On the difference between hardness and security: a comparison of lattice-based signature schemes

Bachelor-Thesis by Patrick Struck

Juni 2015

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Fachgebiet CDC

On the difference between hardness and security:
a comparison of lattice-based signature schemes

Submitted Bachelor-Thesis by Patrick Struck born in Bad Soden

1. Referee: Prof. Dr. Johannes Buchmann
2. Referee: Nina Bindel

Date of Submision:

Affidavit

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, June 14, 2015

_____

(Patrick Struck)

Abstract

The security of currently used signature schemes, e.g. the RSA signature scheme, is threatened through the eventually invention of efficient quantum computers. During the last years, lattice-based cryptography turns out to be a good alternative to currently used cryptography.

In this work we describe the concept of cryptographic security reductions which are used to prove the security of digital signature schemes. Further on, we define the terms bit hardness and bit security which describe how hard a problem and how secure a signature scheme is respectively. To measure the quality of a security reduction, the term tightness is defined. We show that in case of a tight security reduction breaking a signature scheme is at least as hard as solving the underlying problem. However, a non-tight security reduction leaves the possibility that breaking the scheme could be easier than solving the underlying problem.

Further on, we describe selected lattice-based signature schemes and compute the bit security, based on the security reduction of the scheme. Following this, the schemes are compared with regard to the security and the size of the signature and the key sizes.

Contents

1 Introduction

Currently used signature schemes are based on number theory problems, e.g. the factorization problem and the discrete logarithm problem, which are conjectured to be hard for classical computers. Already 20 years ago the mathematician Peter Shor presented an efficient algorithm for solving these problems which uses a quantum computer [20]. So the security of currently used signature schemes is threatened by the eventually invention of efficient quantum computers. Therefore post-quantum cryptography concentrates on finding alternatives to currently used cryptographic primitives which security is not threatened through efficient quantum computers. Due to the progress of quantum computers over the last years, post-quantum cryptography becomes more and more important. One highly promising part of post-quantum cryptography is lattice-based cryptography. Throughout this work we focus on the comparison of provable secure lattice-based signature schemes. This work is divided into three parts which are described below.

In the first part (Chapter 3 and 4) we focus on provable security of signature schemes. First, after a short description of digital signatures, we show how to prove the security of a signature scheme using cryptographic security reductions. The idea of these proofs is to show that an adversary, that is able to break the scheme, can be turned into an algorithm that solves the underlying problem. Further on, to describe how hard it is to solve a problem and how secure a signature scheme is respectively, we define the terms bit hardness and bit security. A relation between these two values is given by the security reduction. The better the security reduction the smaller is the gap between the hardness of the problem and the security of the signature scheme. To measure the quality of a security reduction, the term tightness is defined. Based on this, we finally show how to compute the security of a signature scheme regarding the tightness of the security reduction.

In the second part (Chapter 5 and 6) we compute the security of lattice-based signature schemes. At first, we define (ideal) lattices and the following problems: the small integer solution (SIS) problem, the inhomogeneous small integer solution (ISIS) problem, the learning with errors (LWE) problem, the decisional learning with errors (DLWE) problem, the ring-small integer solution (R-SIS) problem, the ring-learning with errors (R-LWE) problem, the ring-decisional learning with errors (R-DLWE) problem and the decisional compact knapsack (DCK) problem. The hardness of the SIS and the LWE problem is illustrated by two examples. Hereafter, we describe the following lattice-based signature schemes: the signature scheme (LYU12) by Lyubashevsky [16], the signature scheme (BLISS) by Ducas, Durmus, Lepoint and Lyubashevsky [11], the signature scheme (GLP) by Güneysu, Lyubashevsky and Pöppelmann [14], the signature scheme (AFLT) by Abdalla, Fouque, Lyubashevsky and Tibouchi [1], the signature scheme (BG) by Bai and Galbraith [5], the signature scheme (TESLA) by Alkim, Bindel, Buchmann, Dagdelen and Schwabe [4] and the signature scheme (GPV) by Gentry, Peikert and Vaikuntanathan [12]. For each scheme we use the security reduction given by the authors and the results from the first part of this work to compute the bit security of the scheme.

In the third part (Chapter 7), we compare the security and the efficiency of the signature schemes. The efficiency comparison is superficial and only based on the size of the keys and the signature of the signature scheme. We do not consider the runtime of the signing or verification algorithm. The security comparison is more detailed.

2 Notation

Throughout this work we denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of real numbers by $\mathbb{R}$ and the set which contains prime numbers by $\mathbb{P}$. For $q \in \mathbb{N}$ we define the set $\mathbb{Z}_q$ which contains the integers within $\left(-\frac{q}{2}, \frac{q}{2}\right]$, e.g. $\mathbb{Z}_5 = \{-2,-1,0,1,2\}$ and $\mathbb{Z}_4 = \{-1,0,1,2\}$. For $n, q \in \mathbb{N}$ we define the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$. Elements within this ring are represented as polynomials of degree $n-1$ with coefficients in $\mathbb{Z}_q$, e.g. $4x^3 - 2x^2 + 1 \in \mathbb{Z}_7[x]/(x^4+1)$.

We denote vectors by **bold** and matrices by **bold capital** letters. For a vector $\mathbf{v}$ we define the Euclidean norm by $\|\mathbf{v}\|_2$, the maximum norm by $\|\mathbf{v}\|_\infty$, the 1-Norm by $\|\mathbf{v}\|_1$ and the transposed vector by $\mathbf{v}^T$. The inner product of two vectors $\mathbf{v}, \mathbf{w}$ is defined by $\langle \mathbf{v}, \mathbf{w} \rangle$. We denote the $n \times n$ identity matrix by $\mathbf{I}_n$.

For a finite set S we write $a \xleftarrow{\$} S$ to describe that a is chosen uniformly at random from S and for a distribution χ we write $a \leftarrow \chi$ to denote that a is sampled according to the distribution χ. We call a problem $\mathcal{P}$ *hard* if there exist no probabilistic polynomial time (ppt) algorithm which solves $\mathcal{P}$.

For a polynomial $a \in \mathbb{Z}_q[x]/(x^n+1)$ we denote the vector $\mathbf{v} \in \mathbb{Z}_q^n$ which contains the coefficients of a by $\overline{a}$. For polynomials $a_1, a_2, ..., a_k \in \mathbb{Z}_q[x]/(x^n+1)$ we denote the vector $\mathbf{v} \in \mathbb{Z}_q^{k \cdot n}$ which contains the coefficients of the polynomials a_i by $(\overline{a_1}, \overline{a_2}, ..., \overline{a_k})^T$, e.g. $a_1 = x^2 + 3x, a_2 = -2x^2 + 3 \in \mathbb{Z}_7[x]/(x^3+1)$ and $(\overline{a_1}, \overline{a_2})^T = (1,3,0,-2,0,3)^T \in \mathbb{Z}_7^6$.

A function f is called *negligible* in the (security) parameter λ, denoted by $negl(\lambda)$, if it decreases faster than every inverse polynomial function for sufficiently large λ. Below we define the term negligible more formally.

Definition 2.1 *Let D, R be two sets. A function $f : D \to R$ is called negligible in the security parameter λ if $\forall c \in \mathbb{N} : \exists n_0 \in \mathbb{N}$ s.t. $\forall \lambda \geq n_0, f(\lambda) \leq \frac{1}{\lambda^c}$.*

Note that there are further notations which are only necessary for some of the schemes described in Chapter 6. We skip those notations at this point and provide them within the sections in which they are needed.

3 Digital signature schemes

Digital signatures are used to ensure the cryptographic goals of integrity and authenticity. Integrity means that one can check whether received data were manipulated or not. Authenticity means that one can check whether received data were sent by the one who claims it.

An important example of digital signatures is online banking. When the bank receives the instructions to transfer money, it first has to check if the instructions were sent by the owner of the bank account and not by someone else (authenticity). Further on, the bank has to check that the instructions are not corrupted. Otherwise, someone could change the target bank account or the amount of money which should be transferred (integrity).

We assume the following situation: Alice sends a message μ_A to Bob which can be replaced with another message μ_E by a third party called Eve. When Bob receives a message μ he is not able to distinguish whether it is the one by Alice, i.e. $\mu = \mu_A$, or the corrupted one by Eve, i.e. $\mu = \mu_E$.

We change the situation above by using digital signatures, i.e. there exists a pair of two keys (sk, vk). The signing key sk only known by Alice and the verification key vk which is publicly known. With the

signing key sk Alice is able to create a signature σ_A of the message μ_A. With the verification key vk of Alice everyone can check if a signature was created by Alice using her signing key sk.

Alice sends in addition to μ_A the corresponding σ_A created with the signing key. When Bob receives a message μ and a signature σ he is able to use the verification key vk of Alice to check whether σ was created by Alice using her signing key sk or not. Because the signing key is unknown to Eve she should not be able to create a message and a signature which Bob would accept as a message by Alice.

Below we formally define digital signature schemes.

Definition 3.1 *A signature scheme is a triplet (*KeyGen, Sign, Verify*) of polynomial time algorithms.*

- KeyGen *is the key generating algorithm which gets a security parameter λ as input and returns a signing key sk and a verification key vk.*
- Sign *is the signing algorithm which gets a message μ and the signing key sk as input and returns a signature σ of the message μ.*
- Verify *is the verification algorithm which gets a message μ, a signature σ and the verification key vk as input and either accepts (return 1) or rejects (return 0) the signature σ.*

The following equation has to hold for all messages μ.

$$Pr[(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda) \mid \mathsf{Verify}(\mu, \mathsf{Sign}(\mu, sk), vk) = 1] = 1$$

For a signature scheme (KeyGen, Sign, Verify) a signature σ of a message μ is called *valid* if the $Verify$ algorithm accepts the signature.

Definition 3.2 *Let (*KeyGen, Sign, Verify*) be a signature scheme, vk be a verification key and μ be a message. A signature σ of μ is called valid if $Verify(\mu, \sigma, vk) = 1$.*

It is important that an adversary against a signature scheme is not able to compute the signing key with reasonable effort only knowing the public information, e.g. the verification key. Furthermore, it is necessary that an adversary is not able to produce a forgery, i.e. a message-signature pair (μ, σ) such that σ is a valid signature of μ, after seeing arbitrary many message-signature pairs. Below we define the terms *secure* and *strongly unforgeable* in connection with signature schemes.

A signature scheme $\mathcal{S}$ is called *secure* if an adversary $\mathcal{A}$ only knowing the public information of $\mathcal{S}$, after seeing arbitrary many message-signature pairs for messages chosen by himself, is not able to produce a valid signature of a not yet seen message [13].

Definition 3.3 *A signature scheme (*KeyGen, Sign, Verify*) is called secure if for every key pair (sk, vk) and for every ppt adversary $\mathcal{A}$ knowing vk, after seeing q_s message-signature pairs $\{(\mu_1, \sigma_1), (\mu_2, \sigma_2), ..., (\mu_{q_s}, \sigma_{q_s})\}$ with $\mu_1, \mu_2, ..., \mu_{q_s}$ chosen by $\mathcal{A}$ and $\sigma_i = \mathsf{Sign}(\mu_i, sk)$, the probability that $\mathcal{A}$ can produce (μ, σ), such that $\mu \neq \mu_i$ and $\mathsf{Verify}(\mu, \sigma, vk) = 1$, is negligible.*

A signature scheme $\mathcal{S}$ is called *strongly unforgeable* if an adversary $\mathcal{A}$ only knowing the public information of $\mathcal{S}$, after seeing arbitrary many message-signature pairs for messages chosen by himself, is neither able to produce a valid signature of a not yet seen message nor able to produce a different valid signature of a yet seen message [16].

Definition 3.4 *A signature scheme (*KeyGen, Sign, Verify*) is called strongly unforgeable if for every key pair* (sk, vk) *and for every* ppt *adversary* $\mathcal{A}$ *knowing* vk*, after seeing* q_s *message-signature pairs* $\{(\mu_1, \sigma_1), (\mu_2, \sigma_2), ..., (\mu_{q_s}, \sigma_{q_s})\}$ *with* $\mu_1, \mu_2, ..., \mu_{q_s}$ *chosen by* $\mathcal{A}$ *and* $\sigma_i = \mathsf{Sign}(\mu_i, sk)$*, the probability that* $\mathcal{A}$ *can either produce* (μ, σ)*, such that* $\mu \neq \mu_i$ *and* $\mathsf{Verify}(\mu, \sigma, vk) = 1$*, or produce* σ^**, such that* $\exists i \in \{1, 2, ..., q_s\}$ *with* $\sigma^* \neq \sigma_i$ *and* $\mathsf{Verify}(\mu_i, \sigma^*, vk) = 1$*, is negligible.*

We see that the term strongly unforgeable is a stronger notion of the term secure. For practical use it is enough if a signature scheme is secure because possible weakness can be removed by using timestamps. The difference between these terms is more of theoretical interest.

4 Provable Security

In this section we focus on provable security of signature scheme. Therefore, we describe how to prove the security of signature schemes using cryptographic security reductions. First, we give a short description of the random oracle model. Following this, we describe the structure of cryptographic security reductions in the random oracle model as described by Katz and Lindell [15]. Note that we only focus on signature schemes. The concept of security reductions is also used for other cryptographic primitives, e.g. encryption schemes. Finally, we define the terms bit hardness, bit security and tightness to compare different security reductions.

4.1 Cryptographic security reductions in the Random Oracle Model

The random oracle model is a theoretical model to prove the security for cryptographic primitives. The idea is that the used hash functions are modeled as random oracles. A random oracle acts like a black box algorithm. For each query which was not queried yet the oracle outputs a new random value and for each query which already was queried the oracle outputs the same value which was outputted the last time the query was made.

The random oracle model provides an idealized world to prove the security, which facilitates to prove the security of a signature scheme. If the used hash function is indistinguishable from a random oracle then the scheme is secure in the real world. Katz and Lindell [15] describe that there is, at the moment, no justification that hash functions will become indistinguishable from random oracles. In fact, there are examples of cryptographic primitives that are secure in the random oracle model and insecure if implemented in the real world [15].

Let $\mathcal{P}$ be a hard problem and $\mathcal{S}$ be a signature scheme which security is based on $\mathcal{P}$. The goal of a security reduction is to show that $\mathcal{S}$ is secure under the assumption that $\mathcal{P}$ is a hard problem. This can be proven by contradiction.

Assume that $\mathcal{P}$ is hard and $\mathcal{S}$ is not secure, which means there exists an ppt adversary $\mathcal{A}$ who is able to break the scheme by generating a forgery, i.e. (μ, σ) such that σ is a valid signature of μ. Note that for the proof it is neither necessary that such an adversary exists nor that it is known how the adversary breaks the scheme. Based on this, we assume $\mathcal{A}$ as a (hypothetical) black box algorithm which gets the public information (e.g. the verification key) of $\mathcal{S}$ as input and outputs a forgery (μ,σ). Furthermore, $\mathcal{A}$ can query two oracles (hash and sign) which return hash values and signatures respectively. An ppt

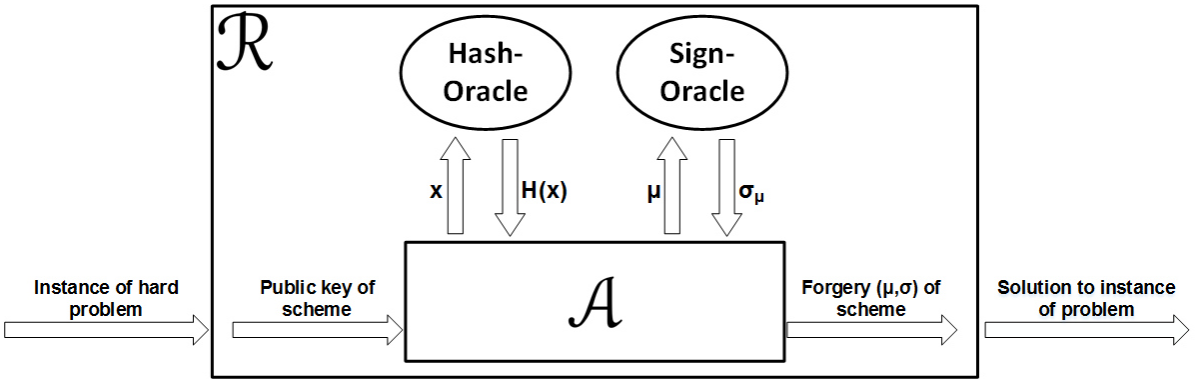

Figure 4.1: Structure of a security reduction

algorithm $\mathcal{R}$ which makes (black-box) use of the algorithm $\mathcal{A}$ to solve the problem $\mathcal{P}$ is constructed. This means that $\mathcal{R}$:

- generates the input of $\mathcal{A}$,
- responds to both the Hash- and Sign- queries of $\mathcal{A}$ and
- has access to the output of $\mathcal{A}$.

The existence of such an algorithm $\mathcal{R}$ leads to a contradiction of the assumption that $\mathcal{P}$ is hard which proves that $\mathcal{S}$ is secure if $\mathcal{P}$ is hard. In that case we call $\mathcal{R}$ the *reduction* of $\mathcal{P}$ to $\mathcal{S}$.

The structure of a security reduction is illustrated in Figure 4.1. The algorithms ($\mathcal{R}$ and $\mathcal{A}$) are represented by the two boxes. The input of each algorithm is represented by the arrow that points to the algorithm while the output is represented by the arrow that points away from the algorithm. The arrows above $\mathcal{A}$ represent the queries that $\mathcal{A}$ makes and the response by each oracle respectively.

4.2 Tightness

Tightness is a measure to describe the quality of a security reduction. A security reduction is called *tight* if both the time and the success probability of the adversary and the reduction are approximately the same. Otherwise, the security reduction is called *non-tight* or *loose*.

Definition 4.1 *([9]) Let $\mathcal{S}$ be a signature scheme with respect to a hard problem $\mathcal{P}$, $\mathcal{A}$ be an adversary against $\mathcal{S}$ and $\mathcal{R}$ be the reduction of $\mathcal{P}$ to $\mathcal{S}$. Furthermore, assume that $\mathcal{A}$ is successfully after time $t_{\mathcal{A}}$ with probability $\epsilon_{\mathcal{A}}$ and $\mathcal{R}$ solves an instance of $\mathcal{P}$ with success probability $\epsilon_{\mathcal{R}}$ in time $t_{\mathcal{R}}$. The security reduction is called tight if $t_{\mathcal{R}} \approx t_{\mathcal{A}}$ and $\epsilon_{\mathcal{R}} \approx \epsilon_{\mathcal{A}}$. Otherwise, if $t_{\mathcal{R}} \gg t_{\mathcal{A}}$ or $\epsilon_{\mathcal{R}} \ll \epsilon_{\mathcal{A}}$, the security reduction is called non-tight or loose. The tightness gap is defined as $\frac{t_{\mathcal{R}} \cdot \epsilon_{\mathcal{A}}}{t_{\mathcal{A}} \cdot \epsilon_{\mathcal{R}}}$.*

For a tight security reduction the tightness gap is ≈ 1 and for a non-tight security reduction it is $\gg 1$. Given two different security reductions with respect to the same signature scheme, the one with the smaller tightness gap is called *tighter* than the other.

Based on the success probability and the runtime of the algorithms, we define the terms *bit hardness* and *bit security*, as described by Alkim et al. [4]. These values define lower boundaries of the effort which is necessary to solve a problem (to break a scheme).

Definition 4.2 *Let $\mathcal{P}$ be a hard problem and $\mathcal{R}$ be an algorithm which solves $\mathcal{P}$ after time $t_{\mathcal{R}}$ with success probability ϵ_R. The bit hardness is the largest integer κ such that*

$$2^{\kappa} \leq \frac{t_{\mathcal{R}}}{\epsilon_{\mathcal{R}}}. \tag{1}$$

The problem $\mathcal{P}$ is said to be κ-bit hard.

Definition 4.3 *Let $\mathcal{S}$ be a signature scheme and $\mathcal{A}$ be an adversary who breaks $\mathcal{S}$ after time $t_{\mathcal{A}}$ with success probability $\epsilon_{\mathcal{A}}$. The bit security is the largest integer γ such that*

$$2^{\gamma} \leq \frac{t_{\mathcal{A}}}{\epsilon_{\mathcal{A}}}. \tag{2}$$

The signature scheme $\mathcal{S}$ is said to be γ-bit secure.

Below we illustrate how the tightness of a security reduction effects the security of a signature scheme given both an example for a tight security reduction and an example for a non-tight security reduction. To simplify matters we assume $\epsilon_{\mathcal{R}} = \epsilon_{\mathcal{A}} = 1$ in both cases.

Example 4.1 *Let $\mathcal{S}$ be a signature scheme which security is based on a hard problem $\mathcal{P}$ with bit hardness κ. Furthermore, let $\mathcal{A}$ be an adversary against $\mathcal{S}$ with runtime $t_{\mathcal{A}}$ and success probability $\epsilon_{\mathcal{A}}$ and $\mathcal{R}$ be the reduction of $\mathcal{P}$ to $\mathcal{S}$ with runtime $t_{\mathcal{R}}$ and success probability $\epsilon_{\mathcal{R}}$. We assume $t_{\mathcal{R}} = t_{\mathcal{A}}$ which, using the assumption above, yields to $t_{\mathcal{A}} = t_{\mathcal{R}} = 2^{\kappa}$. Inserting $\epsilon_{\mathcal{A}} = 1$ and $t_{\mathcal{A}} = 2^{\kappa}$ in* (2) *finally provides $\gamma = \kappa$.*

Example 4.2 *Let $\mathcal{S}$ be a signature scheme which security is based on a hard problem $\mathcal{P}$ with bit hardness κ. Furthermore, let $\mathcal{A}$ be an adversary against $\mathcal{S}$ with runtime $t_{\mathcal{A}}$ and success probability $\epsilon_{\mathcal{A}}$ and $\mathcal{R}$ be the reduction of $\mathcal{P}$ to $\mathcal{S}$ with runtime $t_{\mathcal{R}}$ and success probability $\epsilon_{\mathcal{R}}$. We assume $t_{\mathcal{R}} = t_{\mathcal{A}}^2$. By solving this equation for $t_{\mathcal{A}}$ we obtain $t_{\mathcal{A}} = \sqrt{t_{\mathcal{R}}}$ which, using the assumption above, yields to $t_{\mathcal{A}} = \sqrt{t_{\mathcal{R}}} = \sqrt{2^{\kappa}} = 2^{\frac{\kappa}{2}}$. Inserting $\epsilon_{\mathcal{A}} = 1$ and $t_{\mathcal{A}} = 2^{\frac{\kappa}{2}}$ in* (2) *finally provides $\gamma = \frac{\kappa}{2}$.*

We see that in case of the tight security reduction (Example 4.1) the bit security of the signature scheme and the bit hardness of the problem are the same while in case of the non-tight security reduction (Example 4.2) the bit security of the signature scheme is only half the bit hardness of the problem.

Suppose one wants to use a signature scheme with a bit security $\gamma = 128$. In case of the tight security reduction one can easily obtain the required bit security by choosing the parameters such that $\kappa = 128$ holds. In case of the non-tight security reduction this instantiation only ensures half the required bit security and thus the parameters have to be chosen such that $\kappa = 256$ holds. The disadvantage is that this instantiation might significantly deteriorates the performance of the signature scheme.

There exists a dilemma between security and performance in case of non-tight security reductions. To increase the bit security larger parameters have to be chosen which often leads to worse performance. Higher performance requires smaller parameters which leads to lower bit security. The common method is to make the schemes efficient by instantiate the parameters to obtain the desired bit hardness κ and

to claim that the signature scheme is κ-bit secure. The primary reason of using this method is that there exists the possibility that the bit security and bit hardness are equal because these values define a lower boundary and a tight security reduction might be found in the future. The primary reason against this method is that the scheme is constituted more secure than it can at the time be proven.

Finally, we describe how to compute the bit security of a signature scheme according to the security reduction. Let $\mathcal{P}$ be a problem with bit hardness κ, $\mathcal{S}$ be a signature scheme with bit security γ, $\mathcal{A}$ be an adversary against $\mathcal{S}$, making q_s queries to the Sign-Oracle and q_h queries to the Hash-Oracle, and $\mathcal{R}$ be the reduction of $\mathcal{P}$ to $\mathcal{S}$. Furthermore, let $t_{\mathcal{R}}, \epsilon_{\mathcal{R}}$ describe that $\mathcal{R}$ is successful after time $t_{\mathcal{R}}$ with probability $\epsilon_{\mathcal{R}}$ and $t_{\mathcal{A}}, \epsilon_{\mathcal{A}}$ for $\mathcal{A}$ respectively. By solving (2) for γ we obtain

$$\gamma = \frac{\ln(t_{\mathcal{A}}) - \ln(\epsilon_{\mathcal{A}})}{\ln(2)} \tag{3}$$

as the formula for the bit security. Now we can use the relation between $t_{\mathcal{R}}$, $\epsilon_{\mathcal{R}}$ and $t_{\mathcal{A}}$, $\epsilon_{\mathcal{A}}$, given by the security reduction, to compute the bit security. To simplify matters we use the following assumptions:

$$\epsilon_{\mathcal{R}} = 1, t_{\mathcal{R}} = 2^{\kappa}, \tag{4}$$

$$q_h = 2^{\kappa}, q_s = 2^{\kappa/2}. \tag{5}$$

The first assumption describes, roughly speaking, that the algorithm needs at least 2^{κ} operations to solve the problem with success probability 1. The second assumption is taken from the work of Alkim et al. [4] and describes how often the adversary queries the hash oracle and signing oracle respectively.

5 Hardness assumptions

We define some lattice-based problems on which the security of the signature schemes, described in this work, is based. The problems are divided into standard lattice problems and ideal lattice problems. The standard lattice problems are the small integer solution (SIS) problem, the inhomogeneous small integer solution (ISIS) problem, the learning with errors (LWE) problem and the decisional learning with errors (DLWE) problem. The ideal lattice problems consider the ring-small integer solution (R-SIS) problem, the ring-learning with errors (R-LWE) problem, the ring-decisional learning with errors (R-DLWE) problem and the decisional compact knapsack (DCK) problem. Thereafter, we provide an example for the SIS problem as well as an example for the LWE problem.

5.1 Standard lattice problems

A k-dimensional lattice Λ is a discrete subgroup of the vector space $\mathbb{R}^n$ containing all vectors which can be written as integer linear combinations of k linear independent vectors $\{b_1, b_2, ..., b_k\}$.

Definition 5.1 *Let $n, k \in \mathbb{N}$ and $\{\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_k}\}$ be k linear independent vectors in $\mathbb{R}^n$. The set $\Lambda = \left\{ \sum_{i=1}^{k} a_i \mathbf{r_i} \mid a_i \in \mathbb{Z} \right\}$ is called a k-dimensional lattice with Basis $\{\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_k}\}$.*

The small integer solution problem

The small integer solution problem, which is based on the work by Ajtai [2], is that one is given a random Matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and asked to find a short vector $\mathbf{s} \in \mathbb{Z}_q^m$ such that $\mathbf{As} = 0 \mod q$.

Definition 5.2 *The small integer solution* ($\mathsf{SIS}_{q,n,m,\beta}$) *problem. Let* $n, m, q \in \mathbb{N}$, $m > n$ *and* $\beta \in \mathbb{R}$. *One is given a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ *and asked to find a vector* $\mathbf{s} \in \mathbb{Z}_q^m \backslash \{0\}$, *such that* $\mathbf{As} \equiv 0 \mod q$ *and* $\|\mathbf{s}\|_2 \leq \beta$

The value of the boundary β is a critical part. If β is too small, no solution vector can be found but on the other hand, if β is to big, it is easy to find a solution vector. Lyubashevsky describes that the inequality $\beta \geq \sqrt{m} q^{\frac{n}{m}}$ has to hold in order to ensure the existence of a solution vector [16].

The inhomogeneous small integer solution problem

The inhomogeneous small integer solution problem is that one is given a random Matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{y} \in \mathbb{Z}_q^n$ and asked to find a short vector $\mathbf{s} \in \mathbb{Z}_q^m$ such that $\mathbf{As} = \mathbf{y} \mod q$.

Definition 5.3 *The inhomogeneous small integer solution* ($\mathsf{ISIS}_{q,n,m,\beta}$) *problem. Let* $n, m, q \in \mathbb{N}$, $m > n$ *and* $\beta \in \mathbb{R}$. *One is given a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ *and a random vector* $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^n$ *and asked to find a vector* $\mathbf{s} \in \mathbb{Z}_q^m \backslash \{0\}$, *such that* $\mathbf{As} \equiv \mathbf{y} \mod q$ *and* $\|\mathbf{s}\|_2 \leq \beta$

The inhomogeneous small integer solution problem is a generalization of the small integer solution problem (see Definition 5.2)

The learning with errors problem

The learning with errors problem is that, after a secret vector $\mathbf{s}$ is chosen, one is given arbitrary many samples $(\mathbf{a_i}, \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i)$, such that the vectors $\mathbf{a_i}$ are chosen uniformly at random and e_i are small errors, and asked to find $\mathbf{s}$.

Definition 5.4 *The learning with errors* ($\mathsf{LWE}_{n,q,\chi,\phi}$) *problem. Let* $n, q \in \mathbb{N}$ *and* χ, ϕ *be distributions over* $\mathbb{Z}_q$. *After* $\mathbf{s} \leftarrow \chi^n$ *is chosen, one is given arbitrary many samples* $(\mathbf{a_i}, \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, *such that* $\mathbf{a_i} \xleftarrow{\$} \mathbb{Z}_q^n, e_i \leftarrow \phi$, *and asked to find* $\mathbf{s}$.

The learning with errors problem was first introduced by Regev [18].

The decisional learning with errors problem

The decisional learning with errors problem is that one is given arbitrary many samples $(\mathbf{a_i}, b_i)$ and asked to distinguish whether there exists a secret vector $\mathbf{s}$, such that the samples are of the form $(\mathbf{a_i}, \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i \mod q)$ with random vectors $\mathbf{a_i}$ and small errors e_i, or the samples are chosen randomly.

Definition 5.5 *The decisional learning with errors* ($\mathsf{DLWE}_{n,q,\chi,\phi}$) *problem. Let* $n, q \in \mathbb{N}$ *and* χ, ϕ *be distributions over* $\mathbb{Z}_q$. *One is given arbitrary many samples* $(\mathbf{a_i}, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *and asked to distinguish whether there exists* $\mathbf{s} \leftarrow \chi^n$ *such that the samples are of the form* $(\mathbf{a_i}, \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i \mod q)$ *with* $\mathbf{a_i} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_i \leftarrow \phi$ *or the samples are chosen uniformly at random from* $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

5.2 Ideal lattice problems

Ideal lattices are lattices with additional algebraic structure. One main advantage of ideal lattices is that a n-dimensional ideal lattice can represented by just one vector while a standard lattice requires exactly n vectors.

Definition 5.6 *([8]) Ideal Lattice. Given a lattice Λ, such that $\Lambda \subseteq \mathbb{Z}^n$, a polynomial $f(X) = f_0 + ... + f_{n-1}X^{n-1} + f_nX^n$ and a mapping $\phi_f(v_0, ..., v_{n-1}) \rightarrow v_0 + v_1X + ... + v_{n-1}X^{n-1} + f(X)\mathbb{Z}[X]$. Λ is considered an ideal lattice, if $\phi_f(\Lambda)$ is an ideal in $R_f = \mathbb{Z}[X]/(f(X))$. Likewise, if I is an ideal in R_f, then its image Λ under $\phi_f^{-1}(I)$ is an ideal sublattice of $\mathbb{Z}^n$.*

In cryptography ideal lattices are represented as ideals in polynomial rings.

The ring-small integer solution problem

The ring-small integer solution problem is that one is given random polynomials $a_1, a_2, ..., a_m$ and asked to find small polynomials $s_1, s_2, ..., s_m$ such that $a_1s_1 + a_2s_2 + ... + a_ms_m = 0$.

Definition 5.7 *The ring-small integer solution* ($\text{R-SIS}_{q,n,m,\beta}$) *problem. Let $q \in \mathbb{P}$, $n, m \in \mathbb{N}$, $m > n$, $\beta \in \mathbb{R}$ and $R = \mathbb{Z}_q[x]/(x^n + 1)$. One is given $a_1, a_2, ..., a_m \xleftarrow{\$} R$ and asked to find $s_1, s_2, ..., s_m \in R$ such that $\sum_{i=0}^{m}(a_i \cdot s_i) = 0$ in R and $\|(\overline{s_1}, \overline{s_2}, ..., \overline{s_m})^T\|_2 \leq \beta$.*

The ring-learning with errors problem

The ring-learning with errors problem is that, after a secret polynomial s is chosen, one is given arbitrary many samples $(a_i, a_i \cdot s + e_i)$, such that the polynomials a_i are chosen uniformly at random and e_i are polynomials with small coefficients, and asked to find s.

Definition 5.8 *The ring-learning with errors* ($\text{R-LWE}_{n,q,\chi,\phi}$) *problem. Let $n, q \in \mathbb{N}$, $R = \mathbb{Z}_q[x]/(x^n + 1)$ and χ, ϕ be distributions over R. After $s \leftarrow \chi$ is chosen, one is given arbitrary many samples $(a_i, a_i \cdot s + e_i) \in R \times R$, such that $a_i \xleftarrow{\$} R$, $e_i \leftarrow \phi$, and asked to find s.*

The ring-decisional learning with errors problem

The ring-decisional learning with errors problem is that one is given arbitrary many samples (a_i, b_i) and asked to distinguish whether there exists a secret polynomial s such that the samples are of the form $(a_i, a_i \cdot s + e_i)$ with random polynomials a_i and polynomials with small coefficients e_i or the samples are chosen randomly.

Definition 5.9 *The ring-decisional learning with errors* ($\text{R-DLWE}_{n,q,\chi,\phi}$) *problem. Let $R = \mathbb{Z}_q[x]/(x^n + 1)$, $n, q \in \mathbb{N}$ and χ, ϕ be distributions over R. One is given arbitrary many samples $(a_i, b_i) \in R \times R$ and asked to distinguish whether there exists $s \leftarrow \chi$ such that $b_i = a_i \cdot s + e_i$ with $a_i \xleftarrow{\$} R$, $e_i \leftarrow \phi$ or the samples are chosen uniformly at random from $R \times R$.*

The decisional compact knapsack problem

The decisional compact knapsack problem, described in [14], is a variant of the ring-decisional learning with errors problem in which one is given a sample (a, t) and asked to distinguish whether the sample is of the form $(a, as_1 + s_2)$, such that a is a random polynomial and s_1, s_2 are polynomials with small coefficients, or the sample is chosen randomly.

Definition 5.10 *The decisional compact knapsack* ($\mathsf{DCK}_{p,n}$) *problem. Let* $n \in \{2^k : k \in \mathbb{N}\}$, $p \in \{q \in \mathbb{P} : q \equiv 1 \mod 2n\}$, $R^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$ *and* $R_1^{p^n} = \{z \in R^{p^n} : z = \sum_{i=0}^{n-1} z_i x^i \wedge |z_i| \leq 1\}$. *One is given* $(a, t) \in R^{p^n} \times R^{p^n}$ *and asked to distinguish whether there exist* $s_1, s_2 \in R_1^{p^n}$, *such that* $(a, t) = (a, as_1 + s_2)$, *or* (a, t) *is chosen uniformly at random from* $R^{p^n} \times R^{p^n}$.

The decisional compact knapsack problem is described as the ring-decisional learning with errors problem with "aggressive" parameters by Güneysu et al. [14].

5.3 Examples

In this section we illustrate how the boundary β in case of SIS and the addition of errors e_i in case of LWE makes the problems hard. In order to achieve this, we divide each example in two variants. In the first one, we show how easy it is to find a solution without regarding the boundary β and the errors e_i. In the second one, we describe why the boundary β and the errors e_i makes it much harder to find a solution.

Example: Small Integer Solution problem

We set the parameters as follows: $n = 2$, $m = 4$, $q = 211$ and, as described by Lyubashevsky [16], $\beta = 29.05$.

Variant 1

Let $\mathcal{C}$ be a challenger who chooses a random matrix $\mathbf{A} = \begin{pmatrix} 14 & 46 & -57 & -2 \\ 58 & 80 & -30 & 1 \end{pmatrix} \in \mathbb{Z}_{211}^{2\times 4}$ and asks to find a vector $\mathbf{s} \in \mathbb{Z}_{211}^4 \backslash \{0\}$ such that $\mathbf{As} = 0 \mod 211$ holds.

Using Gaussian elimination provides the matrix $\begin{pmatrix} 1 & 0 & -83 & 98 \\ 0 & 1 & 102 & 16 \end{pmatrix}$ from which we obtain the basis $B = \{\mathbf{b_1}, \mathbf{b_2}\} = \{(83; -102; 1; 0)^T, (-98; -16; 0; 1)^T\}$ of the solution space. There are exactly $211^2 - 1 = 44520$ linear combinations of $\mathbf{b_1}$ and $\mathbf{b_2}$ which are not zero and solve the problem.

Variant 2

Let $\mathcal{C}$ be a challenger who chooses a random matrix $\mathbf{A} = \begin{pmatrix} 14 & 46 & -57 & -2 \\ 58 & 80 & -30 & 1 \end{pmatrix} \in \mathbb{Z}_{211}^{2\times 4}$ and asks to find a vector $\mathbf{s} \in \mathbb{Z}_{211}^4 \backslash \{0\}$ such that $\mathbf{As} = 0 \mod 211$ and $\|\mathbf{s}\|_2 \leq 29.05$ holds.

Using the results from Variant 1 we can use the basis $B = \{\mathbf{b_1}, \mathbf{b_2}\} = \{(83; -102; 1; 0)^T, (-98; -16; 0; 1)^T\}$ to find vectors $\mathbf{s}$ such that $\mathbf{As} = 0 \mod 211$ holds. The only thing that remains is to find a vector $\mathbf{s}$ as a linear combination of $\mathbf{b_1}$ and $\mathbf{b_2}$ such that $\|\mathbf{s}\|_2 \leq 29.05$ holds. By computing $\|\mathbf{b_1}\|_2 \approx 101.79$ and

$\|\mathbf{b_2}\|_2 \approx 91.22$ we see that the trivial linear combinations $1 \cdot \mathbf{b_1} + 0 \cdot \mathbf{b_2}$ and $0 \cdot \mathbf{b_1} + 1 \cdot \mathbf{b_2}$ are no solutions. With the simple linear combination $\mathbf{s_1} = -2 \cdot \mathbf{b_1} + 1 \cdot \mathbf{b_2} = (-53; -23; 2; 1)^T$ with $\|\mathbf{s_1}\|_2 \approx 57.82$ we find a significantly smaller vector but even that one is nearly twice as long as it is claimed by $\mathcal{C}$.

There are exactly 64 linear combinations of $\mathbf{b_1}$ and $\mathbf{b_2}$ which size remains lower than $\beta = 29.05$ (see Table A.1 for all solutions), e.g. $\mathbf{s} = 7 \cdot \mathbf{b_1} + 21 \cdot \mathbf{b_2} = (0; 5; 7; 21)^T$ with $\|\mathbf{s}\|_2 \approx 22.69$. These are merely about 0.144% of all the vectors within the solution space. In Table 5.1 the number of solutions and the percentage of the solution space for different values β are listed. There we see that even for the bigger value $\beta = 40.67 = 1.3 \cdot 29.05$ the number of solutions remains lower than 0.7% of the solution space.

In Table A.2 the number of solutions for different instantiations of the SIS problem with fixed $n = 2$, $m = 4$ and $\beta = \sqrt{4 \cdot q}$ are listed. These values suggest that the percentage of vectors, which are short enough decreases with growing values q.

Table 5.1: Number of solutions for different values for β

β	solutions	percentage	β	solutions	percentage
5.81	0	$= 0$ %	46.48	532	$\approx$ 1.195%
11.62	4	$\approx 0.009\%$	52.29	832	$\approx$ 1.869%
17.43	14	$\approx 0.031\%$	58.10	1256	$\approx$ 2.821%
23.24	34	$\approx 0.076\%$	87.16	6334	$\approx 14.227\%$
29.05	**64**	$\approx \mathbf{0.144}\%$	116.21	19814	$\approx 44.506\%$
34.86	166	$\approx 0.373\%$	145.26	36646	$\approx 82.314\%$
40.67	306	$\approx 0.687\%$	174.31	43864	$\approx 98.527\%$

Example: Learning With Errors problem

We set the parameters as follows: $n = 4$, $q = 211$, χ be the uniform distribution over $\mathbb{Z}_{211}$ and ϕ be the Gaussian distribution with standard deviation $\sigma = 1$.

Variant 1

Let $\mathcal{C}$ be a challenger knowing a secret vector $\mathbf{s}$ which is chosen uniformly at random from $\mathbb{Z}_{211}^4$. $\mathcal{C}$ chooses random vectors $\mathbf{a_0}, \mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}$ uniformly at random from $\mathbb{Z}_{211}^4$ and computes $b_i = \langle \mathbf{a_i}, \mathbf{s} \rangle \mod 211$. Finally, $\mathcal{C}$ outputs the samples

- $(\mathbf{a_0}, b_0) = ((-60; 93; 94; 93)^T, 83)$,
- $(\mathbf{a_1}, b_1) = ((-28; -54; -62; -14)^T, 9)$,
- $(\mathbf{a_2}, b_2) = ((-68; -34; -90; 8)^T, -50)$,
- $(\mathbf{a_3}, b_3) = ((103; 1; -29; 3)^T, -21)$

and asks to find $\mathbf{s}$.

The secret vector $\mathbf{s} = (-56; 33; 53; 96)^T$ for which $\langle \mathbf{a_i}, \mathbf{s} \rangle = b_i$ holds can be easily found using Gaussian elimination.

Variant 2

Let $\mathcal{C}$ be a challenger knowing a secret vector $\mathbf{s}$ which is chosen uniformly at random from $\mathbb{Z}_{211}^4$. $\mathcal{C}$ chooses random vectors $\mathbf{a_0}, \mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}$ uniformly at random from $\mathbb{Z}_{211}^4$ and samples errors e_0, e_1, e_2, e_3, e_4 accord-

ing to the Gaussian distribution with standard deviation $\sigma = 1$. For simplicity, we assume that $\mathcal{C}$ samples the errors until they are within $\{-1, 0, 1\}$. Finally, $\mathcal{C}$ computes $b_i = \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i \mod 211$, outputs the samples

- $(\mathbf{a_0}, b_0) = ((\ 40;\ 17;\ 31;\ 58)^T,\ 6)$,
- $(\mathbf{a_1}, b_1) = ((-94; -51;\ 32;\ 83)^T, -75)$,
- $(\mathbf{a_2}, b_2) = ((-94; -46;\ 100;\ 19)^T, -96)$,
- $(\mathbf{a_3}, b_3) = ((-57; -50; -55; -103)^T,\ 7)$,
- $(\mathbf{a_4}, b_4) = ((-49; -65; -11;\ 39)^T, -19)$

and asks to find $\mathbf{s}$.

Based on the fact that each equation is only correct up to ± 1, Gaussian elimination cannot be used to find the solution. There is exactly one vector $\mathbf{s} = (105; 23; 38; -63)^T$ such that the difference between $\langle \mathbf{a_i}, \mathbf{s} \rangle$ and b_i for all samples is at most 1. Table 5.2 lists the number of solution vectors after seeing each new sample outputted by $\mathcal{C}$. We see that the number of solutions decreases from nearly three billion solutions after the first sample to less than 100 solutions after seeing four samples. After seeing the fifth sample, there is only one solution left. For a unique solution the number of samples has to be bigger than the dimension n. Furthermore, the uniqueness depends on the size of the errors and the random vectors $\mathbf{a_i}$.

Below we describe two algorithms described by Regev [19] for solving the problem. Note that there are more algorithms for the problem, e.g. a maximum likelihood algorithm [19], which we do not describe.

The first algorithm asks continuously for samples until seeing enough of the form $((1; 0; 0; 0)^T, b_i)$ to recover the first component of $\mathbf{s}$. The same procedure is used to recover the other components of $\mathbf{s}$. This algorithm is not efficient because the probability that a sample of the form $((1; 0; 0; 0)^T, b_i)$ is outputted is q^{-n} which is exactly the probability that one randomly guesses the secret vector $\mathbf{s}$. Further on, it is not enough to see just one sample of the form $((1; 0; 0; 0)^T, b_i)$ to recover the first component of $\mathbf{s}$. To illustrate that, assume that $\mathcal{C}$ continuously outputs new samples until we see the following two:

- $((1; 0; 0; 0)^T, 105)$,
- $((1; 0; 0; 0)^T, 104)$.

From the first sample we find out that s_1 is an element of $\{-105, 104, 105\}$ and from the second sample we find out that s_1 has to be within $\{103, 104, 105\}$. Even now we can only limit that the first component of $\mathbf{s}$ is either 104 or 105.

Another algorithm, which is based on the work of Blum et al. [7], has certain similarity to the algorithm above. Instead of asking for samples until seeing a sample of the form $((1; 0; 0; 0)^T, b_i)$, the algorithm tries to find a small set of equations such that the addition of these yields to a sample of the form $((1; 0; 0; 0)^T, b_i)$. To illustrate that, assume that $\mathcal{C}$ outputs new samples until we see the following four:

- $((\ 84;\ 104; -12;\ 43)^T, -97)$,
- $((\ 37;\ 59;\ 98; -79)^T,\ 17)$,
- $((\ 62;\ 7;\ 64; -36)^T, -23)$,
- $((-14;\ 41; -61;\ 72)^T,\ 1)$.

By adding these equations we obtain $((1; 0; 0; 0)^T, -104)$ which also just limits the possible values of the first component of the secret vector.

We see that both algorithms try to recover the first component by using samples of the form $((1;0;0;0)^T, b_i)$. The main difference is that the first algorithm continuously asks for new samples until seeing a sample of that form, while the second algorithm tries to generate a sample of that form from the already seen samples.

Table 5.2: Number of solutions for different number of samples

considered samples	solutions	percentage
$(\mathbf{a_0}, b_0)$	28181793	$\approx 1.4218009479\%$
$(\mathbf{a_0}, b_0), (\mathbf{a_1}, b_1)$	400689	$\approx 0.0202151794\%$
$(\mathbf{a_0}, b_0), (\mathbf{a_1}, b_1), (\mathbf{a_2}, b_2)$	5697	$\approx 0.0002874196\%$
$(\mathbf{a_0}, b_0), (\mathbf{a_1}, b_1), (\mathbf{a_2}, b_2), (\mathbf{a_3}, b_3)$	81	$\approx 0.0000040865\%$
$(\mathbf{a_0}, b_0), (\mathbf{a_1}, b_1), (\mathbf{a_2}, b_2), (\mathbf{a_3}, b_3), (\mathbf{a_4}, b_4)$	1	$\approx 0.0000000505\%$

6 Signature schemes based on lattices

In this section we describe selected lattice-based signature schemes. The considered schemes are the signature scheme by Lyubashevsky [16], the signature scheme by Ducas et al. [11], the signature scheme by Güneysu et al. [14], the signature scheme by Abdalla et al. [1], the signature scheme by Bai and Galbraith [5], the signature scheme by Alkim et al. [4] and the signature scheme by Gentry et al. [12].

For each scheme we describe the public parameters to which everyone has access, the key generation algorithm and the corresponding pair of keys. In addition we describe how to sign a message and verify a signature respectively by describing both the signing and the verification algorithm. Further on, we describe on which problem (see Chapter 5) the security is based and whether the signature scheme is secure (see Definition 3.3) or strongly unforgeable (see Definition 3.4). Finally, we compute the bit security of the scheme as described in Chapter 4.2.

6.1 Signature scheme by Lyubashevsky

In this section we describe the signature scheme (LYU12) by Lyubashevsky [16]. When comparing this scheme with the signature scheme by Ducas et al. [11] we see that they look very familiar. This is mainly due to the fact that Ducas et al. tried to increase the efficiency of this scheme.

Table 6.1 provides an overview of the scheme, which contains the public parameters, the key pair, the random oracle and both the signing and verification algorithm. Table 6.2 presents concrete values of the parameters. Further on, Table 6.2 contains the size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security. Ducas et al. [11] show that the signature scheme has a lower bit security as in the original work by Lyubashevsky [16]. Therefore, we use the parameters provided by Ducas et al. [11].

Description of the signature scheme

Public Parameters: The integers ξ, σ, n, m, k, q, the rational numbers η and M and the distribution D_σ with standard deviation σ. Note that Lyubashevsky [16] uses κ instead of ξ. In this work we use κ as the bit hardness.

Signing Key: The signing key is a $m \times k$ matrix $\mathbf{S}$ such that each component of $\mathbf{S}$ is an integer with absolute value at most d.

Verification Key: The verification key consists of two matrices $\mathbf{A}$ and $\mathbf{T}$ such that $\mathbf{A}$ is sampled uniformly from $\mathbb{Z}_q^{n \times m}$ and $\mathbf{T} = \mathbf{AS} \mod q$.

Key Generation Algorithm: First, a $m \times k$ matrix $\mathbf{S}$ with absolute values at most d is chosen. Afterwards, a $n \times m$ matrix $\mathbf{A}$ with values from $\mathbb{Z}_q$ is chosen uniformly at random. The matrices $\mathbf{A}$ and $\mathbf{S}$ are multiplied yielding the matrix $\mathbf{T}$. Finally the algorithm returns (sk, vk) with $sk = \mathbf{S}$ and $vk = (\mathbf{A}, \mathbf{T})$.

Signing Algorithm: To sign a message μ, a vector $\mathbf{y}$ is sampled from the distribution D_σ^m. Afterwards, the message μ and product of $\mathbf{A}$ and $\mathbf{y}$ are hashed together yielding the vector $\mathbf{c}$. Further on, the vector $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$ is computed. In the last step, to ensure that $\mathbf{z}$ does not leak any information about the signing key $\mathbf{S}$, rejection sampling is applied by returning $(\mathbf{z}, \mathbf{c})$ with probability $\min\left(\frac{D_\sigma^m(\mathbf{z})}{M D_{\mathbf{Sc},\sigma}^m(\mathbf{z})}, 1\right)$. Otherwise, the signing algorithm restarts.

Verification Algorithm: To check whether a given signature $(\mathbf{z}, \mathbf{c})$ of a message μ is valid or not, the verification algorithm checks if the inequality $\|\mathbf{z}\|_2 \leq \eta\sigma\sqrt{m}$ and the equality $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$ are satisfied. If that is true, the algorithm accepts the signature, i.e. return 1. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is based on the SIS problem (see Definition 5.2).

Security Property: The scheme is strongly unforgeable (see Definition 3.4).

Computation of the bit security

Theorem 6.1 *([16]). If there is a polynomial-time forger, who makes at most q_s queries to the signing oracle and q_h queries to the hash oracle H, who breaks the signature scheme with probability $\epsilon_{\mathcal{A}}$, then there is a polynomial-time algorithm who can solve the $SIS_{q,n,m,\beta}$ problem for $\beta = (2\eta\sigma + 2d\kappa)\sqrt{m} = \tilde{O}(dn)$ with probability $\approx \frac{\epsilon_{\mathcal{A}}^2}{2(q_h+q_s)}$. Moreover, the signing algorithm produces a signature with probability $\approx 1/M$ and the verifying algorithm accepts a signature produced by an honest signer with probability at least $1 - 2^{-m}$.*

Theorem 6.1 provides us

$$\epsilon_{\mathcal{R}} \approx \frac{\epsilon_{\mathcal{A}}^2}{2(q_s + q_h)}, \tag{6}$$

$$t_{\mathcal{R}} \approx t_{\mathcal{A}}. \tag{7}$$

Table 6.1: LYU12

Public Parameters	
$\xi, \sigma, n, m, k \in \mathbb{N}$; $\eta, M \in \mathbb{Q}$; $q \in \mathbb{P}$; Gaussian distribution D_σ with standard deviation σ ;	
Signing Key: $\mathbf{S} \xleftarrow{\$} \{-d, ..., 0, ..., d\}^{m \times k}$	
Verification Key: $(\mathbf{A}, \mathbf{T})$ s.t. $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{T} \leftarrow \mathbf{AS}$	
Random Oracle: $H : \{0,1\}^* \rightarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \xi\}$	
Signing Algorithm Input: $(\mu, \mathbf{A}, \mathbf{S})$ Output: $(\mathbf{z}, \mathbf{c})$ 1: $\mathbf{y} \leftarrow D_\sigma^m$ 2: $\mathbf{c} \leftarrow H(\mathbf{Ay}, \mu)$ 3: $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$ 4: return $(\mathbf{z}, \mathbf{c})$ with probability $\min\left(\frac{D_\sigma^m(\mathbf{z})}{M D_{\mathbf{Sc},\sigma}^m(\mathbf{z})}, 1\right)$	Verification Algorithm Input: $(\mu, \mathbf{z}, \mathbf{c}, \mathbf{A}, \mathbf{T})$ Output: $\{0,1\}$ 1: **if** $\|\mathbf{z}\|_2 \leq \eta\sigma\sqrt{m}$ **and** $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$ **then** return 1 2: return 0

Table 6.2: Parameter selection LYU12

Parameter	LYU12 III
n	512
q	2^{33}
d	31
m	3253
k	512
η	1.2
ξ	14
σ	300926
M	2.72
signature size [kb]	≈ 71.29
signing key size [kb]	≈ 8192
verification key size [kb]	≈ 8192
Bit Security	80

By solving (6) for $\epsilon_{\mathcal{A}}$ and (7) for $t_{\mathcal{A}}$ we obtain

$$\epsilon_{\mathcal{A}} \approx \sqrt{2\epsilon_{\mathcal{R}}(q_s + q_h)}, \tag{8}$$

$$t_{\mathcal{A}} \approx t_{\mathcal{R}}. \tag{9}$$

Insertion (8) and (9) in (3) provides us

$$\gamma \approx \frac{\ln(t_{\mathcal{R}}) - \ln\left(\sqrt{2\epsilon_{\mathcal{R}}(q_s + q_h)}\right)}{\ln(2)} \tag{10}$$

as the formula for the bit security.

Ducas et al. [11] indicate a bit security of 80. That actually means that the parameters of the scheme are chosen such that the underlying problem has the bit hardness $\kappa = 80$. By using the assumptions (4) and (5) we obtain $\epsilon_{\mathcal{R}} = 1$, $t_{\mathcal{R}} = 2^{80}$, $q_h = 2^{80}$ and $q_s = 2^{40}$. Insertion in (10) finally yields

$$\gamma \approx \frac{\ln(2^{80}) - \ln\left(\sqrt{2 \cdot 1 \cdot (2^{40} + 2^{80})}\right)}{\ln(2)} \approx 39. \tag{11}$$

We see the bit security is just about half the indicated bit security.

6.2 Signature scheme by Ducas, Durmus, Lepoint and Lyubashevsky

The Bimodal lattice signature scheme (BLISS) by Ducas et al. [11] is the fastest signature scheme, within this work, in running times and sizes of the signature and the keys. The main difference between BLISS and the scheme by Lyubashevsky (see Chapter 6.1) is that BLISS rejects a potential signature with a lower probability than the scheme by Lyubashevsky and that BLISS is based on ideal lattices. To simplify matters we describe the matrix variant instead of the ring variant of the scheme.

Table 6.3 provides an overview of the scheme, while Table 6.4 presents concrete values of the parameters. Further on, Table 6.4 contains the size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security

Throughout this section we define the set $\mathbb{B} = \{0, 1\}$. Further on, we define, for natural numbers n and ξ, the set $\mathbb{B}^n_\xi = \{\mathbf{v} \in \mathbb{B}^n : \|\mathbf{v}\|_1 = \xi\}$ which contains vectors of dimension n such that there are exactly ξ components equal to 1 while all the other components are equal to 0.

Description of the signature scheme

Public Parameters: The integers n, m, ξ, the prime q, the real numbers σ, B_2, M, α and the discrete Gaussian distribution D^m_σ with standard deviation σ. Note that Ducas et al. [11] use κ instead of ξ. In this work we use κ as the bit hardness.

Signing Key: The signing key is a (short) $m \times n$ matrix $\mathbf{S}$ such that each component is chosen from $\mathbb{Z}_{2q}$.

Verification Key: The verification key is a $n \times m$ matrix $\mathbf{A}$ such that each component is chosen from $\mathbb{Z}_{2q}$ and $\mathbf{AS} = q\mathbf{I}_n \mod 2q$.

Key Generation Algorithm: We skip the description of the key generation algorithm at this point because it goes beyond the scope of this work. A detailed description is given in [11].

Signing Algorithm: To sign a message μ, a vector $\mathbf{y}$ is sampled uniformly from a distribution D^m_σ as the first step. In the next step the verification key $\mathbf{A}$ and the vector $\mathbf{y}$ are multiplied modulus $2q$ and

the corresponding vector is hashed together with the message μ yielding the vector $\mathbf{c}$. Afterwards, a bit b is chosen randomly from $\mathbb{B}$ and the vector $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{Sc}$ is computed. To ensure that the distribution of the potential signature $(\mathbf{z}, \mathbf{c})$ is independent from the distribution of the signing key, rejection sampling is applied by returning $(\mathbf{z}, \mathbf{c})$ with probability $1/\left(M \exp\left(-\frac{\|\mathbf{Sc}\|_2^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)\right)$. Otherwise, the signing algorithm restarts.

Verification Algorithm: The verification algorithm accepts, i.e. return 1, a signature $(\mathbf{z}, \mathbf{c})$ for a message μ if the three (in)equalities $\|\mathbf{z}\|_2 \leq B_2$, $\|\mathbf{z}\|_\infty < q/4$ and $c = H(\mathbf{Az} + q\mathbf{c} \mod 2q, \mu)$ are satisfied. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is mainly based on the R-SIS problem (see Definition 5.7).

Security Property: The scheme is strongly unforgeable (see Definition 3.4).

Table 6.3: BLISS

Public Parameters	
$\xi, n, m \in \mathbb{N}$; $q \in \mathbb{P}$; $\sigma, B_2, M, \alpha \in \mathbb{R}$; $\mathbb{B} = \{0,1\}$; $\mathbb{B}_\xi^n = \{\mathbf{v} \in \{0,1\}^n : \|\mathbf{v}\|_1 = \xi\}$; Gaussian distribution D_σ with standard deviation σ ;	
Signing Key: $\mathbf{S} \in \mathbb{Z}_{2q}^{m \times n}$	
Verification Key: $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$ s.t. $\mathbf{AS} = q\mathbf{I}_n \mod 2q$	
Random Oracle: $H : \{0,1\}^* \rightarrow \mathbb{B}_\xi^n$	
Signing Algorithm Input: $(\mu, \mathbf{A}, \mathbf{S})$ Output: $(\mathbf{z}, \mathbf{c})$ 1: $\mathbf{y} \xleftarrow{\$} D_\sigma^m$ 2: $\mathbf{c} \leftarrow H(\mathbf{Ay} \mod 2q, \mu)$ 3: $b \xleftarrow{\$} \mathbb{B}$ 4: $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{Sc}$ 5: return $(\mathbf{z}, \mathbf{c})$ with probability $1/\left(M \exp\left(-\frac{\|\mathbf{Sc}\|_2^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)\right)$	Verification Algorithm Input: $(\mu, \mathbf{z}, \mathbf{c}, \mathbf{A}, \mathbf{T})$ Output: $\{0,1\}$ 1: **if** $\|\mathbf{z}\|_2 \leq B_2$ **and** $\|\mathbf{z}\|_\infty < q/4$ **and** $\mathbf{c} = H(\mathbf{Az} + q\mathbf{c} \mod 2q, \mu)$ **then** return 1 2: return 0

Computation of the bit security

The security of the signature scheme is based on the security reduction by Lyubashevsky [16], thus we can use formula (10) to compute the bit security.

Ducas et al. [11] indicate a bit security of 128. We instantiate the parameters of the underlying problem

Table 6.4: Parameter selection of BLISS

Parameter	BLISS II
n	512
m	≈ 1022
q	12289
ξ	23
B_2	11074
M	–
σ	107
α	0.5
signature size [kb]	5
signing key size [kb]	2
verification key size [kb]	7
Bit Security	128

to obtain a bit hardness $\kappa = 128$. Using the assumptions (4) and (5) provides us $\epsilon_{\mathcal{R}} = 1$, $t_{\mathcal{R}} = 2^{128}$, $q_h = 2^{128}$ and $q_s = 2^{64}$. Insertion in (10) finally yields

$$\gamma \approx \frac{\ln(2^{128}) - \ln\left(\sqrt{2 \cdot 1 \cdot (2^{64} + 2^{128})}\right)}{\ln(2)} \approx 63. \tag{12}$$

We see that the bit security is just about half the indicated bit security.

6.3 Signature scheme by Güneysu, Lyubashevsky and Pöppelmann

In this section we describe the signature scheme (GLP) by Güneysu et al. [14] which is described as "A Signature Scheme for Embedded Systems" by the authors.

Table 6.5 provides an overview of the scheme, while Table 6.6 presents concrete values of the parameters. Further on, Table 6.6 contains the size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security. Throughout this section we define the ring $R^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$ and for $k \in \mathbb{N}$ we define the subset $R_k^{p^n}$ of R^{p^n} which contains all polynomials with integer coefficients within $[-k, k]$.

Description of the signature scheme

Public Parameters: The integers n and k, the prime p and the set D_{32}^n which contains all polynomials d such that there are at most 32 coefficients d_i which are either 1 or -1 while the other coefficients are all 0.

Signing Key: The signing key consists of two polynomials s_1, s_2 chosen uniformly at random from $R_1^{p^n}$.

Verification Key: The verification key consists of a polynomial $a \xleftarrow{\$} R^{p^n}$ and the polynomial $t = as_1 + s_2 \mod (x^n + 1) \in R^{p^n}$.

Key Generation Algorithm: The algorithm chooses two polynomials s_1, s_2 uniformly at random from $R_1^{p^n}$ and the polynomial a uniformly at random from R^{p^n}. Afterwards, the polynomial $t = as_1 + s_2$ is computed. Finally, the algorithm returns (sk, vk) with $sk = (s_1, s_2)$ and $vk = (a, t)$.

Signing Algorithm: To sign a message μ, polynomials y_1, y_2 are chosen uniformly at random from $R_k^{p^n}$. The algorithm hashes the polynomial $ay_1 + y_2$ and the message μ together yielding the polynomial c. In the next step two polynomials z_1, z_2 are computed by setting $z_1 = s_1c + y_1$ and $z_2 = s_2c + y_2$ respectively. Following this, rejection sampling is applied by restarting the algorithm if at least one of the two polynomials z_1 and z_2 is not an element of $R_{k-32}^{p^n}$ until both are elements of $R_{k-32}^{p^n}$. In this case the algorithm returns the signature (z_1, z_2, c).

Verification Algorithm: For a given message μ and a given signature (z_1, z_2, c), the verification algorithm accepts the signature, i.e. return 1, if both z_1 and z_2 are elements of $R_{k-32}^{p^n}$ and the equality $c = H(az_1 + z_2 - tc, \mu)$ is satisfied. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is based on the DCK problem (see Definition 5.10).

Security Property: The scheme is strongly unforgeable (see Definition 3.4).

Table 6.5: GLP

Public Parameters	
$n, k \in \mathbb{N}$; $p \in \mathbb{P}$;	
$D_{32}^n = \{d = \sum_{i=0}^{n-1} d_i x^i \in R^{p^n} : d_i \in \{-1, 0, 1\} \wedge \sum_{i=0}^{n-1} \lvert d_i \rvert \leq 32\}$;	
Signing Key: $s_1, s_2 \xleftarrow{\$} R_1^{p^n}$	
Verification Key: (a, t) s.t. $a \xleftarrow{\$} R^{p^n}, t \leftarrow as_1 + s_2$	
Random Oracle: $H : \{0,1\}^* \rightarrow D_{32}^n$	
Signing Algorithm Input: (μ, a, s_1, s_2) Output: (z_1, z_2, c) 1: $y_1, y_2 \xleftarrow{\$} R_k^{p^n}$ 2: $c \leftarrow H(ay_1 + y_2, \mu)$ 3: $z_1 \leftarrow s_1c + y_1$, $z_2 \leftarrow s_2c + y_2$ 4: **if** $z_1 \notin R_{k-32}^{p^n}$ **or** $z_2 \notin R_{k-32}^{p^n}$ **then** goto step 1 5: return (z_1, z_2, c)	Verification Algorithm Input: (μ, z_1, z_2, c, a, t) Output: $\{0, 1\}$ 1: **if** $z_1, z_2 \in R_{k-32}^{p^n}$ **and** $c = H(az_1 + z_2 - tc, \mu)$ **then** return 1 2: return 0

Computation of the bit security

The security of the signature scheme is based on the security reduction by Lyubashevsky [16], thus we can use (10) as formula to compute the bit security.

Güneysu et al. [14] indicate a bit security of 100. That actually means that the parameters of the

Table 6.6: Parameter selection of GLP

Parameter	GLP I
n	512
p	8383489
k	2^{14}
signature size [kb]	8.74
signing key size [kb]	1.58
verification key size [kb]	11.52
Bit Security	100

scheme are chosen such that the underlying problem has the bit hardness $\kappa = 100$. By using the assumptions (4) and (5) we obtain $\epsilon_{\mathcal{R}} = 1$, $t_{\mathcal{R}} = 2^{100}$, $q_h = 2^{100}$ and $q_s = 2^{50}$. Insertion in (10) finally yields

$$\gamma \approx \frac{\ln(2^{100}) - \ln\left(\sqrt{2 \cdot 1 \cdot (2^{50} + 2^{100})}\right)}{\ln(2)} \approx 49. \tag{13}$$

We see the actual bit security is just about half the indicated bit security.

6.4 Signature scheme by Abdalla, Fouque, Lyubashevsky and Tibouchi

The lattice-based lossy signature scheme (AFLT) by Abdalla et al.[1] is generated via transformation of a lattice-based lossy identification scheme. The signature scheme as well as the identification scheme are described in [1].

Table 6.7 provides an overview of the scheme. Abdalla et al. do not provide how to choose practical parameters.

Description of the signature scheme

Public Parameters: The integers p, n, l, ctr, the real number σ, the ring $R^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$, the distribution $D_{R^{p^n},\sigma}$ over R^{p^n} with standard deviation σ, the sets M and G which are subsets of $R^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$ that contains polynomials which coefficients remain below a given boundary. The symbol $\perp$ which the signing algorithm returns if it fails producing a signature.

Signing Key: The signing key consists of two polynomials $s_1, s_2 \leftarrow D_{R^{p^n},\sigma}$.

Verification Key: The verification key is a tuple (a, t) such that a is chosen uniformly at random from R^{p^n} and $t = as_1 + s_2 \mod (x^n + 1)$.

Key Generation Algorithm: First, the algorithm chooses two polynomials $s_1, s_2 \leftarrow D_{R^{p^n},\sigma}$. Afterwards the algorithm chooses a polynomial a uniformly at random from R^{p^n} and computes the polynomial $t = as_1 + s_2 \mod (x^n + 1)$. Finally, (sk, vk) with $sk = (s_1, s_2)$ and $vk = (a, t)$ is returned.

Signing Algorithm: To sign a message μ, the counter ctr is initialized to zero and polynomials y_1, y_2 are chosen uniformly at random from M. In the next step the polynomial $ay_1 + y_2$ is hashed together with the message μ yielding the polynomial c. Further on, two polynomials z_1, z_2 are computed by setting $z_1 = s_1 c + y_1$ and $z_2 = s_2 c + y_2$. Afterwards, the algorithm checks if z_1 or $z_2 \notin G$ and the current number of attempts ctr is smaller than the boundary l. If it is true, ctr is increased by 1 and the algorithm restarts without the initialization of ctr. Otherwise, the algorithm checks if z_1 or $z_2 \notin G$ and sets the potential signature (z_1, z_2, c) to $(\perp, \perp, \perp)$. Finally, the algorithm returns (z_1, z_2, c).

Verification Algorithm: For a given message μ and a given signature (z_1, z_2, c) the verification algorithm accepts the signature, i.e. return 1, if both $z_1, z_2 \in G$ and $c = H(az_1 + z_2 - tc, \mu)$ are satisfied. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is based on the R-DLWE problem (see Definition 5.9).

Security Property: The scheme is secure (see Definition 3.3). For a bigger value p (see Theorem 6.2) the scheme is strongly unforgeable (see Definition 3.4).

Table 6.7: AFLT

Public Parameters
$p, n, ctr, l \in \mathbb{N}$; $R^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$;
$M = \{\mathbf{g} \in R^{p^n} : \|\mathbf{g}\|_\infty \leq n^{3/2}\sigma \log^3 n\}$; $G = \{\mathbf{g} \in R^{p^n} : \|\mathbf{g}\|_\infty \leq (n-1)\sqrt{n}\sigma \log^3 n\}$;
Distribution $D_{R^{p^n},\sigma}$ with standard deviation σ ;
Signing Key: $s_1, s_2 \xleftarrow{\$} D_{R,\sigma}$
Verification Key: (a, t) s.t. $a \xleftarrow{\$} R, t \leftarrow as_1 + s_2$
Random Oracle: $H : \{0,1\}^* \to \{\mathbf{g} \in R^{p^n} : \|\mathbf{g}\|_\infty \leq \log n\}$

Signing Algorithm	Verification Algorithm
Input: (μ, a, s_1, s_2)	Input: (μ, z_1, z_2, c, a, t)
Output: (z_1, z_2, c)	Output: $\{0, 1\}$
1: $ctr \leftarrow 0$	1: **if** $z_1, z_2 \in G$ **and** $c = H(az_1 + z_2 - tc, \mu)$ **then** return 1
2: $y_1, y_2 \xleftarrow{\$} M$	2: return 0
3: $c \leftarrow H(ay_1 + y_2, \mu)$	
4: $z_1 \leftarrow s_1 c + y_1, z_2 \leftarrow s_2 c + y_2$	
5: **if** ($z_1, z_2 \notin G$ **and** $ctr < l$ **then** $ctr \leftarrow ctr + 1$ goto step 2	
6: **if** $z_1, z_2 \notin G$ **then** $(z_1, z_2, c) \leftarrow (\perp, \perp, \perp)$	
7: return (z_1, z_2, c)	

Theorem 6.2 *([1]). If $p \gg \sigma^{2/\alpha} \cdot n^{3/\alpha+\eta}$ for some $\eta > 0$, and the* R-LWE *problem over R with standard deviation σ is $(\epsilon_{\mathcal{R}}, t_{\mathcal{R}})$-hard, then the signature scheme is $(t_{\mathcal{A}}, q_h, q_s, \epsilon_{\mathcal{A}})$-unforgeable against chosen message attacks in the random oracle model for:*

$$t_{\mathcal{A}} \approx t_{\mathcal{R}} - O(q_s \cdot t_1),$$
$$\epsilon_{\mathcal{A}} \approx \epsilon_{\mathcal{R}} + (q_h + q_s) \cdot negl(n)$$

(where t_1 is the cost of a multiplication in R), and it outputs a valid signature with probability $\geq 1-(1-1/e^2+2/(en))^l$. If, moreover, $p \gg \sigma^{2/\alpha} \cdot n^{4/\alpha+\eta}$ for some $\eta > 0$, the signature scheme is $(t_{\mathcal{A}}, q_h, q_s, \epsilon_{\mathcal{A}})$-strongly unforgeable against chosen message attacks.

The difference to the security reduction by Lyubashevsky [16] is that Abdalla et al. [1] do not use the forking lemma [17] to prove the security. That is the reason why they find a tight security reduction. Theorem 6.2 provides us

$$\epsilon_{\mathcal{A}} \approx \epsilon_{\mathcal{R}} + (q_h + q_s) \cdot negl(n), \tag{14}$$
$$t_{\mathcal{A}} \approx t_{\mathcal{R}} - O(q_s \cdot t_1). \tag{15}$$

Because of the negligibly function in (14) and the constant term in (15) we obtain

$$\epsilon_{\mathcal{A}} \approx \epsilon_{\mathcal{R}}, \tag{16}$$
$$t_{\mathcal{A}} \approx t_{\mathcal{R}}. \tag{17}$$

Insertion (16) and (17) in (3) provides us

$$\gamma \approx \frac{\ln(t_{\mathcal{R}}) - \ln(\epsilon_{\mathcal{A}})}{\ln(2)} \tag{18}$$

as the formula for the bit security.

There is no indicated bit security for the signature scheme given by the authors. Suppose that the parameters of the underlying problem are chosen to obtain a bit hardness κ. By using the assumption (4) we obtain $\epsilon_{\mathcal{R}} = 1$ and $t_{\mathcal{R}} = 2^{\kappa}$. Insertion in (18) yields

$$\gamma \approx \frac{\ln(2^{\kappa}) - \ln(1)}{\ln(2)} \approx \frac{\kappa \cdot \ln(2) - 0}{\ln(2)} \approx \kappa. \tag{19}$$

We see that for every instantiation of the parameters the bit security and the bit hardness are equal.

6.5 Signature scheme by Bai and Galbraith

In this section we describe the signature scheme (BG) described by Dagdelen et. al [10]. This is a variant of the signature scheme by Bai and Galbraith [5] who uses GLP (see Chapter 6.3) to construct a scheme aiming for short signatures to increase the efficiency.

Table 6.8 provides an overview of the scheme while, Table 6.9 presents concrete values of the parameters for both this scheme and TESLA (see Chapter 6.6). Further on, Table 6.9 contains the size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security.

Throughout this section we define, following Bai and Galbraith [5], the function $\lceil a \rfloor_{2^d}$ which get an integer a as input and output the unique integer x within the range $(-2^{d-1}, 2^{d-1}]$ such that a and x have the same remainder modulo 2^d. Based on this we define the function $\lfloor a \rceil_d$ which drops d bits of a. We expand this function to vectors by applying it to each component of the vector.

Description of the signature scheme

Public Parameters: The integers m,n,q,σ,ω,κ,B,L, the real number U, the $m \times n$ matrix $\mathbf{A}$, the Gaussian distribution D_σ with standard deviation σ, and the function F which gets a binary string of length κ as input and generates a n-dimensional vector with weight ω. Furthermore the functions $\lceil \cdot \rfloor_{2^d}$ and $\lfloor \cdot \rceil_d$ which are described above.

Signing Key: The signing key consists of a $n \times n$ matrix $\mathbf{S}$ and a $m \times n$ matrix $\mathbf{E}$ which are chosen from the distributions $D_\sigma^{n \times n}$ and $D_\sigma^{m \times n}$ respectively.

Verification Key: The verification key is a $m \times n$ matrix $\mathbf{T} \leftarrow \mathbf{AS} + \mathbf{E} \mod q$.

Key Generation Algorithm: The algorithm chooses $\mathbf{S} \leftarrow D_\sigma^{n \times n}$ and $\mathbf{E} \leftarrow D_\sigma^{m \times n}$. Afterwards, the algorithm checks if $checkE(\mathbf{E}) = 0$ and restarts if it is true. Otherwise, the algorithm returns (sk, vk) with $sk \leftarrow (\mathbf{S}, \mathbf{E})$ and $vk \leftarrow \mathbf{T}$. A description of the function $checkE$ is given by Dagdelen et al. [10].

Signing Algorithm: To sign a message μ, the algorithm chooses a vector $\mathbf{y}$ uniformly at random from the set $[-B, B]^n$ which is multiplied with the matrix $\mathbf{A}$ to obtain the vector $\mathbf{v}$. The vector $\mathbf{v}$, after dropping the last d bits, and the message μ are hashed together yielding the binary string c which is used as input for F to obtain the vector $\mathbf{c}$. Afterwards, the vectors $\mathbf{z}$ and $\mathbf{w}$ are computed by setting $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$ and $\mathbf{w} \leftarrow \mathbf{v} - \mathbf{Ec}$. Finally, the algorithm returns $(\mathbf{z}, c)$ as a signature if both $|[\mathbf{w}_i]_{2^i}| > 2^{d-1} - L$ and $\|\mathbf{z}\|_\infty > B - U$ holds. Otherwise, the algorithm restarts.

Verification Algorithm: For a given message μ and a given signature $(\mathbf{z}, c)$, the verification algorithm uses c as input for F to obtain the vector $\mathbf{c}$ and computes $\mathbf{w}' \leftarrow \mathbf{Az} - \mathbf{Tc} \mod q$. The vector $\mathbf{w}$, after dropping the last d Bits, and the message are hashed together yielding the binary string c'. Finally, the algorithm accepts the signature, i.e. return 1, if both $c' = c$ and $\|\mathbf{z}\|_\infty \leq B - U$ holds. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is based on the SIS problem (see Definition 5.2) and the DLWE problem (see Definition 5.5).

Security Property: The scheme is strongly unforgeable (see Definition 3.4).

Computation of the bit security

Theorem 6.3 *([5]). Let q be prime. Let parameters n, m, d, κ, B be such that*

$$(2B)^n q^{m-n} \geq (2^{d+1})^m 2^\kappa$$

and suppose that $Pr_{s_1,s_2 \leftarrow \{0,1\}^\kappa}[F(s_1) = F(s_2)] \leq \frac{c_1}{2^\kappa}$ for some constant c_1 holds. Let $D_y = [-B, B]$ with the uniform distribution and let $\mathbf{S}, \mathbf{E}$ have entries chosen from discrete Gaussian distributions with standard deviation $\sigma = \alpha q$. Let $\mathcal{A}$ be a forger against the signature scheme in the random oracle model that makes q_h hash queries, q_s sign queries, runs in time $t_\mathcal{A}$ and succeeds with probability $\epsilon_\mathcal{A}$. Then there is a negligible ϵ and some $0 \leq \delta' \leq \epsilon_\mathcal{A}$ such that $\mathcal{A}$ can be turned into either one of the following two algorithms:

1. *an algorithm, running in time approximately t and with advantage $\epsilon_\mathcal{A} - \delta' - \epsilon$, that solves the (n, m, q, α)-decisional* LWE *problem.*

2. *an algorithm, running in time approximately $2t$ and with success probability $\delta'\left(\frac{\delta'}{q_h} - \frac{1}{2^\kappa}\right)$, that solves the unbalanced $(m+n, m, q)$-*SIS *problem: Given an $m \times (n+m)$ matrix $\mathbf{A}'$ to find a length n vector $\mathbf{y_1}$ and a length m vector $\mathbf{y_2}$ such that $\|\mathbf{y_1}\|_\infty, \|\mathbf{y_2}\|_\infty \leq \max(2B, 2^{d-1}) + 2E'w$ and $\mathbf{A}'\begin{pmatrix}\mathbf{y_1}\\ \mathbf{y_2}\end{pmatrix} \equiv 0 \pmod q$ where E' satisfies*

$$(2E')^{m+n} \geq q^m 2^\kappa$$

The first security reduction is tight while second one is non-tight. To compute the bit security we only need the less tighter security reduction thus we ignore the first one at this point. The second security reduction in Theorem 6.3 provides us

$$\epsilon_\mathcal{R} \approx \delta'\left(\frac{\delta'}{q_h} - \frac{1}{2^\kappa}\right), 0 \leq \delta' \leq \epsilon_\mathcal{A}, \tag{20}$$

$$t_\mathcal{R} \approx 2t_\mathcal{A}. \tag{21}$$

Because of $0 \leq \delta' \leq \epsilon_\mathcal{A}$ there exists a $\rho \in [0, 1]$ such that $\delta' = \rho\epsilon_\mathcal{A}$. Insertion in (20) yields

$$\epsilon_\mathcal{R} \approx \rho\epsilon_\mathcal{A}\left(\frac{\rho\epsilon_\mathcal{A}}{q_h} - \frac{1}{2^\kappa}\right), 0 \leq \rho' \leq 1. \tag{22}$$

By solving (22) for $\epsilon_\mathcal{A}$ and (21), ignoring the linear factor, for $t_\mathcal{A}$ we get

$$\epsilon_\mathcal{A} \approx \frac{\frac{\rho}{2^\kappa} + \sqrt{\frac{-\rho}{2^\kappa}^2 - 4 \cdot \frac{\rho^2}{q_h} \cdot -\epsilon_\mathcal{R}}}{2 \cdot \frac{\rho^2}{q_h}}, \tag{23}$$

$$t_\mathcal{A} \approx t_\mathcal{R}. \tag{24}$$

Insertion (23) and (24) in (3) provides us

$$\gamma \approx \frac{\ln(t_{\mathcal{R}}) - \ln\left(\frac{\frac{\rho}{2^{\kappa}} + \sqrt{\left(\frac{-\rho}{2^{\kappa}}\right)^2 - 4 \cdot \frac{\rho^2}{q_h} \cdot -\epsilon_{\mathcal{R}}}}{2 \cdot \frac{\rho^2}{q_h}}\right)}{\ln(2)} \tag{25}$$

as the formula for the bit security.

The signature scheme has an indicated bit security of 128 [10]. We instantiate the parameters of the underlying problem to obtain a bit hardness $\kappa = 128$. Using the assumptions (4) and (5) provides us $\epsilon_{\mathcal{R}} = 1$, $t_{\mathcal{R}} = 2^{128}$, $q_h = 2^{128}$ and $q_s = 2^{64}$. Furthermore, we assume the best case in which $\rho = 1$. Insertion in (25) finally yields

$$\gamma \approx \frac{\ln(2^{128}) - \ln\left(\frac{\frac{1}{2^{128}} + \sqrt{\left(\frac{-1}{2^{128}}\right)^2 - 4 \cdot \frac{1^2}{2^{128}} \cdot -1}}{2 \cdot \frac{1^2}{2^{128}}}\right)}{\ln(2)} \approx 63. \tag{26}$$

We see that the bit security is just half the indicated bit security.

6.6 Signature scheme by Alkim, Bindel, Buchmann, Dagdelen and Schwabe

The Tightly-secure, Efficient signature scheme from Standard Lattices (TESLA) by Alkim et al. [4] is basically the signature scheme by Bai and Galbraith [5]. The difference of the schemes are the instantiation of the parameters, the security reduction, the hardness assumption and the security property. Hence, we only describe the difference to the signature scheme by Bai and Galbraith (see Chapter 6.5) and do not recall the entire description of the signature scheme.

Table 6.8 provides an overview of the scheme, while Table 6.9 presents concrete values of the parameters for BG and TESLA. Further on, Table 6.9 contains the size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security.

Description of the signature scheme

Hardness Assumption: The security of the scheme is based on the DLWE problem (see Definition 5.5).

Security Property: The scheme is secure (see Definition 3.3).

Computation of the bit security

Theorem 6.4 *([4]). If* LWE *is* $(t_{\mathcal{R}}, \epsilon_{\mathcal{R}})$*-hard, the signature scheme TESLA is* $(t_{\mathcal{A}}, q_h, q_s, \epsilon_{\mathcal{A}})$*- unforgeable against adaptively chosen message attacks in the random oracle model where* $t_{\mathcal{R}} \approx t_{\mathcal{A}} + O(q_s \kappa^3)$ *and* $\epsilon_{\mathcal{R}} \approx \epsilon_{\mathcal{A}}\left(1 - \frac{q_s(q_s+q_h)2^{(d+1)\cdot m}}{(2B+1)^n q^{m-n}}\right) - \frac{q_h 2^{d\cdot n}(2B-2U+2)^n}{q^m} - \frac{(28\sigma+1)^{m\cdot n+n^2}}{q^{m\cdot n}}$.

Table 6.8: BG and TESLA

Public Parameters
$m, n, q, \sigma, \omega, \kappa, \sigma, B, L \in \mathbb{N}$; $U \in \mathbb{R}$; $\mathbf{A} \in \mathbb{Z}^{m \times n}$;
Gaussian distribution D with standard deviation σ;
$[\cdot]_{2^d} : \mathbb{Z} \to \mathbb{Z}, [a]_{2^d} \to x$ s.t. $x \in (-2^{d-1}, 2^{d-1}] \subset \mathbb{Z}$ and $x \equiv a \mod 2^d$;
$\lfloor \cdot \rceil_d : \mathbb{Z} \to \mathbb{Z}, \lfloor a \rceil_d \to (a - [a]_{2^d})/2^d$;
$F : \{0,1\}^\kappa \to \{-1,0,1\}^k$ s.t. $\|F(v)\|_1 = w$ and $Pr[s_1, s_2 \xleftarrow{\$} \{0,1\}^\kappa | (F(s_1) = F(s_2))] \leq \frac{c_1}{2^\kappa}$;
Signing Key: $\mathbf{S} \leftarrow D_\sigma^{n \times n}$ and $\mathbf{E} \leftarrow D_\sigma^{m \times n}$
Verification Key: $\mathbf{T} \leftarrow \mathbf{AS} + \mathbf{E}$
Random Oracle: $H : \{0,1\}^* \to \{0,1\}^\xi$

Signing Algorithm

Input: $(\mu, \mathbf{A}, \mathbf{S}, \mathbf{T})$
Output: $(\mathbf{z}, c)$

1: $\mathbf{y} \xleftarrow{\$} [-B, B]^n$
2: $\mathbf{v} \leftarrow \mathbf{Ay} \mod q$
3: $c = H(\lfloor \mathbf{v} \rceil_d, \mu)$
4: $\mathbf{c} = F(c)$
5: $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$
6: $\mathbf{w} \leftarrow \mathbf{v} - \mathbf{Ec} \mod q$
7: **if** $|[w_i]_{2^d}| > 2^{d-1} - L$ **or** $\|\mathbf{z}\|_\infty > B - U$ **then** Restart
8: return $(\mathbf{z}, c)$

Verification Algorithm

Input: $(\mu, \mathbf{z}, c, \mathbf{A}, \mathbf{T})$
Output: $\{0,1\}$

1: $\mathbf{c} \leftarrow F(c)$
2: $\mathbf{w}' \leftarrow \mathbf{Az} - \mathbf{Tc} \mod q$
3: $c' = H(\lfloor \mathbf{w} \rceil_d, \mu)$
4: **if** $c' = c$ **and** $\|\mathbf{z}\|_\infty \leq B - U$ **then** return 1
5: return 0

Theorem 6.4 provides us

$$\epsilon_{\mathcal{R}} \approx \epsilon_{\mathcal{A}} \left(1 - \frac{q_s(q_s + q_h)2^{(d+1)\cdot m}}{(2B+1)^n q^{m-n}}\right) - \frac{q_h 2^{d \cdot n}(2B - 2U + 2)^n}{q^m} - \frac{(28\sigma + 1)^{m \cdot n + n^2}}{q^{m \cdot n}}, \tag{27}$$

$$t_{\mathcal{R}} \approx t_{\mathcal{A}} + O(q_s \kappa^3). \tag{28}$$

For the parameter values given in Table 6.9, the fractions in (27) are negligibly. This fact and the linear factor in (28) provides us

$$\epsilon_{\mathcal{A}} \approx \epsilon_{\mathcal{R}}, \tag{29}$$

$$t_{\mathcal{A}} \approx t_{\mathcal{R}}. \tag{30}$$

Insertion (29) and (30) in (3) provides us

$$\gamma \approx \frac{\ln(t_{\mathcal{R}}) - \ln(\epsilon_{\mathcal{R}})}{\ln(2)} \tag{31}$$

Table 6.9: Parameter selection of BG and TESLA

Parameter	BG	TESLA
κ	128	128
n	532	416
m	840	800
σ	43	114
α	428	1
k	(14)	–
L	2322	6042
ω	18	20
B	$2^{21}-1$	$2^{23}-1$
U	2554.1	7138
d	23	24
q	$2^{29}-3$	$2^{27}-39$
signature size [kb]	≈ 11.7	10.11
signing key size [kb]	≈ 7127	5407
verification key size [kb]	≈ 12615	8766
Bit Security	128	128

as the formula for the bit security.

Alkim et al. [4] indicate a bit security of 128 using the parameters given in Table 6.9. Hence, we assume that the underlying problem has a bit hardness of $\kappa = 128$. Using assumption (4) provides us $\epsilon_{\mathcal{R}} = 1$ and $t_{\mathcal{R}} = 2^{128}$. Insertion in (31) finally yields

$$\gamma \approx \frac{\ln(2^{128}) - \ln(1)}{\ln(2)} \approx 128. \tag{32}$$

We see that the bit security and the indicated bit security are equal.

6.7 Signature scheme by Gentry, Peikert and Vaikuntanathan

The signature scheme (GPV) by Gentry et al. [12], in contrary to the other schemes, is based on trapdoor functions.

Table 6.10 provides an overview of the scheme, while Table 6.11 provides concrete values of the parameters. Further on, Table 6.11 contains the corresponding size of the signature, the signing key and the verification key (in kilobit [kb]), as well as the indicated bit security. The values of the parameters are taken from the implementation by El Bansarkhani and Buchmann [6].

Throughout this section we define the function $SampleD$ which gets a basis $\mathbf{T}$ of a lattice Λ, a real number s and a vector $\mathbf{t}$ as input and outputs a vector $\mathbf{v}$ within Λ such that the distance between $\mathbf{v}$ and $\mathbf{t}$ is small. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ we define the set $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}_q^m : \mathbf{Ae} \equiv 0 \mod q\}$ which contains all vectors such that the product of the matrix and the vector is 0. Below we define the terms *one-way function* and *trapdoor function*.

A one-way-function is a function which can be efficiently computed but it is very difficult to compute its inverse function.

Definition 6.5 *Let D, R be two sets. A function $f : D \to R$ is called a one-way-Function if for given $x \in D$ it is easy to compute $y = f(x)$ and for given $y \in R$ it is hard to compute $x \in D$ s.t. $f(x) = y$.*

A trapdoor function is a one-way-function which can be efficiently inverted by knowing a secret.

Definition 6.6 *Let D, R be two sets, $f : D \to R$ be a one-way-function and t be a secret. The function f is called a trapdoor function if one, knowing t, is given $y \in R$ and able compute $x \in D$ s.t. $f(x) = y$.*

Description of the signature scheme

Public Parameters: The integers n, m, q, the real number s, the sets D_n, R_n and the function $SampleD$.

Signing Key: The signing key consists of a good basis $\mathbf{T}$ of $\Lambda^{\perp}(\mathbf{A})$. A basis $\mathbf{T}$ is called good if the Gram-Schmidt vectors of $\mathbf{T}$ are short.

Verification Key: The verification key consists of the $n \times m$ matrix $\mathbf{A}$.

Key Generation Algorithm: Gentry et al. [12] describe how to create the key pair using a method by Ajtai [3].

Signing Algorithm: To sign a message μ, the algorithm hashes μ yielding some point y within D. Afterwards, the algorithm chooses an arbitrary vector $\mathbf{t}$ such that $\mathbf{Ay} = \mathbf{t} \mod q$ and computes the vector $\mathbf{v} \leftarrow SampleD(\mathbf{T}, s, -\mathbf{t})$. Finally, the algorithm returns the signature $\mathbf{z} \leftarrow \mathbf{t} + \mathbf{v}$.

Verification Algorithm: The algorithm accepts, i.e. return 1, a given signature $\mathbf{z}$ of a message μ, if both $\mathbf{z} \in D_n$ and the equality $H(\mu) = \mathbf{Az}$ are satisfied. Otherwise, the signature is rejected, i.e. return 0.

Hardness Assumption: The security of the scheme is based on the ISIS problem (see Definition 5.3).

Security Property: The scheme is strongly unforgeable (see Definition. 3.4).

Computation of the bit security

Proposition 6.7 *([12]) The scheme described in Table 6.10 is strongly existentially unforgeable under a chosen-message attack.*

The proof of Proposition 6.7 provides us

$$\epsilon_{\mathcal{R}} \approx \epsilon_{\mathcal{A}} - negl(n), \tag{33}$$

$$t_{\mathcal{R}} \approx t_{\mathcal{A}}. \tag{34}$$

Because of the negligibly function in (33) we obtain

$$\epsilon_{\mathcal{A}} \approx \epsilon_{\mathcal{R}}, \tag{35}$$

$$t_{\mathcal{A}} \approx t_{\mathcal{R}}. \tag{36}$$

Table 6.10: GPV

Public Parameters	
$n, m, q \in \mathbb{N}$; $s \in \mathbb{R}$; $D_n = \{\mathbf{e} \in \mathbb{Z}^m : \|\mathbf{e}\|_2 \leq s\sqrt{m}\}$;	
Signing Key: $\mathbf{T}$ s.t. $\mathbf{T}$ is a good basis of $\Lambda^{\perp}(\mathbf{A})$	
Verification Key: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$	
Random Oracle: $H : \{0,1\}^* \to \mathbb{Z}_q^n$	
Signing Algorithm Input: $(\mu, \mathbf{T})$ Output: $\mathbf{z}$ 1: $\mathbf{y} = H(\mu)$ 2: $\mathbf{t} \leftarrow \mathbb{Z}^m$ s.t. $\mathbf{At} = \mathbf{y} \mod q$ 3: $\mathbf{v} \leftarrow SampleD(\mathbf{T}, s, -\mathbf{t})$ 4: return $\mathbf{z} \leftarrow \mathbf{t} + \mathbf{v}$	Verification Algorithm Input: $(\mu, \mathbf{z}, \mathbf{A})$ Output: $\{0,1\}$ 1: **if** $\mathbf{z} \in D_n$ **and** $H(\mu) = \mathbf{Az}$ **then** return 1 2: return 0

Table 6.11: Parameter selection of GPV

Parameter	GPV
n	512
m	16384
q	2^{30}
s	≈ 4292
signature size [kb]	235.2
signing key size [kb]	96512
verification key size [kb]	222720
Bit Security	108

Insertion (35) and (36) in (3) provides us

$$\gamma \approx \frac{\ln(t_{\mathcal{R}}) - \ln(\epsilon_{\mathcal{R}})}{\ln(2)} \tag{37}$$

as the formula for the bit security.

Gentry et al. [12] indicates a bit security of 128. We instantiate the parameters of the underlying problem to obtain a bit hardness $\kappa = 128$. Using assumption (4) provides us $\epsilon_{\mathcal{R}} = 1$ and $t_{\mathcal{R}} = 2^{128}$. Insertion in (37) finally yields

$$\gamma \approx \frac{\ln(2^{128}) - \ln(1)}{\ln(2)} \approx 128 \tag{38}$$

We see that the bit security and the indicated bit security are equal.

7 Comparison

Within this section we summarize the signature schemes described in Chapter 6 and compare these with regard to security and the size of the signatures and the keys. For convenience Table 7.1 lists all the important information of the signature schemes. The size of the signing key, the verification key and the signature are in kilobit [kb].

There are three schemes with a tight security reduction (AFLT,TESLA,GPV) and four schemes with a non-tight security reduction (LYU12,BLISS,GLP,BG). When comparing the hardness assumption we see that the schemes with a tight security reduction are based on the (R-)DLWE problem and the ISIS problem while the schemes with a non-tight security reduction are based on the (R-)SIS problem and the DCK problem. Note that BG has a tight security reduction based on the DLWE problem which is not mentioned in Table 7.1 as it has no effect to the bit security of the scheme, since the security reduction based on the SIS problem is non-tight.

By comparing the bit hardness and bit security of each scheme, we see that only in case of the schemes with a tight security reduction the bit hardness and bit security are equal. For all the other schemes the bit security is merely half the bit hardness. As described in Chapter 4.2, this leads to lower provable security as it opens the chance that breaking the scheme is easier than solving the problem.

By comparing the security properties, we see that all schemes are strongly unforgeable except for AFTL and TESLA which are secure. As described in Chapter 3, this is more of theoretical interest and does not lead to weaknesses in most real world applications.

For the efficiency comparison[1] we first focus on the size of the signature. We see that BLISS, GLP, BG and TESLA have signature sizes between 5kb and 11.7kb while LYU12 and GPV have signature sizes of 71.29kb and 235.2kb respectively. These values make LYU12 and GPV much less efficient compared to the other four schemes. By comparing the sizes of the keys we see that GPV has by far the biggest keys while BLISS and GLP have the smallest keys. Even BG and TESLA, which have similar signature sizes compared with BLISS and GLP, have key sizes which are more than factor 2^{10} bigger than the keys of BLISS and GLP. This resulted from the fact that BG and TESLA use standard lattices (see Definition 5.1) while BLISS and GLP are based on ideal lattices (see Definition 5.6). By comparing the key sizes of LYU12, BG and TESLA, we see that LYU12 and BG have keys with similar size. TESLA, however, has the smallest keys of these schemes which makes it the most efficient scheme of these three, only outperformed by BLISS and GLP.

It can be summarized that, at the moment, every signature scheme, described in this work, lacks either a tight security reduction or efficiency. The most promising schemes, considering both security and efficiency, are BLISS, GLP and TESLA.

[1] Note that AFLT is not considered in this comparison because there are no values of the parameters given.

Table 7.1: Comparison table

Signature Scheme	Hardness Assumption	Tightness	Bit		Strongly unforgeable	Size [kb]		
			Hardness	Security		Signature	Signing key	Verification key
LYU12 [16]	SIS	Loose	80	39	yes	71.29	8192.00	8192.00
BLISS [11]	R-SIS	Loose	128	63	yes	5.00	2.00	7.00
GLP [14]	DCK	Loose	100	49	yes	8.74	1.58	11.52
AFLT [1]	R-DLWE	Tight	—	—	no	—	—	—
BG [10]	SIS and DLWE	Loose	128	63	yes	11.70	7127.00	12615.00
TESLA [4]	DLWE	Tight	128	128	no	10.11	5407.00	8766.00
GPV [12]	ISIS	Tight	108	108	yes	235.20	96515.00	222720.00

8 Conclusion

During this work we showed that there are crucial differences considering the provable security of lattice-based signature schemes due to the tightness of the security reduction. In case of the schemes with a tight security reduction (AFLT [1], TESLA [4], GPV [12]), the bit security of the schemes and the bit hardness of the underlying problem are equal. In case of the schemes with a non-tight security reduction (LYU12 [16], BLISS [11], GLP [14], BG [10]), the bit security of the scheme is merely half the bit hardness of the underlying problem. On the other hand, it turns out that the schemes with a tight security reduction are significant less efficient compared to schemes like BLISS and GLP.

We also showed that schemes like LYU12 and BG form a good foundation of lattice-based signature schemes. Based on this, the schemes BLISS and TESLA are constructed which are more efficient and ensure higher bit security than LYU12 and BG. But even these schemes require further research and improvements for practical use.

Furthermore, the comparison of the schemes showed that all security reductions based on the (R-)SIS problem are non-tight while all security reductions based on the (R-)DLWE problem are tight.

Based on the fact that the most efficient signature schemes (BLISS and GLP) are based on ideal lattices, a ring-TESLA variant can be a great improvement of lattice-based signature schemes. Also a tight security reduction for BLISS and GLP is desirable to increase their provable security.

Acknowledgement

I would like to thank all who supported me during this work. First of all, Prof. Johannes Buchmann and Nina Bindel for giving me the opportunity to write my work in their field of research. I would also like to thank my family and friends, in particular Anja Bartsch and Martin Scheuerlein, for supporting me during this work. Finally, i am very grateful to Nina Bindel for the useful discussions and the great support which has contributed to the success of this work.

References

[1] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 572–590, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In 28th Annual ACM Symposium on Theory of Computing, pages 99–108, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.

[3] Miklós Ajtai. Generating hard instances of the short basis problem. In Automata, Languages and Programming, pages 1–9. Springer, 1999.

[4] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, and Peter Schwabe. Tesla: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/XXX, 2015.

[5] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, Topics in Cryptology – CT-RSA 2014, volume 8366 of Lecture Notes in Computer Science, pages 28–47, San Francisco, CA, USA, February 25–28, 2014. Springer, Berlin, Germany.

[6] Rachid El Bansarkhani and Johannes Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography, volume 8282 of Lecture Notes in Computer Science, pages 48–67, Burnaby, BC, Canada, August 14–16, 2013. Springer, Berlin, Germany.

[7] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In 32nd Annual ACM Symposium on Theory of Computing, pages 435–440, Portland, Oregon, USA, May 21–23, 2000. ACM Press.

[8] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In Michel Abdalla and Paulo S. L. M. Barreto, editors, Progress in Cryptology - LATINCRYPT 2010: 1st International Conference on Cryptology and Information Security in Latin America, volume 6212 of Lecture Notes in Computer Science, pages 255–272, Puebla, Mexico, August 8–11, 2010. Springer, Berlin, Germany.

[9] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another look at tightness. In Ali Miri and Serge Vaudenay, editors, SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography, volume 7118 of Lecture Notes in Computer Science, pages 293–319, Toronto, Ontario, Canada, August 11–12, 2011. Springer, Berlin, Germany.

[10] Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sanchez, and Peter Schwabe. High-speed signatures from standard lattices. Latincrypt 2014, 2014.

[11] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, Advances in Cryptology – CRYPTO 2013, Part I, volume 8042 of Lecture Notes in Computer Science, pages 40–56, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Germany.

[12] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, 40th Annual ACM Symposium on Theory of Computing, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing, 17(2):281–308, April 1988.

[14] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, Cryptographic Hardware and Embedded Systems – CHES 2012, volume 7428 of Lecture Notes in Computer Science, pages 530–547, Leuven, Belgium, September 9–12, 2012. Springer, Berlin, Germany.

[15] Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography. CRC Press, 2014.

[16] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[17] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, Advances in Cryptology – EUROCRYPT'96, volume 1070 of Lecture Notes in Computer Science, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.

[18] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, 37th Annual ACM Symposium on Theory of Computing, pages 84–93, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[19] Oded Regev. The learning with errors problem. In Proc. of 25th IEEE Annual Conference on Computational Complexity (CCC), 2010.

[20] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In 35th Annual Symposium on Foundations of Computer Science, pages 124–134, Santa Fe, New Mexico, November 20–22, 1994. IEEE Computer Society Press.

A Appendix

Table A.1: All solutions for $\beta = 29.05$

$\mathbf{v}$	$\|\mathbf{v}\|_2$	$\mathbf{v}$	$\|\mathbf{v}\|_2$
$(\ 23; -\ 2;\ \ 4;\ \ 1)^T$	≈ 23.45	$(-23;\ \ 2; -\ 4; -\ 1)^T$	≈ 23.45
$(-22;\ \ 5;\ \ 6;\ \ 1)^T$	≈ 23.37	$(\ 22; -\ 5; -\ 6; -\ 1)^T$	≈ 23.37
$(\ \ 1;\ \ 3;\ 10;\ \ 2)^T$	≈ 10.68	$(-\ 1; -\ 3; -10; -\ 2)^T$	≈ 10.68
$(\ \ 5; -\ 7; -23;\ \ 2)^T$	≈ 24.64	$(-\ 5;\ \ 7;\ 23; -\ 2)^T$	≈ 24.64
$(-\ 7; -27;\ \ 6;\ \ 3)^T$	≈ 28.69	$(\ \ 7;\ 27; -\ 6; -\ 3)^T$	≈ 28.69
$(\ 24;\ \ 1;\ 14;\ \ 3)^T$	≈ 27.96	$(-24; -\ 1; -14; -\ 3)^T$	≈ 27.96
$(-21;\ \ 8;\ 16;\ \ 3)^T$	≈ 27.75	$(\ 21; -\ 8; -16; -\ 3)^T$	≈ 27.75
$(-17; -\ 2; -17;\ \ 3)^T$	≈ 24.31	$(\ 17;\ \ 2;\ 17; -\ 3)^T$	≈ 24.31
$(\ \ 2;\ \ 6;\ 20;\ \ 4)^T$	≈ 21.35	$(-\ 2; -\ 6; -20; -\ 4)^T$	≈ 21.35
$(\ \ 6; -\ 4; -13;\ \ 4)^T$	≈ 15.39	$(-\ 6;\ \ 4;\ 13; -\ 4)^T$	≈ 15.39
$(-16;\ \ 1; -\ 7;\ \ 5)^T$	≈ 18.19	$(\ 16; -\ 1;\ \ 7; -\ 5)^T$	≈ 18.19
$(\ \ 7; -\ 1; -\ 3;\ \ 6)^T$	$\approx\ \ 9.75$	$(-\ 7;\ \ 1;\ \ 3; -\ 6)^T$	$\approx\ \ 9.75$
$(-15;\ \ 4;\ \ 3;\ \ 7)^T$	≈ 17.29	$(\ 15; -\ 4; -\ 3; -\ 7)^T$	≈ 17.29
$(\ \ 8;\ \ 2;\ \ 7;\ \ 8)^T$	≈ 13.45	$(-\ 8; -\ 2; -\ 7; -\ 8)^T$	≈ 13.45
$(-14;\ \ 7;\ 13;\ \ 9)^T$	≈ 22.25	$(\ 14; -\ 7; -13; -\ 9)^T$	≈ 22.25
$(-10; -\ 3; -20;\ \ 9)^T$	≈ 24.29	$(\ 10;\ \ 3;\ 20; -\ 9)^T$	≈ 24.29
$(\ \ 9;\ \ 5;\ 17;\ 10)^T$	≈ 22.25	$(-\ 9; -\ 5; -17; -10)^T$	≈ 22.25
$(\ 13; -\ 5; -16;\ 10)^T$	≈ 23.45	$(-13;\ \ 5;\ 16; -10)^T$	≈ 23.45
$(-\ 9;\ \ 0; -10;\ 11)^T$	≈ 17.38	$(\ \ 9;\ \ 0;\ 10; -11)^T$	≈ 17.38
$(\ 14; -\ 2; -\ 6;\ 12)^T$	≈ 19.49	$(-14;\ \ 2;\ \ 6; -12)^T$	≈ 19.49
$(-\ 8;\ \ 3;\ \ 0;\ 13)^T$	≈ 15.56	$(\ \ 8; -\ 3;\ \ 0; -13)^T$	≈ 15.56
$(\ 15;\ \ 1;\ \ 4;\ 14)^T$	≈ 20.93	$(-15; -\ 1; -\ 4; -14)^T$	≈ 20.93
$(-\ 7;\ \ 6;\ 10;\ 15)^T$	≈ 20.25	$(\ \ 7; -\ 6; -10; -15)^T$	≈ 20.25
$(-\ 3; -\ 4; -23;\ 15)^T$	≈ 27.91	$(\ \ 3;\ \ 4;\ 23; -15)^T$	≈ 27.91
$(\ 16;\ \ 4;\ 14;\ 16)^T$	≈ 26.91	$(-16; -\ 4; -14; -16)^T$	≈ 26.91
$(-\ 6;\ \ 9;\ 20;\ 17)^T$	≈ 28.39	$(\ \ 6; -\ 9; -20; -17)^T$	≈ 28.39
$(-\ 2; -\ 1; -13;\ 17)^T$	≈ 21.52	$(\ \ 2;\ \ 1;\ 13; -17)^T$	≈ 21.52
$(-\ 1;\ \ 2; -\ 3;\ 19)^T$	≈ 19.36	$(\ \ 1; -\ 2;\ \ 3; -19)^T$	≈ 19.36
$(\ \ 0;\ \ 5;\ \ 7;\ 21)^T$	≈ 22.69	$(\ \ 0; -\ 5; -\ 7; -21)^T$	≈ 22.69
$(\ \ 5; -\ 2; -16;\ 23)^T$	≈ 28.53	$(-\ 5;\ \ 2;\ 16; -23)^T$	≈ 28.53
$(\ \ 6;\ \ 1; -\ 6;\ 25)^T$	≈ 26.42	$(-\ 6; -\ 1;\ \ 6; -25)^T$	≈ 26.42
$(\ \ 7;\ \ 4;\ \ 4;\ 27)^T$	≈ 28.46	$(-\ 7; -\ 4; -\ 4; -27)^T$	≈ 28.46

Table A.2: Number of solution of the SIS problem for different value q using $n = 2$, $m = 4$

q	$\mathbf{A}$	β	number of solutions with β	number of solutions without β	percentage
211	$\begin{pmatrix} 14 & 46 & -57 & -2 \\ 58 & 80 & -30 & 1 \end{pmatrix}$	29.05	64	44520	$\approx$ 0.144%
149	$\begin{pmatrix} 16 & 32 & -54 & 10 \\ 40 & 17 & -68 & 71 \end{pmatrix}$	24.41	76	22200	$\approx$ 0.342%
113	$\begin{pmatrix} -26 & 8 & 18 & -21 \\ -53 & -21 & -41 & -42 \end{pmatrix}$	21.26	76	12768	$\approx$ 0.595%
79	$\begin{pmatrix} 1 & 33 & -25 & 38 \\ 4 & -1 & -26 & 29 \end{pmatrix}$	17.78	80	6240	$\approx$ 1.282%
31	$\begin{pmatrix} -11 & -5 & 3 & 4 \\ 9 & -15 & 12 & -14 \end{pmatrix}$	11.14	74	960	$\approx$ 7.708%
17	$\begin{pmatrix} 8 & 7 & 6 & 2 \\ 5 & -8 & 2 & -5 \end{pmatrix}$	8.25	68	288	$\approx$ 23.611%