation (JSON) Data Interchange Format.* RFC 8259 (Internet Standard). RFC. Fremont, CA, USA: RFC Editor, Dec. 2017. DOI: 10.17487/RFC8259. URL: https://www.rfc-editor.org/rfc/rfc8259.txt.

[142] C. Schmitt, B. Stiller, and B. Trammell. *TinyIPFIX for Smart Meters in Constrained Networks.* RFC 8272 (Informational). RFC. Fremont, CA, USA: RFC Editor, Nov. 2017. DOI: 10.17487/RFC8272. URL: https://www.rfc-editor.org/rfc/rfc8272.txt.

[143] M. Sethi, J. Arkko, A. Keranen, and H. Back. *Practical Considerations and Implementation Experiences in Securing Smart Object Networks.* RFC 8387 (Informational). RFC. Fremont, CA, USA: RFC Editor, May 2018. DOI: 10.17487/RFC8387. URL: https://www.rfc-editor.org/rfc/rfc8387.txt.

[144] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols.* RFC 8439 (Informational). RFC. Fremont, CA, USA: RFC Editor, June 2018. DOI: 10.17487/RFC8439. URL: https://www.rfc-editor.org/rfc/rfc8439.txt.

[145] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Aug. 2018. DOI: 10.17487/RFC8446. URL: https://www.rfc-editor.org/rfc/rfc8446.txt.

[146] O. Garcia-Morchon, S. Kumar, and M. Sethi. *Internet of Things (IoT) Security: State of the Art and Challenges.* RFC 8576 (Informational). RFC. Fremont, CA, USA: RFC Editor, Apr. 2019. DOI: 10.17487/RFC8576. URL: https://www.rfc-editor.org/rfc/rfc8576.txt.

[147] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 26.1 (1983), pp. 96–99. ISSN: 00010782. DOI: 10.1145/357980.358017.

[148] Kiki Rizki, Argyro Lamproudi, Marco Tiloca, and Shahid Raza. “Group-IKEv2 for multicast IPsec in the internet of things”. In: *International Journal of Security and Networks* 14.1 (2019), p. 10. ISSN: 1747-8405. DOI: 10.1504/IJSN.2019.098908.

[149] Naoshi Sakamoto. “An efficient structure for LKH key tree on secure multicast communications”. In: *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014.* Ed. by Ju Yeon Jo. Piscataway, NJ: IEEE, 2014, pp. 1–7. ISBN: 978-1-4799-5604-3. DOI: 10.1109/SNPD.2014.6888676.

[150] Fabrizio de Santis, Andreas Schauer, and Georg Sigl. “ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications”. In: *DATE ’17 Proceedings of the Conference on Design, Automation & Test in Europe.* European Design and Automation Association, 2017, pp. 692–697.

[151] Corinna Schmitt. *Secure data transmission in wireless sensor networks.* Vol. 2013,07,2. Network architectures and services. München: Techn. Univ. München, Lehrstuhl Netzarchitekturen und Netzdienste, 2013. ISBN: 9783937201368.

[152] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C.* New York: John Wiley & Sons Incorporated, 2015. ISBN: 978-1-119-09672-6.

[153] C. P. Schnorr. "Efficient signature generation by smart cards". In: *Journal of Cryptology* 4.3 (1991). ISSN: 0933-2790. DOI: `10.1007/BF00196725`.

[154] Jaakko Seppala, Timo Koskela, Tao Chen, and Sami Hakola. "Network controlled Device-to-Device (D2D) and cluster multicast concept for LTE and LTE-A networks". In: *IEEE Wireless Communications and Networking Conference (WCNC), 2011.* Piscataway, NJ: IEEE, 2011, pp. 986–991. ISBN: 978-1-61284-255-4. DOI: `10.1109/WCNC.2011.5779270`.

[155] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Workshop on the Theory and Application of Cryptographic Techniques.* Springer Berlin Heidelberg, 1984, pp. 47–53. DOI: `10.1007/3-540-39568-7_5`.

[156] Zach Shelby and Carsten Bormann. *6LoWPAN: The Wireless Embedded Internet.* Wiley Publishing, 2010. ISBN: 0470747994.

[157] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs.* Ed. by Cryptology ePrint Archive, Report 2004/332. 2004. URL: `https://eprint.iacr.org/2004/332`.

[158] Bluetooth SIG. *Specification of the Bluetooth System, v5.1.* 2019. URL: `https://www.bluetooth.com/specifications/bluetooth-core-specification/` (visited on 09/14/2019).

[159] Ricardo Silva, Jorge Sa Silva, Milan Simek, and Fernando Boavida. "Why should multicast be used in WSNs". In: *IEEE International Symposium on Wireless Communication Systems 2008.* Ed. by Guangzhi Qu. Piscataway, NJ: IEEE, 2008, pp. 598–602. ISBN: 978-1-4244-2488-7. DOI: `10.1109/ISWCS.2008.4726126`.

[160] Nigel P. Smart. *Delivierable 5.4 Algorithms, Key Size and Protocols Report (2018): EU H2020-ICT Project Report.* Ed. by ECRYPT-CSA. 2018. URL: `https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf` (visited on 08/21/2019).

[161] Michael Steiner, Gene Tsudik, and Michael Waidner. "Key Agreement in Dynamic Peer Groups". In: *IEEE Transactions on Parallel and Distributed Systems* 11.8 (2000). DOI: `10.1109/71.877936`.

[162] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. "The First Collision for Full SHA-1". In: *Advances in Cryptology - CRYPTO 2017.* Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture notes in computer science. Cham: Springer, 2017, pp. 570–596. ISBN: 978-3-319-63687-0. DOI: `10.1007/978-3-319-63688-7_19`.

[163] Angus Stevenson. *The Oxford dictionary of English.* 3. ed. Oxford: Oxford Univ. Press, 2010. ISBN: 978-0199571123.

[164] Xun Sun, Jianhua Li, Gongliang Chen, and Shutang Yang. "Identity-Based Directed Signature Scheme from Bilinear Pairings". In: *IACR Cryptology ePrint Archive* 2008 (2008), p. 305.

[165] Niranjan Suri, Mauro Tortonesi, James Michaelis, Peter Budulas, Giacomo Benincasa, Stephen Russell, Cesare Stefanelli, and Robert Winkler. "Analyzing the applicability of Internet of Things to the battlefield environment". In: *2016 International Conference on Military Communications and Information Systems (ICMCIS)*. Piscataway, NJ: IEEE, 2016, pp. 1–8. ISBN: 978-1-5090-1777-5. DOI: 10.1109/ICMCIS.2016.7496574.

[166] Andrew S. Tanenbaum and David Wetherall. *Computer networks*. 5. ed., internat. ed. Safari Tech Books Online. Boston, Mass.: Pearson, 2011. ISBN: 978-0-13-212695-3.

[167] Marco Tiloca, Rikard Hoeglund, Ludwig Seitz, and Francesca Palombini. *Group OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework*. Internet-Draft draft-tiloca-ace-group-oscore-profile-00. Work in Progress. Internet Engineering Task Force, July 2019. 29 pp. URL: https://datatracker.ietf.org/doc/html/draft-tiloca-ace-group-oscore-profile-00.

[168] Marco Tiloca, Kirill Nikitin, and Shahid Raza. "Axiom: DTLS-Based Secure IoT Group Communication". In: *ACM Transactions on Embedded Computing Systems* 16.3 (2017), pp. 1–29. ISSN: 15399087. DOI: 10.1145/3047413.

[169] Marco Tiloca, Jiye Park, and Francesca Palombini. *Key Management for OSCORE Groups in ACE*. Internet-Draft draft-ietf-ace-key-groupcomm-oscore-02. Work in Progress. Internet Engineering Task Force, July 2019. 30 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-02.

[170] Marco Tiloca, Göran Selander, Francesca Palombini, and Jiye Park. *Group OSCORE - Secure Group Communication for CoAP*. Internet-Draft draft-ietf-core-oscore-groupcomm-05. Work in Progress. Internet Engineering Task Force, July 2019. 49 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-05.

[171] Tobias Treutner. "Evaluation effizienter Schlüsselbäume für hierarchische identitätsbasierte Signaturen im IoT". Bachelor Thesis. Munich: Ludwig-Maximilians-Universität, 2018. URL: http://mnm-team.org/pub/Fopras/treu18.

[172] Maarten van Steen and Andrew S. Tanenbaum. *Distributed systems*. Third edition (Version 3.01 (2017)). London: Pearson Education, February 2017. ISBN: 978-15-430573-8-6.

[173] F. Vidal Meca, J. H. Ziegeldorf, P. M. Sanchez, O. G. Morchon, S. S. Kumar, and S. L. Keoh. "HIP Security Architecture for the IP-Based Internet of Things". In: *2013 27th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2013, pp. 1331–1336. ISBN: 978-1-4673-6239-9. DOI: 10.1109/WAINA.2013.158.

[174] Brian Weis and Valery Smyslov. *Group Key Management using IKEv2*. Internet-Draft draft-yeung-g-ikev2-16. Work in Progress. Internet Engineering Task Force, July 2019. 52 pp. URL: https://datatracker.ietf.org/doc/html/draft-yeung-g-ikev2-16.

[175] Jia Xie, Yu-pu Hu, Jun-tao Gao, and Wen Gao. "Efficient identity-based signature over NTRU lattice". In: *Frontiers of Information Technology & Electronic Engineering* 17.2 (2016), pp. 135–142. ISSN: 2095-9184. DOI: 10.1631/FITEE.1500197.

[176] Xuanxia Yao, Zhi Chen, and Ye Tian. "A lightweight attribute-based encryption scheme for the Internet of Things". In: *Future Generation Computer Systems* 49 (2015), pp. 104–112. ISSN: 0167739X. DOI: 10.1016/j.future.2014.10.010.

[177] Rehana Yasmin, Eike Ritter, and Guilin Wang. "An Authentication Framework for Wireless Sensor Networks using Identity-Based Signatures". In: *IEEE 10th International Conference on Computer and Information Technology (CIT), 2010*. Piscataway, NJ: IEEE, 2010, pp. 882–889. ISBN: 978-1-4244-7547-6. DOI: 10.1109/CIT.2010.165.

[178] Denise E. Zheng and William A. Carter. *Leveraging the internet of things for a more efficient and effective military*. CSIS reports. Washington, DC and Lanham, MD: Center for Strategic & International Studies and Rowman & Littlefield, 2015. ISBN: 978-1-4422-5890-7.

[179] Minghui Zheng, Guohua Cui, Muxiang Yang, and Jun Li. "Scalable Group Key Management Protocol Based on Key Material Transmitting Tree". In: *Information security, practice and experience*. Ed. by Ed Dawson and Duncan S. Wong. Vol. 4464. Lecture notes in computer science. Berlin: Springer, 2007, pp. 301–313. ISBN: 978-3-540-72159-8. DOI: `10.1007/978-3-540-72163-5_23`.

[180] Sujing Zhou and Dongdai Lin. "Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps". In: *Cryptology and network security*. Ed. by David Pointcheval, Yi Mu, and Kefei Chen. Vol. 4301. Lecture notes in computer science. Berlin: Springer, 2006, pp. 126–143. ISBN: 978-3-540-49462-1. DOI: `10.1007/11935070_8`.

[181] Joachim von Zur Gathen. *CryptoSchool*. 1st ed. 2015. Berlin, Heidelberg and s.l.: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-48423-4. DOI: `10.1007/978-3-662-48425-8`.

Webography

[W1] *The LoRa Alliance™*. 2019. URL: https://lora-alliance.org/ (visited on 11/14/2019).

[W2] sigfox foundation. *Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)*. 2019. URL: https://www.sigfox.com (visited on 11/21/2019).

[W3] Persistent Systems LLC. *Mobile ad hoc networking solution delivers reliable comms, situational awareness, under rough conditions*. Apr. 24, 2018. URL: https://www.prnewswire.com/news-releases/persistent-systems-successfully-demonstrates-flat-320-radio-mpu5-network-300634923.html (visited on 11/14/2019).

[W4] Tesla ®. *Model S*. 2019. URL: https://www.tesla.com/models (visited on 11/14/2019).

[W5] Jérémy Jean. *TikZ for Cryptographers*. 2016. URL: https://www.iacr.org/authors/tikz/ (visited on 11/14/2019).

[W6] Future Internet Testing Facility. *FIT/IoT-LAB: a very large scale open testbed*. 2019. URL: https://www.iot-lab.info/ (visited on 11/14/2019).

[W7] Riot OS. *RIOT OS – The friendly Operating System for the Internet of Things*. 2019. URL: http://riot-os.org/ (visited on 11/21/2019).

[W8] RIOT OS. *RIOT-OS/RIOT: TI CC3200 SimpleLink RIOT support by gosticks: Pull Request #11866 ·*. 2019. URL: https://github.com/RIOT-OS/RIOT/pull/11866 (visited on 11/17/2019).

[W9] strongSwan. *strongSwan - IPsec VPN for Linux, Android, FreeBSD, Mac OS X, Windows*. 2019. URL: https://www.strongswan.org/ (visited on 11/21/2019).

[W10] D. F. Aranha and C. P. L. Gouvêa. *RELIC is an Efficient LIbrary for Cryptography*. URL: https://github.com/relic-toolkit/relic (visited on 11/14/2019).

[W11] Inc. Cisco Systems. *Cisco Group Encrypted Transport VPN Configuration Guide: GETVPN G-IKEv2*. 2018. URL: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_getvpn/configuration/xe-16/sec-get-vpn-xe-16-book.html (visited on 09/07/2019).

Acronyms

gsk group shared key 39, 41, 46–50, 52, 54, 57, 65, 67, 70, 72, 74, 89, 98, 99, 137

mpk Master Public Key 35, 40, 41, 56, 57, 63, 88, 89, 94, 96, 99, 100

msk Master Secret Key 35, 40, 57, 88, 89

usk User Secret Key 35, 40, 41, 48, 52, 56, 57, 63, 89, 98–100, 108

1m-DLP One-More Discrete Logarithm Problem 70, 71, 76, 77, 135

2KSS Two Key Signature Scheme xvi, 8, 46–51, 53, 55, 57–59, 61, 62, 64–70, 72–76, 78, 80, 92, 96–98, 111, 132, 135, 137

ABE Attribute Based Encryption 35

ABS Attribute Based Signatures 34–36, 95

AEAD Authenticated Encryption with Associated Data 28, 29

AES Advanced Encryption Standard 1, 28–30

BLE Bluetooth Low Energy 16, 18, 84, 85

CA Certificate Authority 1, 2, 16, 17, 37, 45, 94

CAKE Centralized Authorized Key Extension xvi, xvii, 4, 5, 9, 39, 41, 43–46, 52–54, 86–88, 91, 99–101, 105, 108, 137, 139

CDH Computational Diffie-Hellman Problem 23–25, 27, 30, 135

CLM Certificate Lifecycle Managment 2, 4

COSE CBOR Object Signing and Encryption 31

CRL Certificate Revocation List 2, 37, 93, 94

CRT Chinese Remainder Theorem 5, 43, 135

D2D Device-to-Device Communication xv, 3, 12, 13, 18–20, 107

DDH Decisional Diffie-Hellman Problem 24, 27, 28, 135

DLP Discrete Logarithm Problem 2, 23, 25, 28, 29, 71, 77, 78, 107, 135, 139

DSA Digital Signature Algorithm 23, 29, 30

DTLS Datagram TLS 31, 38, 108

ECC Elliptic Curve Cryptography xi, xiii, xv, 2, 4, 8, 21, 25, 27, 29, 30, 35, 38, 55, 57, 63, 78, 107–109

ECDH Elliptic Curve Diffie-Hellmann 17, 30, 105, 137

ECDLP Elliptic Curve Discrete Logarithm Problem 2, 26, 28, 29, 79, 135, 139

ECDSA Elliptic Curve Digital Signature Algorithm xi, xiii, xvii, 17, 29, 30, 35, 89–91, 97–99, 101–103, 106, 108, 137

ESP Encapsulated Security Payload 31, 108

EUF-2KSS-CMA Existential unforgeability under adaptively chosen-message-identity-and-token attacks for Two Key Signature Scheme (2KSS) 72–74, 135

EUF-CMA existentially unforgeable under adaptively chosen-message-and-identity attacks 46–50, 52, 70, 72–75, 108, 111

EUF-KUSS-CMA Existential unforgeability under adaptively chosen-message-identity-and-token attacks for Key Updatable Signature Scheme (KUSS) 75, 135

FIT Future Internet Testing Facility 81

FQDN Fully-Qualified Domain Name 100

G-IKEv2 Group Internet Key Exchange xvii, 40, 43, 44, 53, 86–90, 97, 99, 100, 105, 106, 137, 139

GCKS Group Controller Key Server 32, 33, 40–42, 52, 53, 87, 88, 90, 93, 99, 100, 105

GE group exponentiation 57, 103

gIBS group Identity Based Signature xi, xiii, xvi, xvii, 4, 9, 77, 81, 86, 89–91, 96, 97, 101, 103–106, 108, 137

GKEK Group Key Encryption Key 32, 41–44, 100, 101

GKMP Group Key Management Protocol 5, 6, 9, 39–41, 43, 53, 81, 86, 87, 90, 91, 105, 106

GM Group Member 32, 33, 40–44, 52, 78, 79, 90, 92, 98–100, 105

GSA Group Security Association 32, 40, 41, 52, 53, 87

GSP Group Security Policies 32

GTEK Group Transport Encryption Key 32, 41

H-IBS Hierarchical Identity Based Signature 34, 35, 95, 97, 98

HIP Host Identity Protocol 31, 108

HMAC Keyed-Hash Message Authentication Code 1, 28–30, 56

IBC Identity Based Cryptography xi, xiii, 2, 3, 28, 53, 90, 106–109

IBS Identity Based Signature xi, xiii, xvi, xvii, 2–5, 8, 20, 21, 29, 34–42, 45–57, 61, 63, 70, 71, 75, 78, 79, 81, 86–99, 101–104, 106, 108, 109, 135, 137, 139

IETF Internet Engineering Task Force 3, 8, 31, 36, 108

IKEv2 Internet Key Exchange 30, 31, 33, 86–88, 108

IoT Internet of Things xi, xiii, 2, 11, 31, 84

IPsec IP security protocol 31, 38, 86, 87

ISO International Organization for Standardization 1

KEK Key Encryption Key 32, 33, 40, 42, 43

KEM Key Encaspulation Mechanism 30

KEX Key EXchange 30

KGC Key Generation Center 41, 45, 46, 48, 51, 52, 54, 57, 59, 60, 66, 78

KUC Key Update Center 48, 51, 52, 78

KUSS Key Updatable Signature Scheme xi, xiii, xvi, 4, 8, 9, 46, 50–54, 56, 58–64, 66–70, 74–80, 86, 89, 91, 92, 95–99, 101, 103–106, 108, 109, 111, 132, 135, 137, 139

LKH Logical Key Hierarchy xvi, xvii, 4, 9, 34, 39, 41–46, 52–54, 86–88, 91, 99–101, 105, 106, 108, 137, 139

MANET Mobile Ad-Hoc Network xv, 2, 3, 12, 13, 17–20, 107

ME modular exponentiation 56, 57, 103

MITM Man-in-the-Middle 1, 30

MM modular multiplication 57, 103

MTU Maximum Transfer Unit 13, 14, 31, 100, 101

MWN Munich Scientific Network 82, 86

OCSP Online Certificate Status Protocol 37, 93, 94, 97, 98

PKI Public Key Infrastructure 1, 36, 39, 107

PRF Pseudo Random Function 77

SA Security Association 87, 88

SHA Secure Hash Algorithm 1, 28, 29

SL Secure Lock 43, 44

TLS Transport Layer Security 30, 31

TPM Trusted Platform Module 17

TTP Trusted Third Party 35, 40, 41, 45, 51, 56, 77, 88, 89, 92, 93, 95, 98, 99, 104, 107, 108

U2KSS Updatable Two Key Signature Scheme xvi, 8, 46, 48–53, 56, 58, 59, 61, 62, 64, 66–68, 70, 72, 74, 76–80, 92, 96–98, 135, 137

VANET Vehicular Ad-Hoc Network 13

WSN Wireless Sensor Network xv, 2, 3, 12, 16, 17, 19, 20, 29, 36, 37, 61, 93, 94, 106, 107, 137

List of Lemmata, Theorems, and Definitions

Lemmata

Theorems

Definitions

List of Figures

List of Tables