An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation

Sedat Akleylek[1], Nina Bindel[2], Johannes Buchmann[2], Juliane Krämer[2], and Giorgia Azzurra Marson[2]

[1] Ondokuz Mayis University, Turkey
sedat.akleylek@bil.omu.edu.tr
[2] Technische Universität Darmstadt, Germany
nbindel@cdc.informatik.tu-darmstadt.de,
buchmann@cdc.informatik.tu-darmstadt.de,
jkraemer@cdc.informatik.tu-darmstadt.de, giorgia.marson@cased.de

Abstract. In view of the expected progress in cryptanalysis it is important to find alternatives for currently used signature schemes such as RSA and ECDSA. The most promising lattice-based signature schemes to replace these schemes are BLISS (CRYPTO 2013) and GLP (CHES 2012). Both come with a security reduction from a lattice problem and have high performance. However, their parameters are not chosen according to their provided security reduction, i.e., the *instantiation* is not provably secure. In this paper, we present the first lattice-based signature scheme with good performance when provably secure instantiated. To this end, we provide a tight security reduction for the new scheme from the ring learning with errors problem which allows for provably secure and efficient instantiations. We present experimental results obtained from a software implementation of our scheme. They show that our scheme, when provably secure instantiated, performs comparably with BLISS and the GLP scheme.

Keywords. lattice-based cryptography, tightness, ideal lattices, signatures, ring learning with errors

1 Introduction

Electronic signatures are essential for cybersecurity. For example, they provide authenticity proofs for billions of software downloads daily on the Internet. In recent years, lattice-based signatures such as BLISS [**?**] or the GLP [**?**] signature scheme have become an interesting alternative to the schemes that are currently being used in practice, like RSA and ECDSA. Providing such alternatives is very important in view of the expected progress in cryptanalysis of RSA and ECDSA, in particular by quantum computers.

The lattice-based signature schemes BLISS and GLP have two important properties. They have *good performance*, i.e., they can compete with

RSA and ECDSA. Also, they are *provably secure*: they allow for security reductions from lattice problems that are expected to be hard even in the presence of quantum computers.

Provable security is a very strong security argument. In this paper, we go one step further and present an R-LWE-based signature scheme which has a security property which we consider to be even stronger: *good performance with provably secure instantiation*. By this property we mean that the parameters are chosen according to the security reduction and at the same time allow for good performance. This implies the following: suppose that parameters are constructed for a certain security level. By virtue of the security reduction these parameters correspond to an instance of the ring learning with errors problem (R-LWE). Since the parameters were chosen according to the security reduction, this reduction provably guarantees that our scheme has the selected security level as long as the corresponding R-LWE instance is intractable. In other words, hardness statements for R-LWE instances have a provable consequence for the security levels of our scheme. Currently, both BLISS and GLP do not allow for good performance and provably secure instantiation at the same time. Choosing parameters according to the security reductions for these schemes reduces their performance significantly (see for example [**?**, **?**]).

We note that our scheme has another potential advantage over BLISS. BLISS uses Gaussian sampling, which is generally assumed to be vulnerable to timing attacks [**?**, **?**], while GLP and our scheme use uniform sampling during signature generation which appears to not have this vulnerability.

Our signature scheme is based on the design of Bai and Galbraith [**?**] and its optimizations by Dagdelen *et al.* [**?**]. The reason why our scheme allows for good performance with provably secure instantiation is that we are able to give a tight security reduction from the R-LWE problem to our scheme. The proof of this result is an optimized adaption of the tightness proof in [**?**] to the R-LWE setting which allows for better tightness bounds. To demonstrate that our scheme has good performance, we present experimental results which are based on a software implementation. These results show that our scheme, when provably secure instantiated, performs comparably with BLISS and the GLP scheme without provably secure instantiation.

Related Work. The first lattice-based signature scheme with tight security reduction is the GPV signature scheme [**?**]. Its instantiations are provably secure, but not efficient. Most of the recent lattice-based signature

schemes [?, ?, ?, ?, ?] come neither with a tight reduction nor with provably secure instantiation. The security of all those schemes was proven by applying the powerful Forking Lemma [?], which inherently results in a non-tight security reduction.

Abdalla *et al.* [?] circumvent the Forking Lemma and use an approach inspired by the proof idea introduced by Katz and Wang [?]. However, their tight reduction demands an impractically large choice of the modulus. Recently, Alkim *et al.* [?] also used the approach by Katz and Wang [?] to provide a tight security reduction from the learning with errors problem over standard lattices (LWE) to an improved variant of the Bai-Galbraith signature scheme [?,?]. Instantiations of their scheme are provably secure, but they yield larger key sizes and worse run times than the BLISS and GLP signature scheme.

Organization. After stating notations and definitions in Section ??, we describe the signature scheme in Section ??. In Section ??, we analyze the hardness of R-LWE and we explain the derivation of the parameter sets. Our implementation is described in Section ??. We give our experimental results and compare them with BLISS and GLP in Section ??.

2 Preliminaries

2.1 Notation

Let $k \in \mathbb{N}$. Throughout this paper we define $n = 2^k \in \mathbb{N}$. Let $q \in \mathbb{N}$ be a prime with $q = 1 \pmod{2n}$. We denote by $\mathbb{Z}_q$ the finite field $\mathbb{Z}/q\mathbb{Z}$ and identify an element in $\mathbb{Z}_q$ with its representative in $(-\lfloor q/2 \rfloor, \lceil q/2 \rceil]$, and we write $(\bmod\ q)$ to denote the unique representative in $\mathbb{Z}_q$. We define the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$ and denote the set of its units by $\mathcal{R}^\times$. Further, we define $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$, $\mathcal{R}_{q,[B]} = \{\sum_{i=0}^{n-1} a_i x^i \in \mathcal{R}_q \mid i \in [0, n-1], a_i \in [-B, B]\}$ for $B \in [0, q/2]$, and $\mathbb{B}_{n,\omega} = \left\{\mathbf{v} \in \{0,1\}^n \mid ||\mathbf{v}||^2 = \omega\right\}$ for $\omega \in [0, n]$. We denote polynomials by lower case letters (e.g., p) and (column) vectors by bold lower case letters (e.g., $\mathbf{v}$). Without further mentioning, we use the symbol $\mathbf{p}$ to denote the coefficient vector of a polynomial p. We denote matrices by bold upper case letters (e.g., $\mathbf{M}$) and the transpose of a matrix $\mathbf{M}$ by $\mathbf{M}^T$. We indicate the Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^n$ by $\|\mathbf{v}\|$. All logarithms are in base 2.

Rounding Operators. Let $d \in \mathbb{N}$ and $c \in \mathbb{Z}$. We denote by $[c]_{2^d}$ the unique representative of c modulo 2^d in the set $(-2^{d-1}, 2^{d-1}] \subset \mathbb{Z}$. Let $\lfloor \cdot \rceil_d$ be the rounding operator defined as $\lfloor \cdot \rceil_d : \mathbb{Z} \to \mathbb{Z}, c \mapsto (c - [c]_{2^d})/2^d$. We

naturally extend these definitions to vectors and polynomials by applying $\lfloor \cdot \rceil_d$ and $[\cdot]_{2^d}$ to each component of the vector and to each coefficient of the polynomial, respectively. We abbreviate $\lfloor v \pmod q \rceil_d$ by $\lfloor v \rceil_{d,q}$.

Algorithms and Distributions. If $\mathcal{A}$ is a randomized algorithm we denote by $y \leftarrow \mathcal{A}(x)$ the output of $\mathcal{A}$ on input x and randomly chosen (internal) coins. For an oracle $\mathcal{O}$ we write $\mathcal{A}^{\mathcal{O}}$ to indicate that $\mathcal{A}$ has access to that oracle. Let $\sigma \in \mathbb{R}_{>0}$. The centered discrete Gaussian distribution $\mathcal{D}_\sigma$ on $\mathbb{Z}$ with standard deviation σ is defined as follows: for every $z \in \mathbb{Z}$ the probability of z is given by $\rho_\sigma(z)/\rho_\sigma(\mathbb{Z})$, where $\rho_\sigma(z) = \exp(\frac{-z^2}{2\sigma^2})$ and $\rho_\sigma(\mathbb{Z}) = 1 + 2\sum_{z=1}^{\infty} \rho_\sigma(z)$. We denote by $d \leftarrow \mathcal{D}_\sigma$ the operation of sampling an element d with Gaussian distribution $\mathcal{D}_\sigma$. When writing $\mathbf{v} \leftarrow \mathcal{D}_\sigma^n$ we mean sampling each component of the vector $\mathbf{v}$ with Gaussian distribution. To simplify the notation we indicate sampling all coefficients of a polynomial $a \in \mathcal{R}$ with Gaussian distribution by $a \leftarrow \mathcal{D}_\sigma^n$ as well. Similarly, for a finite set S we write $s \leftarrow \mathcal{U}(S)$, or simply $s \leftarrow_\$ S$, to indicate that an element s is sampled uniformly at random from S.

Lattices and Gaussian Heuristic. Let $n \geq k > 0$. A k-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$ containing all integer linear combinations of k linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\} = \mathbf{B}$, i.e., $\Lambda = \Lambda(\mathbf{B}) = \{\ \mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^k\ \}$. The determinant of a lattice is defined by $\det(\Lambda(\mathbf{B})) = \sqrt{\det(\mathbf{B}^\top \mathbf{B})}$.
Throughout this paper we are mostly concerned with q-ary lattices. $\Lambda \in \mathbb{Z}^n$ is called a *q-ary* lattice if $q\mathbb{Z} \subset \Lambda$ for some $q \in \mathbb{Z}$. Let $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m \times n}$. We define the q-ary lattices $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} = \mathbf{0} \pmod q\}$ and $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n \mid \exists \mathbf{s} \in \mathbb{Z}^m \text{ s.t. } \mathbf{x} = \mathbf{A}^\top \mathbf{s} \pmod q\}$. Furthermore, for $\mathbf{u} \in \mathbb{Z}_q^m$ we define cosets $\Lambda_{\mathbf{u},q}^\perp(\mathbf{A}) = \{x \in \mathbb{Z}^n \mid \mathbf{Ax} = \mathbf{u} \pmod q\}$, i.e., $\Lambda_q^\perp(\mathbf{A}) = \Lambda_{\mathbf{0},q}^\perp(\mathbf{A})$. One can consider $\Lambda_{\mathbf{u},q}^\perp(\mathbf{A})$ as a *shifted lattice* by a vector $\mathbf{u}$, i.e., $\Lambda_{\mathbf{u},q}^\perp(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{y}$ where $\mathbf{y} \in \mathbb{Z}^m$ is an integer solution of $\mathbf{Ax} = \mathbf{u} \pmod q$.

Let S be a measurable set and let $\Lambda \subset \mathbb{Z}^n$ be a lattice. The *Gaussian heuristic* approximates the number of lattice points in the set S by $|S \cap \Lambda| = \frac{vol(S)}{det(\Lambda)}$.

2.2 The Learning with Errors Problem over Rings

Given the isomorphism $\Phi_q : \mathbb{Z}_n \to \mathcal{R}_q$ with $(a_0, ..., a_{n-1}) \mapsto a_0 + a_1 x + ... + a_{n-1} x^{n-1}$, $\mathcal{R}_q$ is isomorphic to $\mathbb{Z}_q^n$ as a $\mathbb{Z}$-module. Therefore, we can identify a polynomial $a = a_0 + a_1 x + ... + a_{n-1} x^{n-1} \in \mathcal{R}_q$ with its

coefficient vector $\mathbf{a} = (a_0, \ldots, a_{n-1})^T$. We define the rotation of a vector $\mathbf{a} = (a_0, \ldots, a_{n-1})^T$ to be the coefficient vector of $ax \in \mathcal{R}_q$, i.e., $\mathrm{rot}(\mathbf{a}) = (-a_{n-1}, a_0, \ldots, a_{n-2})^T$. Furthermore, we define the rotation matrix of a polynomial a as $\mathrm{Rot}(a) = (\mathbf{a}, \mathrm{rot}(\mathbf{a}), \mathrm{rot}^2(\mathbf{a}), \ldots, \mathrm{rot}^{n-1}(\mathbf{a})) \in \mathbb{Z}_q^{n \times n}$. Polynomial multiplication of $a, b \in \mathcal{R}_q$ is equivalent to the matrix-vector multiplication $\mathrm{Rot}(a)\mathbf{b}$ in $\mathbb{Z}_q$. It can be easily shown that $a \in \mathcal{R}_q$ is invertible, i.e., $a \in \mathcal{R}_q^\times$, if and only if $\mathrm{rank}(\mathrm{Rot}(a)) = n$.

We define the learning with errors distribution and the ring learning with errors problem (R-LWE) in the following.

Definition 1 (Learning with Errors Distribution). *Let $n, q > 0$ be integers, $s \in \mathcal{R}_q$, and χ be a distribution over $\mathcal{R}$. We define by $\mathcal{D}_{s,\chi}$ the R-LWE distribution which outputs $(a, \langle a, s\rangle + e) \in \mathcal{R}_q \times \mathcal{R}_q$, where $a \leftarrow_\$ \mathcal{R}_q$ and $e \leftarrow \chi$.*

Since our signature scheme is based on the decisional R-LWE problem, we omit the definition of the search version and state only the decisional learning with errors problem.

Definition 2 (Ring Learning with Errors Problem). *Let $n, q > 0$ be integers and $q = 2^k$ for some $k \in \mathbb{N}_{>0}$ and χ be a distribution over $\mathcal{R}$. Moreover, define $\mathcal{O}_\chi$ to be an oracle, which upon input polynomial $s \in \mathcal{R}_q$ returns samples from the learning with errors distribution $\mathcal{D}_{s,\chi}$. The ring learning with errors problem R-LWE$_{n,m,q,\chi}$ is (t, ε)-hard if for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, running in time t and making at most m queries to its oracle, it holds that*

$$Adv^{\text{R-LWE}}_{n,q,\chi}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{O}_\chi(s)(\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)(\cdot)} = 1\right] \right| \leq \varepsilon \,,$$

where the probabilities are taken over the random choices of $s \leftarrow \mathcal{U}(\mathcal{R}_q)$, the random choice of the distribution $\mathcal{D}_{s,\chi}$, as well as the random coins of $\mathcal{A}$.

The R-LWE assumption comes with a worst-case to average-case reduction to problems over ideal lattices [**?**]. Furthermore, it was shown in [**?**] that the learning with errors problem remains hard if one chooses the secret distribution to be the same as the error distribution. We write R-LWE$_{n,m,q,\sigma}$ if χ is the discrete Gaussian distribution with standard deviation σ.

3 Description and Security of the Signature Scheme

In this section, we present our signature scheme and we prove it to be unforgeable against a chosen-message attack—shortly ufcma-secure (cf. Ap-

pendix **??**, Figure **??**)—as long as R-LWE is computationally hard. We recall basic definitions and notations about signatures schemes in Appendix **??**. We name our scheme ring-TESLA since it is based on the signature scheme TESLA by Alkim *et al.* [**?**].

Our signature scheme is parametrized by the integers $n \in \mathbb{N}_{>0}$, ω, d, B, q, U, L, κ, and the security parameter λ with $n > \kappa > \lambda$, by the Gaussian distribution $\mathcal{D}_\sigma$ with standard deviation σ, by the hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$, and by the encoding function $F : \{0,1\}^\kappa \to \mathbb{B}_{n,\omega}$. The encoding function F takes the (binary) output of the hash function H and maps it to a vector of length n and weight ω. For more information about the encoding function see [**?**]. Furthermore, let $a_1, a_2 \in \mathcal{R}_q^\times$ be two uniformly sampled polynomials which are publicly known as global constants. They can be shared among arbitrary many signers.

$\mathsf{KeyGen}(1^\lambda; a_1, a_2)$:

1 $s, e_1, e_2 \leftarrow D_\sigma^n$
2 If $\mathsf{checkE}(e_1) = 0 \vee \mathsf{checkE}(e_2) = 0$
3 Restart
4 $t_1 \leftarrow a_1 s + e_1 \pmod q$
5 $t_2 \leftarrow a_2 s + e_2 \pmod q$
6 $\mathsf{sk} \leftarrow (s, e_1, e_2)$
7 $\mathsf{pk} \leftarrow (t_1, t_2)$
8 Return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mu; a_1, a_2, s, e_1, e_2)$:

9 $y \leftarrow_\$ \mathcal{R}_{q,[B]}$
10 $v_1 \leftarrow a_1 y \pmod q$
11 $v_2 \leftarrow a_2 y \pmod q$
12 $c' \leftarrow H\left(\lfloor v_1 \rceil_{d,q}, \lfloor v_2 \rceil_{d,q}, \mu\right)$
13 $c \leftarrow F(c')$
14 $z \leftarrow y + sc$
15 $w_1 \leftarrow v_1 - e_1 c \pmod q$
16 $w_2 \leftarrow v_2 - e_2 c \pmod q$
17 If $|[w_1]_{2^d}|, |[w_2]_{2^d}| \notin \mathcal{R}_{2^d - L} \vee z \notin \mathcal{R}_{B-U}$:
18 Restart
19 Return (z, c')

$\mathsf{Verify}(\mu; z, c'; a_1, a_2, t_1, t_2)$

19 $c \leftarrow F(c')$
20 $w_1' \leftarrow a_1 z - t_1 c \pmod q$
21 $w_2' \leftarrow a_2 z - t_2 c \pmod q$
22 $c'' \leftarrow H\left(\lfloor w_1' \rceil_{d,q}, \lfloor w_2' \rceil_{d,q}, \mu\right)$
23 If $c' = c'' \wedge\ z \in \mathcal{R}_{B-U}$:
24 Return 1
25 Else: Return 0

Fig. 1: Specification of the signature scheme ring-TESLA

The secret key sk consists of three small polynomials s, e_1, and e_2; the public key pk is given by two polynomials $t_1 = a_1 s + e_1$ and $t_2 = a_2 s + e_2$.

To ensure that signatures are short and verified correctly, we use a procedure checkE similar to the one introduced by Dagdelen *et al.* [**?**]. Let $max_k(\cdot)$ be a function that takes as input a vector and returns its k-th largest entry. The key polynomials e_1, e_2 are rejected during checkE if $\sum_{k=1}^{\omega} max_k(\mathbf{e_i})$ is greater then L for at least one of e_1 or e_2. Otherwise e_1, e_2 are accepted. To sign a message μ, first a random polynomial $y \in \mathcal{R}_{q,[B]}$ is chosen. Afterwards, the most significant bits of $a_1 y$ and $a_2 y$ and the message are hashed to a value c. The signature of μ consists of the hash value c and the polynomial $z = sc + y$. To hide the secret, rejection sampling is applied. For verification of the signature (c, z), the size of z and the equality of c and $H(\lfloor a_1 z - t_1 c \rceil_d, \lfloor a_2 z - t_2 c \rceil_d, \mu)$ is checked. The signature scheme ring-TESLA is depicted in detail in Figure **??**. We present parameter sets in Table **??** and their derivation in Section **??**.

In our security reduction we follow an idea introduced by Katz and Wang [**?**] that can be summarized at follows: assume there exists an algorithm $\mathcal{A}$ that forges a signature given a valid public key, i.e., an LWE tuple. In contrast, given a random key $\mathcal{A}$ forges a signature only with very small probability. Hence, the security reduction distinguishes whether its own challenge tuple is an LWE tuple or not by the different behavior of the algorithm $\mathcal{A}$.

Theorem 1. *Let n, ω, d, B, q, U, L, and σ be arbitrary parameters satisfying the constraints described in Section* **??**. *Assume that the Gaussian heuristic holds for lattice instances defined by the parameters above. For every* ufcma*-adversary $\mathcal{A}$ that runs in time $t_{\mathcal{A}}$, asks at most q_s and q_h queries to the signing oracle and the hash oracle, respectively, and forges a valid signature of the signature scheme ring-TESLA with probability $\varepsilon_{\mathcal{A}}$, there exists a distinguisher $\mathcal{D}$ that runs in time $t_{\mathcal{D}} = t_{\mathcal{A}} + \mathcal{O}(q_s \kappa^2 + q_h)$ and breaks the R-LWE$_{n,2,q,\sigma}$ problem (in the random oracle model) with success probability*

$$\varepsilon_{\mathcal{D}} \geq \varepsilon_{\mathcal{A}} \left(1 - \frac{q_s q_h 2^{(d+1)2n}}{(2B+1)^n q^n}\right) - \frac{q_h 2^{dn}(2B - 2U + 1)^n + (28\sigma + 1)^{3n}}{q^{2n}} .$$

Proof sketch. We show how to turn any successful forger $\mathcal{A}$ against the signature scheme ring-TESLA into a distinguisher $\mathcal{D}$ for the R-LWE problem. The distinguisher obtains two R-LWE samples from its sampling oracle $\mathcal{O}_\chi(s)$ (cf. Definition **??**) and embeds them into a public key pk. Thus, $\mathcal{D}$ simulates the ufcma game (cf. Figure **??**, Appendix **??**). When $\mathcal{A}$ returns a forgery (μ, σ), $\mathcal{D}$ checks whether σ is a valid signature for message μ under key pk: if so, it outputs 1 as a guess that $\mathcal{O}_\chi(s)$ presented

two R-LWE tuples, otherwise it outputs 0. To derive the explicit relation between $\mathcal{D}$ and $\mathcal{A}$'s success probabilities $\varepsilon_{\mathcal{D}}$ and $\varepsilon_{\mathcal{A}}$ as indicated in the theorem statement, we show that (i) $\mathcal{D}$ provides a good simulation of the ufcma game for $\mathcal{A}$. In particular, we show that the simulated signatures look like genuine ones. And we prove, (ii) $\mathcal{D}$'s simulation does not abort too often. Formal proofs of both facts, (i) and (ii), require several technical lemmas that we state and prove in the full version of this paper [**?**]. For proving fact (i), we observe that $\mathcal{D}$ simulates signatures $\sigma = (z, c')$ by choosing z and c' uniformly at random from appropriate spaces. By applying rejection sampling and the fact that c' is the output of a random oracle, we show that simulated signatures are statistically indistinguishable from genuine ones. Concerning fact (ii), we first note that $\mathcal{D}$'s signing simulation needs to program the random oracle H, which may lead to inconsistencies in case one of $\mathcal{A}$'s signature requests results in programming a hash value $H(x)$ for which x was already queried. Such an occurrence causes a premature termination of the simulation. In [**?**], we prove that the latter happens only with small probability. □

As described in [**?**, Section 3.3], the probability that a polynomial chosen uniformly random in $\mathcal{R}_q$ is in the subset of multiplicative invertible elements of $\mathcal{R}_q$ is given by $\Pr\left[a \in \mathcal{R}_q^{\times}\right] = (1 - 1/q)^n$, where the probability is taken over random choices of $a \leftarrow_{\$} \mathcal{R}_q$. This probability is overwhelming for our choices of q and n in the signature scheme presented in this paper. Thus, it is justified to sample the polynomials a_1 and a_2 uniformly random in $\mathcal{R}_q^{\times}$ instead of $\mathcal{R}_q$ as defined in the R-LWE problem.

Relation to Former Security Reductions. The scheme ring-TESLA is based on the signature scheme by Bai and Galbraith [**?**] with a tight security reduction by Alkim *et al.* [**?**]. Essentially, we convert the scheme by Bai and Galbraith to a scheme over ideal lattices. Our security reduction follows the proof strategy of [**?**]. We emphasize that lifting the security statements for the original (lattice-based) scheme to our (ideal lattice-based) scheme is not trivial. For example, it is unclear whether distributions remain the same when lemmata are applied on rotation matrices instead of matrices chosen uniformly random; in some cases even improvements can be made. Indeed, we could sharpen the bound given in [**?**, Lemma 2]. Our corresponding result is stated in the full version of this paper [**?**]. Moreover, we formulate and prove a similar lemma to [**?**, Lemma 3] for ideal lattices and we state explicitly which property related to the Gaussian heuristic is necessary to prove the statement. Likewise, Bai and Galbraith make use of

the Gaussian heuristic in their corresponding proof. The methods used in our security reduction resemble those formalized by Abdalla *et al.* [**?**]. Abdalla *et al.* define four properties of identification schemes for which they give a black-box-transformation to signature schemes with tight security reduction. Applying their black-box-transformation to a lattice-based signature scheme led to inefficiently large parameters as stated by the authors [**?**]. Hence, we prove unforgeability of ring-TESLA more directly—without passing through an intermediate identification scheme—by following the proof technique introduced by Katz and Wang [**?**]. This yields practical instantiation as we show in Section **??**.

4 Selecting Parameters

The reductionist approach to prove security of a given cryptosystem essentially consists in building an efficient reduction that turns any successful adversary against the cryptosystem into one that solves some computationally hard problem. The hardness of breaking the cryptosystem and of solving the underlying problem are often expressed asymptotically. When a scheme is to be deployed in the real world, however, for a security analysis to be realistic it is essential that run times and success probabilities are estimated in a more explicit way. Moreover, given a (concrete and) tight security reduction, the security of the scheme is about the same as the hardness of the underlying computational assumption when the scheme is instantiated according to the reduction. In contrast, if only a non-tight reduction is available, larger security parameters shall be used in order to achieve a specific level of security. As a consequence, it is often hard to tell whether a provably secure scheme with a non-tight reduction effectively provides the claimed level of security and performance.

In this section, we propose our choice of provably secure parameters for different levels of bit-security for the signature scheme presented in this paper and we explain how we estimate the hardness of the ring learning with errors problem.

4.1 Derivation of Parameters for Different Security Levels

The security reduction given in Section **??** provides a *tight* reduction to the hardness of R-LWE and bounds *explicitly* the forging probability with the success probability of the reduction. More formally, let $\varepsilon_{\mathcal{A}}$ and $t_{\mathcal{A}}$ denote the success probability and the runtime of a forger $\mathcal{A}$ against our signature scheme and let $\varepsilon_{\mathcal{D}}$ and $t_{\mathcal{D}}$ denote analogous quantities for the

reduction $\mathcal{D}$ presented in the proof of Theorem **??**. We can write the explicit relations $\varepsilon_{\mathcal{D}} \geq c_1\varepsilon_{\mathcal{A}} + c_2$ and $t_{\mathcal{D}} \leq c_3 t_{\mathcal{A}} + c_4$, where c_1, c_2, c_3, c_4 are constants which are fixed for a concrete instantiation of the signature scheme. We say that R-LWE is *n-bit hard* if $t_{\mathcal{D}}/\varepsilon_{\mathcal{D}} \geq 2^n$; similarly, we say that the signature scheme is *m-bit secure* if $t_{\mathcal{A}}/\varepsilon_{\mathcal{A}} \geq 2^m$.

Given an explicit security reduction and the assumed bit-hardness of R-LWE, we can compute the bit-security of the signature scheme. In our case, we instantiate the signature scheme such that the constants c_1, c_2, and c_3 are less than $2^{-\lambda}$. Thus, the bit-hardness of the R-LWE instance is the same as the bit-security of our signature instantiated as described below. To ensure both correctness and security of our signature, the following dependencies must hold.

Let λ be the security parameter. We choose a hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$ with $\kappa > \lambda$ to ensure that the hash function gives at least a bit-hardness of λ. We instantiate the hash function for our parameter sets with SHA-256. Furthermore, security relies on the encoding function $F : \{0,1\}^\kappa \to \mathbb{B}_{n,\omega}$. Following Bai and Galbraith [**?**], we require F to be close to an injective function. That means that the probability of mapping two different values to the same output is smaller than or equal to $2^{-\lambda}$. We choose ω such that $2^\kappa \geq |\mathbb{B}_{n,\omega}| = 2^\omega \binom{n}{\omega}$. To use efficient polynomial multiplication, i.e., the number theoretic transform (NTT) in the ring $\mathcal{R}_q$, we restrict ourselves to a polynomial degree of a power of 2, i.e, $n = 2^k$ for $k \in \mathbb{N}$. Choosing the Gaussian parameter σ, we can compute the system parameters to give a concrete instantiation of ring-TESLA with λ-bit security.

To apply rejection sampling we choose $U = 14\sqrt{\omega}\sigma$ and $B \geq 14(n-1)\sqrt{\omega}\sigma$. The rejection probability is given by $M = \left(\frac{2(B-U)+1}{2B+1}\right)^n$. We select the rounding value d to be larger than $\log(B)$ and such that the acceptance probability in the first part of Step 17 in Figure **??** is greater than or equal to 0.4, i.e., $(1 - 2L/2^d)^m \geq 0.4$. The bound L is important during the key generation as well as during the sign procedure. We choose L such that we reject only very few of the possible key pairs in checkE. For example, we achieve an acceptance probability of almost 100% in KeyGen and an acceptance probability of 0.34 in Sign for parameter ring-TESLA-II. At last, the modulus q has to be greater than or equal to $\left(\frac{2^{(d+1)2n+\kappa}}{(2B)^n}\right)^{1/n}$ and greater than or equal to $4B$. The theoretical size of the secret key is given by $3n\lceil\log(14\sigma)\rceil$ bits. The public key is represented by $2n\lceil\log(q)\rceil$ bits and the length of the signature is $n\lceil\log(2B-2U)\rceil + \kappa$ bits. Given the concrete instantiations in Table **??**, we get a signature size of 1,488 byte, a

public key size of 3,328 byte, and a secret key size of 1,920 byte for parameters chosen such that the signature scheme is 128-bit secure. In Table **??** we also propose instantiations for 80 bit of security. For comparison, we depict our signature and key sizes together with the corresponding values of BLISS [**?**] and the GLP [**?**] signature scheme in Table **??**.

Table 1: Parameter sets for our signature scheme in comparison; the hardness of the LWE instance is defined by the dimension n, the modulus q, and the Gaussian parameter σ; derivation of L, ω, B, U, d is explained in Section **??**; pk and sk denote the public and private key, resp.

Parameter selection									
Parameter Set	Security (bit)	n	σ	L	ω	B	U	d	q
ring-TESLA-I	80	512	30	814	11	$2^{21}-1$	993	21	8399873
ring-TESLA-II	128	512	52	2766	19	$2^{22}-1$	3173	23	39960577

		Acceptance prob.		pk Size	sk Size	Signature Size
		KeyGen	Sign	(byte)	(byte)	(byte)
ring-TESLA-I	80	0.5	0.23	3,072	1,728	1,418
ring-TESLA-II	128	0.99	0.34	3,328	1,920	1,488

4.2 Hardness Estimation of the R-LWE Problem

Since the introduction of the learning with errors problem over rings [**?**], it is an open question whether the R-LWE is as hard as the LWE problem. Recently, the cyclic structure of ideal lattices has been exploited by Garg *et al.* [**?**], by Campbell *et al.* [**?**], by Cramer *et al.* [**?**], and by Elias *et al.* [**?**]. However, up to now, these novel results are not known to be directly applicable to most of the proposed ideal-lattice-based signature schemes. Hence, as the R-LWE problem can be seen as an instantiation of the LWE problem, we estimate the hardness of R-LWE via state-of-the-art attacks against LWE. We explain four basic attacks on LWE: the embedding approach, the decoding attack, the algorithm by Blum, Kalai, and Wassermann [**?**], and the Arora-Ge-Algorithm [**?**]. We briefly describe the algorithms next. The most efficient practical approaches to solve LWE are the embedding approach and the decoding attack.

During the *decoding attack*, an LWE instance $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ is seen as an instance of the bounded distance decoding problem (BDD). The idea of the attack is to reduce the lattice by algorithms such as the BKZ

algorithm [?] first, and to find the closest lattice vector to a target vector via the nearest plane algorithm by Babai [?] (or improved variants such as by Linder and Peikert [?] or Liu and Nguyen [?]) afterwards. The closest vector corresponds to $\mathbf{As}$ of the LWE instance, such that the secret can be easily discovered.

The *embedding approach* is to solve an LWE instance by reducing it to an instance of the (unique) shortest vector problem. There are different ways to define a lattice that contains the error term of an LWE instance (e.g., [?,?,?]). In the end, the short error term is found as a shortest vector of the constructed lattice via basis reductions such as BKZ [?] and LLL [?, ?], or directly via sieving algorithms [?, ?] or enumeration [?]. Recent results [?, ?, ?] exploit the cyclic structure of ideal lattices to improve sieving algorithms. However, the improved sieving algorithms are still slower than the enumeration approach on instances currently used for signatures.

Further, there are two non-lattice approaches to solve LWE, namely the attack based on the algorithm by *Blum, Kalai*, and *Wassermann* (BKW) [?] and the algorithm by *Arora and Ge* [?]. Both algorithms require a (very) large number of LWE samples to be applied efficiently. Although the number of required samples was crucially reduced, for both BKW [?, ?, ?] and the Arora-Ge algorithm [?], our proposed instances give far less LWE samples than required for the attacks. Hence, we only take the decoding attack and the embedding approach into account when estimating the bit-security of our instances.

We estimate the hardness of our chosen LWE instances based on [?, ?, ?]. We propose parameters for two different levels of security: 80-bit security (ring-TESLA-I) and 128-bit security (ring-TESLA-II). The embedding attack yields 166 bit of security and the result of the decoding attack is a bit security of 139 on the instances in ring-TESLA-II.

5 Software Implementation

The implementation of the proposed scheme targets the Intel Haswell micro architecture. We perform benchmarks on a machine with an Intel Core i7-5820K (Haswell) CPU at 3300MHz and 16GB of RAM. In our software we use the benefits of AVX2 instructions, where multiplication, addition, and subtraction instructions have one cycle throughput for eight doubles. The software is compiled with gcc-4.7 with optimization code. The experimental results are obtained by using gcc-4.7 with "-Ofast" optimization since it enables all "-O3" optimizations together with turning

on "-ffast-math". This optimization helps us to reduce the timing results significantly. The performance of our implementation mainly depends on the number of rejections during Sign and KeyGen and on the time a single polynomial multiplication takes. The derivation of the number of rejections is explained in Section **??**. We optimized the time for multiplication by choosing the most suitable multiplication algorithm for different cases as it is explained below.

Polynomial Multiplication. In the presented scheme two types of polynomial multiplication occur: standard and sparse polynomial multiplication. For standard polynomial multiplication we use the number theoretic transform (NTT) since NTT performs polynomial multiplication with a quasilinear complexity, i.e., $O(n \log n)$. Thus, the parameter sets are selected in such a way that NTT is applicable, i.e., $q = 1 \pmod{2n}$, where n is a power of 2. In our implementation, we store the integer in double format in a word. Then, after arithmetic operations in NTT, it is expected to fit in a double, i.e., $\log(\log(n)q) + \log(q) < 54$. To avoid an overflow one needs to make extra reduction operations when using ring-TESLA-II because $\log(q)$ is represented by 26 bits. This, of course, results in a drawback of the performance. NTT with extra modulo q reduction would need almost 28383 cycles for $n = 512$ and $\omega = 19$ as chosen in ring-TESLA-II. Without extra reductions, the average cycle count of NTT developed for ring-TESLA-I is 10625. Barrett reduction is preferred over reducing the coefficients because of the modular structure. The hybrid approach of using NTT and sparse polynomial multiplication requires more inverse NTT operations since sparse polynomial multiplication is applicable only in the integer domain.

Input: array $d = [i_1, ..., i_\omega]$, poly $a(x) = \sum_{i=0}^{n-1} a_i x^i$, poly $b(x) = \sum_{i=0}^{n-1} b_i x^i$; with $a_i \in \mathbb{Z}_q$, $b_i \in \{0, 1\}$, $d[k] = i_k$ such that $b_{i_k} = 1$
Output: poly $c(x) = a(x)b(x)$

1 Set all coefficients of $c(x)$ to 0
2 for $i = 0, ..., \omega - 1$:
3 for $j = 0, ..., n - 1$:
4 $c_{j+d[i]} \leftarrow c_{j+d[i]} + a_j$
5 for $i = 0, ..., n - 1$
6 $c_i \leftarrow c_i - c_{i+n} \pmod{p}$
7 Return $c(x)$

Fig. 2: Sparse Polynomial Multiplication

Recall that the weight of c, i.e., the number of 1's, is ω. Then, the multiplication operations in the signature generation phase (Step 14, 15, and 16: sc, e_1c, and e_2c) and in the signature verification phase (Step 20 and 21: t_1c and t_2c) can be considered as sparse polynomial multiplications because of the number of nonzero elements in c. In order to speed up, we use the sparse polynomial multiplication given in Algorithm **??**. The complexity of Algorithm **??** depends on the nonzero coefficients of $b(x)$. Note that polynomial multiplication is performed by using only additions if one of the multiplicands is sparse. The required number of additions and subtractions is $(\omega n + n)$. The last for-loop is designed for polynomial reduction modulo $x^n + 1$. There is only one reduction modulo q of the coefficients. This improves the runtime and complexity. Sparse multiplication requires almost 3650 cycles.

We place our implementation of ring-TESLA in public domain. It can be found under https://www.cdc.informatik.tu-darmstadt.de/cdc/personen/nina-bindel.

6 Performance Analysis

We performed our benchmarks on a machine with an Intel Core i7-5820K (Haswell) CPU at 3300MHz and 16 GB of RAM, while disabling Turbo Boost and hyper threading. In our measurement we considered two parameter sets: ring-TESLA-I and ring-TESLA-II with 80 and 128 bits of security, respectively. Our benchmarks are averaged[3] over 10,000 runs of Sign and Verify. We summarize benchmarks for our proposed parameter sets and state-of-the-art ideal-lattice-based signature schemes in Table **??**. We emphasize once more that our parameter sets are the only ones in Table **??** which are chosen according to the given security reduction, cf. Section **??**. Nevertheless, we achieve good performance with respect to time and space. In the following, we compare sizes and run times for 80 and 128 bits of security.

For *low security of 80-bit*, key and signature sizes of GLP-I are smaller than those of our proposed parameters. Our run time of Sign is a factor of 1.19 faster than GLP. As Table **??** indicates, the software implementations of ring-TESLA and of the GLP signature scheme are optimized for micro architectures. For *medium security of 128-bit* the instantiation of our scheme gives smallest key sizes. Signature sizes are comparably

[3] Sometimes benchmarks are given as the median instead of the average value. Due to the rejection sampling, taking the median value of our experiments would be overly optimistic for Sign.

Table 2: Comparison of our results with the software implementations of the signature schemes BLISS [**?**, **?**] and GLP [**?**, **?**, **?**]. To indicate the considered platforms Intel Core i5-3210M (Ivy Bridge), Intel Core i7-5820K (Haswell), and Intel Core 3.4 GHz we use shortcuts A, B, and C, respectively. Sizes of signatures, signing and verification keys are indicated in Bytes. We abbreviate 'Decisional Compact Knapsack problem' by DCK. In the benchmarks of GLP we include the improvements by Dagdelen et al. presented in [**?**]. In the benchmarks of BLISS we include the improvements by Ducas presented in [**?**].

80-bit security	**GLP [?, ?, ?]**	**ring-TESLA-I (this paper)**
Assumption	DCK	R-LWE
CPU	A	B
Signing key size	256	1,728
Verification key size	1,536	3,072
Signature size	1,186	1,568
Sign cycle counts	452,223	370,880
Verify cycle counts	34,004	94,124
128-bit security	**BLISS [?, ?]**	**ring-TESLA-II (this paper)**
	BLISS-I **BLISS-II**	
Assumption	R-SIS, NTRU	R-LWE
CPU	C	B
Signing key size	2,048 2,048	1,920
Verification key size	7,168 7,168	3,328
Signature size	1,559 1,514	1,568
Sign cycle counts	351,333 582,857	510,981
Verify cycle counts	102,000 102,000	167,791

good. We emphasize that we report the signature size used in the publicly available software implementation of BLISS-I and BLISS-II [4]. Those sizes differ from the theoretical signature sizes presented in [**?**], which are 700 and 625 bytes for BLISS-I and BLISS-II, respectively, because signatures are not compressed in the BLISS software. To our knowledge, there is no implementation of BLISS available that compresses the signature sizes. The signature size of ring-TESLA are also obtained from our imple-

[4] bliss.di.ens.fr

mentation.
The time-optimized implementation of BLISS-I by Ducas [**?**] is only a factor of 1.45 faster than our implementation. We note that our signature scheme uses uniform sampling during Sign. In contrast, BLISS uses Gaussian sampling, which might be vulnerable to timing attacks [**?**, **?**]. Up to now, available implementations of BLISS do not protect against timing-attacks. It would be very interesting to compare our implementation with an optimized and timing-attack-protected implementation of BLISS.

In summary, our signature scheme has good performance compared to state-of-the-art ideal-lattice-based signature schemes, while it is instantiated provably secure. Hence, when real world security matters our presented scheme is a very interesting choice.

Acknowledgment

This work has been cofunded by the DFG as part of project P1 and P2 within the CRC 1119 CROSSING.

A Extended Definitions and Security Notions

A.1 Syntax, Functionality, and Security of Signature Schemes

A signature scheme with key space $\mathcal{K}$, message space $\mathcal{M}$, and signature space $\mathcal{S}$, is a tuple $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of algorithms defined as follows.

- The (probabilistic) key generation algorithm on input the security parameter 1^λ returns a key pair $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$. We write $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and call sk the secret or signing key and pk the public or verification key.
- The (probabilistic) signing algorithm takes as input a signing key sk, a message $\mu \in \mathcal{M}$, and outputs a signature $\sigma \in \mathcal{S}$. We write $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$.
- The verification algorithm, on input a verification key pk, a message $\mu \in \mathcal{M}$, and a signature $\sigma \in \mathcal{S}$, returns a bit b: if $b = 1$ we say that the algorithm accepts, otherwise we say that it rejects. We write $b \leftarrow \mathsf{Verify}(\mathsf{pk}, \mu, \sigma)$.

We require (perfect) correctness of the signature scheme: for every security parameter λ, every choice of the randomness of the probabilistic

algorithms, every key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, every message $\mu \in \mathcal{M}$, and every signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$, $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma) = 1$ holds.

We target the standard security requirement for signature schemes, namely *unforgeability under chosen-message attack* (ufcma). The corresponding experiment involving an adversary $\mathcal{A}$ against a signature scheme Σ is depicted in Figure **??**. Since we prove security of the scheme presented in Section **??** in the random oracle model, we reproduce a corresponding ufcma experiment which grants $\mathcal{A}$ access to a random oracle H. Given the experiment, we say that a signature scheme Σ is (t, q_s, q_h, ϵ)-*unforgeable under chosen-message attack* if every adversary $\mathcal{A}$ which runs in time t and poses at most q_s queries to the signing oracle and q_h queries to the random oracle has advantage

$$\mathrm{Adv}^{\mathsf{ufcma}}_{\Sigma}(\mathcal{A}) = \Pr\left[\mathrm{Expt}^{\mathsf{ufcma}}_{\Sigma,\mathcal{A}} = 1\right] \leq \epsilon \ .$$

$\mathrm{Expt}^{\mathsf{ufcma}}_{\Sigma,\mathcal{A}}(1^\lambda)$:

1 $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2 $(\mu^*, \sigma^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})^{\mathcal{O}_{\mathsf{Sign}}(\cdot), H(\cdot)}$
3 If $\mathsf{Verify}(\mathsf{pk}, \mu^*, \sigma^*) = 1 \wedge \mu^* \notin \mathcal{Q}_S$:
4 Return 1
5 Else: Return 0

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mu)$:

6 $\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{\mu\}$
7 $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$
8 Return σ to $\mathcal{A}$

Fig. 3: Security experiment of unforgeability under chosen-message attack for an adversary $\mathcal{A}$ against a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ in the random oracle model (i.e., all parties including $\mathcal{A}$ have access to a public function H with uniformly distributed output).