INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Towards efficient Revocable Identity-based Signatures on Lattice Problems

Emma Munisamy

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Towards efficient Revocable Identity-based Signatures on Lattice Problems

Emma Munisamy

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Sophia Grundner-Culemann Dr. Tobias Guggemos
Abgabetermin:	9. März 2023

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

München, den 9. März 2023

.....

Emma Munisamy

Abstract

Identity-based Signatures (IBS) are an important technique for light-weight authentication. A downside of this signature scheme is that when a key is revoked, all signing keys in the entire system have to be reissued by one central authority. There exists some approaches to reducing the effort of reissuing these user keys, among those a method of delegating this computationally expensive task from the central authority to each user. This scheme is called Key Updatable Signature Scheme (KUSS) and was designed and implemented for elliptic curve signatures and will not remain secure in the future quantum age. Lattice-based cryptography is a promising candidate for future IBS schemes due to its worst-case hardness assumption, efficient implementation and so-far quantum resistance.

This thesis provides an introduction to lattice-based cryptography, presenting the mechanisms of various cryptographic primitives. Based on this, the requirements for constructing a KUSS from an elliptic curve IBS scheme are analysed and compared with an existing lattice-based IBS scheme. Although no solution for a fully functional lattice-based IBS is found, a candidate KUSS is sketched and the requirements that remain to be met and proven are concluded.

Contents

1 Introduction **1**

2 Mathematical Foundations **5**
2.1 Algebra Basics . . . 5
2.2 Arithmetic in n-dimensional euclidean space . . . 7

3 Cryptographic Foundations **13**
3.1 Information Theoretic Security vs. Computational Security . . . 13
3.2 Computationally Hard Problems . . . 14
3.3 Cryptographic Primitives . . . 14

4 Background **17**
4.1 Elliptic Curves . . . 17
4.1.1 Definition . . . 17
4.1.2 Elliptic curves over finite fields . . . 19
4.1.3 Elliptic Curve Discrete Logarithm Problem . . . 22
4.1.4 Elliptic Curve Schnorr Signatures . . . 22
4.2 Identity-based Cryptography . . . 24
4.2.1 Identity-based Encryption . . . 25
4.2.2 Identity-based Signatures . . . 26
4.2.3 IBS with Elliptic Curves . . . 27
4.3 Related Work on Revocable Identity-based Signatures . . . 29
4.4 Revocation with Key Updatable Signature Schemes . . . 31
4.4.1 Key Updatable Signature Scheme . . . 31
4.4.2 Example KUSS on Elliptic Curves . . . 34
4.5 Summary . . . 37

5 Lattice-based Cryptography **39**
5.1 Lattices . . . 39
5.1.1 Properties of Lattices . . . 41
5.1.2 Q-ary Lattices . . . 43
5.1.3 Lattice Bases in Cryptography . . . 46
5.2 Computational Problems on Lattices . . . 47
5.2.1 Short Integer Solution (SIS) . . . 48
5.2.2 Learning With Errors (LWE) . . . 49
5.3 Cryptography with Lattices . . . 50
5.3.1 Ajtai's Hash Function . . . 51
5.3.2 Secret-Key Encryption . . . 52
5.3.3 Regev Public-Key Cryptosystem . . . 55
5.3.4 Dual Public-Key Cryptosystem . . . 57
5.3.5 Preimage Sampleable Trapdoor Functions . . . 60

5.3.6 Full-Domain Hash Signatures 65
5.3.7 Identity-based Encryption 67
5.3.8 Lyubashevsky Signatures 69
5.3.9 Identity-based Signatures 70
5.4 Related Work on Revocable Identity-based Signatures with Lattices 72
5.5 Summary 74

6 A concept for KUSS using Lattices **75**
6.1 Lattice-based 2KSS and U2KSS 75
6.2 Lattice-based KUSS Requirements Analysis 76
6.3 Findings 80
6.4 Open Problems 82

7 Conclusion and Future Work **83**

List of Abbreviations **85**

List of Figures **87**

Bibliography **89**

1 Introduction

Cryptography is an essential part of modern life. It is used during trivial matters, such as credit card payments or automatic software updates, where it for example protects confidential data, or verifies the identity of digital communication partners. New research shows that quantum computers could solve 2048 bit RSA in 4 days, 256 bit ECDLP only in 9 hours [GRR+23]. As soon as these algorithms are executed on sufficiently error-corrected quantum hardware, the common public-key cryptography will be insecure. Therefore it is important to change the established cryptography as fast as possible to quantum secure schemes.

This way also common digital signature schemes are threatened by quantum algorithms. Digital Signatures are used to ensure the authenticity of digital identities. The process of verifying the identity of a communication participant is called authentication. If for example a software update is digitally signed by the manufacturer, a mobile phone can verify that the update is really from the manufacturer and it can be installed without concern, even without the user noticing.

Identity-based Signatures The common solution for implementing digital signatures is by managing public keys within a Public Key Infrastructure (PKI). The credibility of a signature and thus the security of the authentication results from a chain of certificates, each assigning a public key to a user identity and having its validity through the signature of another user. This chain of trust must be checked, certificate by certificate up to its root, to verify the validity of one single signature.

To avoid these steps, there exists the alternative procedure of Identity-based Signatures (IBS)[Sha84]. Instead of a public key and a signing key, the unique identity of the user is chosen as the public key and the corresponding signing key is calculated corresponding to that. This means that the verification of the certificate chain can be omitted and computing power and network bandwidth can be saved. A constraint of this method is that such a system requires a central authority that would be capable of forging the signatures of all users. Nevertheless, there are many scenarios in which this signature scheme offers great advantages [Mar14], such as in a smart home, where low computing effort and little network communication are desired due to low-power devices, and a central authority managing the system does not raise a major security risk.

Signing Key Revocation An important property that digital signatures must enable is the ability to propagate that a user is no longer trustworthy. From that time on, his signatures must no longer be valid, this means his key has to be revoked in some way. In a company network, this may be the case, when an employee leaves the company. Another reason may be that a user's signing key has been exposed. All signatures created with this signing key can no longer be uniquely assigned to the user and must be detected as invalid. In the case of identity-based signatures, this user cannot simply generate a new signing key since it must match his identity and thus it is unique. To successfully revoke the leaked signing key, verification of signatures created with this signing key must no longer be successful. Therefor

the entire IBS system has to be updated, which is computationally expensive for the central authority and requires high network traffic when distributing the new settings privately to each user. Depending on the size and volatility of the system, despite its advantages an IBS can no longer be used efficiently.

Motivated by this, there are various approaches to make revocation in IBS more efficient. One of these approaches is a mechanism that transforms IBS schemes based on cyclic subsets of elliptic curves into efficiently revocable IBS in three steps, resulting in a so-called Key Updatable Signature Scheme (KUSS) [Gug20]. Their efficiency is accomplished by having the computations for key revocations no longer performed only on the central authority, but distributed over all users. This way also the network can be reduced significantly.

Post-quantum Cryptography Currently used public key cryptography is mainly based on factoring problems such as RSA or discrete logarithm problems such as Diffie-Hellman key exchange and elliptic curve cryptography. With Shor's algorithm [Sho97], these problems could be solved on a sufficiently large and error-corrected quantum computer, which will make the cryptography based on it insecure. For example there exists the already mentioned proposal for the implementation of Shor's algorithm that could break 256 bit elliptic curve logarithm in 9 hours [GRR+23]. Due to the heavy research in quantum computing, funded with $23 billion in total until 2026 [Dar21], the National Institute of Standards and Technology (NIST) expects that a quantum computer capable of breaking current public-key cryptosystems may be available in 20 years or even sooner [Nat23c]. The cryptography used in our everyday lives must therefore be changed to quantum-resistant alternatives as fast as possible. Therefore, in 2016 NIST started a standardisation process for quantum-resistant cryptography [CJL+16], so-called Post-Quantum Cryptography (PQC). Currently, in March 2023, three of the four remaining candidates for the new standard for key encapsulation mechanisms and digital signatures are based on hard problems on lattices [Nat23b].

Lattice-based cryptography is currently considered quantum-resistant, meaning that there is no algorithm that can solve the underlying problem in polynomial time on either classical nor quantum computers. Further advantages are that the arithmetic operations are as efficient as matrix calculations on small integers, keeping keys and signatures relatively small [Vai15b]. Besides, the security of this cryptography is based on worst-case hardness and thus is proven secure. Furthermore research in lattice-based cryptography is promising, because it supports various cryptographic primitives, from hash functions over IBS up to homomorphic encryption.

Contribution Considering the advantages of efficiently revocable identity-based signatures and lattice-based cryptography, this work investigates whether a lattice-based IBS scheme can be transformed into a KUSS by applying the transformation scheme that was constructed for elliptic curves. In this way, a quantum-resistant and efficiently revocable IBS could be achieved.

This work consists of two parts. First, a detailed overview of lattice-based cryptography is given. Different cryptographic primitives are introduced, successively leading to an identity-based signature scheme on lattices. The second part of this thesis analyses whether this IBS can be converted into an efficiently revocable KUSS. In the course of this, a requirements analysis for a transformation into a KUSS is developed, and then compared with the lattice-based IBS. Finally the required properties for such a transformation of a lattice-based IBS

are outlined, leaving the solution satisfying those requirements for future work.

Structure The thesis begins with the necessary mathematical foundations, consisting of algebra basics and arithmetic in multi-dimensional space. This is followed by an introduction to the basics of cryptography. Chapter 4 presents the required background, by first introducing Elliptic Curve Cryptography (ECC) followed by Identity-based Cryptography (IBC). The summary of related work is divided into two parts. The related work to this thesis is divided into general signing key revocation IBS, and signing key revocation in lattice-based IBS. The related work regarding general revocation in IBS is also found in this chapter under 4.3. The end of this chapter brings all former topics together by introducing a Key Updatable Signature Scheme (KUSS) on elliptic curves. Chapter 5 is entirely dealing with lattice-based cryptography, starting with the definition of lattices, computational problems on lattices. Further different cryptographic primitives constructed with lattices are introduced, ending with an IBS scheme. In 5.4 related work on the state of the art in revocation on lattice-based IBS is summarized. Chapter 6 analyzes what requirements make a transformation of an IBS scheme into a KUSS possible, and investigates which properties are fulfilled, summarizing the open problems. Chapter 7 concludes this thesis by summarizing the findings and giving perspectives on further research to be done.

2 Mathematical Foundations

This chapter summarizes the mathematical foundations relevant for this thesis. These are divided into two sections, first algebra basics followed by characteristics and operations in multidimensional euclidean spaces.

2.1 Algebra Basics

The required algebra definitions include groups, subgroups, and cyclic groups, as well as rings and fields, concluding with the set of integers modular prime as essential tool for state of the art cryptography.

2.1.1 Groups

A group $\mathbb{G}$ is a tuple $(G, \circ)$ composed by a set G and an operation $\circ : G \times G \to G$ for which the following 3 group axioms hold [MK18]:

1) The operation $\circ$ is associative, i.e., $a \circ (b \circ c) = a \circ b \circ c = (a \circ b) \circ c \quad \forall a, b, c \in G$
2) The set G contains an identity element or neutral element e, i.e., $\exists e \in G : a \circ e = e \circ a = a \quad \forall a \in G$
3) Every element a in G has an inverse element a^{-1}, i.e., $\forall a \in G \quad \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$

All groups are closed. This "*closedness*" or "*closure*" of a group $\mathbb{G}$ means that the operation $\circ : G \times G \to G$ between any elements of G results again in an element of G [Wei]. Hence, the operation never leads out of the set G.

The group $(G, \circ)$ is additionally called commutative or abelian if the operation $\circ$ is also commutative, i.e., $a \circ b = b \circ a \quad \forall a, b \in G$ [Fis11].

The order of a group $\mathbb{G}$, denoted with $|\mathbb{G}|$ is the number of elements contained in G.

As an example, the integers $\mathbb{Z}$ with the operation $+$ form an abelian group.

2.1.2 Subgroups

If $(G, \circ)$ is a group, then a nonempty subset $U \subset G$ together with $\circ$ is called a subgroup of $\mathbb{G}$ if the following 3 axioms hold [MK18]:

1) The set U contains an identity element e, i.e., $\exists e \in U : a \circ e = e \circ a = a \quad \forall a \in U$
2) Every element a in U has an inverse element a^{-1} in U, i.e., $\forall a \in U \quad \exists a^{-1} \in U : a \circ a^{-1} = a^{-1} \circ a = e$
3) U is "*closed*" under the operation $\circ$, i.e., $a \circ b \in U \quad \forall a, b \in U$

In other words, $(U, \circ)$ is a group and U is a nonempty real subset of G.

As an example, the set of all even integers forms a subgroup of $(\mathbb{Z}, +)$.

2.1.3 Cyclic Groups

A group $\mathbb{G} = (G, \circ)$ is called cyclic if an element $g \in G$ exists that generates the set G, by using the operation $\circ$ multiple times on itself [KM17]. The element g is called the "*generator*" of $\mathbb{G}$. There can be more than one generator in a cyclic group.

$$\mathbb{G} = \langle g \rangle = \{g^k : k \in \mathbb{Z}\} = \{g, g \circ g, ..., \underbrace{g \circ g ... \circ g}_{\text{k-times}}\}$$

As an example, the group $\mathbb{Z}_n$ with the operation $+$ is a cyclic group for any $n \in \mathbb{N}$, with 1 as generator. Also the group of integers modulo prime $\mathbb{Z}_p^*$ (see section 2.1.6) for any prime p forms a cyclic group. As example $\mathbb{Z}_5^* = \langle 2 \rangle = \{2^k : k \in \mathbb{N}_0\} = \{2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8 = 3\} = \{1, 2, 3, 4\}$.

2.1.4 Rings

A ring is a tuple $(R, +, *)$ composed by a set R and the two operations addition $(+)$ and multiplication $(*)$ satisfying the following ring axioms:

1) $(R, +)$ forms an abelian group. Thus the addition $+$ is commutative, i.e., $a + b = b + a \quad \forall a, b \in R$
2) $(R, *)$ is associative, i.e., $(a * b) * c = a * (b * c) \quad \forall a, b, c \in R$
3) There is an identity element for $(R, *)$, denoted by 1, i.e., $a * 1 = 1 * a = a \quad \forall a \in R$
4) The distributive law holds for $(R, +, *)$, i.e., $(a + b) * c = a * c + b * c$, $a * (b + c) = a * b + a * c \quad \forall a, b, c \in R$

The ring is additionally called commutative ring or abelian ring if the multiplication $*$ is also commutative, i.e., $a * b = b * a \quad \forall a, b \in R$ [MK18].

As an example, the integers $\mathbb{Z}$ with the operations $+$ and $*$ form an abelian ring.

2.1.5 Fields

A field $\mathbb{F}$ is a tuple $(F, +, *)$ composed by a set F and the two operations addition $(+)$ and multiplication $(*)$, satisfying the following axioms:

1) $(F, +, *)$ forms a ring
2) $(F \setminus \{0\}, *)$ forms an abelian group. Thus the multiplication $*$ is commutative, i.e., $a * b = b * a \quad \forall a, b \in F \setminus \{0\}$

As an example, the rational numbers $\mathbb{Q}$ together with addition and multiplication form a field.

Finite Fields Finite fields are fields that contain only a finite number of elements.

The set of integers modulo prime $\mathbb{Z}_p$ (see section 2.1.6) together with modular addition and modular multiplication always form a finite field [Fis11]. This special field is denoted by $\mathbb{F}_p = (\mathbb{Z}_p, +, *)$. This kind of finite field is often used in cryptography, i.e., for elliptic curve cryptography.

An example of the finite field $\mathbb{F}_5$ is the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ together with modular addition and multiplication.

2.1.6 The integers modulo n

The **integers mod n** contain $\mathbb{Z}_n = \{0, 1, 2, ..., n-1\}$ for a $n \in \mathbb{N}$, that are all natural numbers that form a group with the *addition mod n*.

The **multiplicative integers mod n** contain $\mathbb{Z}_n^* = \{i \in \mathbb{Z}_n : i \text{ and } n \text{ are } coprime\}$, that are all natural numbers that form a group with the *multiplication mod n*.

The set of **multiplicative integers mod p** where p is prime, has special properties that are useful for cryptography [Len20].

The set contains the elements $\mathbb{Z}_p^* = \{1, ..., p-1\} = \mathbb{Z}_p \setminus \{0\}$, since it contains all smaller natural numbers that are coprime to p. Thus $|\mathbb{Z}_p^*| = p - 1$.

$\mathbb{Z}_p^*$ forms an abelian group with the modular multiplication. Often the group is denoted the same way as the set, namely $\mathbb{Z}_p^* = (\mathbb{Z}_p^*, *)$. Each element of $\mathbb{Z}_p^*$ has an inverse that can be calculated using the extended euclidean algorithm. Additionally it is a cyclic group, that means it contains at least one generator $g \in \mathbb{Z}_p^*$ holding $\mathbb{Z}_p^* = \{g^k : k \in \mathbb{N}\}$.

As mentioned in the previous section, $\mathbb{Z}_p$ together with modular multiplication and modular addition forms the finite field $\mathbb{F}_p$, because $(\mathbb{Z}_p, +, *)$ forms a ring and $(\mathbb{Z}_p^*, *)$ forms an abelian group.

2.2 Arithmetic in n-dimensional euclidean space

This section summarizes important measures and calculations in n-dimensional space R^n. All definitions are from [Fis11] and [MK18] if not explicitly mentioned.

2.2.1 Vectors

A vector or column vector $v \in R^n$ consists of one column with n rows. It can be imagined as a matrix $V \in R^{n \times 1}$. Its elements are denoted as v_j, where $j \in \{0, ..., n\}$.

$$v = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

Row Vectors A vector $v \in R^n$ can be transposed into a row vector $v^\intercal$ that has one row and n columns. This vector can be represented as matrix $v^\intercal = V \in R^{1 \times n}$.

$$v^\intercal = \begin{pmatrix} v_0 & v_1 & v_2 & \dots & v_n \end{pmatrix}$$

A row vector $v^\intercal$ can be transposed to a column vector $(v^\intercal)^\intercal = v$. Unless explicitly mentioned, with "vector" this thesis always refers to a column vector.

Vector Addition, Scalar Multiplication and Multiplication Vector arithmetic works similar as matrix arithmetic on a matrix $A \in R^{n \times 1}$. Regarding vectors $v, w \in R^n$, operations are made as defined in section 2.2.2 with $v = V \in R^{n \times 1}$ and $w = W \in R^{n \times 1}$ resulting in a one-column matrix, thus again in a vector.

Scalar Product The scalar product or dot product is another name for the multiplication operation between vectors. For details see the former paragraph. It is written as $\langle v, w \rangle = v * w$ for $v, w \in R^n$.

Length of Vector The norm, magnitude or length of a vector $v \in R^n$ is defined by: $\|v\| = \sqrt{\langle v, v \rangle} = \sqrt{{v_0}^2 + {v_1}^2 + ... + {v_n}^2}$

Distance of Vectors The distance between two vectors $v, w \in R^n$ is the length of the difference vector $\|v - w\|$.

Linear Independence The vectors $v\_1, v\_2, .., v\_k \in R^n$ are linearly independent if holds:

$$a_1 * v\_1 + a_2 * v\_2 + ... + a_n * v\_k = 0^n \quad \text{, only for } a_i = 0 \in R$$

All coefficients have to be 0 in order to reach the origin. Otherwise $v\_1, v\_2, .., v\_k$ is called linearly dependent.

In other words, the vectors $v\_1, v\_2, .., v\_k$ are linearly independent exactly when none of them can be represented as a linear combination of the others. If two vectors are the same ($v\_1 = v\_2$) or a multiple of each other ($v\_1 = a * v\_2,\ a \in R$) they can not be linearly independent. If there are more vectors in a set than dimensions in space, thus $k > n$, the set of vectors can not be linearly independent.

2.2.2 Matrices

A matrix $A \in R^{n \times m}$ consists of n rows and m columns. It's elements are denoted as a_{ij}, where $i \in \{0, ..., m\}$ is the elements column and $j \in \{0, ..., n\}$ is the elements row.

$$A = \overbrace{\left.\begin{pmatrix} a_{00} & a_{10} & a_{20} & \dots & a_{m0} \\ a_{01} & a_{11} & & & a_{m1} \\ a_{02} & & \ddots & & a_{m2} \\ \vdots & & & & \vdots \\ a_{0n} & a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}\right\}}^{\text{m}} \text{n}$$

Identity Matrix The identity element of matrix multiplications is the identity matrix, denoted as $Id_n \in R^{n \times n}$. It's diagonal elements are 1, the rest is 0:

$$id_{ij} = \delta_{ij}\ \forall i, j \text{ , with } \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

$$Id_n = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & & & 0 \\ 0 & & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

Transposed Matrix Given matrix $A \in R^{n \times m}$, it's transposed matrix is denoted as $A^\intercal \in R^{m \times n}$ and defined by: $a^\intercal_{ij} = a_{ij} + b_{ij} \ \forall i, j$.

Matrix Addition Two matrices $A \in R^{n \times m}$, $B \in R^{x \times y}$ can be added if they have the same dimensions $n = x$ and $m = y$. For the addition $A + B = C$ the elements of the same position are added: $c_{ij} = a_{ij} + b_{ij} \ \forall i, j$. If R forms a group with addition A has an additive inverse $-A = -1 * A$ and subtraction is possible.

Modular Matrix Addition Two matrices $A \in R^{n \times m}$, $B \in R^{x \times y}$ can be added modulo $q \in \mathbb{N}$ if they have the same dimensions $n = x$ and $m = y$. For the modular addition $A + B \mod q = C \in R_q^{n \times m}$ the elements of the same position are added: $c_{ij} = a_{ij} + b_{ij} \mod q \ \forall i, j$. If R forms a group with modular addition and subtraction is possible.

Matrix Scalar Multiplication A scalar $s \in R$ can be multiplied to a matrix $A \in R^{n \times m}$, changing every element of A by the factor s. This is called scaling of a matrix. The scalar multiplication $s * A = B \in R^{n \times m}$ is defined as: $b_{ij} = s * a_{ij} \ \forall i, j$.

Modular Matrix Scalar Multiplication A scalar $s \in R$ can be multiplied to a matrix $A \in R^{n \times m}$ modulo q, changing every element of A by the factor s. The modular scalar multiplication $s * A \mod q = B \in R_q^{n \times m}$ is defined as: $b_{ij} = s * a_{ij} \mod q \ \forall i, j$.

Matrix Multiplication Two matrices $A \in R^{n \times m}$, $B \in R^{x \times y}$ can be multiplied if they have the same dimensions $m = x$, meaning A has the same number of columns as B has rows. Therefore the operation, in contrast to matrix addition and scalar multiplication, are not commutative. The multiplication $A * B = C \in R^{n \times y}$ is defined as: $c_{ij} = \sum_{k=1}^{m} a_{ik} * b_{kj} \ \forall i, j$.

Modular Matrix Multiplication Two matrices $A \in \mathbb{Z}^{n \times m}$, $B \in \mathbb{Z}^{x \times y}$ can be multiplied modulo $q \in \mathbb{N}$ if they have the same dimensions $m = x$, meaning A has the same number of columns as B has rows. It is not commutative. The multiplication $A * B \mod q = C \in \mathbb{Z}_q^{n \times y}$ is defined as: $c_{ij} = \sum_{k=1}^{m} a_{ik} * b_{kj} \mod q \ \forall i, j$.

Basis Matrix A matrix $B \in \mathbb{Z}^{n \times m}$ is called basis matrix, if firstly all its columns are linearly independent and secondly they span the whole space R^m. This means every point in R^m can be reached by combining and scaling the column vectors. Hence for basis matrices it always holds $m = n$.

Length of a Matrix The length of a matrix $A \in R^{n \times m}$ is the norm of its longest column: $\|A\| = max_i(\|a_i\|)$.

Determinant of a Matrix The determinant of a matrix $A \in R^{n \times n}$ is the volume of the parallelepiped spanned by the columns a_i in the space R^m.

$$det(A) = a * d - b * c \quad \text{for } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

For calculating $det(A)$ for higher dimensional matrices see the Laplace expansion of a determinant [Fis11].

The determinant indicates many properties of a matrix, i.e., its sign indicates the orientation of the vectors, or it implies if a matrix is invertible.

For any matrix $A \in R^{n \times n}$ it holds: $det(A) = det(A^\intercal)$.

Rank of a Matrix The (column) rank r of a matrix $A \in R^{m \times n}$ is the number of it's linearly independent columns. Therefore $0 < r \leq n$. If the rank $r = n$ the matrix is called full-rank, meaning its columns are all linearly independent.

Inverse of a Matrix If for a matrix $A \in R^{n \times n}$ it exists an $A' \in R^{n \times n}$, so that $A * A' = Id_n$, A is invertible. A' is the inverse of A and is denoted as A^{-1}.

Not all matrices are invertible, but the following rules hold for matrix $A \in R^{n \times n}$:

- A has full-rank $\leftrightarrow$ A is invertible
- $det(A) \neq 0 \leftrightarrow A$ is invertible
- since $det(A) = det(A^\intercal)$: A is invertible $\leftrightarrow$ $A^\intercal$ is invertible

To calculate the inverse of a matrix A, a linear equation system for $A * A^{-1} = Id$ can be constructed, and solved for A^{-1} with Gaussian elimination [Fis11].

Matrices as linear equation systems A matrix $A \in R^{n \times m}$ with rank r can represent a linear equation system, where n is the number of equations and m is the number of variables.

Given the linear equation system below, it can be solved with gaussian elimination to $x = 2, y = 1, z = 3$.

$$\begin{aligned} 3x + 4y - z &= 10 \\ x - 9y + 2z &= -1 \\ -x + 5y &= 3 \end{aligned}$$

This equation can be written as matrix multiplication $A * v = b$. v contains the value of the variables, that solve the equation.

$$\underbrace{\begin{pmatrix} 3 & 4 & -1 \\ 1 & -9 & 2 \\ -1 & 5 & 0 \end{pmatrix}}_{A} * v = \underbrace{\begin{pmatrix} 10 \\ -1 \\ 3 \end{pmatrix}}_{b} \qquad \text{with } v = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

A linear equation system can only be uniquely solved if it is consistent and there are at least as many linearly independent equations as variables, hence for matrices $n \geq r \geq m$ is required for $A * v = b$ to be solvable.

To solve the equation $A * v = b$, if only A and y are given, the inverse A^{-1} has to be computed and multiplied on the left of b. $A^{-1} * b = A^{-1} * A * v = v$, this way the variables can be retrieved. The inverse A^{-1} can be calculated through constructing $A * A^{-1} = Id$, and solving the corresponding linear equation system with Gaussian elimination. And for this reason A has to be full-rank, to be invertible. It can be seen that more equations, and therefore more rows m of a matrix, leave the equation system solvable as long as it has full rank r.

If $m \geq r > n$ the equation system is underdetermined, and up to $|R|^{n-m}$ many solutions can be found. The calculation of the inverse A^{-1} is not possible any more, and v can not be calculated. Yet it is possible to calculate various left inverse $A_{left}^{-1} \in R^{m \times n}$ solving $A_{left}^{-1} * A = Id_m$, explaining why many solutions of the equation can be found [Tan17].

If the results of the equation, hence the elements of b, are manipulated with small errors the equation system will be inconsistent and not solvable either.

3 Cryptographic Foundations

Information Security defines three main goals, *Confidentiality*, *Integrity* and *Authenticity* [Sch20]. Cryptography enables these goals. Confidentiality through encryption, integrity through hash functions and authenticity through digital signatures.

This section gives a short introduction to the security of cryptography and the preliminaries on its underlying mathematical problems. Further it provides an overview of cryptographic primitives, including hash functions, encryption and digital signatures.

3.1 Information Theoretic Security vs. Computational Security

In cryptography, **information theoretical security** or **perfect secrecy** holds if, given a cipher, it is not possible to make any assumptions about possible underlying plaintexts. This means that the probability that the cipher C originates from the actual plaintext P must be equal to the probability that the plaintext P was encrypted in general [Sha49]. They are statistically independent.

$$Pr(P|C) = Pr(P)$$

The cipher must not reveal any information about the encrypted plaintext, it must not even limit the range of possible plaintexts. The probability of a cipher must be independent of the plaintext. For the set $\mathcal{P}$ of all possible plaintexts $Pr(C|P_1) = Pr(C|P_2)\ \forall P_1, P_2 \in \mathcal{P}$ must hold [Buc16].

An important condition to achieve information theoretical security is a uniform distribution of possible encryption keys [Sha49]. The One-Time Pad (OTP) is a well known perfectly secret encryption system [Buc16]. Thereby uniformly distributed random bit strings are used as secret keys, and added modulo 2 to the binary message, i.e., a bitwise exclusive OR is performed. Given a cipher C it is easy to see that for every possible plaintext P a key exists, which would have generated the given cipher C. The probability $Pr(C|P)$ is therefore equal for each possible plaintext, and the OTP is information theoretically secure.

Since the required keys must be as large as the plaintext, and each key is used only once, the OTP has no real practical use. Yet modern computationally secure cryptography, such as lattice-based cryptography, still uses the basic idea of OTPs and Shannon's Theory.

Computational security or **semantic security** holds if it is theoretically possible to reconstruct information about the plaintext P from the cipher C but the complexity of obtaining this information is too high to be technically feasible [GM84]. It is computationally hard (see chapter 3.2). In this case the difference between $Pr(C|P_0)$ and $P(C|P_1)$ is negligible regarding security, because nevertheless no adversary can compute information from the cipher.

For this reason modern cryptography is based on hard mathematical problems, as for example the Discrete Logarithm Problem (DLP). The cryptographic keys used can be derived, but only with almost infinite computational power. Accordingly, the cryptographic schemes stay secure with the current computational power and satisfy computational security.

3.2 Computationally Hard Problems

All cryptography is based on mathematical problems that are computationally hard to a certain degree, i.e., not solvable in polynomial time. One distinguishes between worst-case and average-case hardness [ZG15].

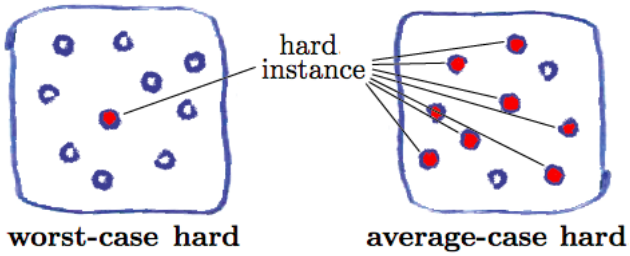


Figure 3.1: Distribution of hard instances in worst-case and average-case problems

Worst-case Hardness Let a problem be given for which there is at least one instance that is not solvable in polynomial time, then this problem is at least **worst-case hard**. However, worst-case hardness of a problem allows all other instances of the problem to still be efficiently solvable. Worst-case hardness is therefore easy to prove, but is not sufficient for cryptography.

An example of such a problem would be the factorization of numbers in general. Even if the factorization of the product of two prime numbers is hard, factorization is possible for more than half of all numbers, just think of the multiples of 2. For this reason a random module n instead of a prime for RSA encryption would not be secure.

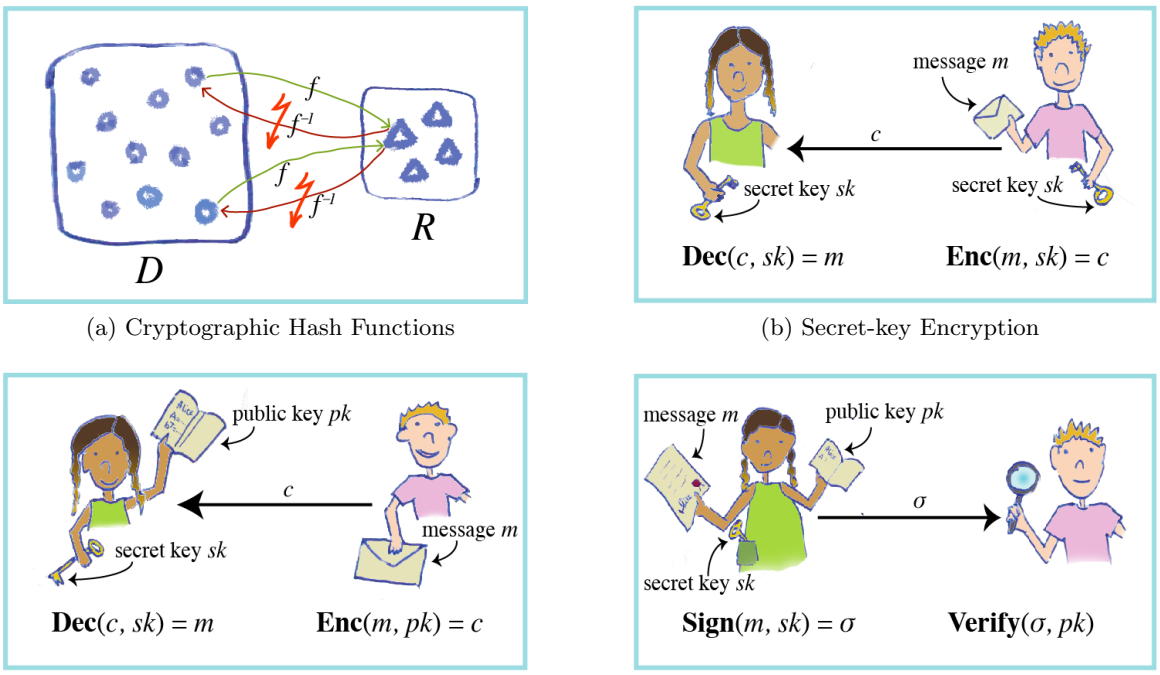
Average-case Hardness Hence, cryptography is based on the class of problems that are at least **average-case hard**. For these problems an instance is not solvable in the average, in other words, a randomly chosen instance of the problem is hard.

An example of this would be the factorization of the product of large prime numbers, as it is used for RSA encryption.

Proving that a problem is average-case hard is much trickier than proving worst-case hardness with one instance. The best approach is to prove the average-case hardness of a problem to be based on the worst-case hardness of another problem, and then prove the worst-case hardness of the other problem by showing it with one instance. This way the average-case hardness of LWE for lattices was proved [Sti19].

3.3 Cryptographic Primitives

As mentioned in the introduction of this chapter, cryptography is used to obtain information security, also nicely explained in [Sch20]. In the following a short classification of the most common cryptographic primitives and their associated security goal is given. Note that there exists a lot more than mentioned here. Figure 3.2 provides an overview of the introduced cryptographic functions.



(a) Cryptographic Hash Functions

(b) Secret-key Encryption

(c) Public-key Encryption

(d) Digital Signatures

Figure 3.2: Cryptographic Primitives

Collision-resistant Hash Functions Collision-resistant hash functions, in cryptographic context often just referenced as **hash functions**, are functions that map a sometimes even infinitely large domain D to a smaller value set R, i.e., $|D| > |R|$ [KM17]. Additionally, two inputs mapping to the same output, so called *collisions*, are hard to find. Thus, given a collision-resistant hash function $f : D \longrightarrow R$, then there is no polynomial algorithm that finds $x, y \in D$ for which holds $f(x) = f(y)$.

Collision-resistant hash functions are used to provide **integrity**. That means, a piece of digital information cannot be manipulated afterwards. For example, to ensure the integrity of a message m, the hash $h = f(m)$ is calculated and appended to the message. The message m is certainly not manipulated, as long as the the result of $f(m)$ is still equal to the appended h, due to the collision resistance of f. An adversary cannot find another message m' with the same hash and pretend that m' was the original message, because m and m' would have to be collisions and they are hard to find.

Symmetric and Asymmetric Encryption Encryption enables **confidentiality** of information. Encrypted information, a so-called *cipher*, can be decrypted to the original *plaintext* only by owners of a secret key and no one else. There are two types of encryption, symmetric and asymmetric [KM17].

For **symmetric encryption** the sender and receiver have the same secret key. This is used by the sender to encrypt the plaintext and by the receiver to decrypt the cipher to the original plaintext. A disadvantage of the method is that the shared secret key must be shared between all participant beforehand over some secret channel.

For **asymmetric encryption** two complementary keys are used for encryption and de-

cryption, forming the so-called asymmetric *key pair*. To encrypt a message, the receiver's public key is used. The resulting cipher can then only be solved with the corresponding secret key of the receiver and by no one else. This method has the advantage that no secret is necessary to encrypt a message. The public key, as the name implies, may be revealed to the public, yet the cipher remains confidential.

Digital Signatures To support **authentication** of the origin of digital information **digital signatures** are used. With their help, the author of a digital message can be identified [KM17].

One example for digital signatures is constructed with asymmetric key pairs. To *sign* a message m, the author encrypts it with his secret key and appends the cipher to the message. The cipher is the digital signature of m. All recipients of the signature can *verify* it by decrypting the cipher with the sender's public key. If the resulting plaintext matches the received message, it was really authored by the owner of the public key. In practice not the entire message m is encrypted and used as a signature but, the collision-resistant hash of the message $f(m)$ is generated and used for signing and verification. This limits the length of the signature to a fixed value and reduces computation. Because of the integrity of the hash, the authentication of the sender remains secure.

A disadvantage of using asymmetric key pairs is, that the ownership of the public key must be ensured. This is also done via a digital signature of a trusted party, which can only be ensured, if the trusted parties public key is valid too. Again the public key of the trusted party has to be verified with a further signature from a higher trusted party and so on. This creates a signature chain, a so-called *chain of trust*, which must be verified all the way to its root [Per99].

Digital signatures based on asymmetric key pairs and a chain of trust are not the only existing schemes to provide authentication. Another concept are identity-based signatures, which are a main part of this thesis, introduced in chapter 4.2.

4 Background

This chapter provides the background upon which chapters 5 and 6 are based. Because the goal of this thesis is to adapt the efficient key revocation mechanism of Key Updatable Signature Schemes (KUSS) from elliptic curves to lattice-based Identity-based Signatures (IBS), this chapter starts with an introduction to Elliptic Curve Cryptography. Next, the idea of Identity-based Cryptography is explained. This chapter also contains related work on key revocation in IBS schemes. Finally, the concept of efficiently revocable KUSS based on elliptic curves is presented. Lattice-based cryptography is presented in an own chapter 5.

4.1 Elliptic Curves

Elliptic curves are an old construct from mathematics, but only in the 1980s the first cryptography on elliptic curves was invented independently by Neil Koblitz [Kob87] and Victor S. Miller [Mil86]. The Elliptic Curve Discrete Logarithm Problem (ECDLP) within such curves enables modern public-key cryptography, such as the key encapsulation mechanism ECDH [Nat18] or the digital signature scheme ECDSA [Nat23a]. The advantages of Elliptic Curve Cryptography (ECC) are smaller key lengths compared to RSA [RSA78] achieving equal security, and simple computations together with low memory usage [ABC+18].

To understand the transformation of an elliptic curve IBS into a KUSS, knowledge of arithmetic on elliptic curves and elliptic curve Schnorr signatures is given.

As basic introduction to ECC [Cor15] is recommended. When not differently stated the definitions are form [ZG15] and [Bro09].

4.1.1 Definition

An elliptic curve $E_{\mathbb{F}}$ is defined over a field $\mathbb{F}$ of characteristics different than 2 and 3, and consists of the set $(x, y) \in \mathbb{F}^2$ which satisfy the following equation, with $a, b \in \mathbb{F}^2$ and $4a^3 + 27b^2 \neq 0$ [ZG15].

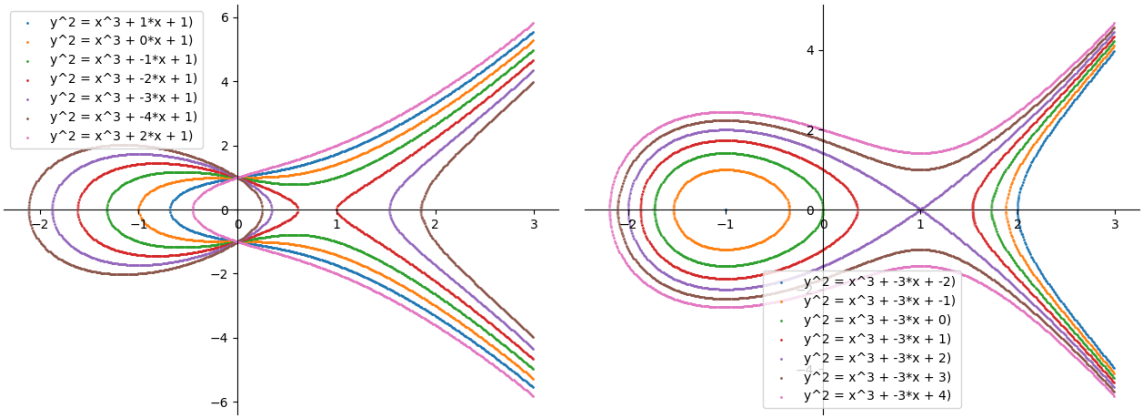
$$\{(x, y) \in \mathbb{F}^2 : y^2 = x^3 + ax + b\} \subseteq \mathbb{F}^2$$

In addition, each elliptic curve contains the *point at infinity* $\mathcal{O}$. Thus an elliptic curve E is defined as follows [ZG15]:

$$E = \{(x, y) \in \mathbb{F}^2 : y^2 = x^3 + ax + b\} \cup \mathcal{O} \text{ with } a, b \in \mathbb{F} \text{ and } 4a^3 + 27b^2 \neq 0$$

An elliptic curve forms an abelian group (see section 2.1). It contains an identity element $\mathcal{O}$. For each element there exists exactly one inverse element. And the associative and commutative addition is defined over it.

Figure 4.1 shows different examples of elliptic curves over $\mathbb{R}$.



(a) a varying from 2 to -4, $b = 1$

(b) $a = -3$, b varying from -2 to 4

Figure 4.1: Examples of elliptic curves

Arithmetic of Elliptic Curves

This section gives a short introduction to the arithmetic on elliptic curves. Because only a basic understanding of elliptic curves is necessary for this thesis, all operations are explained geometrically only. This is sufficient for the following concepts.

Inverse Element Each element of an elliptic curve has exactly one inverse. The inverse $-P$ of a point P is its reflection on the x-axis. See figure 4.2a.

Identity Element The identity element of an elliptic curve is the point at infinity $\mathcal{O}$, which can be thought of as ∞ of the y-axis. A straight line passing through a point P and its inverse $-P$ always intersects also $\mathcal{O}$. The inverse of $\mathcal{O}$ is again $\mathcal{O}$.

$$-\mathcal{O} = \mathcal{O}$$

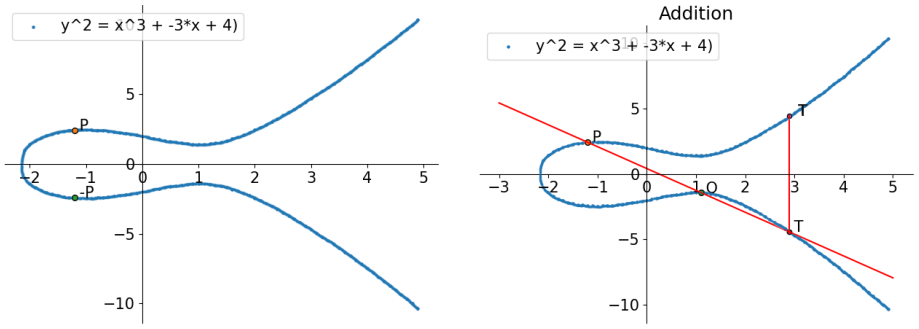
Addition The addition of two points of an elliptic curve can be imagined geometrically as follows. If the sum of point P and point G is wanted, then one draws a line through both points, and for all possible points $P, G \in E$ there is at most one third intersection point of this line with a point T of the elliptic curve. The sum $P + G = -T$, with $-T$ being the third point of intersection of the line with E, mirrored on the x-axis. Geometrically it can be seen that this operation is commutative, i.e., $P + G = G + P$. See figure 4.2b.

This definition is accompanied by a few special rules. First, when a point P is added to $\mathcal{O}$, the result is equal to P, like for any neutral element of a group.

$$P + \mathcal{O} = \mathcal{O} + P = P$$

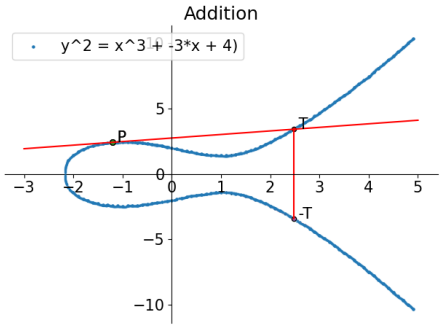
Second, if a point P is added to its inverse $-P$, then the resulting straight line is perpendicular, and runs in an abstract sense through the point at infinity $\mathcal{O}$. For this $-\mathcal{O} = \mathcal{O}$ holds, so the result is $\mathcal{O}$.

$$P - P = -P + P = \mathcal{O}$$

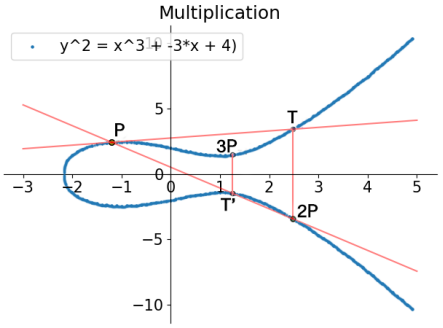


(a) P and it's inverse $-P$

(b) $P + G$



(c) $P + P$



(d) $3 * P$

Figure 4.2: Elliptic curve arithmetic on the curve $y^2 = x^3 - 3x + 4$ over $\mathbb{R}$

And the last special case is the addition of a point P with itself. There are infinitely many possibilities how a line could run through the point P, so a special one, the tangent is used. When drawing the tangent of the elliptic curve in point P, there is again only at most one further intersection point T with the elliptic curve, and $-T$ is the result of the addition. See figure 4.2c.

Scalar multiplication It is also possible to multiply a factor n onto a point P of an elliptic curve. This means an n-fold addition with the point P itself. See figure 4.2d.

$$n * P = \underbrace{P + P + ... + P}_{\text{n- times}} = \sum_{i=0}^{n-1} P$$

4.1.2 Elliptic curves over finite fields

Elliptic curves can also be defined in discrete, finite contexts. Thereby the definition of an elliptic curve over a finite field $\mathbb{F}_p$ is

$$E = \{(x, y) \in \mathbb{F}_p^2 : y^2 \equiv x^3 + ax + b \pmod p\} \cup \mathcal{O} \text{ ,with } a, b \in \mathbb{F} \text{ and } 4a^3 + 27b^2 \not\equiv 0 \pmod p$$

For finite fields $\mathbb{F}_p$ is $(\mathbb{Z}_p, +, *)$ with prime p see chapter 2.1.5.

Examples of modular elliptic curves are shown in figure 4.3. It can be seen that the set contains only single points and does not form a geometric curve anymore. Furthermore, the modular elliptic curve is no longer reflected at $y = 0$, but at $y = \frac{p}{2}$. And the larger p gets, the larger is the order of the elliptic curve, and more messy it seems.

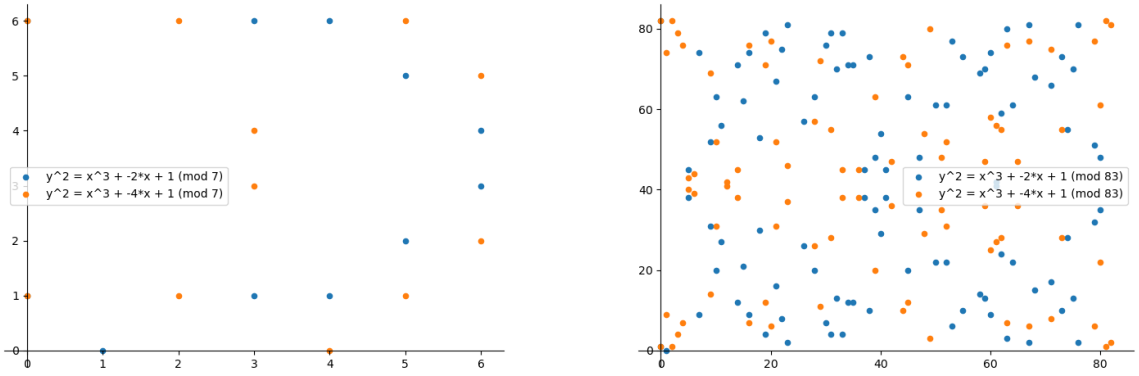


Figure 4.3: Examples of Elliptic Curves over the finite field $\mathbb{F}_7$ (left) and $\mathbb{F}_{83}$ (right)

Arithmetic of Elliptic Curves in $\mathbb{F}_p$

The point at infinity remains the identity element of the elliptic curve, and can still be imagined as ∞ of the y-axis. The inverse, addition, and scalar multiplication on modular elliptic curves must be geometrically imagined in a different way.

Inverse The inverse $-P$ of a point P is its reflection at the constant $y = \frac{p}{2}$. See figure 4.4a.

Addition To add the points $P = (x_P, x_P)$ and $G = (x_G, y_G)$, again a line is imagined through the two points, but here with a magnitude $m = (y_P - y_G) * (x_P - x_G)^{-1} \mod p$. This straight line is defined in the field $\mathbb{F}_p$, so it remains also in the modular space $\mathbb{Z}_p^2$. Again, this line only intersects one other point $T \in E$. The inverse of T is again the sum $-T = P + G$. See figure 4.4b.

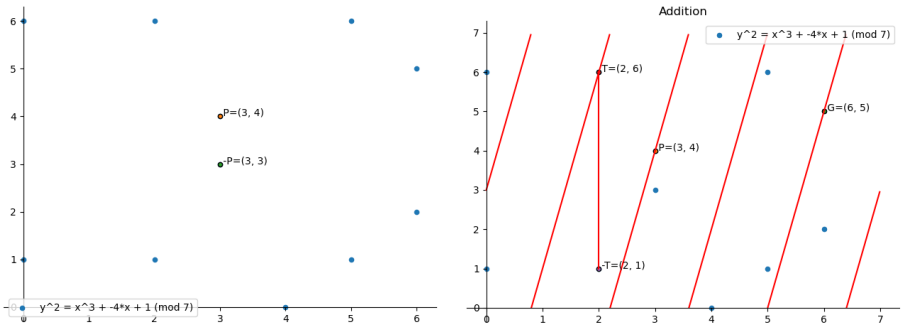
Besides it still holds $P + \mathcal{O} = \mathcal{O} + P = P$ and $P - P = -P + P = \mathcal{O}$.

But to add the point $P = (x_P, y_P)$ to itself, there exists no tangent of a point. Instead, a straight line is drawn through point P with a magnitude $m = (3{x_P}^2 + a) * (2y_P)^{-1} \mod p$, which is again defined in the field $\mathbb{F}_p$. The intersection of this line with the only other point $T \in E$ must be inverted to result in $-T = P + P$. See figure 4.4c.

Scalar multiplication Here the multiplication of a factor n on a point P of the elliptic curve is again the n-fold addition of the point with itself, but using the addition defined for modular elliptic curves.

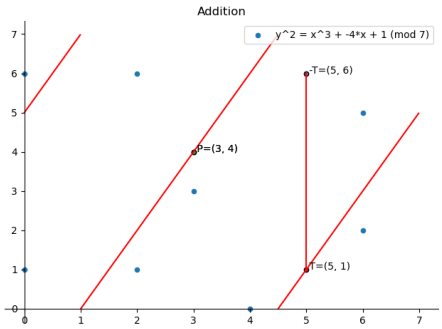
Cyclic Subgroups

Within an elliptic curve over finite fields, there are some points, which, added to themselves, sooner or later end up at themselves again. These points, and the points reached by them, form a cyclic subgroup $\mathbb{G}$ (see section 2.1). The base point is called generator G of this



(a) P and it's inverse $-P$

(b) $P + G$



(c) $P + P$

Figure 4.4: Elliptic curve arithmetic shown on the curve $y^2 = x^3 - 4x + 1$ over $\mathbb{F}_7$

cyclic subgroup, because from it all other points of the subgroup can be generated, denoted as $\langle G \rangle$.

$$\mathbb{G} = \langle G \rangle = \{k * G : k \in \mathbb{Z}\}$$

Figure 4.5 shows an elliptic curve over $\mathbb{F}_{37}$, with $a = -1$ and $b = 3$. Further it's cyclic subgroup generated by $G = (2, 3)$ is scattered. It can be seen, that all elements of the subgroup can be reached by G through scalar multiplications. Remember that the point at infinity $\mathcal{O}$ is also part of any cyclic subgroup of an elliptic curve.

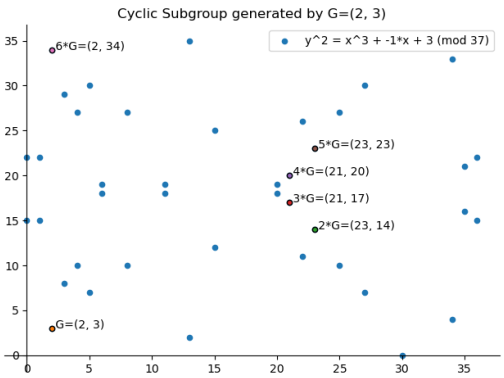


Figure 4.5: Cyclic subgroup of an elliptic curve over a finite field

The order n of a subgroup, is the number of points in the subgroup. This is the smallest positive integer, which when multiplied with the generator G, results in $\mathcal{O}$.

$$n * G = \mathcal{O}$$

To picture this, in figure 4.5, the order would be $n = 7$. The addition of the point $6P$ with P would be the vertical line running through the point at infinity, i.e $7P = 6P + P = \mathcal{O}$.

Thus the elements of a cyclic subgroup can be defined without duplicates by:

$$\mathbb{G} = \langle G \rangle = \{k * G : k = 0, ...n\}$$

Elliptic Curve Cryptography takes place within such subgroups of elliptic curves over finite fields.

4.1.3 Elliptic Curve Discrete Logarithm Problem

As could be seen, scalar multiplication is defined on modular elliptic curves. If a modular elliptic curve E and a point $P \in E$ and $x \in \mathbb{Z}$ is given, then $x * P = G$ is easy to calculate. However, the reverse calculation from G and P to the factor x is not easy for some modular elliptic curves. This problem is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP) problem, and is considered hard since so far no polynomial algorithm is known that can solve it [ZG15]. It is called discrete, because the computations are not defined on infinite domains, but on finite fields, more precisely on the cyclic subgroup with addition and multiplication defined.

In this way, this ECDLP of scalar multiplication on modular elliptic curves is similar to the Discrete Logarithm Problem (DLP) of modular exponentiation, on which also Diffie-Hellman key exchange [DH76] is based on. And similar to the DLP, cryptography can be constructed on top of the Elliptic Curve Discrete Logarithm Problem.

4.1.4 Elliptic Curve Schnorr Signatures

Elliptic Curve Cryptography uses cyclic subgroups of certain secure elliptic curves, such as recommended by the NIST in [Nat23a]. These elliptic curves are classified as secure meaning that the ECDLP problem is average-case hard, and cryptography can be based on them. A benefit of ECC is that, although not as efficient computable as RSA [RSA78], it needs much shorter signature key lengths to be secure. For achieving a 128 bit security ECC needs only 256 bits while RSA would have keys of 3072 bits length [ABC+18].

For elliptic curves a Schnorr signature scheme exists. The Schnorr signature scheme for cyclic groups in general was published in 1991 and allows efficient calculation because a part of the signature can be precomputed [Sch91]. It has been applied to cyclic groups in modular elliptic curves. Figure 4.6 shows the overall procedure of the scheme. The relevant domain parameters $D = (p, a, b, G, n = q)$ include the curve parameters a, b and prime module p, as well as a generator point G and the order n of the cyclic subgroup generated by G. This order has to be prime and therefor is further denoted as q instead of n. The cyclic subgroup will be referenced to as $\mathbb{G}$. Additionally a hash function $H : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_q$ is used, that maps an elliptic curve point and a string to an element of $\mathbb{Z}_q = \{0, ..., q-1\}$.

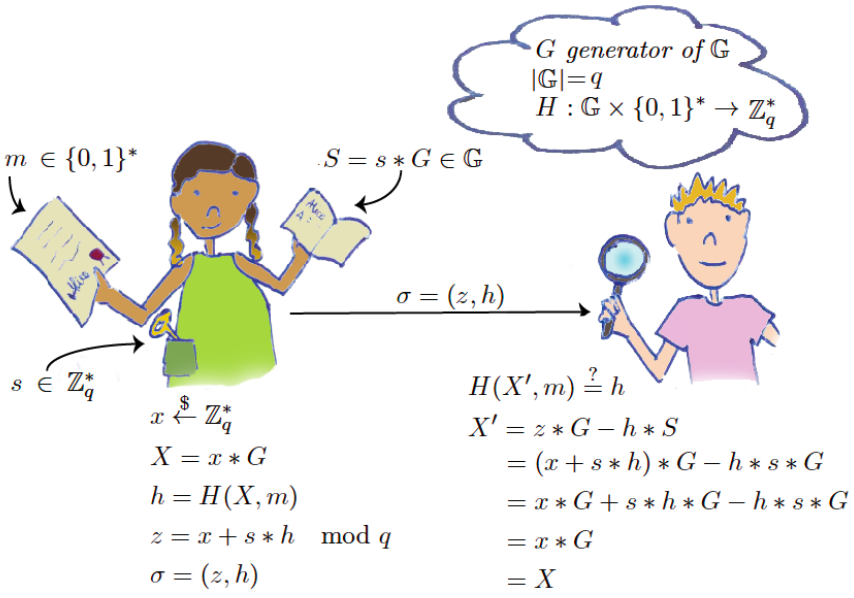


Figure 4.6: Schnorr signature with elliptic curves

As secret key $s \in \mathbb{Z}_q^* = \{1, ..., q-1\}$ is chosen, and the corresponding public key is calculated with $S = s * G \in \mathbb{G}$. According to ECDLP s is hidden in S.

$$\begin{aligned} &\textit{secret key} : s \in \mathbb{Z}_q^* \\ &\textit{public key} : S = s * G \in \mathbb{G} \end{aligned}$$

To sign a message $m = \{0,1\}^*$, a random $x \in \mathbb{Z}_q^*$ is chosen, and multiplied to the generator G. The result is a new point $X \in \mathbb{G}$ of the cyclic subgroup, from which x can no longer be derived. As the first part of the signature, the hash $H(X, m) \in \mathbb{Z}_q^*$ is generated, in which the message but not yet the author is included. Then this hash is multiplied with the secret key s and added to x. This result $z \in \mathbb{Z}_q$ is calculated from the message m, the integer x, and the secret key s of the author. The signature is the tuple $\sigma = (z, h)$.

$$\begin{aligned} &\textbf{Sign}(m) : \\ &x \xleftarrow{\$} \mathbb{Z}_q^* \\ &X = x * G \\ &h = H(X, m) \\ &z = x + s * h \mod q \\ &\sigma = (z, h) \end{aligned}$$

To verify the signature $\sigma = (z, h)$, the verifier reconstructs the X from the signing process, using the public parameters. Therefor the z of the signature is multiplied to the generator G and this result is added to the h of the signature multiplied to the signers public key S. The resulting point is denoted $X' \in \mathbb{G}$. If the hash of X' and m is equal to h the signature

is valid.

$$\begin{aligned}&\textbf{Verify}(\sigma):\\&X' = z * G - h * S\\&H(X', m) \stackrel{?}{=} h\end{aligned}$$

The verification works if the public key $S = s * G$, because then the following equation is true.

$$\begin{aligned}X' &= z * G - h * S\\&= (x + s * h) * G - h * s * G\\&= x * G + s * h * G - h * s * G\\&= x * G\\&= X\end{aligned}$$

This elliptic curve Schnorr signature scheme can be used to construct elliptic curve Identity-based Signatures. Therefor the next section introduces Identity-based Cryptography.

4.2 Identity-based Cryptography

This section introduces Identity-based Cryptography (IBC), consisting Identity-based Encryption (IBE) and Identity-based Signatures (IBS). Exemplarily the elliptic curve Identity-based Signature Scheme by Galindo and Garcia [GG09] is presented.

IBC is an asymmetric cryptosystem that uses the unique identity of the user instead of a user-chosen public key. That way the computationally intensive process of looking up a users public key by verifying a chain of certificates as it is required in PKIs, is omitted.

Identity-based Cryptography was first published by Adi Shamir in 1984 [Sha84]. This work presents the abstract idea of IBE and IBS, and proves the concept of IBS using a variant of the RSA cryptosystem [RSA78]. The first implementation of IBE was only discovered in 2001 independently by Boneh and Franklin [BF01] based on the Weil Pairing on elliptic curves, and by Cocks [Coc01] based on quadratic residues.

The usage of the identity as public key, and yet preventing adversaries to forge others identities, is achieved through a different underlying trust model than when using certificates to link an identity to a public key. For IBC the authenticity of a user is based on the trust in a central authority that issues the keys of all users. Only secret keys, that match the identity of a user and were issued by the PKG are valid.

Thus an identity-based cryptosystem differs to a PKI based one in this ways:

1. In IBC there is a central authority called Private Key Generator (PKG). Its function is to verify the identity of a user, and then generate and distribute the individual secret key for all users. It guarantees the trustworthiness of all users.

2. Instead of a self generated key pair consisting of public key and private key, each user has a unique identity that he uses as public key, and receives a secret key matching this identity from the PKG. This identity can be any string and could range from an

IP or email address to an office number in a company. The only condition is that in the given context it is uniquely associated with a user.

There exist Identity-based Encryption (IBE) and Identity-based Signatures (IBS). In both cases the PKG uses given domain parameters to create its own asymmetric key pair during the so-called **setup** of the system. This is called the Master Key Pair (MKP), consisting of a Master Secret Key (msk) and a Master Public Key (mpk) with the latter one being public. The secret keys of all users will be based on this master key pair.

$$\textbf{Setup()} = (msk, mpk)$$

Furthermore, each user must have a unique attribute assigned to him, his so-called identity id. This id can also be a tuple of attributes.

4.2.1 Identity-based Encryption

Identity-based Encryption (IBE) is defined by four steps, the **setup**, and the **key extraction** performed by the PKG, and the **encryption** and **decryption** used by the users. Figure 4.7 gives an overview of IBE.

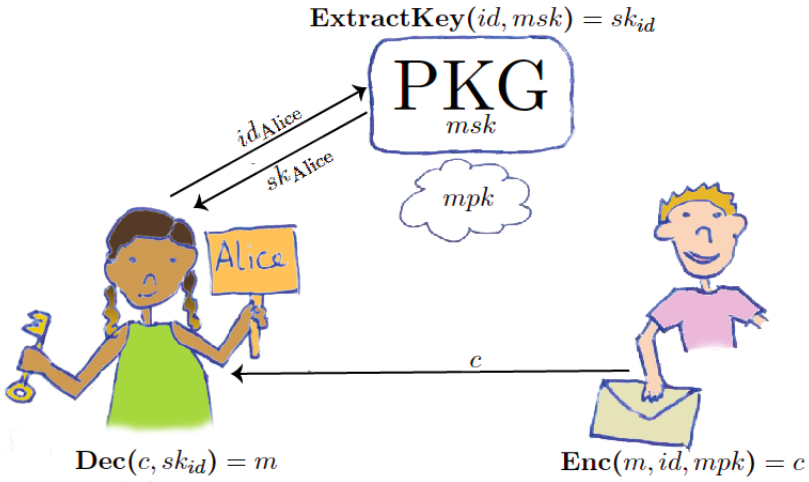


Figure 4.7: Identity-based Encryption

1. **Setup**: The PKG chooses the domain parameters and generates the MKP as described before.

2. **Key Extraction**: Each member of a group has a unique identity id. For encrypted communication a secret key sk_{id} is required that matches this id.

   The calculation of sk_{id} by the PKG is called key extraction. In this process, a user must authenticate himself, then the PKG uses the msk to create the secret key sk_{id} that matches the users id and sends it back to the user over a secure channel. The id and the received secret key sk_{id} complement each other to a asymmetric key pair.

$$\textbf{ExtractKey}(id, msk) = sk_{id}$$

For example the id of a user Alice could be her name, id_{Alice} = "*Alice*". First she authenticates herself towards the PKG. This can happen digitally, but also for example in person with an ID card. The PKG then computes a secret key sk_{Alice} from id_{Alice} with the help of the msk and returns it only to Alice.

3. **Encryption**: To encrypt a message m for another user his identity id and the public mpk is used. Instead of the public key in an asymmetric encryption scheme the id is converted from string to bytes.

$$\textbf{Enc}(m, id, mpk) = c$$

Regarding the example, Bob can encrypt his message m with Alice's id_{Alice} and send the resulting cipher c to her.

4. **Decryption**: To decrypt a cipher c, the recipient must use his secret key sk_{id}.

$$\textbf{Dec}(c, sk_{id}) = m$$

Alice can decrypt the cipher c using her sk_{Alice} resulting in the original message m. The decryption will only work if her id_{Alice} was used for encryption and if her sk_{Alice} was generated with the msk matching the during encryption used mpk. In other words sk_{Alice} has to been issued by the PKG.

4.2.2 Identity-based Signatures

Also Identity-based Signatures (IBS) can be constructed and are defined by four steps, the **setup** and the **key extraction** performed by the PKG the same way as for IBE, and the **signing** and **verification** used by the users. The process is illustrated in Figure 4.8.

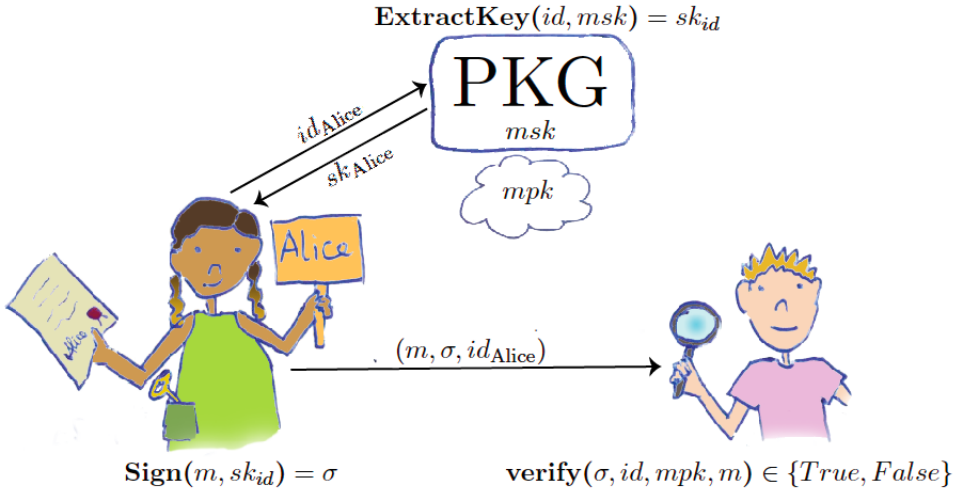


Figure 4.8: Identity-based Signatures

1. **Setup**: This is the same as for IBE.
2. **Key Extraction**: This is the same as for IBE.

3. **Signing**: A user uses his secret key sk_{id} for signing. With his sk_{id} he encrypts a hash of his message m. The result is the signature σ.

$$\textbf{Sign}(m, sk_{id}) = \sigma$$

Assuming Alice wants to sign a message m, she encrypts the hash $h(m)$ with her secret key sk_{Alice}, receiving the signature σ.

4. **Verification**: To verify a signature σ was made for message m by user with id, additionally the mpk is needed to prove, that the user is trusted by the PKG. The verifier decrypts σ using the identity id and the mpk. If resulting plaintext of σ and $h(m)$ are the same he accepts the signature as valid. Of course verification can be successful only if the sk_{id} used to generate the signature is matching id and if it was generated by the msk corresponding to the used mpk.

$$\textbf{verify}(\sigma, id, mpk, m) \in \{True, False\}$$

In case Bob wants to verify Alice's signature σ of the message m, he first computes the hash $h(m)$. Then he decrypts σ using the mpk and id_{Alice}. If this result has the same value as $h(m)$, then he accepts the signature as valid. It really was generated by Alice and signs the message m.

4.2.3 IBS with Elliptic Curves

Different to many other paring-based IBS on elliptic curves like [BF01] and [BLMQ05], D. Galindo and F. Garcia [GG09] designed an Identity-based Signatures scheme in 2009, adapting Schnorrs signatur scheme [Sch91]. It uses a Schnorr signature of a user's id, generated by the PKG, as user secret key. The user then signs by creating a Schnorr scheme of his message together with the signature of the PKG. This procedure is therefore called concatenated Schnorr signature. Figure 4.9 shows the overall procedure.

For the IBS, the PKG creates a key pair for elliptic curve Schnorr signatures, as presented in section 4.1.4. In addition to the domain parameters $D = (p, a, b, G, n = q)$ of ECC, two hash functions are agreed on.

$$\begin{aligned}
&\textbf{Setup()}: \\
&\quad \text{msk} \ : msk \in \mathbb{Z}_q^* \\
&\quad \text{mpk} \ : MPK = msk * G \in \mathbb{G} \\
&\quad H_1 : \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_q \\
&\quad H_2 : \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_q
\end{aligned}$$

To create a user's private key sk_{id}, the PKG generates a random integer $r \in \mathbb{Z}_q^* = \{1, ..., q-1\}$, and multiplies this to the generator G. The result point $R = r*G \in \mathbb{G}$ becomes part of the secret key sk_{id}. To hide the id in the key, first the hash of R together with the id is generated using H_1, then multiplied with msk and added to r. The result $s = (r+H_1(R, id)*msk) \in \mathbb{Z}_n^*$ is also part of the secret key sk_{id}. As can be seen, the secret key is similar to a Schnorr signature of the id generated with the master secret key. The secret key consists of the tuple $sk_{id} = (s, R)$.

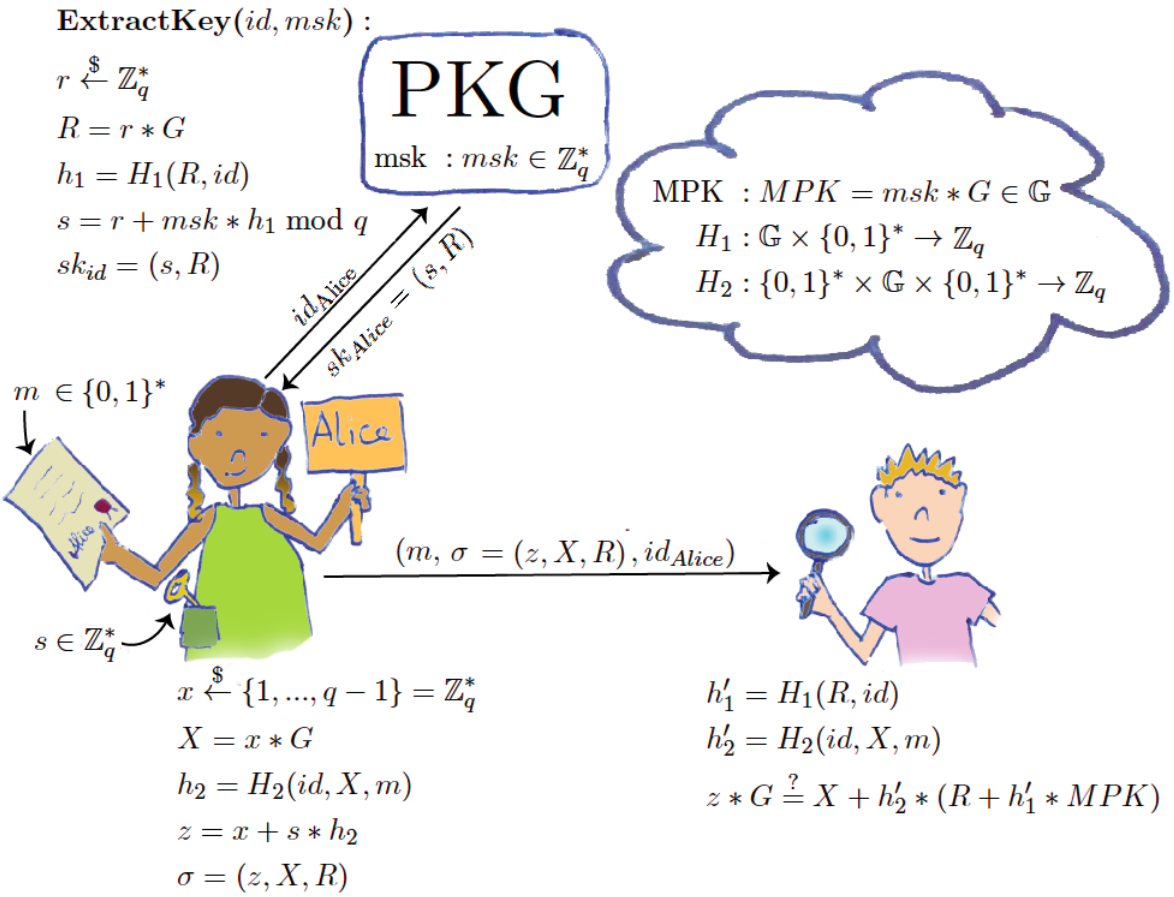


Figure 4.9: Galindo-Garcia Identity-based Signature on elliptic curves

ExtractKey(id, msk) :

$r \xleftarrow{\$} \mathbb{Z}_q^*$

$R = r * G$

$h_1 = H_1(R, id)$

$s = r + msk * h_1 \mod q$

$sk_{id} = (s, R)$

The signature of a message $m \in \{0,1\}^*$ is similar to the key extraction. An $x \in \mathbb{Z}_{||}^*$ is sampled and multiplied on the generator G, resulting in a new point X, from which x can no longer be derived according to ECDLP. Then, using H_2, a hash of the id, the point X and the message m onto $\mathbb{Z}_n$ is generated. Following the Schnorr scheme, this hash h_2 is multiplied to the integer s of the secret key and added to x. Thus the resulting z contains information about the x, the id, the message m and the s of the secret key $sk_{id} = (s, R)$.

The signature consists of the tuple $\sigma = (z, X, R)$.

$$
\begin{aligned}
&\textbf{Sign}(m, sk_{id}) : \\
&x \xleftarrow{\$} \mathbb{Z}_q^* \\
&X = x * G \\
&h_2 = H_2(id, X, m) \\
&z = x + s * h_2 \mod q \\
&\sigma = (z, X, R)
\end{aligned}
$$

Note that the part R of the secret key is allowed to be exposed to the public. It thus serves a similar function as a classic public key, but does not have to be authenticated by a certificate chain. Instead, its authenticity can be verified with the help of the MPK.

To verify the signature $\sigma = (z, X, R)$, first the two hash functions used during the key extraction and signing procedure are recalculated, using the id of the supposed author and parts of the signature. The result h_1' is multiplied to the MPK and added to the received R. This result is multiplied with h_2' and then added to the point X. The signature is valid if this result is equal to $z * G$.

$$
\begin{aligned}
&\textbf{Verify}(\sigma, id, MPK, m) : \\
&h_1' = H_1(R, id) \\
&h_2' = H_2(id, X, m) \\
&z * G \stackrel{?}{=} X + h_2' * (R + h_1' * MPK)
\end{aligned}
$$

The verification works if the message m during signing and verification was the same and if the id used to generate the secret key sk_{id} is the same as the id used to sign and the id used to verify the signature. In other words, it verifies, that the real author of the message is the believed author. Otherwise the hash functions would not result in the same integers and the following equation would no longer hold.

$$
\begin{aligned}
X + h_2' * (R + h_1' * MPK) &= X + H_2(id, X, m) * (R + H_1(R, id) * MPK) \\
&= X + h_2 * (R + h_1 * MPK) \\
&= X + h_2 * (r * G + h_1 * msk * G) \\
&= X + h_2 * (r + h_1 * msk) * G \\
&= x * G + h_2 * s * G \\
&= (x + h_2 * s) * G \\
&= z * G
\end{aligned}
$$

If a user is not trusted by the PKG anymore, there has to be a mechanism to make this verification equation not true anymore. This is discussed in the next section.

4.3 Related Work on Revocable Identity-based Signatures

In cryptography, key revocation means that a key should no longer function, i.e., signatures with this key should no longer be valid because the verification is no longer successful. This

may be necessary if a user is not trusted anymore, or if the signing key of a trusted user is leaked, and thus its generated signatures are not trustworthy anymore.

Contrary to PKI based signatures, where validity of participants is verified through certificates and certificate revocation lists, IBC relies on mathematical revocation mechanisms that cause the verification process to fail [BGK08]. Because the public key of a user is his unique *id*, and therefor can not be chosen newly after compromise, this is necessary. This mathematical revocation is complex for IBS, because signing keys are justified by the Master Key Pair (MKP) and are valid as long as they match the current MKP. Thus no single signing key can be verified, without changing the MKP, which then causes a necessary reissuing of all non-revoked signing keys to match the new MKP.

This chapter outlines how mathematical revocation can be achieved for IBS, leading to so-called Revocable Identity-based Signatures (RIBS).

A basic idea, introduced with the first Identity-based signature scheme by Boneh and Franklin, suggests to append a validity period to the identity string [BF01]. Depending on the particular use case this validity period can for example be the date. Then a signature is only valid if (id || $date$) works as id parameter during verification. If the signing key was issued for another date, thus the key extraction was performed for (id || $other\text{-}date$), the verification will automatically fail, without any communication between verifier and another instance needed. The validity period can be changed to any value, but has to be set for all IBS participants in advance, and all signing keys will automatically only work until end of the validity period. Thus the advantage of IBS not needing certificate verification remains. Nevertheless, as the main disadvantage of this procedure, the PKG has to issue new keys for all participants at every new validity period, e.g. daily. Additionally each user has to authenticate himself again, and a secure channel between the PKG and each user has to be ensured to distribute the new keys. Alternatively the PKG can distribute the new keys by encrypting them with the old master key pair and identity of the former validity period for each user and publishing it for all non-revoked users [BGK08]. In both cases the workload for the PKG, i.e., the generation and secure distribution of the keys increasing linearly with the number of non-revoked users, causes a huge bottleneck.

Another disadvantage is that a compromised key, even if known before, remains valid until the validity period expires, because without certificate authentication no knowledge-based revocation mechanisms are implemented. Furthermore, the original owner cannot easily get a substitute for secure communication until the end of the validity period, such as generating a new public key in a PKI setting, because his ID is unique and he cannot just create any new unique identity until the time expires - since the valid identity for signing remains unique. Therefore the entire IBS setting would have to be updated to a new MKP.

In [BGK08] Boldyreva, Goyal and Kumar still use the idea of Boneh and Franklin to modify the identity to contain a validity period, but computational work of the PKG is decreased logarithmic to the number of non-revoked users without user-PKG interaction needed. To achieve this, they combine fuzzy IBE with binary tree data structures to remain at logarithmic complexity. In a nutshell, it uses a binary tree to assign attributes to each user, with those they can decrypt the fuzzy IBE. This way the computational work for calculating the new keys is only logarithmic to the number of non-revoked users, although individually encrypted. Yet, a computational overhead for the PKG, all in all larger than in the approach of Boneh and Franklin [TT12], remains.

Tseng and Tsai present in [TT12] how key revocation over public channels is possible. Therefore the private key of every user consists of two components, an initial secret key that will remain the same for all validity periods, and a time update key, which is periodically recalculated by the PKG and is allowed to be public, because it only complements the initial secret key to a valid key. Together they allow all features of IBC. This way, the need for an encrypted channel between the PKG and the users after the initial key distribution is eliminated, and computational effort for both the PKG and the users is decreased.

Guggemos introduces the idea of a Key Updatable Signature Scheme (KUSS), where again revocation is performed by updating the signing and master keys, but using only a single broadcast message for all signing key updates [Gug20]. Signing keys are newly issued for validity periods, but the generation of each new signing key is shifted to the individual user, remaining forward secure and secure after signing key exposures. Therefor the PKG selects a secret update token Δ and distributes it privately to all non-revoked users, using a binary tree data structure. Both the PKG and the users update their keys with respect to Δ, resulting in a valid IBS state without the revoked users. The size of the keys and the complexity of signing and verifying stays the same. The number of private channels for the transmission of the random parameter remains only logarithmic to the number of users, and the PKG's workload is decreased, since the key generation for each user is omitted [Gug20]. This procedure was conceptually proved on elliptic curves, and as central topic of this thesis, it is explained in more detail in the next section 4.4.

All in all key revocation in IBC is realized by issuing keys only for certain validity periods. The length of this periods can vary depending on the use case. To revoke a key, it simply will not be re-issued for the next validity period. There are various different approaches for distributing the new keys of a validity period, focusing on decreasing network load and computational workloads for the PKG or the users. The process of re-issuing the signing keys is also called key update.

4.4 Revocation with Key Updatable Signature Schemes

The in [Gug20] proposed Key Updatable Signature Scheme (KUSS) is a solution for key revocation in IBS schemes, where the computational load of the PKG is offloaded on to the users themselves. This way also the required network bandwidth is reduced, because not the whole keys have to be communicated to the users.

This section introduces the general concept of KUSS and then gives a concrete example by showing the Schnorr-like elliptic curve IBS from Galindo and Garcia converted to a KUSS.

4.4.1 Key Updatable Signature Scheme

As discussed in chapter 4.3, key revocation for IBS works by limiting the validity period of keys to a fixed period of time. This means that revoked members and leaked signing keys automatically lose their validity at the end of such a period. The mathematical procedure of verification simply does not work for them anymore, and the revoked members of the IBS do not receive new valid keys anymore. Unfortunately this forces the PKG to generate new keys for all non-revoked users at the beginning of each new validity period and distribute

them to each user via private channels. This leads to a computational bottleneck at sides of the PKG, and due to the encrypted distribution of all new keys also to higher network load.

The idea of KUSS was proposed by Guggemos in [Gug20] and presents a structure of transforming IBS that are constructed on cyclic groups in three steps into more efficient revocable Identity-based Signatures.

The transformation is based on the premise that the PKG chooses a random new master key pair when updating all keys at the beginning of a new validity period. A validity period or *epoch* is referenced with e, thus the next following epoch is $e + 1$.

Regarding IBS on cyclic groups $\mathbb{G}$, for key updates following the naive procedure of [BF01], randomly a msk_{e+1} is chosen from $\mathbb{Z}^*_{|\mathbb{G}|}$ and the master public key $mpk_{e+1} = msk_{e+1} * g$ is calculated by multiplying it to the generator $g \in \mathbb{G}$ of the group. Now, according to the Latin Square property [Gug20], choosing a random element of a group as the new msk_{e+1} is equivalent to choosing a random parameter Δ and multiplying it on a known element of the group, for example the msk_e. Both results are random. Due to the closure of a group, and the uniformly distributed probability of reaching any other element of a cyclic group, the result $\Delta * msk_e = msk_{e+1}$ is also a valid random new master secret key, equivalent to a totally randomly sampled one. In the case that the master public key is a product of the master secret key, multiplying Δ to mpk_e yields the to msk_{e+1} corresponding mpk_{e+1}, since $mpk_{e+1} = msk_{e+1} * g = \Delta * msk_e * g = \Delta * mpk_e$.

Update() :

$$\Delta \xleftarrow{\$} \{1, ..., |\mathbb{G}| - 1\} \text{ or } \mathbb{Z}^*_{|\mathbb{G}|}$$
$$msk_{e+1} = \Delta * msk_e$$
$$mpk_{e+1} = \Delta * mpk_e$$
$$usk_{e+1} = \Delta * usk_e$$

Although this new approach of not choosing the new master key pair completely randomly but updating it by a random parameter leads to the same result as the naive approach, this different procedure has a big advantage. Since the user signing keys usk, in many cryptosystems contain a product of the msk and the generator g, the parameter Δ is sufficient to update the usk_e as well, with $usk_e * \Delta = usk_{e+1}$. By distributing only the new parameter Δ to the non-revoked users, they can update their usks themselves to match the new master key pair. Consequently, the PKG no longer has to calculate the usks itself and then send the entire key encrypted to each user. It is sufficient if it only transmits the *update token* Δ to all non-revoked users. This can even be done logarithmically to the number of non-revoked users by using Logical Key Hierarchy (LKH) or Centralized Authorized Key Extension (CAKE) keying mechanisms [Gug20][GSK+18]. Because the products of Δ and the old keys remain elements of $\mathbb{G}$, the size of the msk, mpk and usks, and therefor the signature size, remains the same. Thus, the computational workload of the PKG and the network load are reduced, because signatures stay at same size while only the small update token $\Delta \in \mathbb{Z}^*_{|\mathbb{G}|}$ has to be broadcasted to all users, and no longer all different usks. A possible downside of the scheme is that the signature and verification, although remaining at the same complexity, may have to be changed slightly.

During this mechanism revocation is happening automatically. The users who shall no longer be part of the IBS, or who illegally use a leaked key, do not receive the update token Δ and therefore cannot calculate the correct usk_{e+1}. This is only holding if Δ can not be

derived from the public mpk_e and $mpk_{e+1} = \Delta * mpk_e$. For example for elliptic curves this is the case because of the ECDLP.

This is the underlying idea of KUSS, used to different extends depending on the properties of the originating IBS. In some cases, as for example for the IBS of Galindo and Garcia not all keys are updated, and therefore the gsk is still needed during verification.

Guggemos presents a technique of transforming an IBS on cyclic groups into a KUSS in three steps, all reducing the computational effort of the PKG and the network traffic. The first step is to transform the IBS into a Two Key Signature Scheme (2KSS), where every epoch a new symmetric group shared key gsk is additionally used during signature creation and signature verification. The second is to make updates of this gsk possible to limit trustworthiness of a key to epochs, resulting in an Updatable Two Key Signature Scheme (U2KSS). And the last step is the Key Updatable Signature Scheme (KUSS), where the gsk is included in the usk, mpk and msk, so that it is used passively during signing and verifying. Note that the use case of [Gug20] was managing group IBS, explaining why the shared secret is called group shared key. This thesis adapts his method for key revocation after epochs, by changing the gsk with every new epoch and treating the entire IBS users as group members.

Two Key Signature Scheme (2KSS) The idea is to prove validity of a user, by including a shared secret gsk into signing and verifying processes as extension to the asymmetric keys. This secret changes with every new epoch $e + 1$ and has to be distributed to all IBS users. Although it can be broadcasted, it has to be encrypted with at least logarithmic complexity to the number of non-revoked users. This way the PKG does not have to update the MKP and all usks and the former bottleneck is eliminated. An IBS is transformable into a 2KSS if the gsk is chosen in such way that it can be included in the mathematical signing process, resulting in a same sized signature as before. Also it must stay as secure as before.

Updatable Two Key Signature Scheme (U2KSS) To update the gsk, instead of distributing it newly for every epoch, an update token Δ is broadcasted. This way the security is increased because the gsk only has to be distributed once, and the network load during an update is decreased from the size of gsk to the size of Δ. The update token Δ is generated by an instance called Key Update Center (KUC) and privately send to the PKG and all non-revoked users. They update the group shared key $gsk_{e+1} = \Delta * gsk_e$. The broadcast message and the signing and verification process remain the same as for 2KSS. By updating the gsk with a secret Δ, forward security and post compromise security is achieved.

Key Updatable Signature Scheme (KUSS) To construct a KUSS from a U2KSS, the gsk is included in the asymmetric keys usk, mpk and optionally the msk. This way users can sign and verify without explicitly using the gsk, because it is included in the keys. It remains updatable by an update token Δ. By multiplying Δ to the keys, the same effect is achieved as if Δ was multiplied to gsk as done in the U2KSS, and then included in the keys. This enables verification without confidential knowledge of an gsk, because it is included in the public mpk. Further the distribution of the gsk to new users is obsolete, because the PKG includes it in the usk during key extraction. The transformability condition which must hold for this step is, that usk and gsk must be part of the signing process, and this part must be precomputed independently of the message. This way the PKG can include gsk in the usk

already during the *extract_key* process, and the gsk itself does not have to be distributed to the users. The same must hold for gsk and the mpk during verification, such that a mpk including the gsk can be used instead of both.

Following these transformation steps, four elliptic curve IBS schemes have been conceptually proven to be convertible to more efficient KUSS in [Gug20]. One of them is the IBS of Galindo and Garcia from section 4.2.3.

4.4.2 Example KUSS on Elliptic Curves

Galindo Garcia's elliptic curve IBS has been introduced in chapter 4.2.3, for an overview see figure 4.9. It is based on the hardness of ECDLP and uses Schnorr signatures on cyclic groups. This section explains how it can be converted into a KUSS according to the three steps from the previous section. In detail it is discussed how the key update works, and how the signature and verification are changed due to the parameter Δ.

2KSS To convert the IBS into a KUSS a symmetric key $gsk \in \mathbb{Z}_q^*$, shared between signer and verifier, must be used in addition to the asymmetric keys. During signature creation this key is multiplied to the former z, resulting in $z = gsk * (x + h_2 * s)$. It is also multiplied during the verification, resulting in $gsk * (X + +h_2' * (R + h_1' * MPK))$. The signature stays the same, $\sigma = (z, X, R)$.

The verification of the signature remains correct because the following equation is satisfied.

$$gsk * (X + h_2' * (R + h_1' * MPK)) = gsk * (x + h_2 * s) * G = z * G$$

Thus, the verification is valid as long as it would have been valid within the IBS and the same gsk was used for signing. As claimed, the security, signatures, and complexity of the procedures remain the same.

U2KSS To convert the 2KSS to a U2KSS, the gsk must be updatable by an update token Δ. Because $\mathbb{Z}_q^*$ is a cyclic group, a random $\Delta \in \mathbb{Z}_q^*$ can be multiplied to the gsk, with the new gsk behaving as if chosen randomly, according to the Latin Square Property. The signature and verification can remain as for 2KSS.

$$gsk_{e+1} = \Delta * gsk_e$$

KUSS The KUSS is constructed from the U2KSS by integrating the gsk into the msk for key extraction, into the usk for signing and into the MPK for verifying. This can then be updated by the token Δ. Figure 4.10 shows its overall procedure.

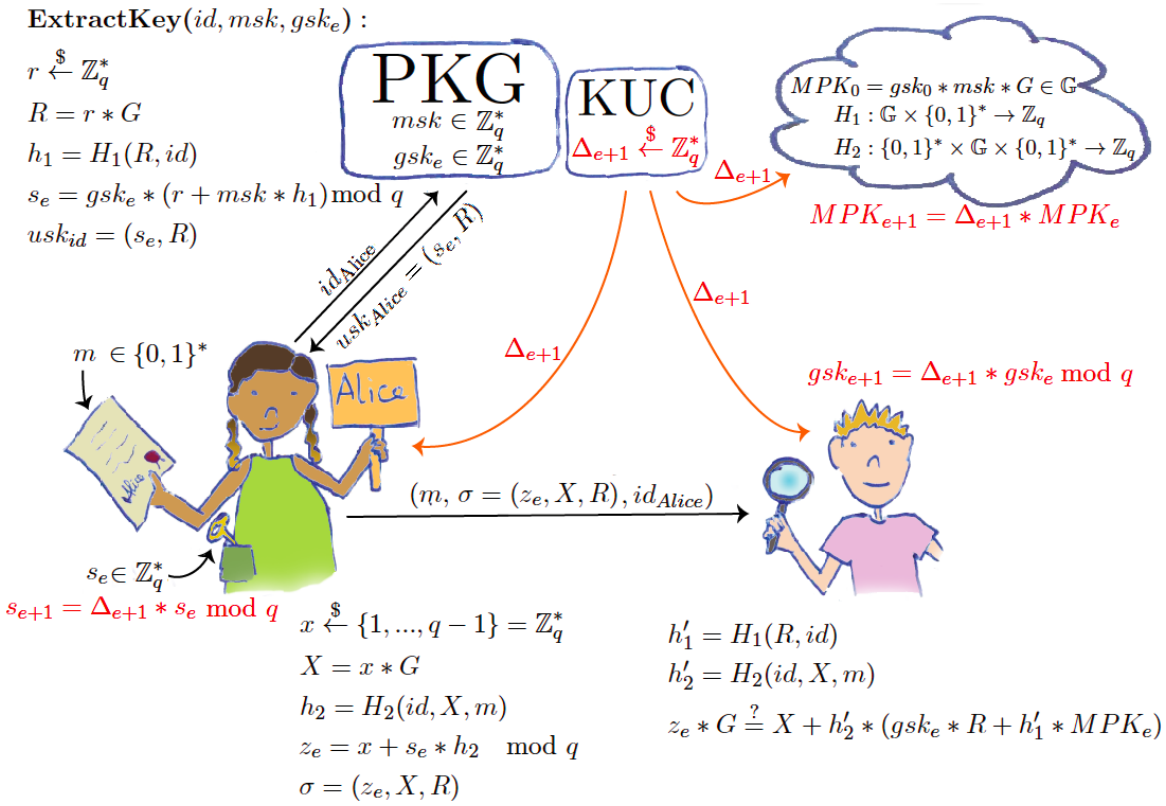


Figure 4.10: Elliptic curve KUSS based on the IBS by Galindo and Garcia (see figure 4.9). The key updates are marked in red.

To make the keys updatable for each epoch e, a $gsk_e \in \mathbb{Z}_q^*$ is included in the master public key, by defining it $MPK_e = gsk_e * msk * G$. In the case of this IBS the $msk \in \mathbb{Z}_q^*$ is not containing the gsk, because else verification will not work.

Setup() :

$$gsk_0 \xleftarrow{\$} \mathbb{Z}_q^*$$
$$\text{msk} : msk \in \mathbb{Z}_q^*$$
$$\text{mpk} : MPK_0 = gsk_0 * msk * G \in \mathbb{G}$$
$$H_1 : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_q$$
$$H_2 : \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_q$$

During key extraction the gsk_e is included in the usk, by multiplying it on the value of s, leading to $s_e = gsk_e * (r + msk * h_1)$. The usk is therefor the tuple $(gsk_e * s, R)$ and matches the secret gsk_e in the MPK_e.

ExtractKey(id, msk, gsk_e) :

$r \xleftarrow{\$} \mathbb{Z}_q^*$

$R = r * G$

$h_1 = H_1(R, id)$

$s_e = gsk_e * (r + msk * h_1) \mod q$

$usk_{id} = (s_e, R)$

At the beginning of each new epoch $e+1$ an update token $\Delta_{e+1} \in \mathbb{Z}_q^*$ is chosen randomly by the KUC, and distributed to the PKG and all users. According to the Latin Square property, the result of $\Delta_{e+1} * gsk_e = gsk_{e+1}$ is also a random element of $\mathbb{Z}_q^*$, and fulfills the same properties as a randomly chosen $gsk_e + 1$ would.

To update the gsk_e in the MPK_e, the PKG multiplies Δ_{e+1} to MPK_e, because $\Delta_{e+1} * MPK_e = \Delta_{e+1} * gsk_e * msk * G = gsk_{e+1} * msk * G = MPK_{e+1}$. Although MPK_{e+1} and MPK_e are public, the parameter Δ remains secret, since according to ECDLP the reduction of $MPK_e * \Delta = MPK_{e+1}$ to Δ is hard.

The users update the gsk_e in the s_e of their usk by multiplying Δ_{e+1} to s_e. This leads to $\Delta_{e+1} * s_e = \Delta_{e+1} * gsk_e * (r + msk * h_1)) = gsk_{e+1} * (r + msk * h_1)) = s_{e+1} \mod q$.

Update() :

$\Delta_{e+1} \xleftarrow{\$} \mathbb{Z}_q^*$

$gsk_{e+1} = \Delta_{e+1} * gsk_e \mod q \in \mathbb{Z}_q^*$

$MPK_{e+1} = \Delta_{e+1} * MPK_e \in \mathbb{G}$

$usk = (\Delta_{e+1} * s_e \mod q \in \mathbb{Z}_q^*, \quad R \in \mathbb{G})$

After the update all users now have valid keys, sufficiently equal to those that would have been calculated and distributed by the PKG with a completely randomly chosen MKP. The keys remain the same size, and the signing can be performed exactly as in the original IBS.

Sign(m, sk_{id}) :

$x \xleftarrow{\$} \mathbb{Z}_q^*$

$X = x * G$

$h_2 = H_2(id, X, m)$

$z_e = x + s_e * h_2 \mod q$

$\sigma = (z_e, X, R)$

The verification for this KUSS is modified slightly to use the gsk_e, unless the value of $gsk_e * R$ is appended to the signature, but this would change the signature size. Otherwise the gsk_e is automatically used, because it is part of the MPK_e.

Verify(σ, id, MPK, m) :

$h_1' = H_1(R, id)$

$h_2' = H_2(id, X, m)$

$z_e * G \stackrel{?}{=} X + h_2' * (gsk_e * R + h_1' * MPK_e)$

The verification is only possible if id and m are the same as during signing, and if the gsk in s_e, g_e and MPK_e is the same. Only then the following equation holds.

$$\begin{aligned} X + h_2' * (gsk_e * R + h_1' * MPK_e) &= X + H_2(id, X, m) * (gsk_e * R + H_1(R, id) * MPK_e) \\ &= X + h_2 * (gsk_e * R + h_1 * MPK_e) \\ &= X + h_2 * (gsk_e * r * G + h_1 * gsk_e * msk * G) \\ &= X + h_2 * gsk_e * (r + h_1 * msk) * G \\ &= x * G + h_2 * s_e * G \\ &= (x + h_2 * s_e) * G \\ &= z_e * G \end{aligned}$$

The benefit of this process is that computation and network traffic is reduced significantly, leading to an overall complexity of signing key revocation of 2 $log\ n$, with the PKGs former computation time of 253ms reduced to 138ms [Gug20]. Achieving such improvements by transforming other schemes into KUSS would be most valuable.

4.5 Summary

This chapter introduced the concept of Identity-based Signatures as alternative to certificate management, leading to the problem of expensive signing key revocation, and a smart solution was found in the Key Updatable Signature Scheme (KUSS). By introducing elliptic curve cryptography, the concept of IBS and KUSS was explained by examples with Schnorr-like IBS on elliptic curves.

Because of the benefits of Identity-based Cryptography and key revocation through KUSS the goal of this thesis is to convert a lattice-based IBS to a KUSS. Beside forward- and post-compromise security, this will additionally achieve quantum resistance, since lattice-based cryptography is still expected to be quantum safe.

The next chapter gives an overview about Lattices and introduces a couple of cryptographic primitives on lattices, starting with the simplest, and step by step leading to an IBS scheme. The transformability of this lattice-based IBS scheme into a KUSS will then be investigated.

5 Lattice-based Cryptography

This chapter is intended to provide a broad overview of lattice-based cryptography. Therefore, first the definition of lattices, their properties and their subgroup of q-ary lattices are presented. Next, their computational hard problems, which make cryptography on lattices possible in the first place, are outlined. Finally, a number of cryptographic primitives using lattices are presented, starting with hash functions, private and public key encryption, over trapdoor functions and digital signatures that are generated with them, up to IBE and IBS. The intention is to give the reader a feeling for the functionality and underlying security of lattice-based cryptography and to allow an easier step-by-step understanding of lattice-based IBS, which is important regarding the desired transformation into KUSS in chapter 6.

5.1 Lattices

The following chapter gives an overview on the mathematical concept of lattices, including it's definition, properties and problems. It is based on [CCSKK15], [ZG15] and [Pol11], whereby the last one is recommended as best introduction for beginners. Besides, Simons Institude provides good introduction videos to lattices with [Vai15b] and [Mic20].

Definition A lattice is a set of regularly arranged points in Euclidean vector space R^n. It is a discrete subgroup of $(R^n, +)$ and closed over addition and subtraction. Usually a lattice is defined by a matrix A. Here holds:

For a lattice Λ in n-dimensional space R^n, be given a set of n generating vectors $a_i \in R^n$. This set $\{a_1, ..., a_\mathrm{m} \in R^n\}$ is also written as a matrix $A \in R^{n \times m}$, where each column of A is a generating vector. If all vectors a_i are linearly-independent the matrix is also called basis matrix B. In this case $m \leq n$.

The lattice Λ is then defined by A as follows:

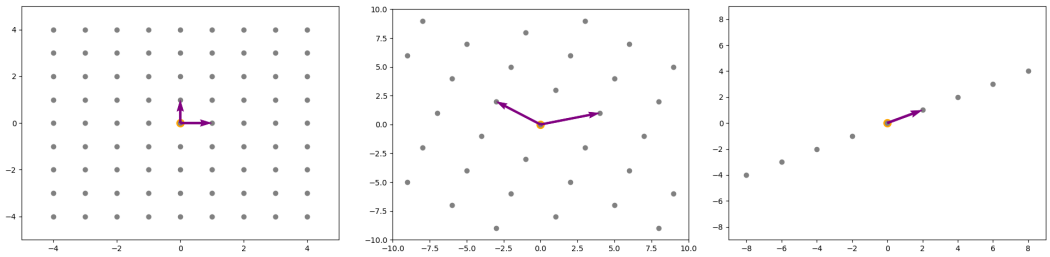
$$\Lambda(A) = \Lambda(a_1, ..., a_\mathrm{m}) = \Big\{ \sum_{i=1}^{m} x_\mathrm{i} a_\mathrm{i} \mid x_\mathrm{i} \in \mathbb{Z} \Big\} = \Big\{ Ax \mid x \in \mathbb{Z}^m \Big\}$$

A lattice Λ thus consists of all points that can be reached with an integer linear combination of its generating vectors.

The resulting lattice can be interpreted as a set of points or as a set of vectors. E.g., the point $(1, 4) \in \mathbb{Z}^2$ is equivalent to the vector $\begin{pmatrix} 1 \\ 4 \end{pmatrix} \in \mathbb{Z}^2$.

Figure 5.1 shows different lattices in two-dimensional space $\mathbb{Z}^2$. The basis vectors are represented by purple arrows, each element of the lattice by a gray dot. Notice that each lattice point can be reached by a combination of the basis vectors, and that all lattice points form a regular pattern.

As mentioned, a lattice is only a subgroup of an euclidean space R^n. It does not have to use the whole euclidean space, meaning it does not have to have the same number of



(a) $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ creates $\mathbb{Z}^2$ (b) $B = \begin{pmatrix} 4 & -3 \\ 1 & 2 \end{pmatrix}$ (c) $B = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

Figure 5.1: Different lattices $\Lambda(B) \subseteq \mathbb{Z}^2$, defined by different B.

dimensions as the space it is defined in. Thus the number of linear independent vectors in A can be smaller than n, i.e it always holds $rank(A) \leq n$.

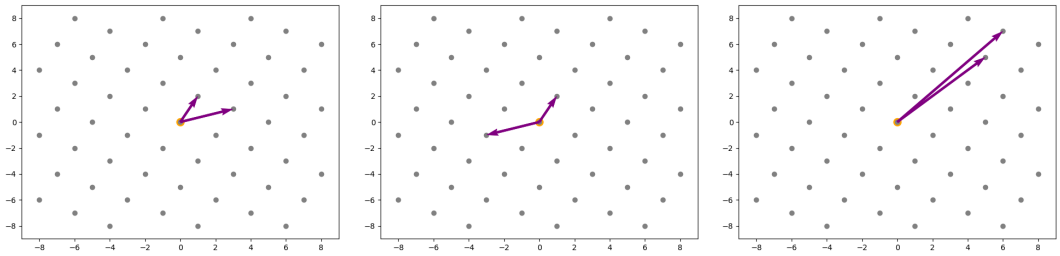
Figure 5.1c shows a lattice defined by a basis $B \in \mathbb{Z}^{2\times 1}$ over the space $\mathbb{Z}^2$, i.e., $m < n$. The basis $B \in \mathbb{Z}^{2\times 1}$. Thus the resulting lattice consists only of linearly dependent vectors, slangy one could say it consists of one linearly-independent vector.

The same lattices Λ can be described by an infinite number of different basis vectors. This means that the basis associated with a lattice is not unique.

The simplest way to generate a generating matrix A' for a given lattice $\Lambda(A)$ is to permute the generating vectors of Λ, which means to swap the columns of the matrix A, or to invert the direction of at least one vector, i.e., $a'_{\mathrm{i}} = -1 * a_{\mathrm{i}}$. Both ways the lattice Λ stays the same.

For a given basis matrix $B \in R^{n\times m}$ one can graphically select a new basis by selecting m lattice points, that span a parallelepiped, which does not contain further lattice points. By reducing the angle between the points, arbitrary long vectors can be chosen. Figure 5.2c might help imagining this.

Figure 5.2 shows the same lattice defined by different basis vectors.



(a) $B = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}$ (b) $B = \begin{pmatrix} 1 & 2 \\ -3 & -1 \end{pmatrix}$ (c) $B = \begin{pmatrix} 5 & 6 \\ 5 & 7 \end{pmatrix}$

Figure 5.2: The same lattice $\Lambda(B) \subseteq \mathbb{Z}^2$, defined by different B.

5.1.1 Properties of Lattices

Lattices have different properties that help to define problems over them. A selection of those being relevant for cryptography is given here.

Fundamental Region The fundamental region of a lattice is the parallelepiped generated by its basis vectors. It is spanned by the dotted grey lines in figures 5.9c and 5.9b.

Its volume Vol can be calculated with the determinant of B [Pol11].

$$Vol(Fundamental\text{-}Region_B) = |det(B)|$$

Given the same lattice Λ created by two different basis B_1 and B_2, the fundamental region is different, but its determinant and hence its volume is staying the same.

$$Fundamental\text{-}region_{B1} \neq Fundamental\text{-}region_{B2}$$
$$Vol(Fundamental\text{-}region_{B1}) = Vol(Fundamental\text{-}region_{B2})$$

The volume of the fundamental region can be used to proof that two lattices are not the same. Further it gives an orientation about the density of a lattice. The greater the volume of the fundamental region, the less dense is the lattice.

Minimum distance and Successive Minima The minimum distance of a lattice Λ is denoted by $\lambda_1(\Lambda)$ and is the length of the shortest non-zero vector in Λ. This is similar to the smallest distance between any two different vectors $v, w \in \Lambda$.

More general, the ith successive minimum $\lambda_i(\Lambda)$ is the smallest radius r around the origin containing i linearly-independent vectors in Λ.

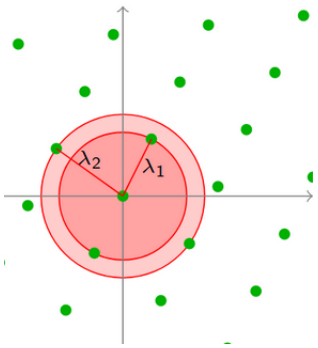


Figure 5.3: The minimum distance λ_1 and the 2nd successive minimum λ_2 of a lattice [Mic20]

For both, the minimum distance and the successive minima, no efficient algorithm is known for higher n-dimensional spaces. This is known as the Shortest Vector Problem (SVP) and discussed in section 5.2.

Distance Function Given a lattice $\Lambda \in R^n$, the distance of a point $t \in R^n$ and Λ is defined as the euclidean distance between t and the next closest lattice point $v \in \Lambda$ [Pol11]. This means if $t \in \Lambda$, then $dist(t, \Lambda) = 0$.

$$dist(t, \Lambda) = min(\|t - v\|) \quad v \in \Lambda$$

Covering radius The covering radius ρ of a lattice $\Lambda \in R^n$ is the largest distance a point $t \in R^n$ can have to any lattice point [GMR04].

$$\rho(\Lambda) = max(dist(t, \Lambda))$$

It can be geometrically imagined as spheres around every lattice point of which the radii are increased. The radius of the spheres at the moment when the whole space R^n is covered is the covering radius. It gives an orientation about the density of a lattice.

Dual Lattice The dual lattice Λ^* of a lattice $\Lambda \in R^n$ is defined as follows [Pol11]:

$$\Lambda^* = \{x \in span(\Lambda) \subseteq R^n : \forall v \in \Lambda,\ \langle x, v \rangle \in \mathbb{Z}\}$$

This means it is the set of all vectors x in $span(\Lambda)$ whose scalar $\langle x, v \rangle$ is an integer, for all $v \in \Lambda$. For example the dual of the n-dimensional integers $(\mathbb{Z}^n)^* = \mathbb{Z}^n$, because all scalars are integers as well.

Figure 5.4 shows a random two dimensional lattice in black and its dual in red. As you can see, the dual of a random lattice looks like a mess and accordingly it is hard to calculate.

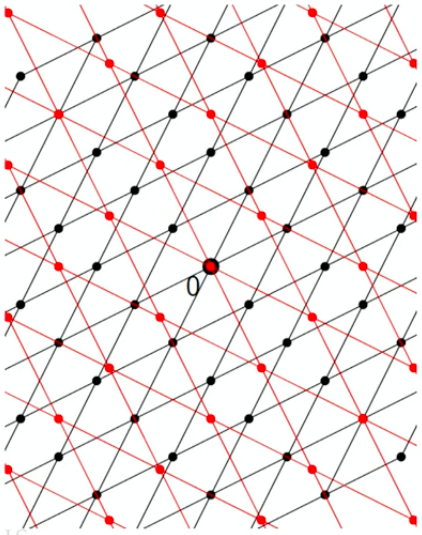


Figure 5.4: A random lattice and its dual [Mic20]

Here are the properties of the dual of a lattice Λ. To avoid confusion, here the symbol $\cdot$ is used instead of $*$ for multiplication.

- $(\Lambda^*)^* = \Lambda$
- the dual of a rotated lattice $Q \cdot \Lambda$ is $(Q \cdot \Lambda)^* = Q \cdot (\Lambda^*)$
- the dual of a scaled lattice $q \cdot \Lambda$ is $(q \cdot \Lambda)^* = \frac{1}{q} \cdot \Lambda^*$
- $\Lambda_1 \subseteq \Lambda_2 \iff {\Lambda_1}^* \supseteq {\Lambda_2}^*$
- If B is a basis of lattice Λ, then $(B^{-1})^\intercal = B^*$ is a basis of its dual lattice Λ^*.

A dual lattice Λ^* can be seen as something close to an inverse to Λ.

5.1.2 Q-ary Lattices

Cryptography is build on **q-ary lattices**, also called **modular lattices**. This q-ary lattices are defined on the integers, and therefor this section defines q-ary lattice not abstractly over the euclidean space R^n, but directly on $\mathbb{Z}^n$. Note that the definitions can be applied to any space with modular multiplication defined. For references see [MR09] and [Pol11].

Given $q \in \mathbb{Z}$ often chosen as prime number or prime power, a q-ary lattice Λ of dimension n satisfies the equation:

$$q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$$

This means any q-ary lattice is an infinite subset of the former introduced lattices $\Lambda \subseteq \mathbb{Z}^n$. And it contains at least all points in space that are multiples of q. In general, it can be said that if vectors $x, y \in \mathbb{Z}^n$ satisfy $x \equiv y \pmod q$, then $x \in \Lambda$ if and only if $y \in \Lambda$.

Given a matrix A it can generate two important q-ary lattices. One, where the kernel of A forms the q-ary lattice $\Lambda_q^\perp(A)$, the other q-ary lattice $\Lambda_q(A)$ contains the image produced by A.

Kernel q-ary Lattices This q-ary lattice $\Lambda_q^\perp(A) \subseteq \mathbb{Z}^m$, also called Ajtai lattice, is defined by a $q \in \mathbb{Z}$ and a basis matrix $A \in \mathbb{Z}_q^{n \times m}$ as followed. Note that the q-ary lattice is a subset of the space $\mathbb{Z}^m$, contrary to the non-modular lattices $\Lambda(A) \subseteq \mathbb{Z}^n$. Figure 5.5 shows the sizes of the matrices.

$$\Lambda_q^\perp(A) = \{x \in \mathbb{Z}^m : A * x = 0 \pmod q\}$$

This way the lattice is defined by the columns of the basis matrix. The q-ary lattice contains all vectors, which multiplied with the basic matrix result in a multiple of q. This means that the basis vectors are combined in such a way that the reached point is a multiple of q in each dimension. This 'combination' is a m-dimensional vector in the q-ary lattice set. Non square matrices always generate Λ and Λ_q in different spaces.

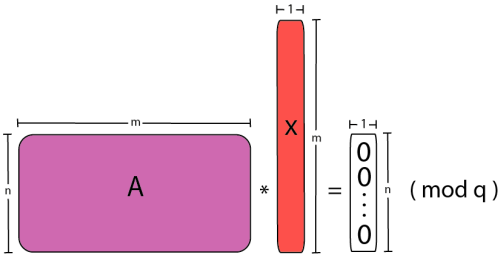
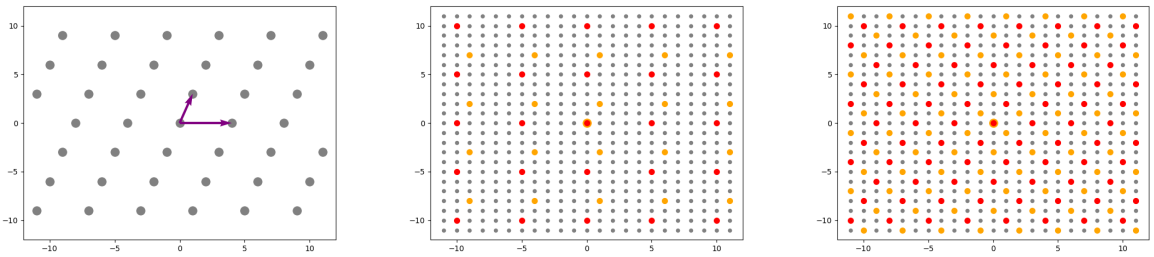


Figure 5.5: Matrix multiplications for calculating the q-ary lattice

For example, let $q = 6$ and $A = \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}$ be given. One vector in $\Lambda_q^\perp(A)$ could be $\begin{pmatrix} 7 & 8 \end{pmatrix}^\intercal$, calculated as follows.

$$\begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix} * \begin{pmatrix} 7 \\ 8 \end{pmatrix} = \begin{pmatrix} 36 \\ 24 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0 \pmod 6$$

Figure 5.6 shows examples for q-ary lattice $\Lambda_q^\perp(A)$ as red dots. The grey dots show $\mathbb{Z}^m$, which is the superset of any q-ary lattice.



(a) $\Lambda(A)$ (b) $\Lambda_5^{\perp}(A)$ (red) and $\Lambda_5^1(A)$ (yellow) (c) $\Lambda_6^{\perp}(A)$ (red) and $\Lambda_6^3(A)$ (yellow)

Figure 5.6: Kernel q-ary lattices for $A = \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}$

Similarly to the previous definition, this q-ary lattice can also be defined by including all vectors that create not multiples of q, but results that are exactly u greater than multiples of q.

$$\Lambda_q^u(A) = \{x \in \mathbb{Z}^m : A * x = u \pmod q\}$$

These are exemplarily marked as yellow dots in figure 5.6.

Image q-ary Lattices A q-ary lattice can also be defined using the rows of a matrix A. This $\Lambda_q(A)$ is defined by a $q \in \mathbb{Z}$ and the matrix $A \in \mathbb{Z}_q^{n \times m}$ as following. Again the lattice vectors are a subset of $\mathbb{Z}^m$ and not $\mathbb{Z}^n$. Figure 5.7 shows the sizes of the matrices.

$$\Lambda_q(A) = \{x \in \mathbb{Z}^m : x = A^\intercal * s \pmod q, \text{ for some } s \in \mathbb{Z}_q^n\}$$

Here the set of vectors is generated by the rows of A, by using $A^\intercal$ as basis to define a new lattice. The lattice set includes all vectors that are equivalent to this lattice's points modulo q.

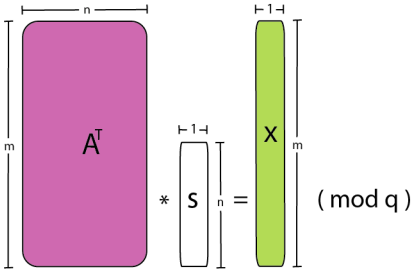
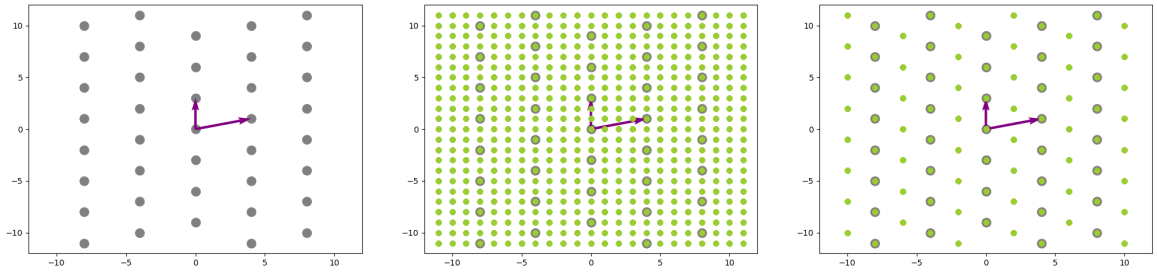


Figure 5.7: Matrix multiplications for calculating the q-ary lattice

Another way to write this would be $\Lambda_q(A) = \{x \in \mathbb{Z}^m : x \mod q = A^\intercal * s \mod q\}$. This means, this q-ary lattice Λ_q contains all vectors v that can be reached when the rows of the matrix are used as basis vectors, and its multiples $\mathbb{Z} * q * v$. Figure 5.8 shows how this subset can be imagined geometrically. The green dots are elements of $\Lambda_q(A)$.



(a) $\Lambda(A^\intercal)$ (b) $\Lambda_5(A)$ (c) $\Lambda_6(A)$

Figure 5.8: Image q-ary lattices with $A = \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}$

Again, let $q = 6$ and $A = \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}$ be given. As example we randomly show the calculation for $s = \begin{pmatrix} 2 & 1 \end{pmatrix}^\intercal$.

$$\begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix} * \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix} \pmod 6$$

This proofs, that $\begin{pmatrix} 3 & 3 \end{pmatrix}^\intercal$ is part of $\Lambda_6(A)$, as well as $\begin{pmatrix} 9 & -3 \end{pmatrix}^\intercal$ and the set $\mathbb{Z}^m * 6 + \begin{pmatrix} 3 & 0 \end{pmatrix}^\intercal = \{..., \begin{pmatrix} -9 & 3 \end{pmatrix}^\intercal, \begin{pmatrix} -3 & -3 \end{pmatrix}^\intercal, \begin{pmatrix} 3 & -3 \end{pmatrix}^\intercal, \begin{pmatrix} -3 & 3 \end{pmatrix}^\intercal, \begin{pmatrix} 3 & 3 \end{pmatrix}^\intercal, \begin{pmatrix} 9 & 3 \end{pmatrix}^\intercal, \begin{pmatrix} 3 & 9 \end{pmatrix}^\intercal, \begin{pmatrix} 3 & 15 \end{pmatrix}^\intercal, \begin{pmatrix} 9 & 15 \end{pmatrix}^\intercal, \begin{pmatrix} 15 & 15 \end{pmatrix}^\intercal, ...\}$, because they are all congruent modulo 6. This way a lattice is constructed, that will repeat after at least q numbers. The example calculation has to be repeated for all $s \in \mathbb{Z}_5^2$ to calculate the whole q-ary lattice.

Properties of q-ary lattices Q-ary lattices are used for cryptography, because they have certain properties, which are summarized here.

First, in average the kernel q-ary lattice is getting less dense for a larger q, while the image q-ary lattice is getting more dense.

Besides, for any lattice Λ the following equivalencies hold.

$$q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m \;\Leftrightarrow\; \Lambda = \Lambda_q(A) \text{ for some } A \;\Leftrightarrow\; \Lambda = \Lambda_q^\intercal(A) \text{ for some } A'$$

This means that all lattices that include $q\mathbb{Z}^n$ can be generated as q-ary lattice by some matrix $A \in \mathbb{Z}^{n \times m}$ as image, and with some other matrix $A' \in \mathbb{Z}^{n' \times m}$ as kernel, holding $\Lambda_q(A) = \Lambda = \Lambda_q^\intercal(A')$. To build the same lattice Λ, A and A' have to be different, because it holds that for any fixed A, the lattices $\Lambda_q(A)$ and $\Lambda_q^\intercal(A)$ are different. This can also be seen by comparing figures 5.6 and 5.8, that are both defined by the same A.

Further the q-ary lattices that can be constructed with the same A are q-scaled duals of each other:

$$\Lambda_q^\perp(A) = q * \Lambda_q(A)^*$$
$$\Lambda_q(A) = q * \Lambda_q^\perp(A)^*$$

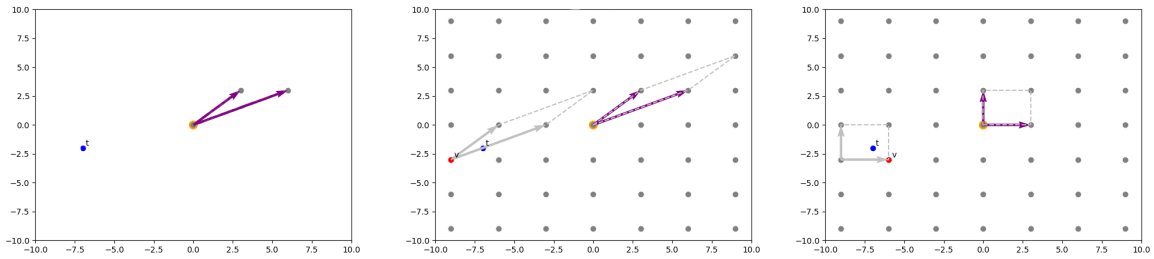
5.1.3 Lattice Bases in Cryptography

This section intends to give the reader an abstract idea of how lattices can be used for cryptography, before defining the problems in the following chapter.

Lattices contain an infinite set of vectors, thus it is not possible to list all elements of a lattice. That is why lattices are not defined by the whole set, but by a basis matrix $B \in \mathbb{Z}^{n \times m}$. Given this, theoretically every vector in the lattice is defined, but with increasing dimensions it is practically unfeasible to calculate the set of all combinations even in only a small range of $\mathbb{Z}^n$. Figure 5.9a gives an impression on how much information about a lattice is provided, if only some random basis vectors are given. It is hard to estimate where the lattice points are, and difficult to work out, what combination of the basis vectors is nearest to the point t. These are the fundamental problems that lattice-based cryptography is based on, and which will be described more formal in the next chapter.

For now it shall be pointed out, that these problems are differently hard depending on the given basis B.

For lattices it can be said, that long basis vectors 'hide' the structure of the real lattice and short vectors reveal the lattice. If a basis B is given it is easy to divide the space into the parallelepipeds that are spanned by B. Given a random point t in space, the surrounding parallelepiped can be calculated by solving the linear equation system $B * x = t$ and rounding the result x off, resulting in the edge point of the surrounding parallelepiped. The figures 5.9b and 5.9c show this parallelepiped in grey. To approximate the closest lattice vector v to the point t, the nearest corner point of the parallelepiped to t is chosen, marked in red. This procedure is the basic idea of Babais Rounding Technique [Gal12] [Bab86]. As can be seen in the figures 5.9b and 5.9c it solves the problem with an orthogonal and short basis, while it returns a wrong "closest" lattice vector when using the long basis.



(a) Space with random basis (b) Rounding with long basis (c) Rounding with short basis

Figure 5.9: The properties of lattice basis

To solve the problem of finding the closest lattice vector a short basis is good, while a long basis is bad. Solving this problem can be used for decryption. If information must be hidden in the space, a long vector is good and a short one is bad. This property is useful for public key encryption. Therefor both qualities are valuable for cryptography, if the problems are average-case hard.

5.2 Computational Problems on Lattices

In the following a few of the existing problems on lattices are introduced. In this context, first the variants of the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) are presented, because their worst-case hardness proves the required average-case hardness of the problems used for lattice-based cryptography. These are the surjective Short Integer Solution (SIS) problem and the injective Learning With Errors (LWE) problem.

Shortest Vector Problem (SVP) The Shortest Vector Problem searches for the shortest non-zero vector in a lattice. For higher dimensional lattices no polynomial time algorithm is known yet. It is NP-hard. The exact definition is the following [Mic16].

> Given a lattice $\Lambda \in R^n$, find a non-zero lattice vector $v \in \Lambda \backslash \{0\}$ such that $\|v\| \leq \|u\|$, for any $u \in \Lambda \setminus \{0\}$.

As every lattice is infinitely periodic, there are infinite vectors with shortest length. Also it shall be recalled, that the shortest length is the first successive minimum λ_1, hence the SVP asks for calculating the successive minima of a lattice.

γ-Approximation of SVP (SVP$_\gamma$) The γ-Approximation of SVP asks for a non-zero lattice vector with a length smaller than $\gamma * \lambda_1$, for a fixed $\gamma \geq 1$. This problem is a weakend definiton of SVP and has polynomial time solutions up to some γ-boundary. The exact definition is the following [Mic16].

> Given a lattice $\Lambda \in R^n$, find a non-zero lattice vector $v \in \Lambda \backslash \{0\}$ such that $\|v\| \leq \gamma * \|u\|$, for any $u \in \Lambda \setminus \{0\}$ and $\gamma \geq 1$.

Although the problem with $\gamma = 1$ (SVP) is NP-hard, for SVP$_\gamma$ it is possible to raise the limiting factor until a solution is found. Indeed it can be solved for $\gamma = (\frac{2}{\sqrt{3}})^n$ with n being the rank of the lattice [LLL82].

Decisional γ-Approximation of SVP (GapSVP) The Decisional γ-Approximation of SVP is not about searching for the shortest vector, but deciding whether there is a shortest vector. This decision is approximated for a parameter gamma. The exact definition goes as follows [Pol11].

> Given a lattice $\Lambda \in R^n$ a $r \in \mathbb{R}$ and $\gamma \geq 1$, return $True$ if $\lambda_1(\Lambda) \leq r$ and return $False$ if $\lambda_1(\Lambda) > \gamma * r$. If $r < \lambda_1(\Lambda) \leq \gamma * r$ $True$ and $False$ can be returned.

The GapSVP is NP hard for a constant γ.

Shortest Independent Vectors Problem (SIVP) The Shortest Independent Vectors Problem asks for the set of linearly independent vectors, that spans the whole space of the lattice, and of which none is larger than a given parameter. It is defined as follows [Pol11].

> Given a lattice $\Lambda \in R^n$ with rank d and $\gamma \geq 1$, find the set of linearly independent vectors $v_1, ...v_d$ satisfying $max(v_i) \leq \gamma * \|\lambda_d(\Lambda)\|$

This problem is weaker than calculating the d successive minima, because all d vectors can be smaller or equal the dth successive minimum, and none must be for example $\leq \gamma * \|\lambda_1(\Lambda)\|$. Besides, they do not have to form a basis of the lattice Λ. The SIVP is hard for $\gamma = n^{\frac{1}{log(logn)}}$ [BS99].

Closest Vector Problem (CVP) The Closest Vector Problem searches for the closest lattice vector to a given point in space. The exact definition is the following [Mic16].

> Given a lattice $\Lambda \in R^n$ and a target point t, find a lattice vector $v \in \Lambda$ such that $\|v - t\| \leq \|u - t\|$, for any $u \in \Lambda$.

Figure 5.9 illustrates the problem. There is no polynomial time algorithm known yet for solving this problem for higher dimensions. It is worst-case hard.

There exist **γ-Approximation of CVP (CVP_γ)** and **Decisional γ-Approximation of CVP (GapCVP)** with definitions similar to the ones for γ-Approximation of SVP and Decisional γ-Approximation of SVP, but not relevant for this thesis. For further information it is referred to [Mic16] [Pol11] and [AR05].

5.2.1 Short Integer Solution (SIS)

The Short Integer Solution problem or Small Integer Solution problem is an average-case hard problem which has been introduced in [Ajt96]. This average-case hardness is based on the worst-case hardness of SVP_γ. Therefore cryptography based on SIS has the advantage of being proven secure, as described in section 3.2.

Unlike the former problems on lattices, SIS is defined on q-ary lattices. It asks for a short non-zero lattice vector in the kernel q-ary lattice $\Lambda_q^\perp(A)$ as defined in section 5.1.2. In one line it states:

$$\Lambda_q^\perp(A) = \{x \in \mathbb{Z}^m : A * x = 0 \pmod q\} \quad \Rightarrow \text{Finding short } x \text{ is hard}$$

The exact definition is the following [MR07] [Pol11].

> Given a matrix $A \in \mathbb{Z}_q^{n \times m}$ with $m \geq n$, prime integer q and a real γ, find a nonzero vector $x \in \mathbb{Z}^m \setminus \{0\}$, such that $A * x = 0$ mod q and $\|x\| \leq \gamma$

This definition is almost the same as the SVP_γ in $\Lambda_q^\perp(A)$ but is only average-case hard. To be sure, that a solution vector x exists at all, the parameters must hold $m \geq n * log(q)$ and $\gamma \geq \sqrt{m} * q^{\frac{n}{m}}$ [MR07].

As small example, solving a SIS problem would be finding a short vector $x \in \mathbb{Z}^3$ solving following equation. Because there are more columns than rows, the linear equation system

presented is underdetermined (see section 2.2.2). There is more than one solution. It can be imagined, that if n and m are sufficiently large, it is not possible to find a solution with small integers among all the numerous possible ones.

$$\textbf{SIS:} \quad \text{Given } \begin{pmatrix} 5 & 2 & 3 \\ 4 & 6 & 9 \end{pmatrix} * x = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \pmod{11} \quad \Rightarrow \text{Find short } x$$

Inhomogeneous Short Integer Solution (ISIS) The Inhomogeneous Short Integer Solution (ISIS) problem states, that it is hard to find a short non-zero vector $x \in \mathbb{Z}^m$ solving $A*x = u \mod q$ [LDSH21]. This is equivalent to searching a non-zero short vector in the kernel q-ary lattice Λ_q^u as defined in section 5.1.2. The vector u is also called *syndrome* of the ISIS instance.

$$\Lambda_q^u(A) = \{x \in \mathbb{Z}^m : A * x = u \pmod q\} \quad \Rightarrow \text{Finding short } x \text{ is hard}$$

This problem is a variation of SIS with the same properties and similar requirements for A and γ. The distribution of $A*x = u \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$ [Vai15b].

5.2.2 Learning With Errors (LWE)

Another average-case hard problem is the Learning With Errors (LWE) problem or Search-LWE. It's average-case hardness is based on the worst-case hardness of the GapSVP and SIVP [Reg09] and the SVP [Pei09]. A good description is provided in [Pol11].

Given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$, with $m > n$, prime $q \geq 2$ and an error distribution χ over $\mathbb{Z}_q^n$ with mean 0. LWE states it is hard to find a uniformly random $s \in \mathbb{Z}_q^n$, if only A and the product $A^\intercal * s + e \in \mathbb{Z}_q^m$ is given of which $e \in \mathbb{Z}_q^m$ are errors from the distribution χ. Note that because all elements of the equation are from $\mathbb{Z}_q$, the addition $+$ and the multiplication $*$ are modular as well without writing this explicitly. Nevertheless in this summarizing equation of the LWE problem it is noted as reminder:

$$\text{Given } A \text{ and } A^\intercal * s + e \pmod q \quad \Rightarrow \text{Finding } s \text{ is hard}$$

Note that $A^\intercal * s + e$ is the same as $s^\intercal * A + e^\intercal$. Further for the case m=1, this equation is equal to $\langle s, a \rangle + e$.

This problem is hard to solve if s and e are unknown. If one is derived, the other can easily be calculated as well. If the errors are small enough, this problem is the same as finding the next closest vector s in the image q-ary lattice $\Lambda_q(A)$ from section 5.1.2.

In [Reg10] LWE is described in terms of linear equation systems. As in chapter 2.2.2, the random matrix $A^\intercal \in \mathbb{Z}^{m \times n}$ can be seen as a set of m linear equations with n unknowns. With a multiplication $A^\intercal * s$, where $s \in \mathbb{Z}_q^n$ represents a random assignment of the unknowns, the result is a vector v of which each element v_i is the solution of the equation i with the unknowns from s. The result of $v = A^\intercal * s$ is therefore the solution of m linear equations, and because $m > n$ the unknowns s can be fully reconstructed from v and A using Gaussian elimination. LWE now states that this is no longer possible if an error e_i is added to each result v_i. The equation system gets inconsistent and Gaussian elimination is not possible anymore. The vector e can be relatively small and is usually chosen randomly from a discrete Gaussian distribution χ with an expectation $\mu = 0$ and a standard deviation $\sigma = \frac{\alpha * q}{\sqrt{2\pi}}$, where $\alpha \in (0, 1)$ is a given parameter of LWE along with n and q. The parameters must be chosen

such that $\alpha - q > 2\sqrt{n}$ so that LWE is average-case hard [Reg09]. The described properties are the same for an arbitrary number of equations, thus the value of m can be arbitrary. Therefore LWE can be defined as follows.

> Given a random matrix $A \in \mathbb{Z}_q^{n \times m}$, $n \geq 2$, prime integer q, a real $\alpha \in (0, 1)$ and arbitrary number of solutions of $A^\intercal * s + e$, where e is any vector from a distribution χ, find the $s \in \mathbb{Z}_q^n$.

As example on small numbers, solving LWE means to find the unknown $s = \begin{pmatrix}3 & 6\end{pmatrix}^\intercal$, when only $A \in \mathbb{Z}_{11}^{2\times 3}$ and $A^\intercal * s + e$ are given, whereby $A^\intercal * s + e$, was calculated using the random error $e = \begin{pmatrix}0 & 1 & -1\end{pmatrix}^\intercal$ as follows:

$$
\begin{aligned}
A &= \begin{pmatrix}5 & 2 & 3\\ 4 & 6 & 9\end{pmatrix} \\
A^\intercal * s + e &= \begin{pmatrix}5 & 4\\ 2 & 6\\ 3 & 9\end{pmatrix} * \begin{pmatrix}3\\ 6\end{pmatrix} + \begin{pmatrix}0\\ 1\\ -1\end{pmatrix} \\
&= \begin{pmatrix}6\\ 9\\ 8\end{pmatrix} + \begin{pmatrix}0\\ 1\\ -1\end{pmatrix} \\
&= \begin{pmatrix}6\\ 10\\ 7\end{pmatrix}
\end{aligned}
$$

$$
\textbf{LWE:} \quad \text{Given } \left(\begin{pmatrix}5 & 2 & 3\\ 4 & 6 & 9\end{pmatrix}, \begin{pmatrix}6 & 10 & 7\end{pmatrix}^\intercal \right) \quad \Rightarrow \text{Find } s = \begin{pmatrix}3 & 6\end{pmatrix}^\intercal \text{ is hard}
$$

Decisional Learning With Errors (DLWE) The Decisional Learning With Errors (DLWE) states that it is not only hard to recover the vector $s \in \mathbb{Z}_q^n$ from $A \in \mathbb{Z}_q^{n \times m}$ and $A^\intercal * s + e \in \mathbb{Z}_q^m$, but the result $s^\intercal * A + e^\intercal$ is not distinguishable from a uniformly random distribution. The security of DLWE is equivalent to LWE if q is prime and $q = poly(n)$ [Reg09]. This means:

$$
\left(A,\ s^\intercal * A + e^\intercal\right) \approx \left(A,\ b\right), \text{ where } b \in \mathbb{Z}_q^m \text{ is uniformly random}
$$

5.3 Cryptography with Lattices

Both problems, Short Integer Solution (SIS) and Learning With Errors (LWE), can be used for one-way functions, because for both problems the generation of an instance is easy, but solving it is proven average-case hard. That way lattices can be used for a wide range of cryptographic applications[1], including Identity-based Signatures.

Although lattice-based cryptography has larger key sizes than classical pre-quantum cryptography, compared to other post-quantum cryptography it uses reasonable key sizes of for

[1] Examples are hash functions, pseudo-random generators, cryptographic hash functions, secret key and public key encryption, signatures, identity-based and attribute-based encryption and signatures, oblivious transfer and fully homomorphic encryption, fully homomorphic signatures, functional encryption and program obfuscation.

example a 2592 Byte public key for Dilithium5, providing the highest NIST Security Level [BDK+21]. Further the implementation of lattice-based cryptography is easy and efficient, because it is based on simple matrix arithmetic. Additionally it is parallelizable [BHWA22]. And another tremendous advantage of lattice-based cryptography is its so-far quantum resistance. Neither for classical nor for quantum computers there is a polynomial time algorithm known that can solve the underlying problems SIS and LWE. Due to the immense research on error-corrected quantum computers, this advantage weighs a lot.

Because the mechanisms of lattice-based cryptography are similar for different primitives, this section starts introducing the most basic ones, getting step by step more complex, intending to make the understanding of the final concept of Identity-based Signatures more intuitive.

Note that in the following, whenever arithmetic is performed on $\mathbb{Z}_q$ the operation $+$ is an addition modulo q and the operation $*$ is a multiplication modulo q without needing this to be written explicitly. Nevertheless it is stated in some equations to remind the reader of this fact.

5.3.1 Ajtai's Hash Function

Together with the Short Integer Solution (SIS) problem Ajtai introduced the first cryptographic usage of lattices in [Ajt96]. This is a cryptographic hash function, the so-called Ajtai Hash Function, based on the hardness of SIS.

For this, let a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$, with prime q and $m > n * log(q)$ be given. The function $h_\mathrm{A} : \{0,1\}^m \to \mathbb{Z}_\mathrm{q}^\mathrm{n}$ is defined by:

$$h_\mathrm{A}(r) = A * r \pmod q$$

As insecure example with small numbers we could calculate the hash of the number 5 with the function parameters $n = 2, m = 4, q = 7$ and $A = \begin{pmatrix} 5 & 1 & 4 & 2 \\ 6 & 3 & 2 & 1 \end{pmatrix}$. The binary value of 5 is 0101, therefor we set r to the value $r = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}^\intercal$.

$$h_\mathrm{A}(r) = A * r = \begin{pmatrix} 5 & 1 & 4 & 2 \\ 6 & 3 & 2 & 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \pmod 7 = \begin{pmatrix} 3 & 4 \end{pmatrix}$$

Figure 5.10 shows graphically what Ajtai's Hash Function does, using the same values as in the former example. The lattice is constructed on A as basis vectors, shown as purple arrows. The hash value $h(t)$ is marked as blue point. It can not easily be seen which other input values would lead to the same point $h(t)$.

Because $m > n * log(q)$, equivalent to $2^m > q^n$ where 2^m is the number of possible inputs and q^n is the number of possible outputs of h_A, the function is compressing. The input r contains more information than the output $h_\mathrm{A}(r)$.

The function is collision-resistant because of the following. Assuming there was a collision

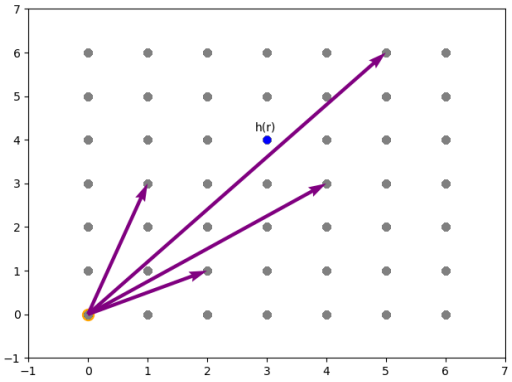


Figure 5.10: $h_A((0\ 1\ 0\ 1)^T)$ with $A = \begin{pmatrix} 5 & 1 & 4 & 2 \\ 6 & 3 & 2 & 1 \end{pmatrix}$

of r and r', this would mean that $h_A(r) = h_A(r')$.

$$\begin{aligned} h_A(r) &= h_A(r') \\ A * r &= A * r' \pmod q \\ A * r - A * r' &= 0 \pmod q \\ A * (r - r') &= 0 \pmod q \end{aligned} \tag{5.1}$$

Because r and $r' \in \{0,1\}^m$ are short as they are only consisting of values 0 or 1, the distance between both vectors must be small, bounded by $\sqrt{m * 1^2} = \sqrt{m}$. That is why finding the vector r' given the vector r would be the solution of the SIS problem, which is hard. So it is also hard to find a r' which would return the same hash value as r, and that is why Ajtai's Hash Function is collision-resistant and a so-called One-way Function or surjective Trapdoor Function.

5.3.2 Secret-Key Encryption

Symmetric encryption, also called secret-key encryption, uses the same key for encryption as for decryption. Because symmetric encryption is less threatened by quantum computers as public key encryption, a lattice-based secret-key encryption is due to its key sizes not relevant in practice. But it helps to understand the concepts of the public-key encryption schemes in the following sections. Therefore here the basic concept of secret-key encryption with lattices is explained [Vai15b], and accompanied by small number examples. Figure 5.11 gives an overview of the scheme.

For secret-key encryption a uniformly random secret key $s \in \mathbb{Z}_q^n$ is chosen, where q is prime and n is the so-called *security parameter*. Additionally a publicly known uniformly random $a \in \mathbb{Z}_q^n$ is used.

$$secret\ key : s \in \mathbb{Z}_q^n$$

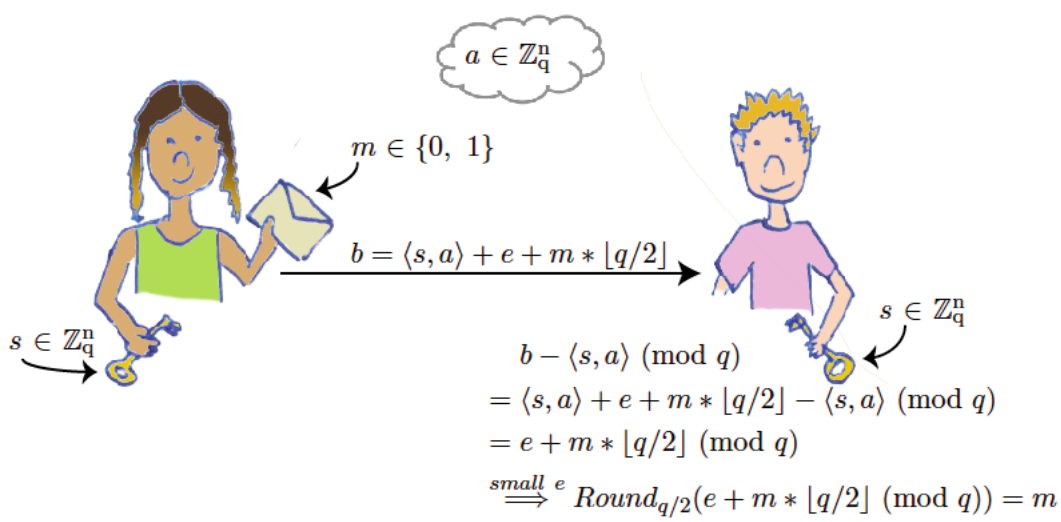


Figure 5.11: Secret-key encryption of a bit

A single bit $m \in \{0,\ 1\}$ can be encrypted as follows. Here a *small* random error e with $\|e\| < q/4 \in \mathbb{Z}$ is chosen from a distribution χ during encryption and deleted directly after, because it is not needed for decryption.

$$\textbf{Enc}(m, s): \quad c = (a,\ b) \quad \text{with } b = \langle s, a\rangle + e + m * \lfloor q/2 \rfloor \pmod q$$

As mentioned in section 5.2.2, $\langle s, a\rangle + e$ is a LWE instance generated by a matrix $a \in \mathbb{Z}^{n\times 1}$. Thus the encryption generates a LWE instance, with a small error vector e or a larger error vector $e + m * \lfloor q/2 \rfloor$ if $m = 1$. Note that the DLWE states that $\langle s, a\rangle + e$ looks uniformly random. Thus this procedure mimics an information theoretically secure OTP encryption (see section 3.1), by adding the message information stored in $m * \lfloor q/2 \rfloor$ to the random looking value of $\langle s, a\rangle + e$, causing the result b to also look random. The vector a has to be part of the resulting cipher because it is needed for decryption.

As example Alice picks the public parameters $q = 7$ and $a = (4\ \ 3)^\intercal$. Further she and Bob share the secret key $s = (2\ \ 6)^\intercal$. For encrypting the message $m \in 0, 1$ she uses $e = 1$ and calculates the cipher c and sends it to Bob:

$$\begin{aligned}
b &= \langle s, a\rangle + e + m * \lfloor q/2 \rfloor \\
&= \langle \begin{pmatrix}2\\6\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}\rangle + 1 + m * \lfloor 7/2 \rfloor \\
&= 2 * 4 + 6 * 3 + 1 + m * 3 \\
&\overset{m=1}{=} 5 + 1 + 1 * 3 = 2 \Rightarrow \quad c = \left(a = \begin{pmatrix}4\\3\end{pmatrix},\ b = 2\right) \\
&\overset{m=0}{=} 5 + 1 + 0 * 3 = 6 \Rightarrow \quad c = \left(a = \begin{pmatrix}4\\3\end{pmatrix},\ b = 6\right)
\end{aligned}$$

What this encryption does geometrically is shown in figure 5.12 using the same vectors as in the former example. The scalar of the vectors s and a is defined by $\left\{\sum_{i=1}^{n} s_i a_i\right\}$ and

because $a_i \in \mathbb{Z}_q$ it is also a point in the lattice spanned by $B = \begin{pmatrix} 4 & 3 \end{pmatrix}$ (see the definition of lattices in section 5.1). The basis vectors are shown as purple arrows and span a one-dimensional q-ary lattice in $\mathbb{Z}_7$. All lattice points are marked as gray dots, marking a possible result of $a = \begin{pmatrix} 4 & 3 \end{pmatrix}^\intercal$ times any possible secret s. The scalar $\langle s, a \rangle$ is marked as green point $x = 5$, the random looking cipher of $m = 0$ being $\langle s, a \rangle + e$ is marked as blue point $b_0 = 6$. And the cipher of $m = 1$ being $b_1 = x' + \lfloor q/2 \rfloor = 2$ is marked as red dot. It can be seen that b_0 nor b_1 give any indication of the values of s or e if x was not known.

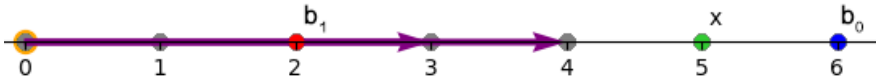


Figure 5.12: Encryption of $m = 1$ as one-dimensional lattice

Decryption uses the same secret key as for encryption:

$$\textbf{Dec}(a, b, s): \quad m = Round_{q/2}(b - \langle s, a \rangle) \pmod q$$

By subtracting the real value of $\langle s, a \rangle$ from $b = \langle s, a \rangle + e + m * \lfloor q/2 \rfloor$ one gets $e + m * \lfloor q/2 \rfloor$, which is either the small error e in the case the message was 0 or $e + \lfloor q/2 \rfloor$, in the case the message was 1. By rounding to $q/2$ we result in the sent message $m \in \{0, 1\}$ because the small e is insignificant.

$$\begin{aligned} & Round_{q/2}(b - \langle s, a \rangle) \\ &= Round_{q/2}((\langle s, a \rangle + e + m * \lfloor q/2 \rfloor) - \langle s, a \rangle) \\ &= Round_{q/2}(e + m * \lfloor q/2 \rfloor) \\ &\approx Round_{q/2}(m * \lfloor q/2 \rfloor) \\ &= m \end{aligned}$$

In the case of Bob, for decrypting the cipher $c = \left(\begin{pmatrix} 4 & 3 \end{pmatrix}^\intercal, \ 6 \right)$ he uses the public $q = 7$ and the same secret key $s^\intercal = \begin{pmatrix} 2 & 6 \end{pmatrix}$ as Alice.

$$\begin{aligned} m &= Round_{q/2}(b - \langle s, a \rangle) \\ &= Round_{7/2}(6 - \langle \begin{pmatrix} 2 \\ 6 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix} \rangle) \\ &= Round_{3.5}(6 - 5) \\ &= Round_{3.5}(1) \\ &= 0 \end{aligned}$$

He calculates the difference between part b of the cipher and the real value of $\langle s, a \rangle$, which is $6 - 5 = 1$. The result 1 is nearer to 0 than to $q/2$ inside $\mathbb{Z}_7$. So the message is 0.

In case of $c = \left(\begin{pmatrix} 4 & 3 \end{pmatrix}^\intercal, \ 2 \right)$ we get $6 - 2 = 4$, which is nearer to 3.5 than to 0. The resulting message is 1.

Figure 5.13 shows the regions of $\mathbb{Z}_7$ in which b would encode the message $m = 0$ in yellow, and regions in which b would encode $m = 1$ in red. The green dot is $x = \langle s, a \rangle \in \mathbb{Z}_7$ from which the secret key s can be calculated.

Figure 5.13: Decryption for $s = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$ and $a = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$

5.3.3 Regev Public-Key Cryptosystem

In 2005 it was demonstrated by Oded Regev in [Reg10] that lattices can also be used for public-key encryption. An overview of the procedure is given in figure 5.14.

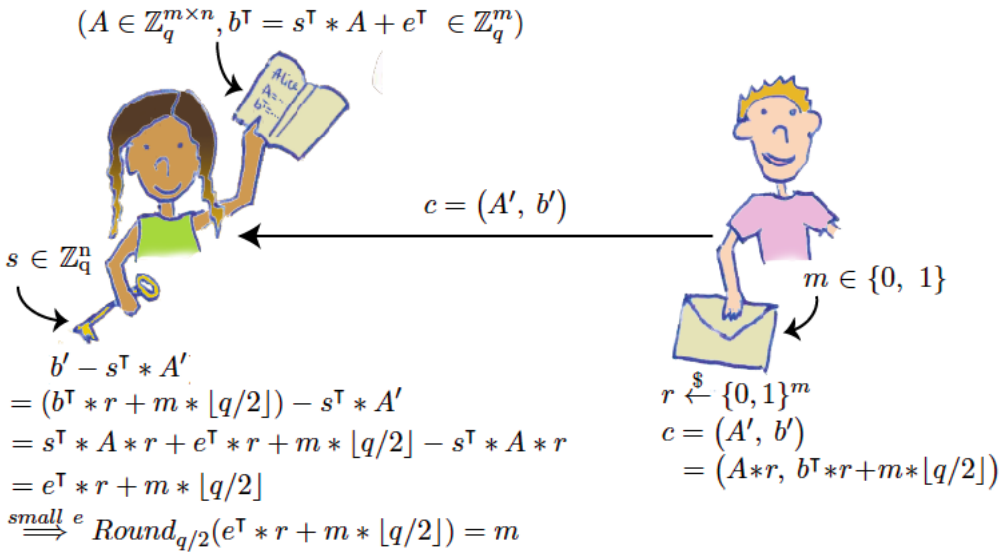


Figure 5.14: Public-key encryption of a bit

For public-key encryption, the previous secret-key encryption is extended. It is also based on LWE and DLWE. The size of the parameters is a possible choice guaranteeing security and correctness after [Reg10].

Again, an equally distributed random vector $s \in \mathbb{Z}_q^n$ is used as secret key, with q being prime, $n \in \mathbb{Z}$ being the *security parameter* and holding $n^2 < q < 2 * n^2$. A uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ and $m = 1.1 * n * log(q)$ is chosen as part of the *public key*. An error vector $e \in \mathbb{Z}_q^m$ from a distribution χ with $\|e\| < q/4$ is used for public key generation. The key pair is computed as follows:

$$\begin{aligned} &\textit{secret key} : s \in \mathbb{Z}_q^n \\ &\textit{public key} : (A \in \mathbb{Z}_q^{n \times m}, \quad b^\intercal \in \mathbb{Z}_q^{1 \times m}) \quad \text{with } b^\intercal = s^\intercal * A + e^\intercal \end{aligned}$$

Part $b^\intercal$ of the public key can be seen as a symmetric encryption of m zeros. Note that $s^\intercal * A + e^\intercal = A^\intercal * s + e \pmod q$, hence it is a LWE instance with small errors, that is located around a lattice vector in $\Lambda_q(A)$. Yet, to the public it looks like uniformly random, according to DLWE.

Suppose Alice chooses $q = 13$, the random uniformly distributed $s = \begin{pmatrix}2 & 9\end{pmatrix}^\intercal$, $A = \begin{pmatrix}2 & 12 & 8\\ 5 & 9 & 10\end{pmatrix}$ and $e = \begin{pmatrix}0 & 1 & -1\end{pmatrix}^\intercal$, her key is composed as follows. Note that the size of q, m and n are not secure, but chosen for having an easier example.

$$\begin{aligned}
\textit{secret key} &= \begin{pmatrix}2\\ 9\end{pmatrix}\\
b^\intercal &= s^\intercal * A + e^\intercal\\
&= \begin{pmatrix}2 & 9\end{pmatrix} * \begin{pmatrix}2 & 12 & 8\\ 5 & 9 & 10\end{pmatrix} + \begin{pmatrix}0 & 1 & -1\end{pmatrix}\\
&= \begin{pmatrix}10 & 1 & 2\end{pmatrix} + \begin{pmatrix}0 & 1 & -1\end{pmatrix}\\
&= \begin{pmatrix}10 & 2 & 1\end{pmatrix}\\
\textit{public key} &= (\begin{pmatrix}2 & 12 & 8\\ 5 & 9 & 10\end{pmatrix}, \begin{pmatrix}10 & 2 & 1\end{pmatrix})
\end{aligned}$$

To encrypt a message $m \in \{0,1\}$, the matrix A is multiplied with a random vector $r \in \{0,1\}^m$, and $b^\intercal$ is multiplied with this r as well. Note that $b^\intercal * r = \langle b, r\rangle$. Similar to symmetric encryption, the resulting $b * r$ is added to $m * \lfloor q/2 \rfloor$. The resulting cipher is composed of $(A * r, b^\intercal * r + m * \lfloor q/2 \rfloor)$.

$$\textbf{Enc}(m, A, b): \quad c = \begin{pmatrix}A', & b'\end{pmatrix} = \begin{pmatrix}A * r, & b^\intercal * r + m * \lfloor q/2 \rfloor\end{pmatrix} \quad \text{with } r \xleftarrow{\$} \{0,1\}^m$$

By multiplying A with r, the sender selects random columns of the matrix, which he will use to encode his message. Since the publicly known $A*r$ of the cipher cannot be decomposed into r according to SIS, an adversary does not know which columns of A were chosen for encryption. Because b looks random by itself, $b^\intercal * r$ also looks random, and $b^\intercal * r + \lfloor q/2 \rfloor$ cannot be decrypted for the same reason as for symmetric encryption.

In case Bob sends the message $m = 1$ to Alice, he chooses for example the random vector $r = \begin{pmatrix}1 & 0 & 1\end{pmatrix}^\intercal$ and computes the cipher as follows.

$$\begin{aligned}
A' = A * r &= \begin{pmatrix}2 & 12 & 8\\ 5 & 9 & 10\end{pmatrix} * \begin{pmatrix}1\\ 0\\ 1\end{pmatrix}\\
&= \begin{pmatrix}10\\ 2\end{pmatrix}\\
b' = b^\intercal * r + m * \lfloor q/2 \rfloor &= \begin{pmatrix}10 & 2 & 1\end{pmatrix} * \begin{pmatrix}1\\ 0\\ 1\end{pmatrix} + 1 * \lfloor 13/2 \rfloor\\
&= 11 + 6 = 4\\
c &= (\begin{pmatrix}10\\ 2\end{pmatrix}, 4)
\end{aligned}$$

To decrypt a cipher, the secret key s is used.

$$\textbf{Dec}(A', b', s): \quad m = Round_{q/2}(b' - s^\intercal * A')$$

The decryption is working, even though r is unknown to the receiver and cannot be derived from either $A*r$ nor $b^\intercal*r$. This is because $b^\intercal*r$ is composed of $(s^\intercal*A+e^\intercal)*r = (s^\intercal*A*r+e*r)$. Because $A' = A*r$ is also sent as part of the cipher, the receiver only needs to multiply his secret key by $A*r$. Subtracting this result from b' yields $r*e^\intercal + m*\lfloor q/2 \rfloor$, of which the error is small enough to discard it. Thus the receiver obtains the message m, encoded in $m*\lfloor q/2 \rfloor$, without needing to know what r has been, in other words, which columns of the public key have been used to encrypt it.

$$\begin{aligned} b' - s^\intercal * A' &= (b^\intercal * r + m * \lfloor q/2 \rfloor) - s^\intercal * A' \\ &= (s^\intercal * A + e^\intercal) * r + m * \lfloor q/2 \rfloor) - s^\intercal * A * r \\ &= s^\intercal * A * r + e^\intercal * r + m * \lfloor q/2 \rfloor - s^\intercal * A * r \\ &= e^\intercal * r + m * \lfloor q/2 \rfloor \\ &\approx m * \lfloor q/2 \rfloor \end{aligned}$$

To decrypt Bob's message $c = (A', b') = (\begin{pmatrix}10\\2\end{pmatrix}, 4)$ Alice must use her secret key $s = \begin{pmatrix}2 & 9\end{pmatrix}$ as following to obtain the original message $m = 1$.

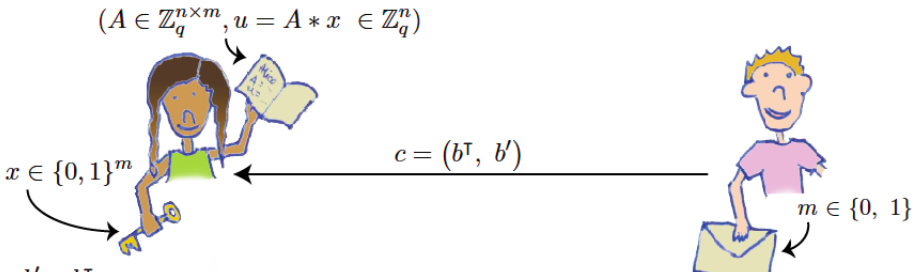
$$\begin{aligned} m &= Round_{q/2}(b' - s^\intercal * A') \\ &= Round_{13/2}(4 - \begin{pmatrix}2 & 9\end{pmatrix} * \begin{pmatrix}10\\2\end{pmatrix}) \\ &= Round_{6.5}(4 - 12) \\ &= Round_{6.5}(5) \\ &= 1 \end{aligned}$$

This cryptosystem has the disadvantage, that all public keys are very close to the lattice points of $\Lambda_q(A)$, and therefor the possible public keys are very sparse. Hence this cryptosystem is not reasonable for IBE, where every id is hashed to public key, because this is not possible for such public keys [GPV08]. Therefore another public-key encryption scheme has been invented.

5.3.4 Dual Public-Key Cryptosystem

The dual cryptosystem introduced in [GPV08] is another kind of asymmetric encryption scheme on lattices. For dual encryption the keys are also based on a uniformly random matrix A, but instead of being generated by an LWE one-way function, the key pair is generated with an ISIS one-way function. This way the range and domain space are exchanged compared to the Regev cryptosystem, and one can be sure that there are enough public keys. This is also where the name "dual" cryptosystem origins from. Its security is based on the ISIS and DLWE problems. Figure 5.15 shows the overall procedure and all mechanisms are accompanied with small number examples.

For key generation, a uniformly random distributed matrix $A \in \mathbb{Z}_q^{n \times m}$is chosen, with q being prime and $m > 2*n*log(q)$. The secret key is a small vector $x \in \mathbb{Z}^m$ sampled from a distribution $D_{\mathbb{Z}^m,\sigma}$. This Gaussian distribution is used to sample m-dimensional integer vectors around 0 with a standard deviation of σ. For more details see [GPV08], but it is enough to know that x is a small vector from $x \in \mathbb{Z}^m$. The public key consists of the matrix



$$
\begin{aligned}
& b' - b^\intercal * x \\
&= (s^\intercal * u + e' + m * \lfloor q/2 \rfloor) - (s^\intercal * A + e^\intercal) * x \\
&= s^\intercal * u + e' + m * \lfloor q/2 \rfloor - s^\intercal * A * x + e^\intercal * x \\
&= s^\intercal * u + e' + m * \lfloor q/2 \rfloor - s^\intercal * u + e^\intercal * x \\
&= e' + m * \lfloor q/2 \rfloor + e^\intercal * x \\
&\overset{small\ e}{\Longrightarrow} Round_{q/2}(e' + m * \lfloor q/2 \rfloor + e^\intercal * x) = m
\end{aligned}
$$

$$
\begin{aligned}
& s \xleftarrow{\$} \mathbb{Z}_q^n \\
& c = (b^\intercal,\ b') \\
& = (s^\intercal * A + e^\intercal,\ s^\intercal * u + e' + m * \lfloor q/2 \rfloor)
\end{aligned}
$$

Figure 5.15: Dual encryption of a bit

A and the solution of the ISIS one-way function $A * x = u \in \mathbb{Z}_q^n$, similar to Ajtai's hash function from section 5.10. Given u and A, x cannot be derived from them.

$$
\begin{aligned}
& secret\ key : x \in \{0,1\}^m \\
& public\ key : (A \in \mathbb{Z}_q^{n \times m},\ \ u \in \mathbb{Z}_q^n) \quad \text{with } A * x = u
\end{aligned}
$$

As a small number example, suppose Alice chooses $q = 13$, the random uniformly distributed $x = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^\intercal$ and $A = \begin{pmatrix} 2 & 12 & 8 \\ 5 & 9 & 10 \end{pmatrix}$, her key is composed as follows.

$$
\begin{aligned}
secret\ key &= \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^\intercal \\
u &= A * x \\
&= \begin{pmatrix} 2 & 12 & 8 \\ 5 & 9 & 10 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 7 \\ 6 \end{pmatrix} \\
public\ key &= \left(\begin{pmatrix} 2 & 12 & 8 \\ 5 & 9 & 10 \end{pmatrix}, \begin{pmatrix} 7 \\ 6 \end{pmatrix} \right)
\end{aligned}
$$

To encrypt a message $m \in \{0,1\}$, a random vector $s \in \mathbb{Z}_q^n$ is chosen. The cipher then consists of the tuple $(b^\intercal = s^\intercal * A + e^\intercal, b' = s^\intercal * u + e' + m * \lfloor q/2 \rfloor)$. The first part is the LWE problem based on A, looking random but containing s. The second part is the LWE problem based on u, looking random but containing s and hiding the message $m * \lfloor q/2 \rfloor$ the same way as in secret-key and public-key encryption. The result looks random, but contains the information of the message m.

$$
\textbf{Enc}(m, A, u): \quad c = \left(b^\intercal = s^\intercal * A + e^\intercal,\ b' = s^\intercal * u + e' + m * \lfloor q/2 \rfloor\right) \quad \text{with } s \xleftarrow{\$} \mathbb{Z}_q^n
$$

In case Bob sends the message $m = 1$ to Alice, he chooses a random vector in $\mathbb{Z}_{13}^2$, for example $s = \begin{pmatrix}2 & 10\end{pmatrix}^\intercal$. Additionally he uses the random small errors $e = \begin{pmatrix}-1 & 0 & 1\end{pmatrix}^\intercal$ and $e' = -1$ and computes the cipher with her public key as follows.

$$\begin{aligned}
b^\intercal = s^\intercal * A + e^\intercal &= \begin{pmatrix}2 & 10\end{pmatrix} * \begin{pmatrix}2 & 12 & 8\\ 5 & 9 & 10\end{pmatrix} + \begin{pmatrix}-1 & 0 & 1\end{pmatrix}\\
&= \begin{pmatrix}2 & 10 & 12\end{pmatrix} + \begin{pmatrix}-1 & 0 & 1\end{pmatrix}\\
&= \begin{pmatrix}1 & 10 & 0\end{pmatrix}\\
b' = s^\intercal * u + e' + m * \lfloor q/2 \rfloor &= \begin{pmatrix}2 & 10\end{pmatrix} * \begin{pmatrix}7\\ 6\end{pmatrix} - 1 + 1 * \lfloor 13/2 \rfloor\\
&= 9 - 1 + 6\\
&= 14\\
c &= \left(\begin{pmatrix}1 & 10 & 0\end{pmatrix}, 14\right)
\end{aligned}$$

To decrypt the cipher $c = (b^\intercal, b')$, the receiver multiplies the secret key x to the first part of the cipher and subtracts the result from the second part of the cipher. This decryption works in the same way as symmetric encryption, except that instead of a shared vector s the product vector of $A * x$ hides the secret.

$$\textbf{Dec}(b', b^\intercal, x): \quad m = Round_{q/2}(b' - b^\intercal * x)$$

The message m is hidden in the random looking $b' = s^\intercal * u + e' + m * \lfloor q/2 \rfloor$. To contain this information one would have to subtract $(s^\intercal * u + e' + m * \lfloor q/2 \rfloor) - (s^\intercal * u + e')$, and because firstly s is unknown and therefore $s^\intercal * u$ cannot be calculated directly, and secondly $s^\intercal * A + e^\intercal$ is given, the necessary subtraction can only be done with the knowledge of x, as shown here.

$$\begin{aligned}
b' - b^\intercal * x &= (s^\intercal * u + e' + m * \lfloor q/2 \rfloor) - (s^\intercal * A + e^\intercal) * x\\
&= s^\intercal * u + e' + m * \lfloor q/2 \rfloor - s^\intercal * A * x + e^\intercal * x\\
&\overset{A*x=u}{=} s^\intercal * u + e' + m * \lfloor q/2 \rfloor - s^\intercal * u + e^\intercal * x\\
&= e' + m * \lfloor q/2 \rfloor + e^\intercal * x\\
&\approx m * \lfloor q/2 \rfloor
\end{aligned}$$

Because the errors are small, by rounding to $q/2$, the original message can be restored to 0, for a result close to 0 or q, or to 1, for a result close to $q/2$.

In our example Alice would decrypt Bob's message $c = \left(\begin{pmatrix}1 & 10 & 0\end{pmatrix}, 14\right)$ by using her secret key $x = \begin{pmatrix}0 & 1 & 1\end{pmatrix}^\intercal$ as following to receive the original message $m = 1$.

$$\begin{aligned}
m &= Round_{q/2}(b' - b^\intercal * x)\\
&= Round_{13/2}(14 - \begin{pmatrix}1 & 10 & 0\end{pmatrix} * \begin{pmatrix}0\\ 1\\ 1\end{pmatrix})\\
&= Round_{6.5}(14 - 10)\\
&= Round_{6.5}(4)\\
&= 1
\end{aligned}$$

5.3.5 Preimage Sampleable Trapdoor Functions

Another cryptographic primitive that can be implemented using lattices are trapdoor functions, that allow an easy way to construct digital signatures and IBE. They were first invented on lattices in [GPV08], based on an earlier work of Ajtai [Ajt99].

Trapdoor Functions are, similar to hash functions, easy to compute in one direction and difficult in the other. But unlike hash functions, for trapdoor functions there is a secret, the so-called trapdoor T, which can be used to easily compute the difficult reverse computation, the preimage.

Lattices allow the construction of trapdoor functions based on LWE and on ISIS. This means the one-way function of LWE and ISIS can be inverted if a secret trapdoor is known. In the following it is first shown how the DLWE and LWE problems can be solved with a given *Type 1 Trapdoor*. Then the concept of a *Gadget Matrix* to generate trapdoors is introduced. Finally it is explained how the Gadget Matrix can be used to create *Type 2 Trapdoors* that can solve ISIS as well. The following outlines the concept from [GPV08]. Also the lecture notes were used for understanding [Vai15a], [Vai15c] and [VG15] can be recommended.

Type 1 Trapdoors for solving LWE So suppose we are given a vector $y \in \mathbb{Z}_q^m$, and a matrix $A \in \mathbb{Z}_q^{n \times m}$. The y looks random, so we don't know if it is actually a random vector, or if it is an LWE sample with the given matrix A, i.e., $y = s^\intercal * A + e^\intercal$.

Now, if we knew a short vector $t \in \Lambda^\perp(A) \subset \mathbb{Z}_q^m$ for which holds $A * t = 0 \mod q$, we could determine whether y is an LWE sample with the matrix A. This means we can solve Decisional Learning With Errors (DLWE). The for this required vector t is nearly impossible to derive from the matrix A, according to SIS. This vector t is our trapdoor for DLWE, this means only if we know t, we can solve DLWE for A.

So, in summary, it is given:

$$y \in \mathbb{Z}_q^m \stackrel{?}{=} s^\intercal * A + e^\intercal \qquad A \in \mathbb{Z}_q^{n \times m} \qquad \text{short } t \in \mathbb{Z}_q^m, \text{ solving } A * t = 0 \bmod q$$

The DLWE problem is solved by calculating the scalar of y and the trapdoor t. If y is just random, the result of $\langle y, t \rangle = y^\intercal * t$ is also random. But if y is a LWE sample of the matrix A, then the result is small since a small error vector multiplied to a small vector t results in a small product.

$$\begin{aligned} y^\intercal * t &= (s^\intercal * A + e^\intercal) * t \\ &= s^\intercal * A * t + e^\intercal * t \\ &\stackrel{A*t=0}{=} s^\intercal * 0 + e^\intercal * t \\ &= e^\intercal * t \text{ is small} \end{aligned}$$

Thus, the result $e^\intercal * t$ lets us solve DLWE, but we cannot recover the error e yet. For this, t would have to have an inverse in $\mathbb{Z}_q$ (for n-dimensional inverses see chapter 2.2.2). To achieve this, instead of a vector t, we need a matrix T that is a short basis for $\Lambda^\perp(A)$. Therefor,

given $A \in \mathbb{Z}_q^{n \times m}$, T_A has to satisfy the following properties:

- $T_A \in \mathbb{Z}_q^{m \times m}$
- $\|T_A\|$ is small
- $A * T_A = 0 \bmod q \quad \text{with } 0 = 0^{n \times m}$
- T_A has rank m over $\mathbb{Z}$

Since the matrix T is full-rank, it is invertible, though only in the space $\mathbb{Z}$ and not $\mathbb{Z}_q$. Nevertheless we can find a T_A^{-1}, so that $T_A * T_A^{-1} = Id_m = 1$. This allows us to reconstruct the error e.

$$\begin{aligned} y^\intercal * T_A &= (s^\intercal * A + e^\intercal) * T_A \\ &= s^\intercal * A * T_A + e^\intercal * T_A \\ &\overset{A*T_A=0}{=} e^\intercal * T_A \\ &\overset{T_A*T_A^{-1}=1}{\Longrightarrow} e^\intercal * T_A * T_A^{-1} \text{ (No modular multiplication because } T_A^{-1} \in \mathbb{Z}^{m \times m}) \\ &= e^\intercal \end{aligned}$$

Since the matrix A consists of more columns than rows, we can also calculate its right-inverse A_{right}^{-1}. With this inverse and the already reconstructed error e, we can recover s, and have broken LWE.

$$\begin{aligned} &\overset{e^\intercal}{\Longrightarrow} y^\intercal - e^\intercal = (s^\intercal * A + e^\intercal) - e^\intercal \\ &\qquad = s^\intercal * A \\ &\overset{A*A_{right}^{-1}=1}{\Longrightarrow} s^\intercal * A * A_{right}^{-1} = s^\intercal \end{aligned}$$

In conclusion, if we have a generating matrix A and an associated trapdoor matrix T_A given, we can use LWE as a trapdoor function now. But the remaining difficulty is that the matrix A must be randomly distributed, and after SIS the computation of an associated trapdoor T is impossible - otherwise everybody could compute the trapdoor and it would not be a trapdoor anymore. This problem is solved by generating A and a corresponding trapdoor in one go - using a special gadget.

The Gadget Matrix for Sampling a Trapdoor Pair To generate the desired random matrix A together with its corresponding trapdoor, a full-rank and uniformly distributed matrix G is used, for which the corresponding trapdoor T_G is already known. It is called the Gadget matrix. As reminder, this matrix cannot be chosen directly as the matrix A because it is not uniformly random, and further its trapdoor is not secret. However, we can use it to sample a matrix A and its trapdoor. How and why this works is explained in this and the next paragraph.

The gadget matrix $G \in \mathbb{Z}^{n \times nlogq}$ is defined by the horizontal vector g.

$$
\begin{aligned}
g &= \begin{pmatrix} 1 & 2 & 4 & 8 ... 2^{\lceil logq \rceil - 1} \end{pmatrix} \in \mathbb{Z}_q^{1 \times logq} \\
G &= g \otimes Id_n \ \in \mathbb{Z}_q^{n \times n * logq} \\
&= \begin{pmatrix} g & 0 & \cdots & 0 \\ 0 & g & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g \end{pmatrix} \\
&= \begin{pmatrix}
1 & 2 & 4 & \cdots & 2^{\lceil logq \rceil - 1} & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 1 & 2 & 4 & \cdots & 2^{\lceil logq \rceil - 1} & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 2 & 4 & \cdots & 2^{\lceil logq \rceil - 1}
\end{pmatrix}
\end{aligned}
$$

The gadget matrix G is a binary recomposition matrix. If $binary: \ \mathbb{Z}_q \to \{0,1\}^{logq}$ is a function that returns the binary composition of its input as vector, G can do the following with $x_i < q$.

$$
G * b = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{for } b = \begin{pmatrix} binary(x_1) \\ \vdots \\ binary(x_n) \end{pmatrix}
$$

For example, given $x_1 = 1, x_2 = 6, x_3 = 5$ and $q = 7$

$$
G * b = \begin{pmatrix} 1 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ 5 \end{pmatrix}
$$

The to G corresponding trapdoor T_G looks as follows.

$$
\begin{aligned}
T_G &= T_g \otimes Id_n \ \in \mathbb{Z}_q^{n \times n*logq} \\
&= \begin{pmatrix} T_g & 0 & \cdots & 0 \\ 0 & T_g & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_g \end{pmatrix} \\
\text{with } T_g &= \left(\begin{array}{ccccc|c} 2 & 0 & 0 & \cdots & 0 & \\ -1 & 2 & 0 & \cdots & 0 & \\ 0 & -1 & 2 & \ddots & \vdots & binary(q) \\ \vdots & \ddots & \ddots & \ddots & 0 & \\ \vdots & \cdots & 0 & -1 & 2 & \\ 0 & \cdots & 0 & 0 & -1 & \end{array}\right) \in \mathbb{Z}_q^{logq \times logq}
\end{aligned}
$$

As an example, chosen $q = 7$ we can prove that T_G is the trapdoor of G.

$$
\begin{aligned}
G * T_G &= \begin{pmatrix} 1 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix} * \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \pmod 7 \\
&= \begin{pmatrix} 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \end{pmatrix} \pmod 7 \\
&= 0 \pmod 7
\end{aligned}
$$

T_g is short, its length is $\sqrt{5}$ or $O(\sqrt{logq})$, and since its determinant $det(T_g) \neq 0$ it is also full-rank over $\mathbb{Z}$, though not over $\mathbb{Z}_q$. From this it follows that also T_G is short, full-rank over $\mathbb{Z}$, and additionally lies in the nullspace of G, which means $G * T_G = 0 \mod q$. Hence, it holds that T_G is a trapdoor of G.

As a resume of this paragraph, we can use G and T_G to solve a LWE sample $y = s^\intercal * G + e \mod q$ exactly as explained in the previous paragraph, only that $A = G$ and $T_A = T_G$. However, this does not help us directly because G is not random, and in addition the trapdoor T_G is public knowledge. However, we can now use G and T_G to construct a uniformly randomly distributed A with a corresponding trapdoor.

Type 2 Trapdoors for solving LWE and ISIS To revert LWE and ISIS our trapdoor requirements can be changed a little. To accomplish our goals, we will also suffice with a trapdoor matrix R that, when multiplied by the random matrix A, yields our gadget matrix G. This allows us to convert our LWE sample respective matrix A to an LWE sample

respective matrix G, and then solve this with our trapdoor T_G. Why and exactly how this works is explained in this paragraph.

Besides it should be mentioned that instead of G and T_G we could use any other equal dimensional matrix G' of which we know a trapdoor T'_G. However, the matrix G is very well suited for this purpose because of the easy to compute power-of-2 calculations and the simple binary recomposition concept.

As before, our new pair of matrices consists on the one side of a linearly independent uniformly random distributed matrix A, and on the other side of the trapdoor matrix R, that is small and thus can not be calculated from A according to ISIS. But now R has to satisfy the following equation, where G is the Gadget matrix from the former paragraph.

$$A * R = G \pmod q \quad \text{and} \quad \|R\| \text{ is small}$$

When A and the corresponding R are known, DLWE can be solved by multiplying the trapdoor R onto the LWE-sample $y = s^\intercal * A + e$. In this way the random matrix A is transformed to the structured matrix G, for which the trapdoor is known. The error of the LWE sample is also changed, but due to e and R being small, the product $e * R = e'$ remains small as well. The DLWE problem can now be decided for the new LWE sample y' with respect to G, in the same way as in the first paragraph, using the trapdoor T_G from the second paragraph. If y' is small, then y was a LWE sample with respect to A.

$$\begin{aligned}
\text{Transform } A \text{ to } G \implies y^\intercal * R &= (s^\intercal * A + e^\intercal) * R \\
&= s^\intercal * A * R + e^\intercal * R \\
&\overset{A*R=G}{=} s^\intercal * G + e^\intercal * R \\
&= s^\intercal * G + e'^\intercal \\
&= y' \\
\text{Solve DLWE regarding } G \implies y'^\intercal * T_G &= (s^\intercal * G + e^\intercal) * T_G \\
&= s^\intercal * G * T_G + e'^\intercal * T_G \\
&\overset{G*T_G=0}{=} s^\intercal * 0 + e'^\intercal * T_G \\
&= e'^\intercal * T_G \text{ is small}
\end{aligned}$$

Further it is possible to solve the LWE problem from the former result. To do so, we reconstruct the error e' of our y' as before. If this is known we can reconstruct s as well.

$$\begin{aligned}
y'^\intercal * T_G &= e'^\intercal * T_G \\
\text{Calculate } e' &\overset{T_G * T_G^{-1} = 1}{\implies} e'^\intercal * T_G * T_G^{-1} \pmod q \\
&= e'^\intercal \\
\text{Reconstruct } s &\overset{e'}{\implies} y^\intercal - e'^\intercal \\
&= (s^\intercal * G +' e^\intercal) - e'^\intercal \\
&= s^\intercal * G \\
&\overset{G * G^{-1}_{right} = 1}{\implies} s^\intercal * G * G^{-1} \\
&= s^\intercal
\end{aligned}$$

Thus the matrices A and R are sufficient for a trapdoor function based on LWE. However, with their help we can also solve the ISIS problem $A * x = u \mod q$. Therefore we have to proceed the other way round.

Given a solution x' for the ISIS problem $G * x' = u \pmod q$ regarding G, we can multiply R on the left side of the solution. This product $R * x' = x$ is also small, because R and x' are small, and it solves the ISIS problem in respect to A, because $A * R = G \mod q$.

$$\begin{aligned} u &= G * x' \\ &= A * R * x' \\ \overset{R*x'=x}{\Longrightarrow} u &= A * x \end{aligned}$$

And because G is the binary decomposition matrix, a possible solution for x' is easy to find. For this, the row concatenation of the results of the *binary* function from paragraph 5.3.5 can be used for each element of v. This vector consists only of 0 and 1, is accordingly short and results multiplied with G in the vector v, as shown in the previous paragraph.

In summary, for lattice-based trapdoor functions we do not necessarily need a pair of matrices A and T_A, satisfying $A * T_A = 0 \mod q$, which is mathematically hard to find. It's enough to know a pair A and R, for which $A * R \mod q = G$ is true. How to generate this is quite simple and works as follows.

Sampling Trapdoors We want to find a pair of matrices $A \in \mathbb{Z}_q^{n \times m_0 + nlogq}$ and $R \in \mathbb{Z}_q^{m_0 + nlogq \times nlogq}$ that satisfies $A * R = G \mod q$, for $G \in \mathbb{Z}_q^{n \times nlogq}$. At the same time the matrix A must be uniformly random, while the length of R must be small. Therefore we first choose a linear independent equally random distributed matrix $A_0 \in \mathbb{Z}^{n \times m_0}$. In addition, we sample a matrix $R_0 \in \{0,1\}^{m_0 \times nlogq}$ with $Bernoulli(\frac{1}{2})$ probability. Our final matrix A is then composed of the columns of A_0 concatenated with the result of $-A_0 * R_0 + G$. And R is selected as concatenation of the rows of R_0 and the identity matrix Id_{nlogq}.

$$A_0 \xleftarrow{\$} \mathbb{Z}^{n \times m_0} \qquad R_0 \xleftarrow{\$} \{0,1\}^{m_0 \times nlogq}$$

$$A = \left(A_0 \mid -A_0 * R_0 + G\right) \qquad R = \left(\begin{array}{c} R_0 \\ \hline Id_{nlogq} \end{array}\right)$$

Because the elements of A_0 multiplied with those of R_0 are erased by the sum of $-A_0 * R_0$ in $-A_0 * R_0 + G$, the product $A * R \mod q$ yields the gadget matrix G. Figure 5.16 illustrates the size and composition of the matrices.

With this procedure, uniformly random matrices A along with a corresponding trapdoor can be generated for LWE or ISIS trapdoor functions. And although A is public, the secret trapdoor R cannot be computed from $-A_0 * R_0 + G$ according to the ISIS problem, the result of $-A_0 * R_0 + G$ is even looking statistically random.

5.3.6 Full-Domain Hash Signatures

The trapdoor functions presented in the previous chapter make it possible to implement further cryptographic primitives with lattices. The simplest of these are digital signatures according to the hash-and-sign paradigm, which were also presented in [GPV08].

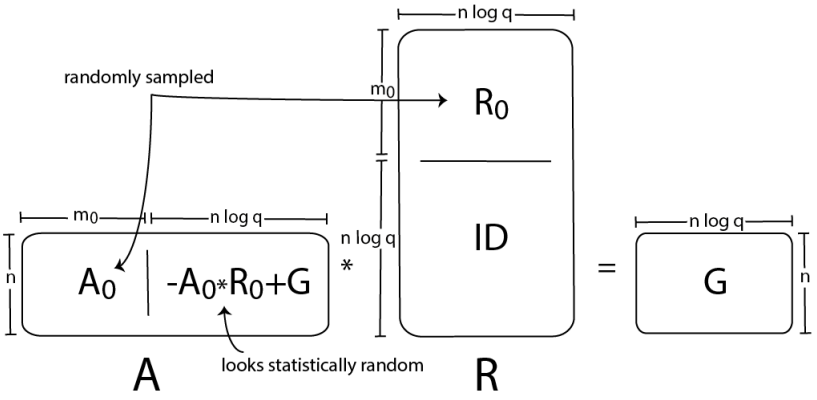


Figure 5.16: The Matrix A and its trapdoor R

Here, the message to be signed is hashed into the value space of an ISIS trapdoor function, and the trapdoor function's preimage of the hash is used as the signature. This signing mechanism is also called Full-Domain Hash (FDH). The verifier can then check the signature by applying it to the trapdoor function and proving that the message originates from the owner of the trapdoor. This way, the trapdoor function f serves as public key, and the inverse function f^{-1} with the help of the trapdoor as signing key.

This section describes how this signature scheme on lattices is working. Figure 5.17 shows an overview of the procedure.

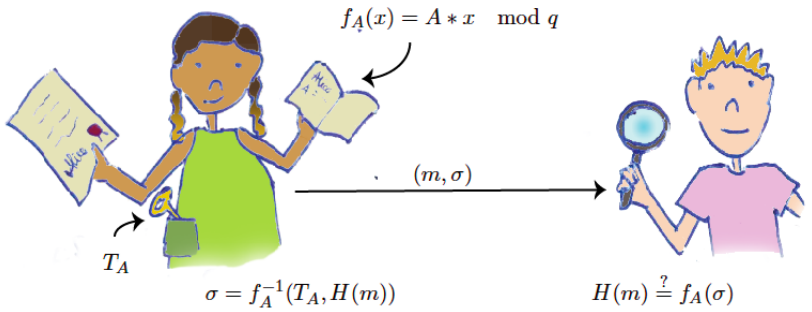


Figure 5.17: Lattice-based FDH Signatures

For the signature keys, a trapdoor function is generated, meaning that a pair of matrices A and T_A is generated, for which $A * T_A = G \mod q$ holds, according to paragraph 5.3.5. The public key then is the trapdoor function $f_A(x) = A * x \mod q$, the signing key is the inverse function $f_A^{-1}(T_A, y) = SolveISIS$. $SolveISIS$ refers to the ISIS solution procedure from chapter 5.3.5. The set of small solution vectors that solve ISIS is denoted with $D_n = \{x : \|x\| \leq \sigma * \omega\sqrt{logm}\} \subset \mathbb{Z}_q^m$. For more details about the parameters σ, ω see [GPV08], but it is enough to know that all elements from D_n are small enough to generate ISIS instances. In addition, as public parameter a hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ exists that maps any

message to the value space of the trapdoor function f_A.

$$\begin{aligned} \textit{public key}: &\quad f_A(x) = A * x \mod q && f_A : \mathbb{Z}_q^m \to \mathbb{Z}_q^n \\ \textit{signing key}: &\quad f_A^{-1}(T_A, y) = SolveISIS && f_A^{-1} : \mathbb{Z}_q^n \to D_n \subset \mathbb{Z}_q^m \\ H: &\quad \{0,1\}^* \to \mathbb{Z}_q^n \end{aligned}$$

To sign a message m, its hash $H(m)$ is computed. The signature σ is the preimage of the trapdoor function for this hash, which is a small vector $x \in D_n$ satisfying $A * x = H(m)$. This can only be computed using the trapdoor T_A, i.e., only by the owner of the public key f_A who is the creator of the matrix pair $A,\ T_A$.

$$\textbf{Sign}(m, T_A,): \quad \sigma = f_A^{-1}(T_A, H(m))$$

Verifying a signature works simply by solving the public trapdoor function of the claimed author for σ and checking if $\|\sigma\|$ is small enough to be an ISIS instance, hence if $x \in D_n$. If the result is small and equal to the hash of the message, the owner of the public key created the signature.

$$\begin{aligned} \textbf{Verify}(\sigma, m, A): \quad & H(m) \stackrel{?}{=} f_A(\sigma) = A * \sigma \mod q \\ & \|\sigma\| \stackrel{?}{\in} D_n \Leftrightarrow \|\sigma\| \stackrel{?}{=} small \end{aligned}$$

The verification can be accomplished by simple matrix-vector multiplication and is an example of what makes lattice-based cryptography that promising. Besides this signature scheme enables lattice-based IBE.

5.3.7 Identity-based Encryption

Identity-based Encryption was introduced also in [GPV08]. It uses ISIS trapdoor functions from section 5.3.5 for the key extraction, and for the encryption it uses the dual public-key encryption scheme from section 5.15. For an introduction to Identity-based Cryptography (IBC) see chapter 4.2. Figure 5.18 gives an overview of the scheme.

For the IBE scheme, the PKG generates a trapdoor function f_A with matrices $A \in \mathbb{Z}_q^{n \times m}$ and the corresponding T_A, according to chapter 5.3.5. A is used as master public key MPK and $T_A \in \mathbb{Z}_q^{m \times nlogq=k}$, as master secret key MSK. In addition, the system requires a publicly known hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ that maps any string to the result space of the ISIS problem $A * x = u \mod q$. Thus, every id can be mapped to an ISIS syndrome u.

$$\begin{aligned} &\textbf{Setup}(): \\ &MPK : A \in \mathbb{Z}_q^{n \times m} \\ &MSK : T_A \in \mathbb{Z}_q^{m \times k} \ \text{ with } A * T_A = G \bmod q \ \text{ and } \ \|T_A\| \text{ is small} \\ &H : \{0,1\}^* \to \mathbb{Z}_q^n \end{aligned}$$

The private key sk_{id} of a user is computed using the trapdoor T_A. For this, the PKG solves the ISIS problem $f_A(x) = A*x \mod q = H(id)$ as described in section 5.3.5. Thus the result is from the set $D_n \subset \mathbb{Z}^m$ that contains all short vectors x solving $A*x \mod q = H(id)$, and

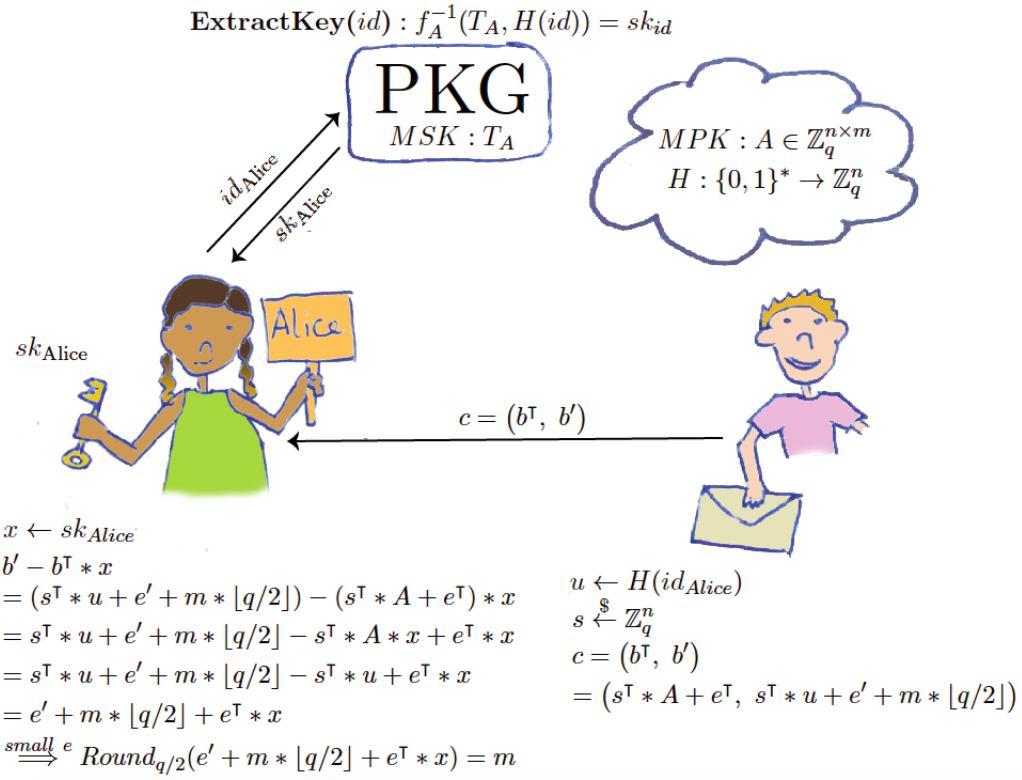


Figure 5.18: Lattice-based Identity-based Encryption

is chosen as sk_{id}. Hence, the sk_{id} is a FDH signature of the id signed by the PKG. For FDH signatures see section 5.3.6.

$$\textbf{ExtractKey}(id) : f_A^{-1}(T_A, H(id)) = sk_{id} \in D_n \subset \mathbb{Z}^m$$

To send a message $m \in \{0,1\}$ to a user with identity id, the dual cryptosystem from section 5.3.4 is used, except that the public key u is the result of $H(id)$. Thus, the cipher consists of the tuple $c = (b^\intercal, b')$.

$$\textbf{Enc}(m, id, A) :\ c = \left(b^\intercal = s^\intercal * A + e^\intercal,\ b' = s^\intercal * H(id) + e' + m * \lfloor q/2 \rfloor\right) \text{ with } s \xleftarrow{\$} \mathbb{Z}_q^n$$

For decryption also the procedure of the dual cryptosystem is used, whereby the secret key sk_{id} is used as x.

$$\textbf{Dec}(b', b^\intercal, sk_{id}) :\quad m = Round_{q/2}(b' - b^\intercal * sk_{id})$$

For an explanation of why the dual cryptosystem works, as long as $A * x = u \mod q$ and $\|x\|$ is small, see chapter 5.3.4. The usage of the Regev Public-Key Encryption from chapter 5.3.3 is not possible, because there would be less possible public keys, and so a function H could not be realized [Vai15b].

5.3.8 Lyubashevsky Signatures

Another signature scheme was presented by Lyubashevsky in [Lyu12]. It has average-case security based on the worst-case hardness of SIVP, and it has similarities to Schnorr signatures [Sch91] and the elliptic curve Schnorr signature scheme from section 4.1.4. Figure 5.19 gives an overview of the signature scheme.

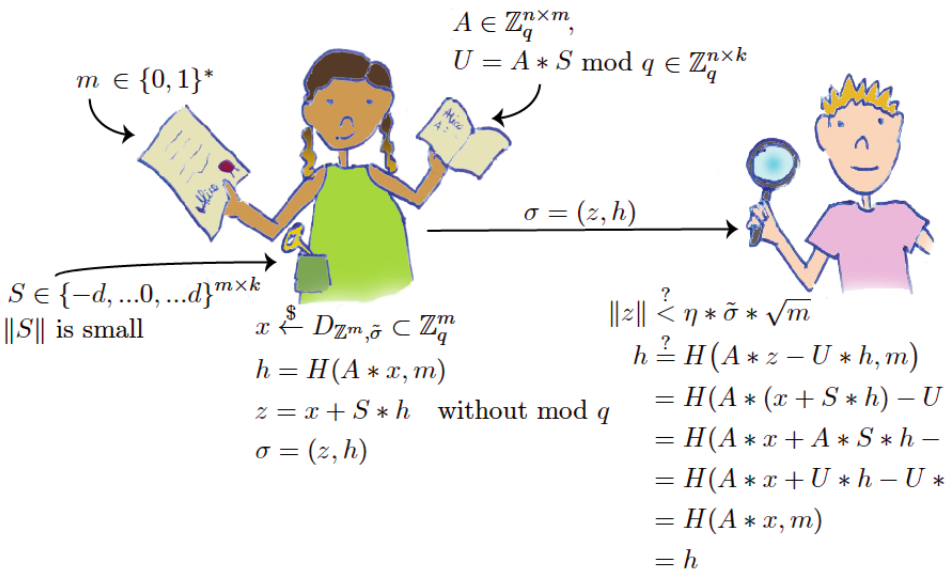


Figure 5.19: Lattice-based Lyubashevsky Signatures

For the key pair, a small sized matrix $S \in \{-d,...0,...d\}^{m\times k}$ is chosen as the secret key. The elements of S all have absolute values smaller than a parameter d. Thus the matrix consists of k short vectors from $D_n \subset \mathbb{Z}_q^m$. The public key consists of a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$ and the product of A and S, named U. Based on the ISIS assumption, the small sized S cannot be reconstructed from A and U, and remains secret. Additionally a hash function is used, that maps binary values to short vectors from $\{v : v \in \{-1,0,1\}^k \wedge \|v\| \leq \kappa\} \subset \mathbb{Z}_q^k$. For the size of the constant κ see [Lyu12], but it is enough to know that the results of the hash function are short.

$$
\begin{aligned}
&\textit{secret key} : S \in \{-d,...0,...d\}^{m\times k} \text{ such that } \|S\| \text{ is small}\\
&\textit{public key} : \left(A \in \mathbb{Z}_q^{n\times m},\ U = A * S \in \mathbb{Z}_q^{n\times k}\right)\\
&\qquad H : \mathbb{Z}_q^n \times \{0,1\}^* \to \{v : v \in \{-1,0,1\}^k \wedge \|v\| \text{ is small}\}
\end{aligned}
$$

To sign a message $m \in \{0,1\}^*$, a random but short vector $x \in \mathbb{Z}^m$ is sampled from a distribution $D_{\mathbb{Z}^m,\tilde{\sigma}}$. This Gaussian distribution is used to sample small m-dimensional integer vectors with a standard deviation of $\tilde{\sigma}$. For more details see [Lyu12], but it is enough to know that x is a small vector from $x \in \mathbb{Z}^m$. Then, typical for Schnorr signatures, this x is hidden in the public part A and hashed together with the message m to the value $h \in \mathbb{Z}^k$. The short h is multiplied to the short secret key S and the short x is added to the product. This is done without modulo reduction. The result, called z, together with h form

the signature $\sigma = (z, h)$.

$$\begin{aligned}\textbf{Sign}(m, S): \quad & x \xleftarrow{\$} D_{\mathbb{Z}^m, \tilde{\sigma}} \subset \mathbb{Z}_q^m \\ & h = H(A * x, m) \\ & z = x + S * h \quad \text{without mod } q \\ & \sigma = (z, h)\end{aligned}$$

To verify the signature σ the part A of the public key is multiplied to z. Then U is multiplied to h, and subtracted from the previous result. The hash of this together with the apparent message m must equal h. Additionally the length of z must be smaller than $\eta * \tilde{\sigma} * \sqrt{m}$ for the signature to be valid, because only then one can be sure, that the original small S was used during signing. Otherwise an adversary could have generated some larger S' satisfying the equation $A * S' = U$ which is not hard, and signed in the name of the owner of the public key A, U. For details about the constant η see [Lyu12].

$$\begin{aligned}\textbf{Verify}(\sigma, m, A, U): \quad & h \stackrel{?}{=} H\big(A * z - U * h, m\big) \\ & \|z\| \stackrel{?}{<} \eta * \tilde{\sigma} * \sqrt{m}\end{aligned}$$

The verification works as long as the same message m, public key A and secret key S are used for signing and verifying, because only then the following equation holds.

$$\begin{aligned}H(A * z - U * h, m) &= H(A * (x + S * h) - U * h, m) \\ &= H(A * x + A * S * h - U * h, m) \\ &= H(A * x + U * h - U * h, m) \\ &= H(A * x, m) \\ &= h\end{aligned}$$

5.3.9 Identity-based Signatures

As the final stage of this lattice-based cryptography chapter the focus of this thesis, the Identity-based Signatures (IBS) on lattices have been reached. These were developed some years after IBE, for example in [Rüc10], [THY13] or [LHZL13], all using the trapdoors from [Vai15b] and lattice basis delegation [CHKP12] to generate the signing keys.

The idea of Tian and Huang in [TH14] requires only trapdoor functions with lattices to generate the signing keys. It uses the FDH signatures from section 5.3.6 to create the user secret keys, and the signatures are following the Lyubashevsky signature scheme from section 5.3.8. Figure 5.20 shows the overall procedure.

The IBS requires a trapdoor function from chapter 5.3.5, where $A \in \mathbb{Z}_q^{n \times m}$ is used as master public key MPK, and $T_A \in \mathbb{Z}_q^{m \times nlogq=k}$ as master secret key MSK. It must hold, that $m > 5 * nlogq$ and $q \geq 3$ is prime. In addition, a hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_q^{n \times k}$ that maps any string to a matrix, and a hash function $H : \{0,1\}^* \to \{v : v \in \{-1, 0, 1\}^k \wedge \|v\| \leq \eta\}$

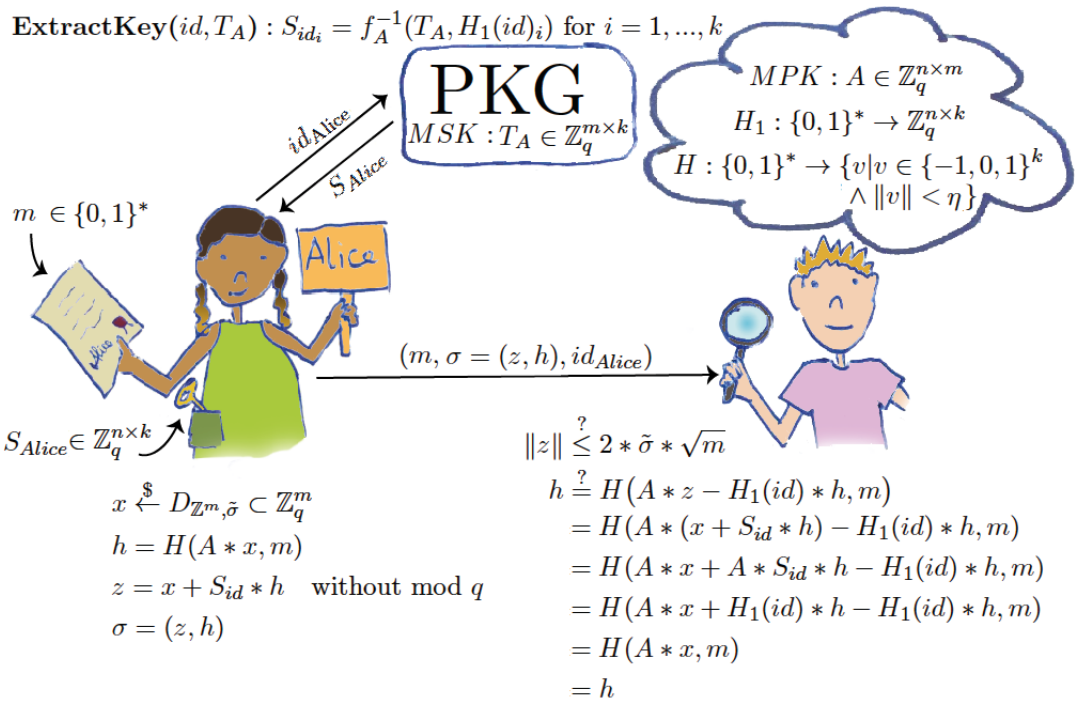


Figure 5.20: Lattice-based Identity-based Signatures

that maps any string to a short vector with length of at most the constant $\eta \in \mathbb{N}$, is needed.

Setup() :

$MPK : A \in \mathbb{Z}_q^{n \times m}$

$MSK : T_A \in \mathbb{Z}_q^{m \times k}$ with $A * T_A = G \bmod q$ and $\|T_A\|$ is small

$H_1 : \{0,1\}^* \to \mathbb{Z}_q^{n \times k}$

$H : \{0,1\}^* \to \{v : v \in \{-1,0,1\}^k \wedge \|v\| \text{ is small}\}$

To assign a private signing key sk_{id} to a user with identity id, the PKG calculates the hash $H_1(id)$ and solves the ISIS problem $A * x_i = u_i \mod q$ for all $i = 1, ...k$ columns of $H_1(id)$. This means the PKG creates k FDH signatures. Each result $x_i \in D_n = \{x : \|x\| \leq \hat{\sigma} * \omega\sqrt{logm}\} \subset \mathbb{Z}_q^m$ is then concatenated to a new matrix $S_{id} \in \mathbb{Z}_q^{m \times k}$, which will satisfy $A * S_{id} = H_1(id) \mod q$ and has a small length $\|S_{id}\| \leq \hat{\sigma} * \omega\sqrt{logm}$. For the value of $\hat{\sigma}, \omega$ see [TH14]. This will be the user's signing key. Note that only the PKG can calculate such short vectors with the help of the trapdoor $T_A = MSK$.

ExtractKey$(id, T_A) : S_{id_i} = f_A^{-1}(T_A, H_1(id)_i)$ for $i = 1, ..., k$

$S_{id} \in \mathbb{Z}_q^{m \times k}$

$\|S_{id}\|$ is small

To sign a message $m \in \{0,1\}$ the Schnorr-like signatures of Lyubashevsky are used. A random but short vector $x \in \mathbb{Z}^m$ is sampled from a distribution $D_{\mathbb{Z}^m, \tilde{\sigma}}$. This Gaussian distribution is used to sample small m-dimensional integer vectors with a standard deviation

of $\tilde{\sigma}$. For the value of $\tilde{\sigma}$ see [TH14]. Then the hash $h = H(A * y, m)$ is computed, which hides the x on the one hand because of the collision-resistance of H, and on the other hand because of the ISIS assumption, according to which one cannot find a small x from $A * x$ mod q. Also, the same x is hidden in the equation $z = S_{id} * h + x$. The signing key S_{id} can not be reconstructed from z because of LWE. The final signature consists of the tuple $\sigma = (z, h)$.

$$\begin{aligned} \textbf{Sign}(m, S_{id}): \quad & x \xleftarrow{\$} D_{\mathbb{Z}^m, \tilde{\sigma}} \subset \mathbb{Z}_q^m \\ & h = H(A * x, m) \\ & z = x + S_{id} * h \quad \text{without mod } q \\ & \sigma = (z, h) \end{aligned}$$

To verify a signature, the h is recomputed by the receiver using the public $MPK = A$ and id of the supposed author, and compared to the h that was received. Additionally it checks whether $\|z\| \leq 2 * \tilde{\sigma} * \sqrt{m}$. This way it can be assured, that S_{id} was short, and really assigned to the user by the PKG.

$$\begin{aligned} \textbf{Verify}(\sigma, m, id, A): \quad & h \stackrel{?}{=} H\big(A * z - H_1(id) * h, m\big) \\ & \|z\| \stackrel{?}{\leq} 2 * \tilde{\sigma} * \sqrt{m} \end{aligned}$$

By multiplying the MPK A to the value z we get the value of $A * S_{id} * h + A * x$. According to the properties of our signing key, this is equal to $H_1(id) * h + A * x$. By subtracting $H_1(id) * h$, which can also be calculated using the public parameters, we get $A * x$ allowing the comparison $H(A * x, m) = h$. This signature cannot be misused by strangers, because during verification $A * S_{id} = H(id)$ must hold and z remains short. Moreover, the signing key remains hidden under the ISIS assumption.

$$\begin{aligned} H(A * z - H_1(id) * h, m) &= H(A * (x + S_{id} * h) - H_1(id) * h, m) \\ &= H(A * x + A * S_{id} * h - H_1(id) * h, m) \\ &= H(A * x + H_1(id) * h - H_1(id) * h, m) \\ &= H(A * x, m) \\ &= h \end{aligned}$$

As now can be seen, this lattice-based IBS scheme would have to create a completely new trapdoor function as master key pair to revoke a single user key. Additionally it would have to create new signing keys for all non-revoked users and distribute them over secret channels. Also for lattice-based IBS in particular there are approaches of making signing key revocation more efficient. The next section gives an overview about the state of the art on lattice-based Revocable Identity-based Signatures (RIBS).

5.4 Related Work on Revocable Identity-based Signatures with Lattices

There has been research on key revocation specifically for identity-based signature schemes based on lattices. The ideas are based on the general revocation ideas for identity-based

signature schemes, that were already presented in section 4.3. It shall be mentioned, that every IBS supports user key revocation, by adding validity period timestamps to the identity strings, following the proposal of Boneh and Franklin [BF01]. The ideas presented in the following enable more efficient revocation. Unlike earlier IBS schemes based on pairings like [GG09], the lattice-based IBS schemes are considered to be quantum secure, and are therefore more promising. The IBS schemes that enable efficient user key revocation are called Revocable Identity-based Signatures (RIBS).

The first lattice-based RIBS scheme was introduced by Xiang in 2015 [Xia15]. The lattice-based IBS issues new signing keys for every validity period following the procedure of Boldyreva et al. [BGK08], that already was introduced in section 4.3. Here validity periods are created by appending a timestamp to the identity string as proposed by Boneh and Franklin in [BF01], and encrypting the new keys within a binary data structure. Therefore the workload for the PKG is reduced to the logarithm of the number of non-revoked users. This scheme requires private channels for distributing the key update, and the size of signing keys grows logarithmically, which makes the scheme inefficient [HTH17].

In 2017, Hung et al. [HTH17] designed a RIBS on lattices that adapts the idea of Tseng and Tsai [TT12], also already introduced before. The signing key consists of two components, a fixed initial signing key and a changing time update key. At the beginning of each new validity period, new time update keys are transmitted to each non-revoked user over a public channel. The verification will only work if the signing key and the new time update key were used for the signature. This RIBS scheme is based on the SIS problem. Both keys are chosen with Gaussian sampling over NTRU lattices, and the signatures follow Lyubaschevski's rejection sampling technique from section 5.3.8. In this case, instead of one short signing key, a tuple of two such signing keys is used. The time update key is another tuple with the same properties, but with the time stamp of the validity period hashed inside, it is regularly calculated and distributed by the PKG. The signature uses both signing keys in combination with both time update keys, resulting in a larger signature, a three tuple, compared to the only naive revocable IBS from chapter 5.3.9. An advantage of this method is that signature keys and signatures have still quite small sizes, and instead of calculating all signing keys newly and distributing them privately, only the time update keys have to be calculated and can be public, which reduces computational effort same as required bandwidth. Yet, this method is insecure under signing key exposures [XWW20].

In 2018, Langois et al. [LLNW14] published an IBS scheme with verifier-local revocation (VRL), claiming to have constructed the first quantum-safe VRL IBS. Only the verifiers receive revocation information, and not the signers. The idea follows the premise that less users verify compared to those that sign, and thus computation time and network traffic can be reduced, when only they need update tokens for new validity periods. The scheme is based on the ISIS problem and uses Bonsai tree hard lattices [CHKP12] to create zero-knowledge proofs as signatures. Users receive group secret keys and sign with them in combination with the public MPK. For verification, beside the master public key revocation tokens are used. Although those need to be securely transmitted to verifiers, it is claimed that computation and network bandwidth are still reduced compared to alternative RIBS.

In 2020, Xie et al. [XWW20] present a RIBS based on the tree structure of Boldyreva et

al. [BGK08] and the left-right lattices and delegation technology of Agrawal et al. [ABB10a], but this is updatable via public channels. It is considered the first RIBS on lattices with signing key exposure resistance. At the beginning of each validity period t an update key KU_t is generated and distributed to all users via public channels. The users can update their signing key SK_{ID} with basis delegation [ABB10b] using the KU_t. Signing and verification is performed using the validity period t and the current signing and public keys. The keys and signature sizes of this RIBS are larger than those of Hung et al. [XWW20]. Similar as in [CLL+12], a binary tree revocation list is maintained by the PKG for choosing the update key KU_t. Because of this logarithmic complexity to the number of users combined with the distribution of the update keys over a public channel, this method provides a highly scalable solution for RIBS compared to the previous approaches.

In 2022, He et al. [HQG+22] present a generic construction method of forward-secure RIBS applicable on lattice-based IBS constructed on the SIS problem. Here, similar to [Gug20], the signing keys of the users are updated by a token. The validity period t is included in the signature and verification process. The disadvantage of this approach is that the key lengths grow logarithmically in relation to the number of validity periods.

Beside all the presented approaches the Key Updatable Signature Schemes (KUSS) of Guggemos from [Gug20] that were introduced in section 4.4, would preserve the size of all keys, without changing the complexity of the signature and verification procedures. Although a private channel is needed to distribute the update tokens, this could give an advantage to the RIBSs mentioned here. For this reason, the following chapter investigates how an existing IBS on lattices can be transformed into a KUSS, even if lattices do not form cyclic groups as it is the case for existing elliptic curve KUSS.

5.5 Summary

This section gave an introduction to lattices and their geometrical meaning. Furthermore the computational hard problems on lattices were introduced, of which SIS and LWE are even proven average-case hard, exactly as needed for cryptography. Based on this, several cryptographic schemes were explained, finally leading to an Identity-based Signature scheme constructed on lattices. Besides, existing efficient revocable IBS schemes were presented.

The next section will finally deal with the main goal of this thesis. It discusses possibilities of how the lattice-based IBS can be converted to an efficiently revocable KUSS.

6 A concept for KUSS using Lattices

After a lot of background knowledge was introduced, this chapter finally addresses the main question of this thesis, namely how the lattice-based IBS from Tian and Huang [TH14], which was introduced in section 5.3.9, can be converted into an efficiently revocable Key Updatable Signature Scheme (KUSS) from section 4.4.

The advantages of KUSS are that they allow signing key revocation that reduces the computational load of the PKG and lowers the network traffic, while it preserves the key and signature sizes and therefor the security of the original IBS. Further this revocation mechanism achieves forward and post compromise security. The advantages of lattice-based cryptography lie in its so-far quantum resistance and its lightweight matrix operations. By combining the schemes, all those beneficial properties could be achieved in one.

Guggemos demonstrated the concept of KUSS among others on the elliptic curve IBS by Galindo and Garcia [GG09], which uses the concept of Schnorr signatures. The lattice-based IBS by Tian and Huang was chosen for the following investigation because it uses Lyubashevsky signatures. These are similar to the Schnorr signature scheme, as can be easily seen by comparing figures 4.6 and 5.19. Therefore also both IBS schemes are similar in some ways, as can be seen by comparing figures 4.9 and 5.20.

At first, this chapter explains possibilities to transform the lattice-based IBS scheme into 2KSS and U2KSS schemes, following the idea of Guggemos in [Gug20] on how to transform elliptic curve IBS schemes into KUSS. Since lattices do not form cyclic groups, the transformation into KUSS according to the approach of Guggemos turned out to be difficult. Thus, in the second section, the requirements which must be fulfilled to ensure an intact KUSS are analysed and are compared with the lattice-based IBS. Finally, the chapter concludes by summarizing the findings of the requirements analysis, pointing out what still is needed for obtaining a fully functional KUSS.

6.1 Lattice-based 2KSS and U2KSS

In order to achieve signing key revocation, it is possible to include an additional symmetric gsk in the signing and verification process of the IBS by Tian and Huang that changes with every epoch, the same way as for the IBS by Galindo and Garcia from section 4.4.2. This way a lattice-based Two Key Signature Scheme (2KSS) and Updatable Two Key Signature Scheme (U2KSS) is constructed.

2KSS By multiplying a symmetric secret gsk to the result of $A * y$ during signing, leading to the signature $\sigma = (h = H(gsk * A * y, m), z = S_{id} * h + y)$, a 2KSS is achieved by changing the first step of the verification to $h \stackrel{?}{=} H(gsk * (A * z - H_1(id) * h * h), m)$. This way the size of a valid signature stays the same, holding $\|z\| \leq 2 * \tilde{\sigma} * \sqrt{m}$, because only the input to the hash function changes, and the output remains same sized. The signature can still only be generated by the owner of a short S_{id} and thus is trusted by the PKG. Further verification

is only successful if also the gsk was the same for signing and verifying, as the following equation shows.

$$\begin{aligned} H(gsk * (A * z - H_1(id) * h), m) &= H(gsk * (A * (S_{id} * h + y) - H_1(id) * h), m) \\ &= H(gsk * (A * S_{id} * h + A * y - H_1(id) * h), m) \\ &= H(gsk * (H_1(id) * h + A * y - H_1(id) * h), m) \\ &= H(gsk * A * y, m) \\ &= h \end{aligned}$$

Because the result is mapped to a hash, the dimension of $gsk * A * y$ can be different to $A * y$ without violating the transformability condition from section 4.4.1. As long as the output of the hash function remains from the same set, the signature stays the same. Thus the gsk can be a scalar $\in \mathbb{Z}$, a horizontal vector $\in \mathbb{Z}^n$ or a matrix $\in \mathbb{Z}^{n \times n}$, as long as the hash function H is collision-resistant.

U2KSS The gsk_e is updated by a token $\Delta_{e+1} \in \mathbb{Z}_q$ no matter what dimensions gsk_e has.

$$gsk_{e+1} = \Delta_{e+1} * gsk_e$$

Because gsk_e is not part of any public parameter, but only the input of a cryptographic hash function, it does not matter that the knowledge of gsk_e and gsk_{e+1} allow the calculation of Δ_{e+1}. Two signatures σ_e and σ_{e+1} will not reveal any information about Δ_{e+1} because of the collision-resistance of the hash function.

The difficulty within transforming the lattice-based IBS into a KUSS is that the keys are not correlated in such way that one is a multiple of the other, as it is the case for IBS schemes on cyclic groups like the one from Galindo and Garcia using elliptic curves. There, because usk and mpk are multiples of the msk and the multiplication is a transitive relation, the gsk and update tokens Δ could be multiplied on all of them, to make the scheme work in the same way with new but still matching keys. The master key pair of the lattice-based IBS consists of a random matrix and its trapdoor as introduced in section 5.3.5. They behave rather similar to inverse of each other. Multiplying the same token on both will break it's trapdoor properties, and therefor the whole functionality of the scheme. To construct a KUSS on such an IBS the requirements will be analyzed from top to bottom, and checked for applicability.

6.2 Lattice-based KUSS Requirements Analysis

The following section analyzes what properties are required for revocation with token updates in a KUSS, and examines how and to what degree these properties are met in the lattice-based IBS of Tian and Huang [TH14] from section 5.3.9. In particular it is discussed what set the update token Δ must be from, and where it must be applied, to fulfill all requirements. The gsk_0 will then be set to a suitable element of the same set. Table 6.1 summarizes which possible Δ meets which requirements for a KUSS.

In a nutshell it is searched for an additional secret key gsk_e that is passively used in the signing and verification process, by including it in the calculation of the master public key

	$gsk, \Delta \in$	$\mathbb{Z}_q$	$(\mathbb{Z}_q^m)^\intercal$	$\mathbb{Z}_q^{m\times k}$	$\mathbb{Z}_q^{m\times m}$	$\{0,1\}^{m\times m}$
	Operation	$*$	$*$	$+$	$*$	$*$
S_{id} remains same sized	$S_{id_{e+1}} \in \mathbb{Z}_q^{m\times k}$	✓	✗	✓	✓	✓
$A * S_{\text{id}}$ remains $H_1(id)$	$A_{e+1} * S_{id_{e+1}} = H_1(id)$	✓		✗	✓	✓
Δ hidden in $A * \Delta^{-1}$	$A_e, A_e * \Delta^{-1} \not\Rightarrow \Delta^{-1}$	✗			✓	✓
Extract S_{id} in later epoch	$extract\_key(id, epoche)$				✓	✓
Verification equation	$h_e \stackrel{?}{=} H(A_e * z_e - H_1(id) * h_e, m)$				✓	✓
$\|S_{id}\|$ remains small	$\|x + S_{id_{e+1}} * h\| \leq 2 * \tilde{\sigma} * \sqrt{m}$				✗	?

Table 6.1: Possible Δs and their properties

$MPK = A \in \mathbb{Z}_q^{n\times m}$ and the individual signing keys $usk = S_{id} \in \mathbb{Z}_q^{m\times k}$. This must then be updatable by a token Δ_{e+1}. This means that the keys are manipulated according to $\Delta e + 1$ in such way that the included value of gsk_e is changed to a new value gsk_{e+1}. Because gsk_0 is the start of a chain of updated gsk_{e+1}, it will be set to the identity element of the group where Δ is chosen from, and all properties will be discussed only for Δ, representative for the properties of gsk_e as well, because all gsk_e will be from the same group as Δ. Note that within a KUSS the key updates must be achieved in a way that the keys, the signatures, the complexity of the signature and verification processes, and the security of the IBS remain the same. This is only possible if the following requirements are met.

S_{id} remains same sized The first condition for a KUSS from [Gug20] is that the keys of the scheme remain the same size. Given the signing key $S_{id} \in \mathbb{Z}_q^{m\times k}$, in order to get $\Delta * S_{id}$ with the same dimensions, it is only possible to choose Δ either as a scalar from $\mathbb{Z}_q$, or as a square matrix from $\mathbb{Z}_q^{m\times m}$. A vector or a matrix of other dimensions would result in a $\Delta * S_{id}$ with a different size. In order to get an updated signing key $\Delta + S_{id}$ from $\mathbb{Z}_q^{m\times k}$, the Δ must have the same size as S_{id}. Thus, when *multiplying* Δ to S_{id}, it can not be a vector or a differently dimensioned matrix than $\mathbb{Z}_q^{m\times m}$. When *adding* Δ, it must be from the set $\mathbb{Z}_q^{m\times k}$.

$A * S_{id}$ remains $H_1(id)$ Another condition is that the signing key S_{id} multiplied to the right of the public key A must result in the hash $H_1(id)$ for the verification to work. In other words, the equation $A * S_{id} = H_1(id)$ must still be satisfied after the key update, therefor $A_{e+1} * S_{e+1} = A_{e+1} * \Delta * S_{id_e} = H_1(id)$ or $A_{e+1} * S_{e+1} = A_{e+1} * (\Delta + S_{id_e}) = H_1(id)$. As is seen in the equation, the public key A needs to be updated as well to meet this condition, but contrary to the procedure for elliptic curves, the same Δ cannot simply be multiplied to A as well. The equation holds if two inverse elements are placed around the multiplication, leading to $A_e * \Delta^{-1} * \Delta * S_{id_e} = H_1(id)$. During an update the $MSK = A$ is also updated by calculating $A_{e+1} = A_e * \Delta^{-1}$. Because Δ^{-1} and Δ have to be inverse regarding multiplication, an update token from the group $(\mathbb{Z}_q^{m\times k}, +)$ is no option anymore.

Δ hidden in $A*\Delta$ To ensure that the update token Δ remains a secret only between non-revoked users although Δ^{-1} is contained in the public key A, it must be ensured that it is hidden in A, i.e., it can not be calculated from A_e and A_{e+1}. Otherwise revoked users could compute Δ from Δ^{-1} and the public parameters, and update their keys secretly, although they are not trusted by the PKG anymore. Thus, it is necessary that if A_e and

$A_{e+1} = A_e * \Delta^{-1}$ are given, the Δ must not be derivable from them. The operation $A * \Delta^{-1}$ must therefore be a one-way function.

Suppose Δ is a scalar from $\mathbb{Z}_q$, then the Δ can be easily recovered from $A_{e+1} = A_e * \Delta^{-1}$ and A_e, by simply dividing any element of matrix A_{e+1} through the same positioned element of matrix A_e, i.e $A_{e+1_{ij}} * (A_{e_{ij}})^{-1} = A_e * \Delta^{-1}$. Because the elements of the matrix are from the field $(\mathbb{Z}_q, +, *)$, there exist inverse elements for multiplication. Thus a $\Delta \in \mathbb{Z}_q$ is no candidate for the KUSS.

Another option for Δ is a full-rank matrix $M \in \mathbb{Z}_q^{m \times m}$ that therefor is invertible. In this case $A_{e+1} = A * M^{-1}$ and A_e must not allow reconstruction of M^{-1}. According to the ISIS assumption from section 5.2.1, if $A \in \mathbb{Z}_q^{n \times m}$ and $A * S = U \in \mathbb{Z}_q^{m \times k}$ with small $\|S\|$ are given, then it is hard to find any short S. So a $\Delta = M \in Z_n^{m \times m}$ with small $\|M^{-1}\|$ would be a candidate for the KUSS.

Further the ISIS problem implies, that it is possible to find some S solving $A * S = U$, but not such with small $\|S\|$. Based on this it is assumed that if there are several solutions S' for an equation $A * S = U$, then there also are several solutions $M^{-1'}$ for the equation $A_e * M^{-1} = A_{e+1}$ independent of the size of $\|M^{-1}\|$. Another point of view on the same assumption is, that with A being not quadratic, but having more columns than rows, i.e., $m > n$, the linear equation system presented by this matrix has more unknown variables than equations (see section 2.2.2). Now to reconstruct M^{-1}, one would solve the equation system, but since there are more unknowns than equations there is more than one possible solution for M^{-1}. Roughly approximated, with $A \in \mathbb{Z}_q^{n \times m}$ there are up to q^{m-n} possible solutions, because the unknowns that are not solved during Gaussian elimination could take any value of the definition range $\mathbb{Z}_q$. Of these multiple possibilities $M^{-1'}$ the actual $\Delta^{-1} = M^{-1}$ is only one, so the probability for adversaries to determine the actual Δ^{-1} is $\frac{1}{q^{m-n}}$ and is considered negligible. Accordingly, a $\Delta = M \in \mathbb{Z}_q^{m \times m}$ with an arbitrary large $\|M^{-1}\|$ remains as well candidate for the KUSS, so far allowing any full-rank $M \in \mathbb{Z}_q^{m \times m}$. In order to support this assumption it must be investigated whether the M^{-1} can be reconstructed using a left inverse of A by calculating $A_{el}eft^{-1} * A_{e+1} = A_{e_{left}}^{-1} * A_e * M^{-1} = M^{-1}$.

Since it is possible to hide the Δ in the public key, according to the ISIS assumption, it is investigated if this is also feasible with LWE. LWE hides an $S \in \mathbb{Z}_q^{k \times n}$ and $E \in \mathbb{Z}_q^{k \times m}$ in the sum $S * A + E$, even if $A \in \mathbb{Z}_q^{n \times m}$ and $S * A + E \in \mathbb{Z}_q^{k \times m}$ are known. In a KUSS that updates A with a $\Delta^{-1} = (S, E)$ by calculating $A_{e+1} = S * A_e + E$, the S_{id}s must be updated in a way satisfying $A_{e+1} * S_{id_{e+1}} = (S * A_e + E) * S_{id_{e+1}} = H_1(id)$. This approach seems much more complicated and therefore has not been explored further.

Extract S_{id} in later epoch Furthermore the KUSS must allow the issuing of new signing keys at a later epoch for newly joining users. Thus key extraction using the master secret key must be possible also after the master public key was updated.

In the lattice-based IBS the master secret key T_A is a trapdoor for the ISIS problem with respect to the master public key A, holding $A * T_A = G$ and small $\|T_A\|$. After an update, the new master public key is $A * M^{-1}$. To preserve the functionality of key extraction, the master secret key T_A would have to be manipulated to $T_{A_{e+1}}$, now satisfying the trapdoor properties in respect to A_{e+1}. To fulfill the equation $A_{e+1} * T_{A_{e+1}} = A_e * M^{-1} * T_{A_{e+1}} = G$, the matrix M could be multiplied to the left side of T_{A_e}. However, depending on the choice of M, the size of $\|T_A\|$ will grow during an update, causing the $T_{A_{e+1}}$ to lose its trapdoor properties. Whether such an update of the master secret key T_A is possible would therefore

depend on the selection of $\Delta = M$, which will be discussed later in more detail.

A more straightforward alternative to enable key extraction in later epochs is to let the PKG use the same master secret key T_A over the whole time, and store the first master public key A_0 and all previous update tokens $(\Delta_1, ..., \Delta_e)$. When a new user joins, the PKG computes the S_{id_0} for the user using the original A_0 and the trapdoor T_A, and then updates S_{id_0} using the stored update tokens up to the current validity period e, before transferring it to the user. This is the same as storing the gsk and updating it according to every new Δ_{e+1}, by calculating $gsk_{e+1} = \Delta_{e+1} * gsk_e = \Delta_{e+1} * \Delta_e * ... * \Delta_1 * gsk_0$. This mechanism does not further restrict the choice of Δ.

At this stage it was found that a KUSS with an update token $\Delta = M \in Z_q^{m \times m}$ and M being full-rank could be possible. Another condition of a KUSS is that the verification is working after at most slight modifications preserving the same complexity, and after the token updates the same security must be guaranteed as before. The verification in the lattice-based IBS by Tian and Huang is performed in two steps. Both are investigated separately.

Verification equation holds One step of the verification is checking that $h = H(A * z - H_1(id) * h, m)$. After an update, any non-revoked user signing key is $S_{id_{e+1}} = \Delta * S_{id_e}$. The verification equation still works with the updated keys.

$$\begin{aligned} H(A_{e+1} * z_{e+1} - H_1(id) * h_{e+1}, m) &= H(A_{e+1} * (S_{id_{e+1}} * h_{e+1} + y) - H_1(id) * h_{e+1}, m) \\ &= H(A_{e+1} * S_{id_{e+1}} * h_{e+1} + A_{e+1} * y - H_1(id) * h_{e+1}, m) \\ &= H(H_1(id) * h_{e+1} + A_{e+1} * y - H_1(id) * h_{e+1}, m) \\ &= H(A_{e+1} * y, m) \\ &= h \end{aligned}$$

The verification only works if $S_{id_{e+1}}$ fulfills $A_{e+1} * S_{id_{e+1}} = H_1(id)$, and the message during signing and verification is the same. Additionally, for users with not updated keys, the function does not hold and the verification fails, because $A_{e+1} * S_{id_e} \neq H_1(id)$. Thus the sketched KUSS works regarding this requirement.

S_{id} remains short The second part of the verification is to check that $\|z\| \leq 2 * \tilde{\sigma} * \sqrt{m}$. Because during signing h and x are short, $z = x + S_{id} * h$ is also short as long as S_{id} is short. And according to ISIS, only the PKG can calculate a short matrix S_{id} by using the master secret key, the trapdoor T_A. Thus the requirement of $\|z\| \leq 2 * \tilde{\sigma} * \sqrt{m}$ proves the PKG's trust in the user and gives him his legitimacy. If $\|z\|$ was greater, then the signature was created with a longer S_{id}, and could have been self-created by an adversary. Therefore, the security of the IBS is based on the condition that S_{id} must not exceed a certain size.

For the KUSS this means that a signing key also after an update $\Delta * S_{id_e}$ must not exceed a certain size either. Consequently, the Δ must be a matrix $M \in \mathbb{Z}_q^{m \times m}$ which fulfills the property that the product of it with a small matrix results in a matrix of same or smaller size. This means that the individual elements of the matrices must in average remain the same size or converge towards 0.

If a random $\Delta = M \in \mathbb{Z}_q^{m \times m}$ is chosen, then $\|S_{id}\|$ also grows randomly after an update. This is not allowed.

If $\Delta = M \in \{0,1\}^{m \times m}$, thus consisting only of small elements, then the size of the product remains smaller than in the previous scenario, but still grows. Since each element of $S_{id_{e+1}}$ is the sum of m small elements from S_{id_e}, each new element can grow up to m times the previous elements size. After i updates, the size of an actually legitimate key would already be m^i times larger than originally. Thus, after some updates it will exceed the requirement of $\|S_{id_{e+1}}\|$ being small enough to satisfy $\|z\| \leq 2 * \tilde{\sigma} * \sqrt{m}$ and does not provide security anymore. It remains to be investigated how many iterations of updates are possible until the signing key size exceeds the limit, and the system is not secure.

With this all possible candidates for the Δ were discussed, leaving the solution of an $M \in \{0,1\}^{m \times m}$ and the number of possible updates until violating the length restriction to be investigated.

6.3 Findings

In summary, the lattice-based IBS of Tian and Huang would transform into a KUSS if the update tokens Δ were square full-rank matrices, which multiplied to S_{id} result again in a small matrix. Thus the following properties must be fulfilled.

$$\Delta \in \mathbb{Z}_q^{m \times m} \quad \wedge \quad rank(\Delta) = m \quad \wedge \quad \Delta_{e+1} * S_{id_e} = S_{id_{e+1}}, \text{ with } \|x + S_{id_{e+1}} * h\| \leq 2 * \tilde{\sigma} * \sqrt{m}$$

Assuming such an update token Δ exists, the KUSS works as following. Note that because the set where Δ is from is not defined more precise yet, it is denoted with $\mathbb{Z}_q^{m \times m}$ to specify its dimensions. Further the gsk of the KUSS is chosen as the identity element of the group of which Δ is from, thus $gsk = gsk^{-1} = Id_m$

The MSK of the KUSS still is an ISIS trapdoor T_A with respect to a random A. The MPK_0 is the product of A and the $gsk_0^{-1} = Id_m$, which is the neutral element for multiplication.

Setup() :

$$gsk_0 = Id_m \in \mathbb{Z}_q^{m \times m}$$
$$MPK_0 = A * gsk_0^{-1} \in \mathbb{Z}_q^{n \times m}$$
$$MSK = T_A \in \mathbb{Z}_q^{m \times k} \text{ with } A * T_A = G \bmod q \text{ and } \|T_A\| \text{ is small}$$

If a new key needs to be issued in any epoch e, the PKG can use the trapdoor T_A matching the original matrix $A = MPK_0$ to calculate the key S_{id} for the user, and updates it with the current $gsk_e = \Delta_e * ... * \Delta_1 * gsk_0$. Note that because $MPK_0 = A * gsk_0^{-1}$, for the KUSS $S_{id_0} = gsk_0 * S_{id}$.

ExtractKey(id, T_A, gsk_e) :

$$S_{id_i} = f_A^{-1}(T_A, H_1(id)_i) \text{ for } i = 1, ..., k$$
$$S_{id_e} = gsk_e * S_{id} \in \mathbb{Z}_q^{m \times k}$$

During an update all users compute $\Delta_{e+1} * S_{id_e} = \Delta_{e+1} * \Delta_e * ... * gsk_0 * S_{id} = S_{id_{e+1}}$ to reissue their key. The PKG updates the master public key by computing the inverse of Δ_{e+1}, and multiplying that to the right of MPK_e, leading to $MPK_e * \Delta_{e+1}^{-1} = A * gsk_0^{-1} * ... * \Delta_e^{-1} * \Delta_{e+1}^{-1} =$

MPK_{e+1}. Additionally it calculates the new $gsk_{e+1} = \Delta_{e+1} * gsk_e = \Delta_{e+1} * \Delta_e * ... * \Delta_1 * gsk_0$, that is needed for the key extraction.

Update() :

$$\Delta_{e+1} \xleftarrow{\$} \mathbb{Z}_q^{m \times m}$$
$$MPK_{e+1} = MPK_e * \Delta_{e+1}^{-1}$$
$$gsk_{e+1} = \Delta_{e+1} * gsk_e$$
$$S_{id_{e+1}} = \Delta_{e+1} * S_{id_e}$$

Signing and Verifying will remain the same as before, only now using the S_{id_e} instead of S_{id}, and the MPK_e instead of $MPK = A$. With a Δ fulfilling the requirements from above, first $\|x + S_{id_{e+1}} * h\| \leq 2 * \tilde{\sigma} * \sqrt{m}$ holds, and second the following equation holds.

$$\begin{aligned}
&MPK_e * S_{id_e} \\
&= A * gsk_0^{-1} * \Delta_1^{-1} * ... * \Delta_e^{-1} * \Delta_e * ... * \Delta_1 * gsk_0 * S_{id} \\
&= A * Id_m * S_{id} \\
&= A * S_{id} \\
&= H_1(id)
\end{aligned}$$

Figure 6.1 shows the overall procedure of the outlined lattice-based KUSS.

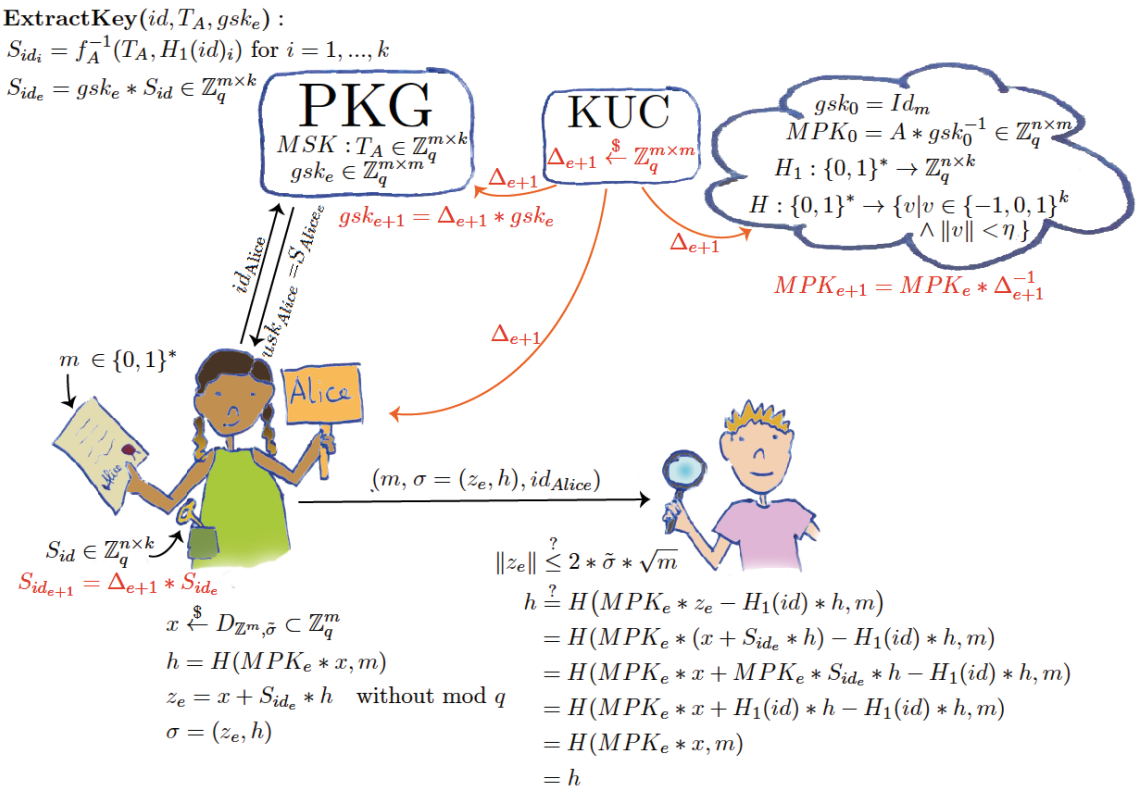


Figure 6.1: KUSS constructed from the lattice-based IBS by Tian and Huang (see figure 5.20). The key updates are marked in red.

6.4 Open Problems

After having sketched an incomplete concept for a KUSS constructed from the lattice-based IBS of Tian and Huang from section 5.3.9, it is summarized what remains to be completed for the KUSS to be fully functional. Furthermore, the assumptions that need to be proven to ensure the security of the system are stated, and an outlook is given on the next steps to be taken in the event that a satisfactory Δ is found and a working KUSS is constructed.

The main goal will be to find a Δ that fulfills the characteristics that have been worked out in the previous requirements analysis, and were summarized in section 6.3. It is necessary to find a matrix that is full-rank, and multiplied to the left of a matrix of small size results in a matrix whose size remains the same, decreases, or grows at most negligibly. In case the matrix grows, then it has to be estimated how many updates can be performed before verification is no longer possible, because the maximum size of S_{id} is exceeded. This is as well to be investigated for the solution with a Δ consisting only from 0s and 1s. Alternatively it must be proven that no such Δ exists.

Further, for the security of the KUSS it is necessary to prove whether, after updating the master public key A_e with a $\Delta = M \in \mathbb{Z}_q^{m \times m}$ to $A_{e+1} = A_e * M^{-1}$, the specific M^{-1} can no longer be recovered from the public matrices A_e and A_{e+1}. For this, it may be necessary to investigate how many inverse, left inverse and pseudo inverse of A exist.

Optionally, it can be investigated whether the issuing of keys for new users can be simplified by updating the msk as well with every key update, instead of using the gsk during key extraction. This way key extraction could be simplified.

Once a Δ is found that matches all requirements needed for the KUSS, the next step is to implement the KUSS as a proof of concept. Based on this implementation, it can be compared whether there actually is a reduction in network load and the PKG's computational effort compared to the original IBS and other revocation mechanisms from chapter 5.4. Alongside a security proof of the system must be made.

7 Conclusion and Future Work

Identity-based Signatures are an alternative to authentication using certificates. Instead of a public key, the unique identity of a user is used for verification, whereby the overhead arising from verifying public key ownership is omitted. The downside of this method is that revocation of user keys causes a large overhead, with the bottleneck in the PKG. Because this is not suitable in dynamic settings, mechanisms exist to make revocation in IBS more efficient, resulting in so-called RIBS.

One of them is the Key Updatable Signature Scheme (KUSS) from [Gug20] by Guggemos, which allows revocation by adding a secret to the keys used during signing and verification processes. For verification this secret is updated by all non-revoked users, using a secret update token. Since the users update their keys themselves, the computational effort of the PKG is decreased and network traffic is reduced. Guggemos presents three steps to transform four different elliptic curve IBS, first into 2KSS, then into U2KSS, and finally into KUSS.

These constructed KUSS rely on the average-case hardness of ECDLP. Due to the intensive research on sufficient error-corrected quantum computers, the security of these systems is threatened. Lattice-based Cryptography is a so-far quantum resistant alternative to common public-key cryptography.

This thesis investigates whether an efficient revocation method for IBS schemes, the so-called KUSS [Gug20], can be applied on a quantum-safe IBS scheme on lattices in order to achieve an efficiently revocable and quantum-safe IBS.

For this purpose, this thesis gives a broad introduction to the underlying concepts of lattice-based cryptography. Based on the consequential understanding of lattice-based signature schemes, the functionality of the lattice-based IBS by Tian and Huang from [TH14] is explained. It uses an ISIS trapdoor function for the key extraction, and Lyubashevsky signatures for the signing and verifying. This IBS is used to analyse how far it can be transformed towards an efficiently revocable KUSS following the proposal of Guggemos for elliptic curve IBS.

The lattice-based IBS is transformed into a 2KSS and a U2KSS, by multiplying an additional shared secret to the input of the hash function during signing and verification. The further transformation from the U2KSS is not possible straightforward after the elliptic curve schemes, because regarding the lattice master key pair, one is not the multiple of the other. Therefore the KUSS is analysed with respect to which requirements allow signing key updates with tokens. The identified requirements are compared with the properties of the lattice-based IBS, limiting the number of candidate update tokens to a quadratic matrix with full rank consisting of integers modulo prime. Additionally this matrix, when multiplied modulo on a short matrix, must result in another short matrix. Such a concrete matrix is not found nor has its existence been disproven. Under the assumption that such a matrix exists and is used as update token, a lattice-based KUSS is designed, where signing key updates are performed by multiplying the matrix to the user signing keys, and the inverse

of the matrix is multiplied to the master public key, and in such way allows revocation after the idea of Guggemos. Nevertheless, the concrete update token matrix fulfilling the required properties needs to be found in order to realize the quantum resistant lattice-based KUSS.

Future Work

On the basis of this work, it is first essential to find a possible update token that meets the requirements that are specified during this thesis, and which enables a complete transformation of a lattice-based IBS into a KUSS. Alternatively, it can be investigated whether the token can be chosen in such a way that at least a limited number of updates is possible.

If such a suitable update token is found, allowing unlimited revocation or only a limited number of revocations, the KUSS has to be implemented as a proof of concept. In addition, it needs to be investigated whether this lattice-based KUSS compared to other lattice-based revocable IBS actually results in the expected performance advantages that were found for elliptic curve KUSS.

Furthermore, the designed KUSS is partially based on assumptions which must be proven, and which are summarised in section 6.4.

In any case, it can be investigated whether the first and second transformation of the lattice-based IBS into 2KSS and U2KSS already bring any performance advantages.

In addition, the basic understanding of lattice-based cryptography conveyed through this thesis serves as a starting point for investigation on whether other lattice-based IBS, for example those using basis delegation, are suitable for transforming into KUSS.

List of Abbreviations

2KSS Two Key Signature Scheme

CAKE Centralized Authorized Key Extension

CVP Closest Vector Problem

CVP$_\gamma$ γ-Approximation of CVP

DLP Discrete Logarithm Problem

DLWE Decisional Learning With Errors

ECC Elliptic Curve Cryptography

ECDLP Elliptic Curve Discrete Logarithm Problem

FDH Full-Domain Hash

GapCVP Decisional γ-Approximation of CVP

GapSVP Decisional γ-Approximation of SVP

IBC Identity-based Cryptography

IBE Identity-based Encryption

IBS Identity-based Signatures

ISIS Inhomogeneous Short Integer Solution

KUC Key Update Center

KUSS Key Updatable Signature Scheme

LKH Logical Key Hierarchy

LWE Learning With Errors

MKP Master Key Pair

mpk Master Public Key

msk Master Secret Key

NIST National Institute of Standards and Technology

OTP One-Time Pad

PKG Private Key Generator

PKI Public Key Infrastructure

PQC Post-Quantum Cryptography

RIBS Revocable Identity-based Signatures

SIS Short Integer Solution

SIVP Shortest Independent Vectors Problem

SVP Shortest Vector Problem

$\mathbf{SVP}_\gamma$ γ-Approximation of SVP

U2KSS Updatable Two Key Signature Scheme

List of Figures

3.1 Distribution of hard instances in worst-case and average-case problems 14
3.2 Cryptographic Primitives 15

4.1 Examples of elliptic curves 18
4.2 Elliptic curve arithmetic on the curve $y^2 = x^3 - 3x + 4$ over $\mathbb{R}$ 19
4.3 Examples of Elliptic Curves over the finite field $\mathbb{F}_7$ (left) and $\mathbb{F}_{83}$ (right) . . . 20
4.4 Elliptic curve arithmetic shown on the curve $y^2 = x^3 - 4x + 1$ over $\mathbb{F}_7$ 21
4.5 Cyclic subgroup of an elliptic curve over a finite field 21
4.6 Schnorr signature with elliptic curves 23
4.7 Identity-based Encryption 25
4.8 Identity-based Signatures 26
4.9 Galindo-Garcia Identity-based Signature on elliptic curves 28
4.10 Elliptic curve KUSS based on the IBS by Galindo and Garcia (see figure 4.9). The key updates are marked in red. 35

5.1 Different lattices $\Lambda(B) \subseteq \mathbb{Z}^2$, defined by different B. 40
5.2 The same lattice $\Lambda(B) \subseteq \mathbb{Z}^2$, defined by different B. 40
5.3 The minimum distance λ_1 and the 2nd successive minimum λ_2 of a lattice [Mic20] 41
5.4 A random lattice and its dual [Mic20] 42
5.5 Matrix multiplications for calculating the q-ary lattice 43
5.6 Kernel q-ary lattices for $A = \begin{pmatrix} 4 & 10 & 3 \end{pmatrix}$ 44
5.7 Matrix multiplications for calculating the q-ary lattice 44
5.8 Image q-ary lattices with $A = \begin{pmatrix} 4 & 10 & 3 \end{pmatrix}$ 45
5.9 The properties of lattice basis 46
5.10 $h_{\mathrm{A}}(\begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}^{\intercal})$ with $A = \begin{pmatrix} 5 & 1 & 4 & 26 & 3 & 2 & 1 \end{pmatrix}$ 52
5.11 Secret-key encryption of a bit 53
5.12 Encryption of $m = 1$ as one-dimensional lattice 54
5.13 Decryption for $s = \begin{pmatrix} 26 \end{pmatrix}$ and $a = \begin{pmatrix} 43 \end{pmatrix}$ 55
5.14 Public-key encryption of a bit 55
5.15 Dual encryption of a bit 58
5.16 The Matrix A and its trapdoor R 66
5.17 Lattice-based FDH Signatures 66
5.18 Lattice-based Identity-based Encryption 68
5.19 Lattice-based Lyubashevsky Signatures 69
5.20 Lattice-based Identity-based Signatures 71

6.1 KUSS constructed from the lattice-based IBS by Tian and Huang (see figure 5.20). The key updates are marked in red. 81

Bibliography

[ABB10a] Agrawal, Shweta ; Boneh, Dan ; Boyen, Xavier: Efficient lattice (h) ibe in the standard model. In: *Eurocrypt* Bd. 6110 Springer, 2010, S. 553–572

[ABB10b] Agrawal, Shweta ; Boneh, Dan ; Boyen, Xavier: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30* Springer, 2010, S. 98–115

[ABC+18] Abdalla, Michel ; Bjørstad, Tor E. ; Cid, Carlos ; Gierlichs, Benedikt ; Hülsing, Andreas ; Luykx, Atul ; Paterson, Kenneth G. ; Preneel, Bart ; Sadeghi, Ahmad-Reza ; Spies, Terence u. a.: Algorithms, key size and protocols report. In: *ECRYPT-CSA, Tech. Rep. H2020-ICT-2014–Project* 645421 (2018)

[Ajt96] Ajtai, Miklós: Generating hard instances of lattice problems. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, S. 99–108

[Ajt99] Ajtai, Miklós: Generating hard instances of the short basis problem. In: *International Colloquium on Automata, Languages, and Programming* Springer, 1999, S. 1–9

[AR05] Aharonov, Dorit ; Regev, Oded: Lattice problems in NP ∩ coNP. In: *Journal of the ACM (JACM)* 52 (2005), Nr. 5, S. 749–765

[Bab86] Babai, László: On Lovász'lattice reduction and the nearest lattice point problem. In: *Combinatorica* 6 (1986), S. 1–13

[BDK+21] Bai, Shi ; Ducas, Léo ; Kiltz, Eike ; Lepoint, Tancrède ; Lyubashevsky, Vadim ; Schwabe, Peter ; Seiler, Gregor ; Stehlé, Damien: CRYSTALS-Dilithium. In: *Algorithm Specifications and Supporting Documentation (Version 3.1)* (2021)

[BF01] Boneh, Dan ; Franklin, Matt: Identity-based encryption from the Weil pairing. In: *21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings* Springer, 2001, S. 213–229

[BGK08] Boldyreva, Alexandra ; Goyal, Vipul ; Kumar, Virendra: Identity-based encryption with efficient revocation. In: *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, S. 417–426

[BHWA22] Bandara, Harshana ; Herath, Yasitha ; Weerasundara, Thushara ; Alawatugoda, Janaka: On Advances of Lattice-Based Cryptographic Schemes and Their Implementations. In: *Cryptography* 6 (2022), Nr. 4, S. 56

[BLMQ05] Barreto, Paulo S. L. M. ; Libert, Benoît ; McCullagh, Noel ; Quisquater, Jean-Jacques: Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: *Asiacrypt* Bd. 3788 Springer, 2005, S. 515–532

[Bro09] Brown, D: Standards for efficient cryptography, SEC 1: elliptic curve cryptography. In: *Released Standard Version* 1 (2009)

[BS99] Blömer, Johannes ; Seifert, Jean-Pierre: On the complexity of computing short linearly independent vectors and short bases in a lattice. In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, S. 711–720

[Buc16] Buchmann, Johannes: *Einführung in die Kryptographie*. Bd. 6. Springer, 2016

[CCSKK15] Chi, Dong P. ; Choi, Jeong W. ; San Kim, Jeong ; Kim, Taewan: Lattice based cryptography for beginners. In: *Cryptology ePrint Archive* (2015)

[CHKP12] Cash, David ; Hofheinz, Dennis ; Kiltz, Eike ; Peikert, Chris: Bonsai trees, or how to delegate a lattice basis. In: *Journal of cryptology* 25 (2012), Nr. 4, S. 601–639

[CJL+16] Chen, Lily ; Jordan, Stephen ; Liu, Yi-Kai ; Moody, Dustin ; Peralta, Rene ; Perlner, Ray A. ; Smith-Tone, Daniel: Report on post-quantum cryptography. In: *Technical Report NIST Internal or Interagency Report (NISTIR) 8105* (2016). `https://nvlpubs.nist.gov/nistpubs/ir/2016/nist.ir.8105.pdf`

[CLL+12] Chen, Jie ; Lim, Hoon W. ; Ling, San ; Wang, Huaxiong ; Nguyen, Khoa: Revocable identity-based encryption from lattices. In: *Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings 17* Springer, 2012, S. 390–403

[Coc01] Cocks, Clifford: An identity based encryption scheme based on quadratic residues. In: *IMA international conference on cryptography and coding* Springer, 2001, S. 360–363

[Cor15] Corbellini, Andrea: *Elliptic Curve Cryptography: a gentle introduction.* `https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/`. Version: 2015. – last accessed on 2022/06/14

[Dar21] Dargan, James: *15 Leading Quantum Computing Countries with National Initiatives.* `https://thequantuminsider.com/2021/04/29/leading-quantum-computing-countries/`. Version: 2021. – last accessed on 2023/02/22

[DH76] Diffie, Whitfield ; Hellman, Martin E.: New Directions in Cryptography. In: *IEEE Transactions on Information Technology* 22 (1976), Nr. 6

[Fis11] Fischer, Gerd: *Lernbuch Lineare Algebra und Analytische Geometrie.* 4. Springer, 2011

[Gal12] Galbraith, Steven: *Algorithms for the closest and shortest vector problem.* Cambridge University Press, 2012

[GG09] Galindo, David ; Garcia, Flavio D.: A Schnorr-like lightweight identity-based signature scheme. In: *International Conference on Cryptology in Africa* Springer, 2009, S. 135–148

[GM84] Goldwasser, Shafi ; Micali, Silvio: Probabilistic encryption. In: *Journal of Computer and System Sciences* 28 (1984), Nr. 2, 270-299. `https://www.sciencedirect.com/science/article/pii/0022000084900709`

[GMR04] Guruswami, Venkatesan ; Micciancio, Daniele ; Regev, Oded: The complexity of the covering radius problem on lattices and codes. In: *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.* IEEE, 2004, S. 161–173

[GPV08] Gentry, Craig ; Peikert, Chris ; Vaikuntanathan, Vinod: Trapdoors for hard lattices and new cryp-tographic constructions. In: *Proceedings of the 40th annual ACM symposium on Theory of computing* ACM, 2008, S. 197–206

[GRR+23] Gouzien, Élie ; Ruiz, Diego ; Régent, Francois-Marie L. ; Guillaud, Jérémie ; Sangouard, Nicolas: Computing 256-bit Elliptic Curve Logarithm in 9 Hours with 126133 Cat Qubits. In: *arXiv preprint arXiv:2302.06639* (2023)

[GSK+18] Guggemos, Tobias ; Streit, Klement ; Knüpfer, Marcus ; Felde, Nils gentschen ; Hillmann, Peter: 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)*, 2018

[Gug20] Guggemos, Tobias: *Efficient signature verification and key revocation using identity based cryptography*, lmu, Diss., 2020

[HQG+22] He, Yan ; Qin, Baodong ; Gao, Wen ; Zheng, Dong ; Zhao, Qianqian u. a.: Generic Construction of Forward-Secure Revocable Identity-Based Signature and Lattice-Based Instantiations. In: *Security and Communication Networks* 2022 (2022)

[HTH17] Hung, Ying-Hao ; Tseng, Yuh-Min ; Huang, Sen-Shan: Revocable ID-based signature with short size over lattices. In: *Security and Communication Networks* 2017 (2017)

[KM17] Karpfinger, Christian ; Meyberg, Kurt: *Algebra: Gruppen-Ringe-Körper.* Springer-Verlag, 2017

[Kob87] Koblitz, Neal: Elliptic curve cryptosystems. In: *Mathematics of computation* 48 (1987), Nr. 177, S. 203–209

[LDSH21] Le Dévéhat, Anaëlle ; Shizuya, Hiroki ; Hasegawa, Shingo: On the Higher-Bit Version of Approximate Inhomogeneous Short Integer Solution Problem. In: *International Conference on Cryptology and Network Security* Springer, 2021, S. 253–272

[Len20] Lenze, Burkhard: *Basiswissen Angewandte Mathematik – Numerik, Grafik, Kryptik*. Springer, 2020

[LHZL13] Liu, Zhenhua ; Hu, Yupu ; Zhang, Xiangsong ; Li, Fagen: Efficient and strongly unforgeable identity-based signature scheme from lattices in the standard model. In: *Security and communication networks* 6 (2013), Nr. 1, S. 69–77

[LLL82] Lenstra, Arjen K. ; Lenstra, Hendrik W. ; Lovász, László: Factoring polynomials with rational coefficients. In: *Mathematische annalen* 261 (1982), Nr. ARTICLE, S. 515–534

[LLNW14] Langlois, Adeline ; Ling, San ; Nguyen, Khoa ; Wang, Huaxiong: Lattice-based group signature scheme with verifier-local revocation. In: *Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17* Springer, 2014, S. 345–361

[Lyu12] Lyubashevsky, Vadim: Lattice signatures without trapdoors. In: *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31* Springer, 2012, S. 738–755

[Mar14] Markmann, Tobias: Performance Analysis of Identity-based Signatures. In: *Project Report, Aug* (2014)

[Mic16] Micciancio, Daniele: Shortest Vector Problem. In: *Encyclopedia of Algorithms* (2016), 1974–1977. https://doi.org/10.1007/978-1-4939-2864-4_374

[Mic20] Micciancio, Daniele: *Mathematics of Lattices*. https://simons.berkeley.edu/talks/basic-mathematics-lattices. Version: 2020 (Lattices: Algorithms, Complexity, and Cryptography Boot Camp). – Accessed on 2022/10/11

[Mil86] Miller, Victor S.: Use of Elliptic Curves in Cryptography. In: Williams, Hugh C. (Hrsg.): *Advances in Cryptology — CRYPTO '85 Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1986. – ISBN 978–3–540–39799–1, S. 417–426

[MK18] Modler, Florian ; Kreh, Martin: *Tutorium Analysis 1 und Lineare Algebra 1*. 4. Springer, 2018

[MR07] Micciancio, Daniele ; Regev, Oded: Worst-case to average-case reductions based on Gaussian measures. In: *SIAM Journal on Computing* 37 (2007), Nr. 1, S. 267–302

[MR09] Micciancio, Daniele ; Regev, Oded: Lattice-based Cryptography. In: *Post-Quantum Cryptography* (2009), 147–191. https://doi.org/10.1007/978-3-540-88702-7_5

[Nat18] National Institute of Standards and Technology : Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. In: *NIST Special Publication 800-56A* (2018). https://doi.org/10.6028/NIST.SP.800-56Ar3

[Nat23a] National Institute of Standards and Technology : Digital Signature Standard (DSS). In: *Federal Information Processing Standard (FIPS) 186-5* (2023). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

[Nat23b] National Institute of Standards and Technology: *Post-Quantum Cryptography - Selected Algorithms 2022: Public-key Encryption and Key-establishment Algorithms.* Computer Security Resource Center. https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms-2022. Version: 2023. – last accessed on 2023/02/22

[Nat23c] National Institute of Standards and Technology: *Post-Quantum Cryptography Standardization.* Computer Security Resource Center. https://csrc.nist.gov/projects/post-quantum-cryptography. Version: 2023. – last accessed on 2023/02/22

[Pei09] Peikert, Chris: Public-key cryptosystems from the worst-case shortest vector problem. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, S. 333–342

[Per99] Perlman, Radio: An overview of PKI trust models. In: *IEEE network* 13 (1999), Nr. 6, S. 38–43

[Pol11] Pol, Joop van d.: Lattice-based cryptography. In: *Eindhoven University of Technology, Department of Mathematics and Computer Science* (2011)

[Reg09] Regev, Oded: On lattices, learning with errors, random linear codes, and cryptography. In: *Journal of the ACM (JACM)* 56 (2009), Nr. 6, S. 1–40. – Preliminary version in Proc. of STOC 2005

[Reg10] Regev, Oded: The learning with errors problem. (2010), S. 191–204

[RSA78] Rivest, Ronald L. ; Shamir, Adi ; Adleman, Leonard: A method for obtaining digital signatures and public-key cryptosystems. In: *Communications of the ACM* 21 (1978), Nr. 2, S. 120–126

[Rüc10] Rückert, Markus: Strongly Unforgeable Signatures and Hierarchical Identity-Based Signatures from Lattices without Random Oracles. In: *PQCrypto* 6061 (2010), S. 182–200

[Sch91] Schnorr, Claus-Peter: Efficient signature generation by smart cards. In: *Journal of cryptology* 4 (1991), Nr. 3, S. 161–174

[Sch20] Schwenk, Jürgen: *Sicherheit und Kryptographie im Internet: Theorie und Praxis.* Springer, 2020

[Sha49] Shannon, Claude E.: Communication theory of secrecy systems. In: *The Bell system technical journal* 28 (1949), Nr. 4, S. 656–715

[Sha84] Shamir, Adi: Identity-based cryptosystems and signature schemes. In: *Workshop on the theory and application of cryptographic techniques* Springer, 1984, S. 47–53

[Sho97] Shor, Peter: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In: *SIAM journal on computing (Print)* 26 (1997), Nr. 5, S. 1484–1509

[Sti19] Stinson, Douglas R.: *Cryptography: theory and practice*. Bd. 4. Chapman and Hall/CRC, 2019

[Tan17] Tan, Liansheng: 3 - Generalized inverse of matrix and solution of linear system equation. Version: 2017. http://dx.doi.org/https://doi.org/10.1016/B978-0-08-101946-7.00003-2. In: Tan, Liansheng (Hrsg.): *A Generalized Framework of Linear Multivariable Control*. Butterworth-Heinemann, 2017. – DOI https://doi.org/10.1016/B978–0–08–101946–7.00003–2. – ISBN 978–0–08–101946–7, 38-50

[TH14] Tian, Miaomiao ; Huang, Liusheng: Efficient Identity-Based Signature from Lattices. In: *ICT Systems Security and Privacy Protection*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014, S. 321–329

[THY13] Tian, Miaomiao ; Huang, Liusheng ; Yang, Wei: Efficient hierarchical identity-based signatures from lattices. In: *International Journal of Electronic Security and Digital Forensics* 5 (2013), Nr. 1, S. 1–10

[TT12] Tseng, Yuh-Min ; Tsai, Tung-Tso: Efficient revocable ID-based encryption with a public channel. In: *The Computer Journal* 55 (2012), Nr. 4, S. 475–486

[Vai15a] Vaikuntanathan, Vinod: *Lattice Based Crypto - Worst-case to Average-case Reduction for LWE*. https://people.csail.mit.edu/vinodv/6876-Fall2015/L15.pdf. Version: 2015 (Advanced Topics in Cryptography: Lattices). – Lecture Notes, Accessed on 2023/01/11

[Vai15b] Vaikuntanathan, Vinod: *The Mathematics of Lattices*. https://simons.berkeley.edu/talks/mathematics-lattices. Version: 2015 (Cryptography Boot Camp). – Accessed on 2022/10/11

[Vai15c] Vaikuntanathan, Vinod: *Sampling Lattice Trapdoors*. https://people.csail.mit.edu/vinodv/6876-Fall2015/L16.pdf. Version: 2015 (Advanced Topics in Cryptography: Lattices). – Lecture Notes, Accessed on 2023/01/11

[VG15] Vaikuntanathan, Vinod ; Grinman, Alex: *Lecture 17*. https://people.csail.mit.edu/vinodv/6876-Fall2015/L17.pdf. Version: 2015 (Advanced Topics in Cryptography: Lattices). – Lecture Notes, Accessed on 2023/01/11

[Wei] Weisstein, Eric W.: *Closed*. https://mathworld.wolfram.com/Closed.html, . – Accessed on 2022/10/10

[Xia15] Xiang, X: Adaptive secure revocable identity-based signature scheme over lattices. In: *Computer Engineering* 41 (2015), Nr. 10, S. 126–129

[XWW20] Xie, Congge ; Weng, Jian ; Wen, Jinming: Scalable revocable identity-based signature scheme with signing key exposure resistance from lattices. In: *Security and Communication Networks* 2020 (2020), S. 1–11

[ZG15] Zur Gathen, Joachim v.: *CryptoSchool*. Springer, 2015