TESLA: Tightly-Secure Efficient Signatures from Standard Lattices .

Erdem Alkim · Nina Bindel · Johannes Buchmann · Özgür Dagdelen · Peter Schwabe

Date: 2016-10-05

Abstract Generally, lattice-based cryptographic primitives offer good performance and allow for strong security reductions. However, the most efficient current lattice-based signature schemes sacrifice (part of their) security to achieve good performance: first, security is not based on the worst-case hardness of lattice problems. Secondly, the security reductions of the most efficient schemes are *non-tight*; hence, their choices of parameters offer security merely heuristically. Moreover, lattice-based signature schemes are instantiated for classical adversaries, although they are based on presumably quantum-hard problems. Yet, it is not known how such schemes perform in a *post-quantum world*.

We bridge this gap by proving the lattice-based signature scheme TESLA to be tightly secure based on the learning with errors problem over lattices in the random-oracle model. As such, we improve the security of the original proposal by Bai and Galbraith (CT-RSA'14) twofold: we tighten the security reduction and we minimize the underlying security assumptions. Remarkably, by enhancing the security we can greatly improve TESLA's performance. Furthermore, we are first to propose parameters providing a security of 128 bits against *both* classical and quantum adversaries, for a lattice-based signature scheme. Our implementation of TESLA competes well with state-of-the-art lattice-based signatures and SPHINCS (EUROCRYPT'15), the only signature scheme instantiated with quantum-hard parameters so far.

Keywords: signature scheme, lattice cryptography, tight security, efficiency, quantum security

Mathematics Subject Classification: 11T71, 94A60

This work has been supported by the the German Research Foundation (DFG) as part of project P1 within the CRC 1119 CROSSING, by TÜBITAK under 2214-A Doctoral Research Program Grant and 2211-C PhD Scholarship, by Ege University under project 2014-FEN-065, by the European Commission through the ICT program under contract ICT-645622 (PQCRYPTO), and by the Netherlands Organisation for Scientific Research (NWO) through Veni 2013 project 13114. Permanent ID of this document: `4f2d342337db6ce6cc0c7b44159b4598`

Erdem Alkim
Department of Mathemathics, Ege University, Bornova - İzmir, Turkey,
E-mail: erdemalkim@gmail.com

Nina Bindel and Johannes Buchmann
Department of Computer Science, Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany,
E-mail: {nbindel,buchmann}@cdc.informatik.tu-darmstadt.de

Özgür Dagdelen
BridgingIT GmbH, Solmsstraße 4, 60486 Frankfurt/Main, Germany,
E-mail: oezdagdelen@googlemail.com

Peter Schwabe
Digital Security Group, Radboud University, PO Box 9010, 6500 GL Nijmegen, The Netherlands,
E-mail: peter@cryptojedi.org

1 Introduction

Since Shor presented a polynomial-time quantum algorithm for factoring integers and solving the discrete-logarithm problem [70], it is clear that all public-key cryptography that is in wide use today will fall, once an efficient quantum computer can be built. Post-quantum public-key cryptography addresses this by offering alternatives that—as far as we know—will not be broken by a quantum computer in polynomial time.

Unfortunately, the current state-of-the-art is far from offering "drop-in replacements" for schemes such as RSA, DSA, or elliptic-curve-based schemes. One reason is that many schemes suffer from practical issues such as large key sizes, long computation times, large ciphertext expansion, or large signature sizes.

In the realm of lattice-based signature schemes, recent approaches to overcome this efficiency problems define schemes over ideal-lattices. To gain further speed-ups, security is not based on worst-to-average-case hardness of (ideal) lattice problems. For instance, the signature scheme BLISS [38, 39] generates its secret keys in a similar way as NTRU [47] and the security of the signature scheme by Güneysu et al. [44, 45] is based on the learning with errors problem with ternary secret and error. Following Peikert [67], to renounce worst-case hard instantiations means to sacrifice important security guarantees for lattice-based cryptography. Additionally, most of the performance-optimized lattice-based signature schemes are not instantiated according to their given security reduction. The reason is that the relation of the hardness of the underlying assumption and the security of the signature schemes are not (yet) proven to be linear, i.e., the security reductions are not tight. In order to provably provide a certain target security level, signature schemes with a non-tight security reduction must be instantiated with much larger parameters to compensate for the loose reduction. The importance of tight security reductions is extensively discussed, for instance, by Bellare and Rogaway [17] and Chatterjee, Menezes, and Sarkar [31].

Another problem with most "post-quantum" proposals is that the security analysis of instantiations with concrete parameters is based on attacks that exclude attacks by quantum computers. In fact, only very few papers explicitly propose and analyze parameters that offer security against attacks by quantum computers [8, 20]. We discuss and compare security and performance properties of lattice-based and other post-quantum signature schemes from the literature in Sec. 6.

Our contributions. We address the above issues by presenting a lattice-based signature scheme, which we call TESLA: "**T**ightly-secure, **E**fficient signature scheme from **S**tandard **LA**ttices". We note that the novelty of TESLA does not rely on its construction. In fact, the design of TESLA is essentially the signature scheme by Bai and Galbraith [13], who base the security of their scheme on *both* the learning with errors (LWE) and the small integer solution (SIS) problem with a *loose* security reduction. However, our contributions introduce significant theoretical and practical improvements to TESLA (and beyond). Our contributions are summarized as follows:

- We give a *tight* security reduction from the LWE problem on *standard*-lattices in the random-oracle model. Our novel security reduction is inspired by a method of Katz and Wang [48][1]. The previous reduction by Bai and Galbraith [13] is not tight and relies on the hardness of LWE *and* SIS.
- We give an instantiation of TESLA according to our security reduction, called TESLA-416, that offers 128 bits of security against all known attacks, *excluding* attacks by a quantum computer. Our implementation of TESLA-416 is significantly faster than the "high-speed" implementation of the same scheme given in [75] and even key and signature sizes are smaller than in [75].
- We give an instance of TESLA, called TESLA-768, that offers 128 bits of security against all known attacks, *including* attacks by a quantum adversary. After the stateless hash-based signature scheme SPHINCS [20], proposed at Eurocrypt 2015, TESLA-768 is the second signature scheme (and the first lattice-based signature scheme) to propose and analyze parameters for this level of security.
- We apply a standard conversion to the Bai-Galbraith scheme to turn TESLA into a deterministic signature scheme. Thus, TESLA does not rely on the security of an external random-number generator to replace oracles during the signing algorithm.
- We present software implementations of TESLA-416 and TESLA-768 targeting Intel Haswell CPUs. This software has the same level of timing-attack protection as the software presented in [75] and outperforms all previous standard-lattice-based signature schemes at comparable security levels. The software performance

[1] This technique was also used by Abdalla, Fouque, Lyubashevsky, and Tibouchi [1], leading to the only other Fiat-Shamir lattice-based signature with tight reduction. Unfortunately, this imposes further conditions on their parameters yielding a less efficient signature scheme than the original scheme by Lyubashevsky [57].

of TESLA relies on the careful choice of parameters, which is enabled by the tight security reduction from LWE and a novel implementation technique that reduces penalties from insufficient cache throughput. This technique might be of independent interest. The software is in the public domain and available at https://cryptojedi.org/crypto/#tesla.

– In Appendix A we present a tight security reduction from LWE to *a variant of* TESLA in the *quantum random oracle model* (QROM) [25][2]. One step in this reduction requires a chameleon hash function where TESLA-768 simply uses SHA-256. We *could* have decided to implement and use a chameleon hash function instead of SHA-256 in TESLA-768, but that would have meant to significantly sacrifice performance without gaining protection against any known attack. We believe that the need for a chameleon hash function is merely an artifact of the proof and we would be surprised if our decision to use SHA-256 could be exploited in an attack. Clearly, such an attack would be a major contribution to the community's understanding of security reductions in the quantum random oracle model.

Organization of this paper. Sec. 2 briefly gives some necessary background and establishes notation. Sec. 3 presents the signature scheme TESLA and our novel security reduction. Sec. 4 analyzes the best known attacks against LWE and derives the parameters for TESLA-416 and TESLA-768. Sec. 5 gives some details of our software implementation. Finally, Sec. 6 presents performance results and concludes with a comparison between TESLA and results from the literature.

2 Preliminaries

In this section we fix notations and recall definitions of signature schemes in general, lattices, the learning with errors and the short integer solution problem, and tight security reductions.

2.1 Notation

Throughout this paper q is a prime integer (if not stated otherwise) and the elements in the ring $\mathbb{Z}_q$ are represented by the set of integers $(-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$. We denote a column vector $\mathbf{v}$ by bold lower case letters and a matrix $\mathbf{M}$ by bold upper case letters. The transpose of a vector or a matrix is denoted by $\mathbf{v}^T$ or $\mathbf{M}^T$, respectively. We denote by $||\mathbf{v}||$ the Euclidean norm of a vector $\mathbf{v}$, and by $||\mathbf{v}||_\infty$ its infinity norm. All logarithms are base 2. In this work, we mainly consider the uniform distribution and the centered discrete Gaussian distribution. For a finite set S we associate $s \leftarrow_\$ \mathscr{U}(S)$ to sample the element s uniformly from S (sometimes we simply write $s \leftarrow_\$ S$). The centered discrete Gaussian distribution for $x \in \mathbb{Z}$ is defined to be $\mathscr{D}_\sigma = \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$, where $\sigma > 0$, $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$, and $\rho_\sigma(\mathbb{Z}) = 1 + 2\sum_{x=1}^{\infty} \rho_\sigma(x)$. We denote by $d \leftarrow_\$ \mathscr{D}_\sigma$ sampling a value d randomly according to the distribution $\mathscr{D}_\sigma$.

Following the notation of [13], we define the rounding operator $\lfloor \cdot \rceil_d$ for some $d \in \mathbb{N}$ to be $\lfloor \cdot \rceil : \mathbb{Z} \to \mathbb{Z}, c \mapsto (c - [c]_{2^d})/2^d$, where $[c]_{2^d}$ denotes the unique integer in the set $(-2^{d-1}, 2^{d-1}] \subset \mathbb{Z}$ such that $c = [c]_{2^d} \pmod{2^d}$. We abbreviate $\lfloor a \pmod q \rceil$ by $\lfloor a \rceil_{d,q}$. The definition is easily extended to vectors by applying $\lfloor \cdot \rceil_d$ for each component.

A function is called *negligible* in the security parameter λ, denoted by $negl(\lambda)$, if it decreases faster than the inverse of every polynomial in λ, for sufficiently large λ. For an algorithm $\mathscr{A}$, the value $y \leftarrow \mathscr{A}(x)$ denotes the output of $\mathscr{A}$ on input x; if $\mathscr{A}$ uses randomness then $\mathscr{A}(x)$ is a random variable. Also, $\mathscr{A}^{\mathscr{O}}$ denotes that $\mathscr{A}$ has access to oracle $\mathscr{O}$. An algorithm $\mathscr{A}$ is in probabilistic polynomial-time (PPT) if $\mathscr{A}$ is randomized — uses internal random coins — and, for any input $x \in \{0,1\}^*$, the computation of $\mathscr{A}(x)$ terminates in at most $poly(|x|)$ steps. A problem is called hard if there exist no polynomial time algorithm which solves the problem.

2.2 Signature Schemes

A signature scheme Π is defined as a tuple of the following algorithms: KeyGen, Sign, and Verify, where KeyGen and Sign are randomized algorithms.

[2] There are only two other lattice-based signature schemes proven secure in QROM which are GPV [42] and a variant of Lyubashevsky's scheme [58] proven secure by Dagdelen, Fischlin, and Gagliardoni [35].

Upon input the security parameter κ the algorithm KeyGen outputs a publicly known verification key vk and a secret signing key sk. The algorithm Sign receives as input sk, vk, and a message μ. It returns a valid signature σ for the message μ. The third algorithm Verify gets the verification key vk, the signature σ, and the message μ as input and checks if σ is a valid signature for the message μ. In this case Verify accepts and outputs 1, otherwise the algorithm outputs 0. We require as usual that the signature scheme has to be correct, meaning that for any message μ, any $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ we have $\mathsf{Verify}(\mathsf{vk},\mu,\mathsf{Sign}(\mathsf{sk},\mu)) = 1$.

Let $\Pi = (\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$ be a signature scheme and $\mathscr{A}$ be a probabilistic polynomial-time (PPT) adversary which upon input the public verification key vk forges a signature. Then the scheme Π is $(t_{\mathscr{A}},q_h,q_s,\varepsilon_{\mathscr{A}})$-unforgeable if any algorithm $\mathscr{A}$ running in time $t_{\mathscr{A}}$, making at most q_s queries to a singing oracle and q_h hash queries to the random oracle outputs a forgery (μ^*,σ^*), such that μ^* was not queried to the signing oracle and $\mathsf{Verify}(\mathsf{vk},\mu^*,\sigma^*) = 1$ is at most $\varepsilon_{\mathscr{A}}$.

2.3 Lattices

A k-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$ containing all integer linear combinations of k linearly independent vectors $\{\mathbf{b}_1,\ldots,\mathbf{b}_k\}$ with $k \leq n$ and $n \geq 0$. More formally, we have $\Lambda = \{\, \mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^k \,\}$. The determinant of a lattice is the value $\det(\Lambda(\mathbf{B})) = \sqrt{\det(\mathbf{B}^\top\mathbf{B})}$. We note that a basis is not unique for a lattice and moreover, the determinant of a lattice is independent of the basis. That is, a different basis for the same lattice will yield the same determinant with the above formula.

Throughout this paper we are mostly concerned with q-ary lattices $\Lambda_q^\perp(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$, where integer $q > 0$ denotes the modulus and $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ is a uniformly random chosen matrix. Lattices $\Lambda_q^\perp(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are defined by

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} = \mathbf{0} \,(\mathrm{mod}\ q)\},$$
$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n \mid \exists \mathbf{s} \in \mathbb{Z}^m \text{ s.t. } \mathbf{x} = \mathbf{A}^\top\mathbf{s} \,(\mathrm{mod}\ q)\}.$$

Furthermore, for any $\mathbf{u} \in \mathbb{Z}_q^m$ we can define cosets $\Lambda_{\mathbf{u},q}^\perp(\mathbf{A}) = \{x \in \mathbb{Z}^n \mid \mathbf{Ax} = \mathbf{u} \,(\mathrm{mod}\ q)\}$, 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 $\mathbf{Ax} = \mathbf{u} \,(\mathrm{mod}\ q)$. We note that for a uniformly chosen matrix $\mathbf{A}$ the determinant of the above modular lattices coincide with $\det(\Lambda_{\mathbf{u},q}^\perp(\mathbf{A})) = \det(\Lambda_q^\perp(\mathbf{A})) = q^{\mathrm{rank}(A)}$ for any $\mathbf{u} \in \mathbb{Z}_q^m$. Note that the rank of a uniformly random chosen matrix equals $\min(m,n)$ with high probability.

Given a measurable set S and a lattice $L \subset \mathbb{Z}^n$, the *Gaussian heuristic* approximates the number of lattice points in S by $|S \cap L| = \frac{\mathrm{vol(S)}}{\det(L)}$. Especially, if L is a q-ary lattice $\Lambda_q^\perp(\mathbf{A})$ and $S = [-\delta,\delta]^n$ and $m >> n$, the *Gaussian heuristic* is given by $|S \cap \Lambda_q^\perp(\mathbf{A})| = \frac{(2\delta+1)^n}{q^n}$.

2.4 The Learning with Errors Problem

In the following we recall the learning with errors problem (LWE) with a bounded number of LWE-samples. To this end we define the learning with errors distribution first.

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

Since our signature scheme is based on decisional LWE problem we omit the definition of the search version in the following definition.

Definition 2 (Learning with Errors Problem) Let $n,m,q > 0$ be integers and χ be a distribution over $\mathbb{Z}$. Moreover, define $\mathscr{O}_\chi$ to be an oracle, which upon input vector $\mathbf{s} \in \mathbb{Z}_q^n$ returns samples from the distribution $\mathscr{D}_{\mathbf{s},\chi}$. The decisional learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is (t,ε)-hard if for any algorithm $\mathscr{A}$, running in time t and making at most m queries to its oracle, we have

$$\left|\Pr\left[\mathscr{A}^{\mathscr{O}_\chi(\mathbf{s})}(\cdot) = 1\right] - \Pr\left[\mathscr{A}^{\mathscr{U}(\mathbb{Z}_q^n\times\mathbb{Z}_q)}(\cdot) = 1\right]\right| \leq \varepsilon\,,$$

where the probabilities are taken over $s \leftarrow_{\$} \mathscr{U}(\mathbb{Z}_q^n)$ and the random choice of the distribution $\mathscr{D}_{\mathbf{s},\chi}$, as well as the random coins of $\mathscr{A}$.

We note that the hardness of LWE is retained even if the secret vector $\mathbf{s}$ is sampled according to the error distribution χ, known as the "normal form" [10, 60]. We use the notation $\mathsf{LWE}_{n,m,q,\sigma}$ if χ is distributed according to $\mathscr{D}_\sigma$. The LWE assumption comes with a worst-to-average-case reduction [28, 66, 69]; breaking certain average instances of LWE allows one to break all instances of certain standard lattice problems (namely GapSVP and SIVP).

Following the proposal by Bai and Galbraith [13], we use a matrix variant of LWE in this work. That means, we work on pairs of matrices $(\mathbf{A}, \mathbf{AS}+\mathbf{E} \pmod q)$ with matrices $\mathbf{S} \in \mathbb{Z}^{n\times n}$ and $\mathbf{E} \in \mathbb{Z}^{m\times n}$ instead of $(\mathbf{A}, \mathbf{As}+\mathbf{e} \pmod q)$ as LWE-tupels. This matrix variant of LWE is not easier than the LWE problem from Definition 2 [13, 55]. We refer to the matrix variant of LWE also as $\mathsf{LWE}_{n,m,q,\chi}$.

2.5 Small Integer Solution problem

The small integer solution (SIS) problem is formally defined (in the Euclidean norm) as follows.

Definition 3 (Small Integer Solution Problem) Let $n, m, q > 0$ be integers and $\beta \in \mathbb{R}_{>0}$. The small integer solution problem $\mathsf{SIS}_{n,m,q,\beta}$ is (t,ε)-hard if for any algorithm $\mathscr{A}$, running in time t, we have

$$\Pr\left[||\mathbf{x}|| \leq \beta \wedge (\mathbf{Ax} = \mathbf{0} \pmod q) \mid \mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}; \mathbf{x} \leftarrow \mathscr{A}(\mathbf{A})\right] \leq \varepsilon,$$

where the probability is taken over the random choices of matrix $\mathbf{A}$ as well as the random coins of the algorithm $\mathscr{A}$.

Similarly, the average-case instance of the SIS problem is hard as long as worst-case instances of GapSVP as shown in [2, 61].

2.6 Tightness

Let Π be a cryptographic scheme with the security based on a hard problem $\mathscr{P}$, e.g., SIS or LWE. Let $\mathscr{A}$ be an algorithm which breaks the security of the scheme Π—with respect to a security model—in time $t_\mathscr{A}$, and with a success probability of $\varepsilon_\mathscr{A}$. Let $\mathscr{R}$ be an algorithm, also called reduction, which solves the underlying problem $\mathscr{P}$ in time $t_\mathscr{R}$ with success probability $\varepsilon_\mathscr{R}$ by internally running the algorithm $\mathscr{A}$ (in a black-box way). We say the reduction of $\mathscr{P}$ to Π is *tight* if $\varepsilon_\mathscr{A} \approx \varepsilon_\mathscr{R}$ and $t_\mathscr{A} \approx t_\mathscr{R}$. Otherwise, we call the reduction *loose* or *non-tight*. The term $(t_\mathscr{R}\varepsilon_\mathscr{A})/(t_\mathscr{A}\varepsilon_\mathscr{R})$ denotes the tightness gap. We call a problem $\mathscr{P}$ *n-bit hard* if $t_\mathscr{R}/\varepsilon_\mathscr{R} \geq 2^n$, and a scheme Π *m-bit secure* if $t_\mathscr{A}/\varepsilon_\mathscr{A} \geq 2^m$. Note that a scheme Π is not necessarily n-bit secure if its security is reduced to an n-bit hard problem P, in particular, if the given reduction is non-tight.

3 The Signature Scheme TESLA

In this section, we present the lattice-based signature scheme TESLA which we prove unforgeable assuming the hardness of the LWE problem. While its construction was originally proposed by Bai and Galbraith [13] and later revisited by Dagdelen, El Bansarkhani, Göpfert, Güneysu, Oder, Pöppelmann, Sánchez, and Schwabe [75], we are able to enhance its security, minimize the underlying assumptions, remove the requirement of a secure random-number generator for signing, and improve the performance even further. Moreover, in our security reduction we get rid of the Forking Lemma which is in general an obstacle when proving quantum security [25, 35].

We (re-)name the signature scheme by Bai and Galbraith with its modifications from [75] to emphasize the various properties which we show in this paper. Throughout this paper we call it TESLA (Tightly-secure, Efficient signature scheme from Standard LAttices).

3.1 Description of the Signature Scheme

For easy reference the signature scheme $\mathsf{TESLA} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is depicted in Fig. 1. The concrete parameter sets we propose can be found in Table 1 (and their derivation in Sec. 4).

Algorithm KeyGen

INPUT: $1^\lambda; \mathbf{A}, n, m, q, \sigma$
OUTPUT: $((\mathbf{S}, \mathbf{E}, s), \mathbf{T})$

1. $\mathbf{S} \leftarrow_\$ D_\sigma^{n \times n}$
2. $\mathbf{E} \leftarrow_\$ D_\sigma^{m \times n}$
3. **if** $\mathsf{checkE}(\mathbf{E}) = 0$
   then Restart
4. $s \leftarrow_\$ \{0,1\}^\kappa$
5. $\mathbf{T} \leftarrow \mathbf{AS} + \mathbf{E} \pmod q$
6. $\mathsf{sk} \leftarrow (\mathbf{S}, \mathbf{E}, s), \mathsf{vk} \leftarrow (\mathbf{T})$
7. **return** $(\mathsf{sk}, \mathsf{vk})$

Algorithm Sign

INPUT: $\mu, q, \mathbf{A}, \mathbf{S}, \mathbf{E}, s$
OUTPUT: $(\mathbf{z}, c)$

$j \leftarrow 0$
1. $\mathbf{k} \leftarrow \mathsf{PRF}_1(s, \mu)$
2. $\mathbf{y} \leftarrow \mathsf{PRF}_2(\mathbf{k}, j)$
3. $\mathbf{v} \leftarrow \mathbf{Ay} \pmod q$
4. $c \leftarrow \mathsf{H}(\lfloor \mathbf{v} \rceil_{d,q}, \mu)$
5. $\mathbf{c} \leftarrow F(c)$
6. $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$
7. $\mathbf{w} \leftarrow \mathbf{v} - \mathbf{Ec} \pmod q$
8. **if** $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - L$,
   or $|\mathbf{w}_i| > \lfloor q/2 \rfloor - L$
   or $||\mathbf{z}||_\infty > B - U$
   then $j \leftarrow j + 1$ and go to Step 1
9. **return** $(\mathbf{z}, c)$

Algorithm Verify

INPUT: $\mu, q, \mathbf{z}, c, \mathbf{A}, \mathbf{T}$
OUTPUT: {0,1}

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

Fig. 1 Specification of the signature scheme $\mathsf{TESLA} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$; for details of the function checkE see the explanation of the key-generation algorithm.

Public Parameters. TESLA is parameterized by the integers n, m, α, κ, and the security parameter λ with $m > n > \kappa \geq \lambda$; by the matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$; by the hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$, by the pseudo-random function $\mathsf{PRF}_1 : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$, and the pseudo-random generator $\mathsf{PRF}_2 : \{0,1\}^\kappa \times \mathbb{Z} \to [-B, B]^n$. The remaining values, i.e., the standard deviation σ of the centered discrete Gaussian distribution $\mathscr{D}_\sigma$, ω, d, B, q, and L, are derived as shown in Table 1 and described in Sec. 4.1. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be a uniformly random sampled matrix which is publicly known as a global constant and can be shared among arbitrarily many signers.
The algorithms make use of a hash function H which maps a bit string of arbitrary length to a bit string of length κ[3]. Furthermore, let $\mathscr{B}_{n,\omega} = \{(v_1, ..., v_n)^T \in \{0,1\}^n \text{ with } |\{v_i \neq 0\}| = \omega\}$ and let F be an encoding function $F : \{0,1\}^\kappa \to \mathscr{B}_{n,\omega}$ which takes the binary output of the hash function and produces a vector of length n and weight ω. For more information about the encoding function see [44]. Additionally, we employ a pseudo-random function $\mathsf{PRF}_1 : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$, which maps the message to-be-signed and parts of the secret key to a pseudo-random value, which is used to key $\mathsf{PRF}_2 : \{0,1\}^\kappa \times \mathbb{Z} \to [-B, B]^n$. Doing so, all randomness in the signature-generation process is deterministically derived.

Key Generation. At first, secret matrices $\mathbf{S} \in \mathbb{Z}^{n \times n}$ and $\mathbf{E} \in \mathbb{Z}^{m \times n}$ are sampled from the discrete Gaussian distributions $D_\sigma^{n \times n}$ and $D_\sigma^{m \times n}$, respectively. The matrix $\mathbf{E}$ has to satisfy certain constraints to ensure that the signatures are correct and short. These constraints are checked by the function checkE which has been introduced in [75, Sec. 3.2]. The check algorithm works as follows: for a matrix $\mathbf{E}$, define $\mathbf{e_h}$ to be the h-th row of $\mathbf{E}$. The function $\max_k(\cdot)$ returns the k-th largest entry of a vector. The key pair is rejected if for any row of $\mathbf{E}$ it holds that $\sum_{k=1}^{\omega} \max_k(\mathbf{e_h})$ is greater than some bound L. Furthermore, a secret key s is sampled uniformly random from $\{0,1\}^\kappa$. Finally, the signing key $\mathsf{sk} = (\mathbf{S}, \mathbf{E}, s)$ and public verification key $\mathsf{vk} = \mathbf{T} = \mathbf{AS} + \mathbf{E} \pmod q$ are returned.

[3] As it is common for signatures derived by the Fiat-Shamir transform, we instantiate a signature of bit security λ using a random oracle which outputs λ bits. As in [58], finding a collisions in the random oracle does not constitute a break.

Signing Algorithm. During signing of a message μ, generate a secret seed $\mathbf{k} = \mathsf{PRF}_1(s,\mu)$ first. Afterwards, a pseudo-random vector $\mathbf{y}$ is obtained by computing $\mathbf{y} = \mathsf{PRF}_2(\mathbf{k}, j) \in [-B, B]^n$ and multiplied by $\mathbf{A}$ in $\mathbb{Z}_q$. The value j is simply a counter that is increased until signing succeeds. Afterwards, the higher order bits of $\mathbf{v} = \mathbf{A}\mathbf{y} \pmod q$ are hashed together with the message μ yielding the hash value c. Applying the encoding function to c we obtain a value $\mathbf{c} = F(c)$. Further on, we compute $\mathbf{z} = \mathbf{S}\mathbf{c} + \mathbf{y}$ in $\mathbb{Z}$. Now, rejection sampling is applied to make sure that the signature does not leak any information about the secret $\mathbf{S}$ and that the signature verifies for the applied compression. That is, if either $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - L$ or $|\mathbf{w}_i| > \lfloor q/2 \rfloor - L$, or $||\mathbf{z}||_\infty > B - U$, with $\mathbf{w} = \mathbf{v} - \mathbf{E}\mathbf{c} \pmod q$, then the signing algorithm discards $(\mathbf{z}, c)$ and repeats all steps. Finally, it returns the signature $(\mathbf{z}, c)$ on the message μ.

Verification Algorithm. The algorithm, upon input of a message μ and a signature $(\mathbf{z}, c)$, first computes $\mathbf{c} = F(c)$ to obtain $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \pmod q$, and returns 1 if $c = \mathsf{H}(\lfloor \mathbf{w}' \rceil_{d,q}, \mu)$ and $||\mathbf{z}||_\infty \leq B - U$ are both satisfied; otherwise, it returns 0.

In contrast to earlier proposals [13,75], we add an additional check during the signature generation. Namely, we also check that the absolute value of every coordinate of $\mathbf{w}$ is less or equal than $\lfloor q/2 \rfloor - L$ to ensure correctness of the scheme.

3.2 Security Reduction

Bai and Galbraith [13] prove the security of their signature scheme assuming the hardness of *both* LWE and (unbalanced) SIS. The proof follows the standard way of proving Fiat-Shamir-type signatures, namely using the Forking Lemma proposed by Pointcheval and Stern [68]. As mentioned earlier, although the Forking Lemma is a powerful and actively used tool to prove security of signatures, the obtained security reductions are not tight. Furthermore, the use of the Forking Lemma makes it difficult to prove security of schemes against quantum adversaries.

To avoid this lemma we use a reduction method introduced by Katz and Wang [48]. The underlying idea is to give a hypothetical adversary against the signature scheme either a genuinely generated public key of the system or a fake – i.e., random – one. The underlying assumption is that those keys cannot be distinguished easily. Now, if the scheme guarantees that for fake public keys the existence of valid signatures is statistically bounded by a negligible probability, the hypothetical adversary will simply fail to forge in case a fake public key is given. On the other hand, in case a public key is generated honestly, the adversary will output a forgery by assumption. This different behavior of the adversary helps the security reduction to distinguish between the two samples.

The following theorem shows that the signature scheme TESLA is unforgeable. Specifically, our security reduction is tight and the security of TESLA relies solely on the decisional learning with errors problem in the random oracle model.

Theorem 4 *Let the parameters $n, m, \omega, d, B, q, U, L, \sigma$ be arbitrary but satisfying the bounds in Table 1. Assume that the Gaussian heuristic holds for lattice instances defined by the parameters above. If $\mathsf{LWE}_{n,m,q,\sigma}$ is $(t_{\mathscr{D}}, \varepsilon_{\mathscr{D}})$-hard, the signature scheme* TESLA *in Fig. 1 is $(t_{\mathscr{A}}, q_h, q_s, \varepsilon_{\mathscr{A}})$-unforgeable against adaptively chosen-message attacks in the random oracle model where $t_{\mathscr{D}} = t_{\mathscr{A}} + \mathcal{O}(q_s \lambda^3)$ and*

$$\varepsilon_{\mathscr{D}} \geq \left(\varepsilon_{\mathsf{PRF}} + (1 - \varepsilon_{\mathsf{PRF}}) \left(\varepsilon_{\mathscr{A}} - \varepsilon_{\mathscr{A}} \frac{q_s(q_s+q_h)2^{(d+1)m}}{(2B+1)^n q^{m-n}} - \frac{q_h 2^{dn}(2B-2U+1)^n}{q^m} - \frac{(28\sigma+1)^{mn+n^2}}{q^{mn}} \right) \right) \delta_{check} \,,$$

where $\varepsilon_{\mathsf{PRF}}$ is the prf-advantage[4] *of $\mathsf{PRF}_2 \circ \mathsf{PRF}_1$ and δ_{check} is the probability that the procedure* checkE *accepts a value $\mathbf{E} \leftarrow_\$ D_\sigma^{m \times n}$ in Fig. 1.*

Before proving our main theorem we build some supplementary results. First, we recall a useful lemma stated by Bai and Galbraith [13, Lemma 3] which gives us information about the number of possible values for $\lfloor \mathbf{A}\mathbf{y} \rceil_{d,q}$. To prove their statement, Bai and Galbraith make use of the Gaussian heuristic.

[4] The prf-advantage ε of a family of keyed functions $\mathscr{F} = \{F_s : D \to C\}_{s \in S}$ is defined as $\varepsilon = |\Pr[s \leftarrow_\$ S : \mathscr{A}^{F_s}() = 1] - \Pr[f \leftarrow_\$ \Gamma : \mathscr{A}^{f}() = 1]|$ for all polynomial-time adversaries $\mathscr{A}$, where Γ is the set of all functions which map from D to C.

Lemma 5 *Let the parameters n,m,d,B, and q and the assumptions be as in Theorem 4. Furthermore, let $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$. Then for $\mathbf{y_1}, \mathbf{y_2} \leftarrow_\$ [-B,B]^n$ it holds that*

$$\Pr\left[\lfloor \mathbf{Ay_1} \rceil_{d,q} = \lfloor \mathbf{Ay_2} \rceil_{d,q} \pmod q \right] \leq \frac{2^{(d+1)m}}{(2B+1)^n q^{m-n}},$$

where the probability is taken over random choices of $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{A}$.

During the reduction we use that for randomly chosen matrices $\mathbf{A}, \mathbf{T} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$, where n,m are chosen with respect to the security parameter λ, the probability that there exist matrices $\mathbf{S}$ and $\mathbf{E}$ with "small" entries such that $\mathbf{AS}+\mathbf{E} = \mathbf{T} \pmod q$ is negligible in λ. This statement is captured in the following lemma.

Lemma 6 *Let n, m, q, and σ be as defined in Theorem 4. Furthermore, let $\mathbf{A}, \mathbf{T} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$. The probability that there exist matrices $\mathbf{S} \in [-k\sigma, k\sigma]^{n\times n}$ and $\mathbf{E} \in [-k\sigma, k\sigma]^{m\times n}$, for some $k \in \mathbb{N}$, such that $\mathbf{AS}+\mathbf{E} = \mathbf{T} \pmod q$, is bounded by the following term*

$$\begin{aligned}&\Pr\left[\ \exists\, \mathbf{S} \in [-k\sigma, k\sigma]^{n\times n}, \mathbf{E} \in [-k\sigma, k\sigma]^{m\times n} \text{ such that } \mathbf{AS}+\mathbf{E} = \mathbf{T} \pmod q \right] \\ \leq\ & \frac{(2k\sigma+1)^{mn+n^2}}{q^{mn}},\end{aligned}$$

where the probability is taken over random choices of $\mathbf{A}$ and $\mathbf{T}$.

Proof First, we bound the probability that there exist vectors $\mathbf{s}$ and $\mathbf{e}$ with small entries such that $\mathbf{As}+\mathbf{e} = \mathbf{t}$ for $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$ and $\mathbf{t} \leftarrow_\$ \mathbb{Z}_q^m$, i.e., we show that

$$\begin{aligned}\delta :=\ & \Pr[\ \exists\, \mathbf{s} \in [-k\sigma, k\sigma]^n,\ \mathbf{e} \in [-k\sigma, k\sigma]^m \mid\ \mathbf{As}+\mathbf{e} = \mathbf{t} \pmod q] \\ \leq\ & \frac{(2k\sigma+1)^{m+n}}{q^m}.\end{aligned}$$

Since $\mathbf{A}$ and $\mathbf{t}$ are chosen uniformly random the probability δ can be bound by the ratio of the number of possible vectors $\mathbf{As}+\mathbf{e}$ with $\mathbf{s} \in [-k\sigma, k\sigma]^n$ and $\mathbf{e} \in [-k\sigma, k\sigma]^m$, and the number of possible vectors for $\mathbf{t} \leftarrow_\$ \mathbb{Z}_q^m$, i.e.,

$$\delta \leq \frac{|\{\mathbf{As}+\mathbf{e} \mid \mathbf{s} \in [-k\sigma, k\sigma]^n \text{ and } \mathbf{e} \in [-k\sigma, k\sigma]^m\}|}{|\{\mathbf{t} \in \mathbb{Z}_q^m\}|}.$$

It holds that

$$\begin{aligned}|\{\mathbf{t} \in \mathbb{Z}_q^m\}| &= q^m, \\ |\{\mathbf{As} \mid \mathbf{s} \in [-k\sigma, k\sigma]^n\}| \leq |\{\mathbf{s} \in [-k\sigma, k\sigma]^n\}| &\leq (2k\sigma+1)^n \text{ since } \mathit{rank}(\mathbf{A}) = n, \\ |\{\mathbf{e} \in [-k\sigma, k\sigma]^m\}| &\leq (2k\sigma+1)^m.\end{aligned}$$

Thus,

$$\begin{aligned}|\{\mathbf{As}+\mathbf{e} \mid \mathbf{s} \in [-k\sigma, k\sigma]^n \text{ and } \mathbf{e} \in [-k\sigma, k\sigma]^m\}| &\leq (2k\sigma+1)^n (2k\sigma+1)^m \\ &= (2k\sigma+1)^{n+m}.\end{aligned}$$

Now, assume $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$ and $\mathbf{T} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$ are given. Let $\mathbf{s_i}$, $\mathbf{e_i}$, and $\mathbf{t_i}$ be the columns of $\mathbf{S}$, $\mathbf{E}$, and $\mathbf{T}$, respectively. Since the columns $\mathbf{s_i}$ and $\mathbf{s_j}$, and $\mathbf{e_i}$ and $\mathbf{e_j}$, are independent from each other for all $i, j \in \{1, \cdots, n\}$, and $i \neq j$, it holds that

$$\begin{aligned}&\Pr\left[\ \exists\, \mathbf{S} \in [-k\sigma, k\sigma]^{n\times n} \text{ and } \mathbf{E} \in [-k\sigma, k\sigma]^{m\times n} \mid \mathbf{AS}+\mathbf{E} = \mathbf{T}\ \right] \\ &\leq \Pr[\ \forall i \in \{1, \cdots, n\}\ \exists\, \mathbf{s_i} \in [-k\sigma, k\sigma]^n,\ \mathbf{e_i} \in [-k\sigma, k\sigma]^m \mid \mathbf{As_i}+\mathbf{e_i} = \mathbf{t_i}\ ] \\ &\leq \delta^n \\ &\leq \frac{(2k\sigma+1)^{mn+n^2}}{q^{mn}}.\end{aligned}$$

□

Remark 7 By [58, Lemma 4.4], we know that $\forall\ k > 0$ $\Pr[\,|z| > k\sigma \text{ for } z \leftarrow_{\$} D_\sigma] \leq 2\exp(-k^2/2)$. Hence, the probability that all entries of the matrices $\mathbf{S} \leftarrow_{\$} D_\sigma^{n\times n}$ and $\mathbf{E} \leftarrow_{\$} D_\sigma^{m\times n}$ are in the interval $[-k\sigma, k\sigma]$ is at least $\left(1-2\exp(-k^2/2)\right)^{2n(m+n)}$. For $k = 14$ and our choice for m and n given in Table 1 this probability is overwhelming, i.e., $\left(1-2\exp(-\frac{k^2}{2})\right)^{2n(m+n)} \geq 1-2^{-\lambda}$ for $\lambda = 128$.

In our security reduction we show that if an algorithm $\mathscr{A}$ is given a randomly chosen tuple $(\mathbf{A}, \mathbf{T})$, then there exists a valid signature only with negligible probability. To this end, we utilize the following lemmata.

Lemma 8 *Let $\mathbf{A}, \mathbf{T} \in \mathbb{Z}_q^{m\times n}$, with rank$(\mathbf{A}) = n$ and such that $\mathbf{T} \neq \mathbf{AS} + \mathbf{E} \pmod q$ for any $\mathbf{S} \in [-14\sigma, 14\sigma]^{n\times n}$ and $\mathbf{E} \in [-14\sigma, 14\sigma]^{n\times m}$. Furthermore, let $\mathbf{v} \in \mathbb{Z}_q^m$, $\delta \in \mathbb{Q}_{>0}$, and q, n, m and the assumptions be as in Theorem 4. Then for $c \leftarrow_{\$} \{0,1\}^\kappa$ it holds that*

$$\Pr\left[\ \exists\ \mathbf{z} \in [-\delta, \delta]^n \text{ such that } \lfloor \mathbf{Az} \rceil_{d,q} = \mathbf{v} + \lfloor \mathbf{Tc} \rceil_{d,q} \pmod q\right] \leq \frac{2^{dn}(2\delta+1)^n}{q^m},$$

where the probability is taken over random choices of c with $\mathbf{c} = F(c)$.

Proof Recall that the rounding operator $\lfloor \cdot \rceil_{d,q}$ essentially drops component-wise the least d bits of a vector. Thus, we can bound the probability in the theorem statement by the probability that there exists an $\mathbf{z} \in [-\delta, \delta]^n$ such that $\mathbf{Az} = \mathbf{v} + \lfloor \mathbf{Tc} \rceil_{d,q} \pmod q$ multiplied by the factor of 2^{dn}, where the probability is again taken over random choices of c with $\mathbf{c} = F(c)$. By assumption, the rank of $\mathbf{A}$ is equal to n, thus the map defined through $\mathbf{A}$ is injective. Hence, there are at most 2^{dn} possible values for $\mathbf{Az}$ such that $\lfloor \mathbf{Az} \rceil_{d,q} = \mathbf{v} + \lfloor \mathbf{Tc} \rceil_{d,q} \pmod q$. It remains to show that for fixed $\mathbf{v}$ and $c \leftarrow_{\$} \{0,1\}^\kappa$ with $\mathbf{c} = F(c)$ it holds that

$$\Pr\left[\ \exists\ \mathbf{z} \in [-\delta, \delta]^n \mid \mathbf{Az} = \mathbf{v} + \lfloor \mathbf{Tc} \rceil_{d,q} \pmod q\right] \leq \frac{(2\delta+1)^n}{q^m}, \tag{1}$$

where the probability is taken over random choices of c.

Let $\mathbf{u} = \mathbf{v} + \lfloor \mathbf{Tc} \rceil_{d,q} \pmod q$ and let $\mathbf{A}$ be of the form $\mathbf{A} = \begin{pmatrix} \mathbf{A_1} \\ \mathbf{A_2} \end{pmatrix}$ where $\mathbf{A_1} \in \mathbb{Z}_q^{n\times n}$ and $\mathbf{A_2} \in \mathbb{Z}_q^{m-n\times n}$, and $\mathbf{u} = \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \end{pmatrix}$ where $\mathbf{u_1} \in \mathbb{Z}_q^n$ and $\mathbf{u_2} \in \mathbb{Z}_q^{m-n}$. Without loss of generality – by shifting the rows of $\mathbf{A}$ and $\mathbf{b}$ simultaneously – we can assume that $\mathbf{A_1}$ is a square matrix of rank$(\mathbf{A_1}) = n$.

We stress that there exists a unique vector $\bar{\mathbf{z}} = (\bar{z}_1, \cdots, \bar{z}_n) \in \mathbb{Z}_q^n$ such that $\mathbf{A_1}(\bar{z}_1, \cdots, \bar{z}_n)^T = \mathbf{u_1} \pmod q$. We prove Equation (1), by pointing on two things: **(i)** the probability that $\bar{\mathbf{z}}$ is in $[-\delta, \delta]^n$ can be bounded by $(2\delta+1)^n/q^n$, and **(ii)** the probability that $\mathbf{A_2}\bar{\mathbf{z}} = \mathbf{u_2} \pmod q$ is bounded by $1/q^{m-n}$. Let us have a closer look at these two points.

(i) Define the set $S_{u_1} = \{\mathbf{z} \in [-\delta, \delta]^n \mid \mathbf{A_1 z} = \mathbf{u_1} \pmod q\}$. Since $\mathbf{A_1}$ is of rank n, it defines a shifted random lattice $\Lambda^\perp_{u_1,q}(\mathbf{A_1}) = \{\mathbf{z} \in \mathbb{Z}^n \mid \mathbf{A_1 z} = \mathbf{u_1} \pmod q\}$ where $\det(\Lambda^\perp_{u_1,q}(\mathbf{A_1})) = q^n$. Via the *Gaussian heuristic* we approximate the number of lattice vectors in the set S_{u_1} upper bounding the probability for **(i)**. We obtain

$$|[-\delta, \delta]^n \cap \Lambda^\perp_{u_1,q}(\mathbf{A_1})| = \frac{\mathrm{vol}([-\delta, \delta]^n)}{\det(\Lambda^\perp_{u_1,q}(\mathbf{A_1}))} = \frac{(2\delta+1)^n}{q^n}\ .$$

(ii) Let $a_{i,j}$ and u_i denote the elements of $\mathbf{A}$ and $\mathbf{u}$, respectively. To bound the probability that $\mathbf{A_2}\bar{\mathbf{z}} = \mathbf{u_2} \pmod q$ holds, we basically bound the probability that $a_{i,1}\bar{z}_1 + \cdots + a_{i,n}\bar{z}_n = u_i$ holds for every $i = n+1, \cdots, m$. Note that this equation can be rewritten into the form $u_i - a_{i,1}\bar{z}_1 - \ldots - a_{i,n-1}\bar{z}_{n-1} = a_{i,n}\bar{z}_n$. Let $\bar{z}_1, \ldots, \bar{z}_{n-1}$ be arbitrary. Then, the probability that $\bar{z}_n$ will satisfy the above equation is $1/q$. Since index i ranges from $n+1$ to m, we get $1/q^{m-n}$ in total.

Taking the bounds given by **(i)** and **(ii)** together we show Equation (1), and obtain the theorem statement by multiplying with 2^{dn}. □

We reformulate the lemma and obtain the following corollary useful for the proof of our main theorem.

Corollary 9 *Let $\mathbf{A},\mathbf{T}\in\mathbb{Z}_q^{m\times n}$, with rank$(\mathbf{A})=n$ and such that $\mathbf{T}\neq\mathbf{AS}+\mathbf{E}\pmod q$ for any $\mathbf{S}\in[-14\sigma,14\sigma]^{n\times n}$ and $\mathbf{E}\in[-14\sigma,14\sigma]^{n\times m}$. Furthermore, let $\mathbf{v}\in\mathbb{Z}_q^m$, $\delta\in\mathbb{Q}_{>0}$, and q,n,m and the assumptions be as in Theorem 4. Then for $c\leftarrow_{\$}\{0,1\}^\kappa$ it holds that*

$$\Pr\left[\ \exists\,\mathbf{z}\in[-\delta,\delta]^n\ \text{ such that }\ \mathbf{v}=\lfloor\mathbf{Az}-\mathbf{Tc}\rceil_{d,q}\pmod q\ \right]\leq\frac{2^{dn}(2\delta+1)^n}{q^m},$$

where the probability is taken over random choices of c with $\mathbf{c}=F(c)$.

We are now ready to prove Theorem 4.

Proof (Theorem 4) We prove the theorem in two games: first, we show that using a pseudo-random function instead of a truly random function in the signing process increases the success probability of the reduction $\mathscr{D}$ only minor; secondly, we prove security of the signature scheme where $\mathsf{PRF}_2\circ\mathsf{PRF}_1$ is replaced by a truly random function.

Game 1: Define the signature scheme $\mathsf{SIG}'=(\mathsf{KeyGen}',\mathsf{Sign}',\mathsf{Verify}')$, where KeyGen' and Verify' are the same as KeyGen and Verify, respectively. Sign' differs from Sign as it chooses $\mathbf{y}\leftarrow_{\$}[-B,B]^n$ instead of computing it by PRF_1 and PRF_2. Let $\varepsilon'_{\mathscr{D}}$ be the success probability to break the unforgeability of SIG'.

The success probability against TESLA, i.e, $\varepsilon_{\mathscr{A}}$, is upper bounded by the sum of the success probability where the adversary distinguishes $\mathsf{PRF}_2\circ\mathsf{PRF}_1$ from a truly random function and the success probability where she does not distinguish the two functions but breaks the unforgeability of SIG'. Moreover, the success probability against TESLA is depending on the acceptance probability of checkE, since this function reduces the key space. Hence we have, $\varepsilon_{\mathscr{A}}\leq\left(\varepsilon_{\mathsf{PRF}}+\varepsilon'_{\mathscr{D}}(1-\varepsilon_{\mathsf{PRF}})\right)\delta_{check}$, where δ_{check} is the probability that checkE accepts an error matrix $\mathbf{E}$.

Game 2: Now we derive the success probability to win the unforgeability game against SIG'. Let $\mathscr{A}$ be an algorithm which runs in time $t_{\mathscr{A}}$, makes q_h hash queries and q_s sign queries, and forges a signature with probability $\varepsilon_{\mathscr{A}}$. We show how to build a distinguisher $\mathscr{D}$ solving $\mathsf{LWE}_{n,m,q,\sigma}$ in time $t_{\mathscr{D}}$ with probability $\varepsilon'_{\mathscr{D}}$ as stated in the theorem.

Algorithm $\mathscr{D}$ upon input tuple $(\mathbf{A},\mathbf{T})$ has to decide whether $\mathbf{T}$ is a matrix sampled uniformly from $\mathbb{Z}_q^{m\times n}$ or whether it is of the form $\mathbf{T}=\mathbf{AS}+\mathbf{E}$ for some $\mathbf{S}\leftarrow_{\$}D_\sigma^{n\times n}$ and $\mathbf{E}\leftarrow_{\$}D_\sigma^{m\times n}$. That means, $\mathscr{D}$ outputs 1 if $\mathbf{T}=\mathbf{AS}+\mathbf{E}$ and 0 otherwise. The algorithm $\mathscr{D}$ uses $\mathscr{A}$ as a black-box. Algorithm $\mathscr{A}$ expects as input the public key. To this end, $\mathscr{D}$ hands over its own challenge tuple $(\mathbf{A},\mathbf{T})$. The responses to the hash and sign queries made by $\mathscr{A}$ are simulated as follows:

Hash queries: Algorithm $\mathscr{D}$ answers with values c uniformly sampled from $\{0,1\}^\kappa$; however, if an input to the oracle repeats, we keep being consistent and reply with the same hash value as before.

Sign queries: Upon input a message μ, $\mathscr{D}$ simulates a signature $(\mathbf{z},c)$ on μ by the following steps: $\mathscr{D}$ chooses uniformly random $c\leftarrow_{\$}\{0,1\}^\kappa$ and vector $\mathbf{z}\leftarrow_{\$}[-B+U,B-U]^n$, computes $\mathbf{c}=F(c)$ and $\mathbf{w}=\mathbf{Az}-\mathbf{Tc}\pmod q$, and checks whether $|[\mathbf{w}_i]_{2^d}|<2^{d-1}-L$ for all $i\in\{1,\cdots,m\}$. If the latter is not fulfilled, $\mathscr{D}$ chooses c and $\mathbf{z}$ again and repeats. Moreover, if the oracle was queried before on that input, namely on $(\lfloor\mathbf{w}\rceil_d,\mu)$, then $\mathscr{D}$ aborts the simulation. Otherwise, $\mathscr{D}$ returns $(\mathbf{z},c)$.

Eventually, $\mathscr{A}$ outputs a forgery $(\tilde{\mathbf{z}},\tilde{c})$ on some message $\tilde{\mu}$ that was not queried to the sign oracle. We also assume that $\mathscr{A}$ made the hash-query $\mathsf{H}(\lfloor\mathbf{A}\tilde{\mathbf{z}}-\mathbf{T}\tilde{\mathbf{c}}\rceil_{d,q},\tilde{\mu})$. If $\mathsf{Verify}(\mathsf{vk},\mu,(\tilde{\mathbf{z}},\tilde{c}))=1$, algorithm $\mathscr{D}$ returns 1; else $\mathscr{D}$ returns 0. In the following, we distinguish between two cases where $\mathbf{T}$ follows the LWE distribution — i.e, $\mathbf{T}=\mathbf{AS}+\mathbf{E}$ — or $\mathbf{T}$ was sampled uniformly.

1st case, $\mathbf{T}=\mathbf{AS}+\mathbf{E}$: By [13, Lemma 4], we know that for the algorithm $\mathscr{A}$ the response to hash or sign queries from the simulation by $\mathscr{D}$ is indistinguishable from responses made by a real hash function and signer. Thus, the only possibility that $\mathscr{D}$ falsely outputs 0 is when $\mathscr{D}$ aborts during a sign query or when the algorithm $\mathscr{A}$ fails. The probability that $\mathscr{A}$ fails is $1-\varepsilon_{\mathscr{A}}$. By Lemma 5, the probability that $\mathscr{D}$ will abort during the simulation of $\mathscr{A}$'s environment is $q_s(q_h+q_s)\frac{2^{(d+1)m}}{(2B+1)^nq^{m-n}}$. Thus, $\mathscr{D}$ outputs 1 correctly with probability at least $\varepsilon_{\mathscr{A}}\left(1-q_s(q_h+q_s)\frac{2^{(d+1)m}}{(2B+1)^nq^{m-n}}\right)$.

2nd case, $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_q^{m\times n}$: First by Remark 7, we can bound the entries of matrices $\mathbf{S}$ and $\mathbf{E}$ with high probability by 14σ, since they are Gaussian distributed with standard deviation σ. We stress that by Lemma 6 the probability that there exist matrices $\mathbf{S} \in [-14\sigma, 14\sigma]^{n\times n}$ and $\mathbf{E} \in [-14\sigma, 14\sigma]^{m\times n}$ such that $\mathbf{T} = \mathbf{AS} + \mathbf{E} \pmod q$ for $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_q^{m\times n}$ is smaller or equal $\frac{(28\sigma+1)^{mn+n^2}}{q^{mn}}$. Assume $\mathbf{T}$ is not of this form (which is the case with probability greater than $1-2^{\lambda}$ with our choice of parameters).
Since we assumed that for every forgery $(\tilde{c}, \tilde{\mathbf{z}})$ on message $\tilde{\mu}$ by $\mathscr{A}$ the hash oracle was queried on $(\lfloor \mathbf{A}\tilde{\mathbf{z}} - \mathbf{T}\tilde{\mathbf{c}} \rceil_{d,q}, \tilde{\mu})$ with $\tilde{\mathbf{c}} = F(\tilde{c})$, it suffices to show the following:
For any hash query $c = \mathsf{H}(\mathbf{v}, \mu)$ by $\mathscr{A}$ the probability that there exists a vector $\mathbf{z}$ such that $(c, \mathbf{z})$ is a valid signature for message μ is negligible with our choice of parameters. That means, letting $\mathbf{v} \in \mathbb{Z}_q^m$ be fixed we want to show that the probability, taken over random choices of $c \leftarrow_{\$} \{0,1\}^{\kappa}$, that there exists a vector $\mathbf{z} \in \mathbb{Z}_q^n$ with $||\mathbf{z}||_{\infty} \leq B - U$ such that $\mathbf{v} = \lfloor \mathbf{Az} - \mathbf{Tc} \rceil_{d,q} \pmod q$, with $\mathbf{c} = F(c)$, is negligible. By Corollary 9, the probability that such a vector $\mathbf{z}$ exists is smaller than $\frac{2^{dn}(2(B-U)+1)^n}{q^m}$ and $q_h \frac{2^{dn}(2(B-U)+1)^n}{q^m}$ is negligible with our choice of parameters.
By definition of a forgery $(\tilde{\mathbf{z}}, \tilde{c})$ on a message $\tilde{\mu}$, $\tilde{\mu}$ was not used in a sign query by $\mathscr{A}$. Hence, with probability $1-2^{-\kappa}$ the value $\tilde{c}$ has not been used in any of the signatures seen by the forger. Thus, $\mathscr{A}$ cannot exploit the simulated signatures to simulate a forgery. Hence, we have an upper bound on the probability of $\mathscr{D}$ falsely returning 1 of $q_h \frac{2^{dn}(2(B-U)+1)^n}{q^m} + \frac{(28\sigma+1)^{mn+n^2}}{q^{mn}}$.

Finally,

$$\begin{aligned}&\left|\Pr\left[\mathbf{S} \leftarrow_{\$} D_{\sigma}^{n\times n}, \mathbf{E} \leftarrow_{\$} D_{\sigma}^{m\times n} : \mathscr{D}(\mathbf{A}, \mathbf{AS}+\mathbf{E}) = 1\right] - \Pr\left[\mathbf{T} \leftarrow_{\$} \mathbb{Z}_q^{m\times n} : \mathscr{D}(\mathbf{A}, \mathbf{T}) = 1\right]\right| \\ &= \varepsilon'_{\mathscr{D}} \geq \varepsilon_{\mathscr{A}} \left(1 - \frac{q_s(q_s+q_h)2^{(d+1)m}}{(2B+1)^n q^{m-n}}\right) - \frac{q_h 2^{dn}(2B-2U+1)^n}{q^m} - \frac{(28\sigma+1)^{mn+n^2}}{q^{mn}}.\end{aligned}$$

It remains to show that the time $t_{\mathscr{D}}$ is close to the running time of $\mathscr{A}$. The running time $t_{\mathscr{D}}$ of $\mathscr{D}$ includes the running time $t_{\mathscr{A}}$ of the algorithm $\mathscr{A}$. Besides, $t_{\mathscr{D}}$ is dominated by the computational time of answering sign queries during the simulation, which essentially consists of a constant (small) number of a matrix-vector multiplication. Such a multiplication runs in complexity $\mathscr{O}(\lambda^3)$. In every simulation of a signature query the probability that the simulated signature is returned, i.e., $|[\mathbf{w}_i]_{2^d}| < 2^{d-1} - L$, can be bounded by $\left(\frac{2^d - L}{2^d}\right)^m$. Because $\mathbf{A}$, $\mathbf{T}$, c, and $\mathbf{z}$ are chosen uniformly random, $\mathbf{w}$ is uniformly random distributed in $\mathbb{Z}_q^m$. Since q is a large integer, also the $d-$least significant bits are distributed (closely to) uniformly random distributed in the range of $[-2^d, 2^d]$. Thus, the ratio of $2^d - L$ and 2^d gives the probability we are looking for. Since L is always greater than 2, the probability that $|[\mathbf{w}_i]_{2^d}| < 2^{d-1} - L$ is always greater than $1/2$. Therefore, we get $t_{\mathscr{D}} \approx t_{\mathscr{A}} + \mathscr{O}(q_s \lambda^3)$. □

Remark 10 (Deterministic signature) Note, that signing is deterministic for each message μ since the randomness is determined by the vector $\mathbf{y}$ which is deterministically computed by the secret key and the message to-be-signed. In the original scheme by Bai and Galbraith [13] the vector $\mathbf{y}$ was sampled uniformly random in $[-B, B]^n$. As long as we assume that PRF_1 and PRF_2 are pseudo-random functions, the prf-advantage of $\mathsf{PRF}_2 \circ \mathsf{PRF}_1$ is negligible in the security parameter. Hence, the reduction given in Theorem 4 is tight with our choice of parameters. The idea to use a pseudo-random function to generate signatures deterministically was deployed several times before [15, 19, 48, 65, 71].

Remark 11 (Strongly unforgeability against chosen-message attacks.) We remark that we prove TESLA unforgeable against chosen-message attacks. Alternatively, we could aim for a stronger security model, namely strong unforgeability. Roughly speaking, a signature scheme is strongly unforgeable if an adversary is not able to forge a new signature which she has not received from the signing oracle. In contrast to standard unforgeability, an adversary also succeeds if she finds a forgery for a message she has already queried to the signing oracle (as long as the signature is different). In order to prove TESLA strongly unforgeable we would need to be based on the SIS assumption to prevent an adversary to find collisions in the hash input. We note that other lattice-based signature schemes, like the GLP signature [44] as stated in [39], are also proven (standard) unforgeable.

Table 1 Concrete Instantiation TESLA-416 and TESLA-768 of 128-bit of security against classical and quantum adversaries, respectively; comparison with the instantiation proposed in [75]; sizes are given in kilo byte [KB]; sizes are theoretic sizes for fully compressed keys and signatures, for sizes used by our software see Table 3

Parameter selection				
Parameter	Bound	Dagdelen et al. [75]	TESLA-416	TESLA-768
λ		128	128	256
κ		256	256	256
n		532	416	768
m		840	800	1376
σ	$> 2\sqrt{n}$, see Sec. 4.1	43	114	55
α		128	1	1
L	empirically, see Sec. 4.1	2322	6042	5535
ω	$2^{\omega}\binom{n}{\omega} \geq 2^{\kappa}$	18	20	39
B	$\geq 14(n-1)\sqrt{\omega}\sigma$	$2^{21}-1$	$2^{23}-1$	$2^{23}-1$
U	$14\sqrt{\omega}\sigma$	2554.1	7138	4809
d	$(1-2L/2^d)^m \geq 0.4$	23	24	24
q	$\geq \left(\frac{2^{(d+1)m+\alpha}}{(2B)^n}\right)^{1/(m-n)}, \geq 4B$	$2^{29}-3$	$2^{27}-39$	$2^{28}-57$
Prob. of acc.,KeyGen (δ_{check})	empirically, see Sec. 4.1	0.99	0.35	0.99
Prob. of acc., Sign		0.314	0.357	0.217
A	see Sec. 4.2			
H	$\{0,1\}^* \to \{0,1\}^{\kappa}$	SHA-256	SHA-256	SHA-256
PRF_1	$\{0,1\}^{\kappa} \times \{0,1\}^* \to \{0,1\}^{\kappa}$	-	SHA-256	SHA-256
PRF_2	$\{0,1\}^{\kappa} \times \mathbb{Z} \to [-B,B]^n$	-	ChaCha20	ChaCha20
public-key size	$mn\lceil\log_2(q)\rceil$	1581.97 KB	1096.88 KB	3612 KB
secret-key size	$(n^2+nm)\lceil\log_2(14\sigma)\rceil$	890.99 KB	679.25 KB	2010 KB
signature size	$n\lceil\log_2(2B)\rceil+\kappa$	1.4 KB	1.25 KB	2.28 KB

4 Selecting Parameters for TESLA

In this section we propose parameters for TESLA such that the signature scheme is 128-bit secure in both the classical and post-quantum setting. In particular, we deduce parameters using state-of-the-art methods on assessing the concrete hardness of LWE against classical and against quantum adversaries.

4.1 Optimizing for Software Efficiency

We proceed similarly to Dagdelen et al. [75] when selecting parameters for TESLA. However, our security reduction for TESLA minimizes the underlying assumptions which allows us to choose secure parameters from a greater set of choices. More precisely, our parameters do not have to involve a 128-bit hard instance of the SIS assumption. In fact, running a lattice-reduction algorithm on our proposed instances yields a bit security of 108 bits for the SIS assumption for TESLA-416 and 227 bits in case of TESLA-768 (see Table 2). Nevertheless, since we base security only on LWE (instead of on both, LWE and SIS as it was done by Bai and Galbraith [13] before) our parameters yield an 128-bit secure signature scheme.

Table 1 illustrates our concrete choice of parameters for 128-bit security. Note that we introduce a new parameter α, which is not present in previous descriptions of the scheme [13, 75]. This value gives an upper bound for the probability that a forger $\mathscr{A}$ in the security game can distinguish the simulated signatures from honestly generated ones in the security reduction (see Sec. 3.2). More precisely, the distinguishing probability is bounded by $\frac{q_s(q_s+q_h)2^{(d+1)m}}{(2B+1)^n q^{m-n}}$. Bai and Galbraith [13] chose parameters, in particular the modulus q such that the distinguishing probability is negligible in the security parameter, i.e., smaller than $2^{-\lambda}$. We stress however, that it suffices if the forger $\mathscr{A}$ cannot differentiate them in at least half of the (complete) runs. In that case, the reduction has to run $\mathscr{A}$ twice (expected) to succeed. Doing so, one has to take the loss of security due to this distinguishing probability into account. Nonetheless, if $\mathscr{A}$ succeeds to distinguish genuine from fake signatures in less than 50% of the cases, the signature instantiation merely loses one bit of security.

Another important parameter of the signature scheme is the value L. In the original work [13], Bai and Galbraith set $L = 7\omega\sigma$, whereas Dagdelen et al. [75] chose $L = 3\omega\sigma$. We propose a different way to choose parameter L. Specifically, we select L such that the function checkE accepts an error matrix $\mathbf{E}$ with probability higher than 2^{-2}; in [13, 75] $\mathbf{E}$ is accepted with probability close to 1. This way we are able to select L more aggressively. We note that the smaller the value L, the higher the probability of acceptance in the signature algorithm (Line 8, first part) becomes. The probability for acceptance is $(1-2L/2^d)^m$. Our choice of parameters yields a concrete acceptance probability in Line 8, first part, of Sign for TESLA-416 of approximately 56.2% and for TESLA-768 of approximately 37%. Note that choosing $\delta_{check} = 0.35 \geq 2^{-2}$, i.e., rejecting 65% of secret key instances, lowers the bit security of TESLA-416 again by two bits. Overall, a signature $(c, \mathbf{z})$ is accepted with probability 35.7% (resp. 21.7%) which is larger than in [75].

Also the standard deviation σ is, considering the bound given in the table, optimized experimentally with respect to performance.

In summary, our parameters yield a secret key (resp. public key) size of 0.66 MB (resp. 1.07 MB) for TESLA-416 and 1.96 MB (resp. 3.5 MB) for TESLA-768. These sizes come off since we rely on lattice problems on standard lattices without introducing any structure being potentially exploited in future. Our instantiation of TESLA-416 gives signature sizes of 1.25 KB which improves the proposal by [13] (albeit their shortest instantiation of 1.39 KB of signature size provides 83 bits of security as pointed out by [75]).

4.2 Generating the Matrix $\mathbf{A}$

Most previous proposals of (R)-LWE-based schemes set the matrix (or, in the context of R-LWE, polynomial) $\mathbf{A}$ as a system parameter that is chosen uniformly at random [13, 27, 44, 74, 75]. Typically, those proposals do not specify what this system parameter actually is, which may raise concerns about back doors in concrete instantiations of the scheme. One solution to this problem is to let each user generate their own parameter as part of the public key as described in the context of R-LWE-based key exchange in [8, Section 3]. However, for schemes using standard lattices such as TESLA, this either incurs a huge overhead in key size or in key-expansion time during signing and verification.

As pointed out by [41, Sec. 3], in reality the matrix $\mathbf{A}$ is most likely generated by some PRG. In the following we describe how we generated the matrix $\mathbf{A}$ for TESLA using the ChaCha8 stream cipher [18], which generates pseudo-random output given a 32-byte key and an 8-byte nonce. More specifically, for TESLA-416, we run ChaCha8 with the fixed 32-byte key "Generate matrix A for TESLA-416!"; for TESLA-768, we run ChaCha8 with the fixed 32-byte key "Generate matrix A for TESLA-768!". To generate row i of $\mathbf{A}$, ChaCha8 is using the nonce i (as 8-byte little-endian integer) to generate $4m$ bytes of output. Those bytes are considered as a sequence of m 4-byte little-endian signed integers. Each of those integers is reduced into the range $[-2^{26}, 2^{26}-1]$ for TESLA-416 and into the range $[-2^{27}, 2^{27}-1]$ for TESLA-768. If the resulting integer is not in the range $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ it is discarded. However, in TESLA-416, it turns out that for the fixed key stated above, no integer needs to be discarded – this is a result of choosing q just below a power of 2.

As discussed in [41], generating $\mathbf{A}$ pseudo-randomly from a short seed may also allow small embedded devices to generate TESLA signatures. They never have to store the whole matrix $\mathbf{A}$ but can generate it on the fly. The software implementations described in Sec. 5 do not make use of the pseudo-random generation of $\mathbf{A}$; they would work at the same speed with any other matrix $\mathbf{A}$.

4.3 Concrete Bit Security of TESLA Against Classical Adversaries

To estimate the hardness of LWE we consider state-of-the-art lattice attacks. There are mainly two families of algorithms for solving LWE. First, there is the decoding attack which goes back to 1986 where Babai [12] proposed the nearest-plane algorithm. Today, improved versions by Linder and Peikert [55] and Liu and Nguyen [56] are used. Here, the given lattice basis is first reduced by a lattice reduction algorithm (for instance, the BKZ algorithm [33]) and then given the nearest plane algorithm (or faster variants) as input to find the closest vector to a target vector.

The second approach is to convert the LWE instance to the (unique) shortest vector instance. This is called the embedding approach. There are two ways to define the underlying lattice for which the solution of the

Table 2 Estimation of the security of TESLA-416 and TESLA-768 against the decoding attack and the embedding approach, comparison of the security our instantiations to the parameter set proposed by Dagdelen et al. [75]

Problem	Attack	Bit Security		
		Dagdelen et al. [75]	TESLA-416	TESLA-768
LWE	Decoding	276	132	286
	Embedding	134	131	257
SIS	Lattice reduction	157	108	234

(unique) shortest vector instance is of interest. Depending on the number of given LWE samples the appropriate lattice is chosen. We refer to [75] for a formal description of the embedding attacks. We estimate the hardness of our chosen LWE instances based on [6, 13, 55]. The resulting bit hardness (estimated by our script `estimationBitSecurity.sage`) of both attacks is shown in Table 2.

Alternatively, one could use non-lattice algorithms to solve the LWE problem. There is, for instance, the algorithm by Blum, Kalai, and Wassermann [24] (BKW). It received a lot of attention in the last years [4, 5, 22, 40, 46, 54]. During the algorithm run one tries to find a transformation matrix $\mathbf{B}$ to introduce more structure to the LWE matrix $\mathbf{A}$. To this end, one needs a large number of LWE samples. Although the number of samples necessary was crucially reduced by Duc et al. [37] and even further by Kirchner and Fouque [49], it is still not applicable (by far) for our instances. The same drawback exists in the algorithm by Arora and Ge [11]. The main idea here is to transform a noisy linear system of equations to a noise-free non-linear system of equations, via linearization [11] or via Gröbner bases [3]. Again, their approach requires far more LWE samples than given in our LWE instance. For this reason, their runtimes are not considered here. We note that asymptotically those algorithms are the fastest algorithms to solve LWE but not yet applicable to current LWE-based signatures.

Another approach is Distinguishing attack [62] which solves the decisional LWE instead of the search LWE. Linder and Peikert state that the Decoding attack is always more efficient than the Distinguishing attack [55]. Hence, we do not consider this attack in our estimation.

Comparison with other LWE Estimation Tools. Recently, online tools to estimate the hardness of LWE became available, e.g., the *LWE-Estimator* by Albrecht et al. [7] and the *LWE security estimation tool* by De Meyer [59]. The latter tool estimates the hardness via the BKW algorithm, the Decoding attack, and the Distinguishing attack. Additionally to the mentioned attacks, Albrecht et al. consider the standard and dual embedding approach. Both tools do not take the given number of LWE samples into account but compute the hardness for different LWE attacks for an optimal number of LWE samples. Hence, estimating the hardness of our parameters with those tools gives less security than estimated by our script `scripts/estimationBitSecurity.sage`[5] which considers the number of given samples, similarly to [75].

4.4 Concrete Bit Security of TESLA Against Quantum Adversaries

To the best of our knowledge, there is no quantum algorithm known to solve learning with errors directly. Instead quantum speedups on the building blocks of aforementioned attacks are investigated.

In most efficient LWE-solvers, the shortest vector problem (SVP) is rather solved on sublattices of lower dimension, also called block size b, than on the lattice defined by an LWE matrix. For instance the BKZ algorithm [34], queries to an SVP-oracle on dimension b. State-of-the-art quantum attacks on LWE mainly make (black-box) use of Grover's quantum search algorithm [43] to speed up classical algorithms. Laarhoven et al. [53] investigate and compare the impact of Grover's quantum search algorithm on different SVP-solvers. In [53] a new, faster quantum-SVP solver with a runtime of

$$time_{\text{q-sieve}}(n) = 2^{0.268n+o(1)}. \tag{2}$$

is proposed. Furthermore, in a recent approach [8] Alkim et al. point out that since enumeration is a backtracking algorithm, it might benefit from applying the quantum algorithm by Montanaro [64].

[5] Our script is available at https://cryptojedi.org/crypto/#tesla

In [8], the *core SVP hardness* is estimated, i.e., the hardness during a single call to an SVP-oracle of a blocksize $b < n$ during BKZ. The authors argue that for their instances the estimations for the quantum enhanced sieving algorithm given in Equation (2) is even faster than an enumeration algorithm with quadratic speed-up.

However, this is not to be expected for our instances. To the best of our knowledge the currently fastest estimations for enumeration are given by Kuo et al. [52][6]. Kuo et al.'s implementation observes a runtime of

$$time_{enum} = \begin{cases} 2^{0.0138b^2-2.2b+93.2}, & \text{if } b \leq 97, \\ 2^{0.0064b^2-0.92b+38.4}, & \text{if } 98 \leq b \leq 104, \\ 2^{0.001b^2+0.034b-2.8}, & \text{if } 105 \leq b \leq 111, \\ 2^{0.00059b^2+0.11b-5.8}, & \text{if } 112 \geq b. \end{cases} \tag{3}$$

Unfortunately, their implementation does not make use of the fastest building blocks known today, such as BKZ 2.0 [33] and Progressive BKZ [9]. Hence, we consider this estimation to be not reliable for choosing parameters anymore. Since BKZ is a building block of the (dual and standard) embedding approach described in Sec. 4.3, we make the (very) conservative assumptions that the best possible embedding attack benefits with a quadratic speed-up from the quantum enhanced enumeration algorithm.

We believe that there is a big gap in the literature in investigating quantum hardness of lattice problems in general. We see a huge potential in this research field and thus pose the challenge to find better quantum algorithms on LWE that would make a revision of our parameters necessary and would lead to more practical parameter choice for 128-bits of security.

Nevertheless, the bit security of TESLA does not only depend on the bit hardness of LWE but also on the security of the used hash function. As stated by Bernstein et al. [20] we have to increase the output length of the hash function to at least 2λ to receive a bit security of λ. However, we already use SHA-256 in both parameter sets TESLA-416 and TESLA-768.

5 Software Implementation

To demonstrate the efficiency of our proposed parameter set we present a software implementation targeting the Intel Haswell microarchitecture. The starting point for our implementation is the software presented by Dagdelen et al., which we obtained from the authors. Our software offers the same level of protection against timing attacks as the software presented in [75]. The software makes use of the fast AVX2 instructions on vectors of 4 double-precision floating-point numbers. The reason that signing and verification is faster with the parameters proposed for TESLA might be obvious from the fact that n, m, and q are smaller than the parameters proposed by Dagdelen et al [75]. However, signing with our parameters needs slightly more attempts to find a valid signature and the slowdown from the increased number of average iterations might be larger than the effect from smaller cost per iteration. The following two modifications to the software proposed by Dagdelen et al. address this issue.

Parallel Matrix-Vector Multiplication. The most costly operation during signing is the computation of $\mathbf{Ay}$, which requires nm multiplications in $\mathbb{Z}_q$, most of those followed by accumulation. Intel Haswell processors can perform two multiply-accumulate instructions on 256-bit vectors of double-precision floating point numbers every cycle. As we represent elements of $\mathbb{Z}_q$ as double-precision floating point numbers, one obtains a lower bound of $nm/8$ cycles per matrix multiplication. This lower bound corresponds to 41600 cycles for TESLA-416 and to 132096 cycles for TESLA-768.

Already Dagdelen et al. [75] pointed out that their actual performance is much lower. The reason is that each coefficient from $\mathbf{A}$ needs to be loaded from (at best) L2 cache, because the whole matrix $\mathbf{A}$ does not fit into the 32KB of level-1 cache. The actual performance of matrix-vector multiplication presented by Dagdelen et al. was a factor of 5 slower than the lower bound.

However, on average, signing computes multiple of those matrix-vector multiplications, all with the constant matrix $\mathbf{A}$. Our software always samples $k = 3$ vectors y, then performs 3 matrix-vector multiplications and then proceeds to investigate whether one of the 3 results is usable for the signature. The disadvantage of this technique is that most of the time we compute some matrix-vector products that are not used in the end. The advantage

[6] The algorithm by Micciancio and Walter [63] has an asymptotic complexity of $2^{O(n\log(n))}$. However, practically Kuo et al. still outperforms the algorithm given in [63] for our instances.

is that matrix coefficients, once loaded from L2 cache are used k times instead of just once. The optimal value of k depends on the parameter set. We wrote scripts that generate an optimized assembly routine for different parameters and different values of k. With those scripts we generated and benchmarked code for many different combinations that all offered the targeted security and then picked the fastest one for TESLA-416 and TESLA-768.

Note that signature verification cannot use the k-times-parallel matrix-vector multiplication for the computation of $\mathbf{Az}$. For this task we use an approach similar to [75].

Lazy Reductions. The coefficients of $\mathbf{A}$ are in the interval $(-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ and coefficients of $\mathbf{y}$ are in the interval $[-B, B]$. For our parameter choices of q and B in TESLA-416, each product of coefficients in the matrix vector multiplication has up to 49 bits. The mantissa of a double-precision floating-point value has 53 bits, so when accumulating $n = 416$ such products in the matrix-vector multiplication we need to reduce modulo q several times. This was no different for the software by Dagdelen et al. [75], which is "overly conservative" and reduces after seven multiply-accumulates. We reduce only after 16 multiply-accumulates. In TESLA-768 the products reach up to 50 bits and so we have to reduce after 8 multiply-accumulates; these frequent reductions account for a large portion of the performance difference between TESLA-416 and TESLA-768.

Resistance Against Timing Attacks. Our implementation offers the same level of side-channel protection as the software presented in [75]. More specifically, the software samples $\mathbf{y}$ in constant time and also the rejection sampling (step 8. in Algorithm Sign of Fig. 1) does not leak information about $\mathbf{y}$. However, the probability for rejecting $\mathbf{w}$ is not independent of the private-key matrix $\mathbf{E}$. An attacker might be able to average timing over many signatures and gain some knowledge about $\mathbf{E}$. The same "high-level" timing leakage is present in the "timing-attack-protected" software presented in [75] and a similar timing leakage is also present in the "timing-attack-protected" software presented in [45].

6 Results and Comparison

Table 3 gives benchmarking results of TESLA-416 and TESLA-768 and compares those benchmarks to state-of-the art results from the literature. As indicated in the table, we obtain all our benchmarks on an Intel Core-i7 4770K (Haswell) processor. We followed the standard practice of disabling Turbo Boost and hyperthreading. The standard practice for benchmarking (cryptographic) software is to compute the median of a large amount of cycle counts. Other than the average, this filters out individual results that were influenced by, for example, an interrupt from an incoming network packet. For TESLA *signing*, considering the median would be overly optimistic, because the rejection sampling creates "outliers" that are actually legitimate part of the computation. Consequently, benchmarks of TESLA for signing are averaged over $100,000$ signatures; benchmarks of TESLA for verification are the median of 100 verifications.

For all software results we report the sizes of keys and signatures actually produced by the software, not the theoretically smallest possible sizes with full compression. We make an exception for BLISS. The authors of the software obviously did not spend any effort on reducing the size of signatures and keys; we report sizes with "trivial" compression through choosing native data types of appropriate sizes. As for most other schemes (including TESLA) it is possible to compress signatures and keys further, but only at the cost of additional computation.

Both our TESLA-416 software and the software presented in [75] use the same construction and target the 128-bit pre-quantum security level. There are two reasons that our software is almost two times faster on the same microarchitecture: First, the reduction to LWE (instead of LWE and SIS) allows us to choose more efficient parameters. Secondly, the parallel matrix-vector multiplication technique offers additional speedup.

The two ideal-lattice-based schemes listed in Table 3 are still faster than TESLA. However, the GLP software from [45] offers less than 80 bits of security and all available BLISS software leaks timing information in the Gaussian sampling, which is unavoidable for signing (or at least expensive) [29]. We expect that optimization of BLISS software can outperform the current results, but it would be very interesting to see how large the penalty is for a timing-attack-protected implementation of BLISS.

In the (as of yet quite small) realm of signatures that offer 128 bits of post-quantum security, TESLA-768 offers an interesting alternative to SPHINCS. Public and secret keys of TESLA-768 are much larger than

SPHINCS keys, but TESLA-768 is much faster for signing, significantly faster for verification, and signatures are more than an order of magnitude smaller.

The post-quantum multivariate-based signature scheme Rainbow5640 [32,36] performs best among all listed schemes but unfortunately, comes with no security reduction to its underlying problem.

Table 3 Overview of state-of-the-art post-quantum signature schemes; sizes are given in byte [B]; the column "ROM?, tight?" states whether the scheme has a security reduction in the random oracle model and whether this reduction is tight; "QROM?, tight?" states the same for the quantum random oracle model; "Security (PreQ)" lists the claimed pre-quantum security level; "Security (PostQ)" lists the claimed post-quantum security level, if available

Scheme/Software	Comp. Assumptions	ROM?, Tight?	QROM?, Tight?	Security (PreQ)	Security (PostQ)	CPU	Size (bytes)		cycles	
Selected signature schemes over standard lattices										
GPV [14, 42]	SIS	yes, yes	yes, yes	100	?	AMD Opteron 8356 (Barcelona)	vk: sk: sig:	24,772,608 11,501,568 28,058	sign: verify:	287,500,000 48,300,000
BG [13, 75]	SIS, LWE	yes, no	-	128	?	Intel Core i7-4770K (Haswell)	vk: sk: sig:	1,619,940 912,380 1,495	sign: verify:	1,203,924 335,072
TESLA-416 (this paper)	LWE	yes, yes	no[a]	128	?	Intel Core-i7-4770K (Haswell)	vk: sk: sig:	1 331 200 1 011 744 1,280	sign: verify:	729 990 247 158
TESLA-768 (this paper)	LWE	yes, yes	no[a]	> 128	128	Intel Core-i7-4770K (Haswell)	vk: sk: sig:	4 227 072 3 293 216 2,336	sign: verify:	2 366 541 873 742
Selected signatures schemes over ideal lattices										
GPV-poly [14, 42]	R-SIS	yes, yes	yes, yes	100	?	AMD Opteron 8356 (Barcelona)	vk: sk: sig:	48,435 24,474 30,822	sign: verify:	71,300,000 9,200,000
GLP [44, 45, 75][b]	DCK	yes, no	-	$75-80$	?	Intel Core i5-3210M (Ivy Bridge)	vk: sk: sig:	1,536 256 1,186	sign: verify:	452,223 34,004
BLISS-BI [38, 39][c,c]	R-SIS, NTRU	yes, no	-	128	?	"Intel Core 3.4GHz"	vk: sk: sig:	7,168 2,048 1,559	sign: verify:	$\approx 358,400$ 102,000
Selected other post-quantum signature schemes										
SPHINCS-256 [20]	Red. to hash collisions, tight Red. to 2nd preimages, not tight Both in the standard model			> 128	128	Intel Xeon E3-1275 (Haswell)	vk: sk: sig:	1,056 1,088 41,000	sign: verify:	51,636,372 1,451,004
Rainbow5640 [32, 36]	MQ, EIP[f]	-	-	80	?	Intel Xeon E3-1275 (Haswell)	vk: sk: sig:	44,160 86,240 37	sign: verify:	42,700[e] 36,072[e]

[a] The auxiliary material of this paper gives a tight security reduction of a *variant* of TESLA in the QROM.
[b] In the benchmarks we include the improvements by Dagdelen et al. presented in [75].
[c] In the benchmarks we include the improvements by Ducas presented in [38].
[d] We report sizes of keys and signatures with "trivial" compression as explained in the text.
[e] Benchmark on Haswell CPU from [21].
[f] The security of Rainbow5640 is based on the Multivariate Quadratic polynomial (MQ) and the Extended Isomorphism of Polynomials (EIP) problem, but no security reduction has been given yet.

Acknowledgement

We thank Dennis Hofheinz and Vadim Lyubashevsky for initial discussions and Marc Fischlin and Florian Göpfert for helpful comments. Furthermore, we thank Gus Gutoski who made us aware of the additional needed check during the signature generation.

References

1. M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly-secure signatures from lossy identification schemes. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, 2012. 2
2. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996)*, pages 99–108. ACM, 1996. 5
3. M. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, and L. Perret. Algebraic algorithms for LWE problems. Cryptology ePrint Archive, Report 2014/1018, 2014. http://eprint.iacr.org/2014/1018/. 14
4. M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, and L. Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74(2):325–354, 2015. 14
5. M. R. Albrecht, J.-C. Faugère, R. Fitzpatrick, and L. Perret. Lazy modulus switching for the BKW algorithm on LWE. In H. Krawczyk, editor, *Public-Key Cryptography*, volume 8383 of *LNCS*, pages 429–445. Springer, 2014. 14
6. M. R. Albrecht, R. Fitzpatrick, and F. Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In H.-S. Lee and D.-G. Han, editors, *Information Security and Cryptology - ICISC 2013*, volume 8565 of *LNCS*, pages 293–310. Springer, 2013. 14
7. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 14
8. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange – a new hope. In *Proceedings of the 25th USENIX Security Symposium*. USENIX Association, 2016. http://cryptojedi.org/papers/#newhope. 2, 13, 14, 15
9. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi. Improved progressive bkz algorithms and their precise cost estimation by sharp simulator, 2016. 15
10. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, 2009. 5
11. S. Arora and R. Ge. New algorithms for learning in presence of errors. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Automata, Languages and Programming*, volume 6755 of *LNCS*, pages 403–415. Springer, 2011. 14
12. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. 13
13. S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In J. Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, 2014. 2, 3, 5, 7, 10, 11, 12, 13, 14, 18
14. R. E. Bansarkhani and J. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In T. Lange, K. Lauter, and P. Lisonek, editors, *Selected Areas in Cryptography*, volume 8282 of *LNCS*, pages 48–67. Springer, 2013. 18
15. G. Barwood. Digital signatures using elliptic curves. message 32f519ad.19609226@news.dial.pipex.com posted to sci.crypt, 1997. http://groups.google.com/group/sci.crypt/msg/b28aba37180dd6c6. 11
16. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93 Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993. 21
17. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996. 2, 21
18. D. J. Bernstein. Chacha, a variant of salsa20. In *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*, 2008. http://cr.yp.to/papers.html#chacha. 13
19. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *LNCS*, pages 124–142. Springer, 2011. http://cryptojedi.org/papers/#ed25519. 11
20. D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In M. Fischlin and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 368–397. Springer, 2015. 2, 15, 18
21. D. J. Bernstein and T. Lange. eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to (accessed 2015-05-19). 18
22. D. J. Bernstein and T. Lange. Never trust a bunny. In J.-H. Hoepman and I. Verbauwhede, editors, *Radio Frequency Identification. Security and Privacy Issues*, volume 7739 of *LNCS*, pages 137–148. Springer, 2013. 14
23. O. Blazy, S. Kakvi, E. Kiltz, and J. Pan. Tightly-secure signatures from chameleon hash functions. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, volume 9020 of *LNCS*, pages 256–279. Springer, 2015. 23
24. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC 2000)*, pages 435–440. ACM, 2000. 14
25. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, 2011. 3, 5, 21, 22

26. D. Boneh and M. Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *LNCS*, pages 361–379. Springer, 2013. 21, 23
27. J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, 2015. http://eprint.iacr.org/2014/599. 13
28. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC 2013)*, pages 575–584. ACM, 2013. 5
29. L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *LNCS*, pages 323–345. Springer, 2016. https://eprint.iacr.org/2016/300/. 16
30. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010. 23
31. S. Chatterjee, A. Menezes, and P. Sarkar. Another look at tightness. In A. Miri and S. Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *LNCS*, pages 293–319. Springer, 2011. 2
32. A. I.-T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E. L.-H. Kuo, F. Y.-S. Lee, and B.-Y. Yang. SSE implementation of multivariate PKCs on modern x86 CPUs. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *LNCS*, pages 33–48. Springer, 2009. 17, 18
33. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, 2011. 13, 15
34. Y. Chen and P. Q. Nguyen. Bkz 2.0: Better lattice security estimates. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, 2011. 14
35. Ö. Dagdelen, M. Fischlin, and T. Gagliardoni. The Fiat-Shamir transformation in a quantum world. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 62–81. Springer, 2013. 3, 5, 21
36. J. Ding and D. Schmidt. Rainbow, a new multivariable polynomial signature scheme. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 164–175. Springer, 2005. 17, 18
37. A. Duc, F. Tramèr, and S. Vaudenay. Better algorithms for LWE and LWR. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 173–202. Springer, 2015. 14
38. L. Ducas. Accelerating bliss: the geometry of ternary polynomials. Cryptology ePrint Archive, Report 2014/874, 2014. http://eprint.iacr.org/2014/874/. 2, 18
39. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013. 2, 11, 18
40. M. Fossorier, M. Mihaljević, H. Imai, Y. Cui, and K. Matsuura. An algorithm for solving the lpn problem and its application to security evaluation of the hb protocols for rfid authentication. In R. Barua and T. Lange, editors, *Progress in Cryptology - INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 48–62. Springer, 2006. 14
41. S. Galbraith. Space-efficient variants of cryptosystems based on learning with errors, 2012. https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf. 13
42. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC 2008)*, pages 197–206. ACM, 2008. 3, 18
43. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996)*, pages 212–219. ACM, 1996. 14
44. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, 2012. 2, 6, 11, 13, 18
45. T. Güneysu, T. Oder, T. Pöppelmann, and P. Schwabe. Software speed records for lattice-based signatures. In P. Gaborit, editor, *Post-Quantum Cryptography*, volume 7932 of *LNCS*, pages 67–82. Springer, 2013. 2, 16, 18
46. Q. Guo, T. Johansson, and P. Stankovski. Coded-bkw: Solving lwe using lattice codes. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *LNCS*, pages 23–42. Springer, 2015. 14
47. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In J. P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998. 2
48. J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *CCS '03 Proceedings of the 10th ACM conference on Computer and communications security*, pages 155–164. ACM, 2003. 2, 7, 11
49. P. Kirchner and P.-A. Fouque. An improved bkw algorithm for lwe with applications to cryptography and lattices. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *LNCS*, pages 43–62. Springer, 2015. 14
50. N. Koblitz and A. J. Menezes. Another look at "provable security". *jcrypto*, 20(1):3–37, 2007. 21
51. H. Krawczyk and T. Rabin. Chameleon signatures. In *Network and Distributed System Security Symposium*. IEEE, 2000. http://eprint.iacr.org/1998/010. 22
52. P.-C. Kuo, M. Schneider, Ö. Dagdelen, J. Reichelt, J. Buchmann, C.-M. Cheng, and B.-Y. Yang. Extreme enumeration on GPU and in clouds – how many dollars you need to break SVP challenges -. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *LNCS*, pages 176–191. Springer, 2011. 15
53. T. Laarhoven, M. Mosca, and J. van de Pol. Finding shortest lattice vectors faster using quantum search. *Designs, Codes and Cryptography*, 2015. Online-first at http://link.springer.com/article/10.1007/s10623-015-0067-5. 14
54. E. Levieil and P.-A. Fouque. An improved lpn algorithm. In R. De Prisco and M. Yung, editors, *Security and Cryptography for Networks*, volume 4116 of *LNCS*, pages 348–359. Springer, 2006. 14
55. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011. 5, 13, 14

56. M. Liu and P. Nguyen. Solving BDD by enumeration: An update. In E. Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, 2013. 13
57. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009. 2
58. V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012. 3, 6, 9
59. L. D. Meyer. Security of lwe-based cryptosystems. Master's thesis, Katholieke Universiteit Leuven, 2015. https://www.cosic.esat.kuleuven.be/LWESecurity/. 14
60. D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In J. H. Silverman, editor, *Cryptography and Lattices*, volume 2146 of *LNCS*, pages 126–145. Springer, 2001. 5
61. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science, 2004. Proceedings.*, pages 372–381. IEEE, 2004. 5
62. D. Micciancio and O. Regev. Lattice-based cryptography. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post Quantum Cryptography*, pages 147–191. Springer, 2009. 14
63. D. Micciancio and M. Walter. Fast lattice point enumeration with minimal overhead. In *SODA '15 Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 276–294, 2015. 15
64. A. Montanaro. Quantum walk speedup of backtracking algorithms. arXiv preprint arXiv:1509.02374, 2016. http://arxiv.org/pdf/1509.02374v2. 14
65. D. M'Raïhi, D. Naccache, D. Pointcheval, and S. Vaudenay. Computational alternatives to random number generators. In S. Tavares and H. Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *LNCS*, pages 72–80. Springer, 1998. 11
66. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC 2009)*, pages 333–342. ACM, 2009. 5
67. C. Peikert. How (not) to instantiate ring-lwe. Cryptology ePrint Archive, Report 2016/351, 2016. http://eprint.iacr.org/2016/351. 2
68. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996. 7
69. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC 2005)*, pages 84–93. ACM, 2005. 5
70. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1997. 2
71. J. Wigley. Removing nedd for rng in signatures. message 5gov5dpad@wapping.ecs.soton.ac.uk posted to sci.crypt, 1997. http://groups.google.com/group/sci.crypt/msg/a6da45bcc8939a89. 11
72. M. Zhandry. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 679–687. IEEE, 2012. 22
73. M. Zhandry. Secure identity-based encryption in the quantum random oracle model. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, 2012. 21
74. J. Zhang, Z. Zhang, J. Ding, M. Snook, and Ö. Dagdelen. Authenticated key exchange from ideal lattices. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 719–751. Springer, 2015. 13
75. Özgür Dagdelen, R. E. Bansarkhani, F. Göpfert, T. Güneysu, T. Oder, T. Pöppelmann, A. H. Sánchez, and P. Schwabe. High-speed signatures from standard lattices. In D. F. Aranha and A. Menezes, editors, *Progress in Cryptology – LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 84–103. Springer, 2015. 2, 5, 6, 7, 12, 13, 14, 15, 16, 18

A TESLA_Q: A Variant of TESLA Secure in the Quantum Random Oracle Model

We have proven the security of the signature scheme TESLA in the random oracle model (ROM) [16]. The random oracle is widely recognized as relevant for security reductions of cryptographic schemes and, in particular, extensively used for real-world cryptosystems, such as the signature schemes RSA-FDH [16, 17] and RSA-PSS [17]. Security in the ROM shows that in order to break the protocol, one must exploit some weaknesses of the hash function [50].

While security reductions in the ROM may be reasonable for classical adversaries, Boneh et al. [25] argue that this model potentially is inappropriate when considering quantum adversaries. In fact, a quantum algorithm could implement the concrete choice of hash function — the instantiation of the random oracle — and run it in superposition on exponentially many inputs. To capture this capability of a quantum adversary, Boneh et al. introduced the quantum(-accessible) random oracle model (QROM). Here, adversaries are allowed to query the random oracle in superposition, simulating the real world capabilities of an adversary. Proofs in the QROM are significantly harder to obtain as common classical proof techniques do not easily transfer to the quantum setting, such as adaptive programmability, preimage awareness, lazy-sampling, and rewinding.

There are a couple of post-quantum schemes proven secure in the ROM which remain secure in the QROM [25, 26, 73]; however, it does not hold in general. In particular, rewinding the adversary is a delicate issue (such as in proofs of signatures derived via the Fiat-Shamir transformation [35]).

In this section, we show that a slight modification of TESLA facilitates the security reduction in the QROM. More precisely, we replace the input of the random oracle by its hash — for which we use a chameleon hash function. A chameleon hash function allows one to find efficiently collisions in its output if he is in possession of a trapdoor. In our security reduction the simulator makes use of such a trapdoor in order to simulate genuine signatures. In fact, by doing so, we are able to show that our security reduction of TESLA can be upgraded to be *history-free*; a sufficient property to claim security in the QROM even based on classical random oracles [25].

We call this modified construction TESLA_Q to emphasize that the construction provides security in the quantum random oracle model as opposed to TESLA. However, we do not rule out the possibility that TESLA could be proven secure against quantum adversaries in QROM, since, intuitively, this additional value is just introduced to overcome a technical dilemma and does not seem to add any security to the scheme. Moreover, we avoid proof techniques which are generally problematic in the quantum setting.

In the following we first recall some essential definitions and statements which will be of necessity to understand the security of TESLA_Q against quantum adversaries, and present its construction and corresponding security reduction afterwards.

A.1 Preliminaries for the Quantum World

Quantum Random-Oracle Model. In the classical random oracle model a classical adversary has access to a random (hash) function $H : \{0,1\}^* \to \{0,1\}^\kappa$ which is model by a classical random oracle O_c. Thus, the adversary gets only the hash value at the classical state it asks for. In case the hash function is replaced by a concrete hash function instead of a random oracle, a quantum attacker can evaluate the concrete hash function on quantum states. To adjust the model to such an adversary Boneh et al. [25] introduce the *quantum(-accessible) random oracle model*. That means, the adversary is allowed to evaluate the random oracle in superposition, i.e., the adversary can submit quantum states $|\phi\rangle = \sum \alpha_x |x\rangle$ to the oracle O_c and receives the hash value $\sum \alpha_x |O_c(x)\rangle$. Note that the quantum-accessible oracle can also be used as a classical oracle. Furthermore, honest parties and algorithms of the signature scheme are still classical and thus, access O_c only via classical bit strings.

History-Free Reduction. The concept of history-free reduction, introduced by Boneh et al. [25], defines a subclass of security reductions for signature schemes in the classical random oracle model which imply security in the quantum model. Informally, history-freeness of a (classical) reduction proof means that the simulated response to a hash or sign query is independent of the (number of) responses to queries before, i.e., it is history-free. Unfortunately, only few signature schemes are known for which there are history-free reductions, and all those schemes follow the hash-and-sign paradigm. In contrast, our signature scheme TESLA_Q has a history-free reduction but follows the Fiat-Shamir transform – thus, does not require any trapdoor known to the signer. Unfortunately, chameleon hash functions involve trapdoors and will have a significant impact on the running time even though the trapdoor is merely used in our construction to simulate successfully signatures in our security reduction.

The following definition [25, Def. 4] captures history-free reductions formally.

Definition 12 (History-free Reduction [25])
A signature scheme $\mathscr{S} = (\mathsf{KeyGen}, \mathsf{Sign}^O, \mathsf{Verify}^O)$ in the classical random oracle model has a history-free reduction from a hard problem $\mathscr{P}$ if there is a proof of security that uses a classical PPT adversary $\mathscr{A}$ for $\mathscr{S}$ to construct a classical PPT algorithm $\mathscr{B}$ for problem $\mathscr{P}$ such that statements (a)-(e) are satisfied:

(a) $\mathscr{B}$ for $\mathscr{P}$ consists of four explicit classical algorithms: $\underline{\mathsf{Start}}$, $\underline{\mathsf{Rand}}^{O_c}$, $\underline{\mathsf{Sign}}^{O_c}$, and $\underline{\mathsf{Finish}}^{O_c}$. The latter three algorithms have access to a shared classical random oracle O_c. These algorithms, except for $\underline{\mathsf{Rand}}^{O_c}$, may also make queries to the challenger for problem $\mathscr{P}$. The algorithms are used as follows:
  (i) Upon input an instance x for problem $\mathscr{P}$, algorithm $\mathscr{B}$ first runs $\underline{\mathsf{Start}}(x)$ to obtain $(\mathsf{vk}, \mathsf{st})$ where vk is a signature verification key and st is private state to be used by $\mathscr{B}$. Algorithm $\mathscr{B}$ sends vk to $\mathscr{A}$ and plays the role of the challenger to $\mathscr{A}$.
  (ii) When $\mathscr{A}$ makes a classical random oracle query to $O(r)$, algorithm $\mathscr{B}$ responds with $\underline{\mathsf{Rand}}^{O_c}(r, \mathsf{st})$. Note that $\underline{\mathsf{Rand}}^{O_c}$ is given the current query as input, but is unaware of previous queries and responses.
  (iii) When $\mathscr{A}$ makes a classical signature query $\mathsf{Sign}(\mathsf{sk}, \mu)$, $\mathscr{B}$ responds with $\underline{\mathsf{Sign}}^{O_c}(\mu, \mathsf{st})$.
  (iv) When $\mathscr{A}$ outputs a signature forgery candidate (μ, σ), the algorithm $\mathscr{B}$ outputs $\underline{\mathsf{Finish}}^{O_c}(\mu, \sigma, \mathsf{st})$.

(b) There is an efficiently computable function $\underline{\mathsf{Instance}}(\mathsf{vk})$ which produces an instance x of problem $\mathscr{P}$ such that $\underline{\mathsf{Start}}(x) = (\mathsf{vk}, \mathsf{st})$ for some st. Consider the process of first generating $(\mathsf{sk}, \mathsf{vk})$ from $\mathsf{KeyGen}(1^\kappa)$, and then computing $x = \underline{\mathsf{Instance}}(\mathsf{vk})$. The distribution of x generated in this way is negligibly close to the distribution of x generated as a real instance to $\mathscr{P}$.

(c) For fixed st, consider the classical random oracle $O(r) = \underline{\mathsf{Rand}}^{O_c}(r, \mathsf{st})$. Define a quantum oracle O_{quant}, which transforms a basis element $|x, y\rangle$ into $|x, y \oplus O(x)\rangle$. We require that O_{quant} is computationally indistinguishable from a random oracle for quantum algorithms.

(d) $\underline{\mathsf{Sign}}^{O_c}$ either aborts (and hence $\mathscr{B}$ aborts) or it generates a valid signature relative to the oracle $O(r) = \underline{\mathsf{Rand}}^{O_c}(r, \mathsf{st})$ with a distribution negligibly close to the correct signing algorithm. The probability that none of the signature queries abort is non-negligible.

(e) If (μ, σ) is a valid signature forgery relative to the public key vk and oracle $O(r) = \underline{\mathsf{Rand}}^{O_c}(r, \mathsf{st})$ then the output of $\mathscr{B}$ (i.e. $\underline{\mathsf{Finish}}^{O_c}(\mu, \sigma, \mathsf{st})$) causes the challenger for problem $\mathscr{P}$ to output 1 with non-negligible probability.

Boneh et al. state that history-free reductions imply security in the quantum settings as in the following theorem:

Theorem 13 ([25]) *Let $\mathscr{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. Suppose $\mathscr{S}$ has a history-free reduction from a problem $\mathscr{P}$. Further, assume that $\mathscr{P}$ is hard for polynomial-time quantum algorithms. Then, $\mathscr{S}$ is secure in the quantum-accessible random oracle model.*

Originally, Boneh et al. assumed also the existence of quantum-accessible pseudorandom functions. This assumption has been made obsolete by Zhandry [72] who presented secure constructions of such functions.

Chameleon Hash Functions. Chameleon hash functions has been introduced by Krawczyk and Rabin [51]. Roughly speaking, chameleon hash functions are like collision-resistant hash functions but come with a trapdoor that allows one to find collisions efficiently if in possession. The following captures the definition of chameleon hash functions formally.

Definition 14 (Chameleon hash functions [26]) A chameleon hash function consists of three PPT algorithms $\mathscr{CH} = (\mathsf{Gen}, \mathsf{CH}, \mathsf{CH}^{-1})$ defined as follows. Upon input the security parameter 1^κ, the probabilistic key generation algorithm Gen outputs a key pair $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa)$ where ek is the evaluation key and td the corresponding trapdoor. The evaluation algorithm CH upon input an evaluation key ek, a message $m \in \mathsf{M}_{\mathsf{ek}}$, and randomness $r \in \mathsf{R}_{\mathsf{ek}}$ outputs $\mathsf{CH}(\mathsf{ek}, m, r) \in \mathsf{Y}_{\mathsf{ek}}$. Chameleon hash functions satisfy the following properties:

Chameleon hash property The function CH^{-1} upon input trapdoor td, messages $m, m' \in \mathsf{M}_{\mathsf{ek}}$, and randomness $r \in \mathsf{R}_{\mathsf{ek}}$ outputs $r' \leftarrow \mathsf{CH}^{-1}(\mathsf{td}, m, m', r)$ with $\mathsf{CH}(\mathsf{ek}, m, r) = \mathsf{CH}(\mathsf{ek}, m', r')$.

Uniform distribution There exists a distribution $\mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$ such that for all $m \in \mathsf{M}_{\mathsf{ek}}$, the distributions $(\mathsf{ek}, \mathsf{CH}(\mathsf{ek}, m, r)$ and (ek, y) are computationally indistinguishable where $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa)$, $r \leftarrow_\$ \mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$ and $y \leftarrow \mathscr{U}(\mathsf{Y}_{\mathsf{ek}})$.

Randomness indistinguishability Let $m, m' \in \mathsf{M}_{\mathsf{ek}}$ be arbitrary, and $r \leftarrow_\$ \mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$. The distribution $\mathsf{CH}^{-1}(\mathsf{td}, m, m', r)$ is negligible close to the distribution $\mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$ where $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa)$.

Collision freeness For any PPT algorithm $\mathscr{A}$, we have

$$\Pr[(m,r) \neq (m',r') \wedge \mathsf{CH}(\mathsf{ek}, m, r) = \mathsf{CH}(\mathsf{ek}, m', r') \mid (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa); (m, r, m', r') \leftarrow \mathscr{A}(1^\kappa, \mathsf{ek})\,] \leq \mathsf{negl}(\kappa)\,.$$

Several constructions of chameleon hash functions based on the lattice problems, such as the SIS assumption, are known [23, 30]. Any of those could be used as an instantiation in TESLA_Q as described in the next subsection.

A.2 The Signature Scheme TESLA_Q

Our signature scheme TESLA_Q is similar to TESLA, but the value $\lfloor \mathbf{Ay} \rceil_{d,q}$ is first input to a chameleon hash function CH with fresh randomness $r \leftarrow_\$ \mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$ before it is hashed (together with the message) by the random oracle. More formally, $\mathsf{TESLA}_Q = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is defined as follows.

KeyGen. The key generation is performs the same as steps as in the key generation algorithm of TESLA as described in Sec. 3.1.In addition, it generates a key pair of the chameleon hash function, $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sets $\mathsf{sk} = (\mathbf{S}, \mathbf{E})$ and $\mathsf{vk} = (\mathbf{T} = \mathbf{AS} + \mathbf{E}, \mathsf{ek})$.

Sign. Upon input secret matrices $\mathbf{S}, \mathbf{E}$ and a message μ the signing algorithm performs the following. First, it samples vectors $\mathbf{y} \leftarrow_\$ [-B, B]^n$ and $r \leftarrow_\$ \mathscr{D}_{\mathsf{R}_{\mathsf{ek}}}$, and computes $\mathbf{v} \leftarrow \mathbf{Ay} \pmod q$. Afterwards, the hash value $c = H(\mathsf{CH}(\lfloor \mathbf{v} \rceil_{d,q}, r), \mu)$ is used to compute $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$ where $\mathbf{c} = F(c)$. Let $\mathbf{w} \leftarrow \mathbf{v} - \mathbf{Ec}$. If $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - L$ for some $i \in \{1, \cdots, m\}$ or $||\mathbf{z}||_\infty > B - U$, then the algorithm restarts; else, it outputs the signature $(\mathbf{z}, c, r)$.

Verify. Upon input the public parameters, a message μ and signature $(\mathbf{z}, c, r)$, it first computes $\mathbf{c} = F(c)$ to obtain $\mathbf{w}' = \mathbf{Az} - \mathbf{Tc} \pmod q$, and returns 1 if $c = H(\mathsf{CH}(\lfloor \mathbf{w}' \rceil_{d,q}, r), \mu)$ and $||\mathbf{z}||_\infty \leq B - U$ are both satisfied; otherwise, it returns 0.

Correctness. It is easy to verify the correctness of TESLA_Q. In the signature algorithm the value $\mathsf{CH}(\lfloor \mathbf{Ay} \rceil_{d,q}, \mathrm{r})$ is hashed instead of $\lfloor \mathbf{Ay} \rceil_{d,q}$ as in TESLA. Since the value $r \in \mathsf{R}_{\mathsf{ek}}$ is part of the signature, the verification algorithm is always able to verify a given valid signature.

A.3 Security Reduction in the Quantum Setting

We show that by incorporating chameleon hash functions into TESLA, we are able to give a history-free reduction from LWE to TESLA_Q. Together with Theorem 13 we conclude that TESLA_Q is a quantum-secure lattice-based signature scheme in the quantum random oracle model. History-freeness is shown in the following lemma.

Lemma 15 *There exists a history-free reduction from $\mathsf{LWE}_{n,m,q,\sigma}$ to the signature scheme $\mathsf{TESLA}_Q = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The tightness gap is equal to the signature scheme TESLA as stated in Theorem 1.*

Proof Assume there is a quantum polynomial-time algorithm $\mathscr{A}$ which forges a valid signature in time $t_{\mathscr{A}}$ and probability $\varepsilon_{\mathscr{A}}$. We show how to construct a PPT algorithm $\mathscr{B}$ which solves the $\mathsf{LWE}_{n,m,q,\sigma}$ problem which internally makes black-box use of algorithm $\mathscr{A}$. That means, $\mathscr{B}$ upon an instance $(\mathbf{A}, \mathbf{T})$ decides whether matrices $\mathbf{A}$ and $\mathbf{T}$ are chosen uniformly random or whether they are of the form $\mathbf{AS} + \mathbf{E} = \mathbf{T}$ for some $\mathbf{S}$ and $\mathbf{E}$ with Gaussian-distributed entries. Following the idea of the classical reduction proof of Theorem 1, the adversary can forge (up to negligible probability) a signature only if a tuple $(\mathbf{A}, \mathbf{T})$ with $\mathbf{AS} + \mathbf{E} = \mathbf{T}$ is given. Furthermore, we show that our reduction is history-free.

A history-free reduction includes five (classical) algorithms $\underline{\mathsf{Start}}$, $\underline{\mathsf{Rand}}^{O_c}$, $\underline{\mathsf{Sign}}^{O_c}$, $\underline{\mathsf{Finish}}^{O_c}$, and $\underline{\mathsf{Instance}}$, as in Definition 12, where $\underline{\mathsf{Rand}}^{O_c}$, $\underline{\mathsf{Sign}}^{O_c}$, and $\underline{\mathsf{Finish}}^{O_c}$ have access to the same classical random oracle O_c.

$\underline{\mathsf{Start}}$: Algorithm $\mathscr{B}$ upon input $(\mathbf{A}, \mathbf{T})$ samples key pairs for the chameleon hash function $(\mathsf{ek}, \mathsf{td}) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$ and defines $\mathsf{vk} = (\mathbf{T}, \mathsf{ek})$ and state $\mathsf{st} = (\mathsf{vk}, \mathsf{td})$. It returns $(\mathsf{vk}, \mathsf{st})$. Matrix $\mathbf{A}$ is set as the global system parameter.

$\underline{\mathsf{Rand}}^{O_c}$: When $\mathscr{A}$ queries $O(h, \mu)$ for some hash value $h \in \mathsf{Y}_{\mathsf{ek}}$, $\mathscr{B}$ responds with $\underline{\mathsf{Rand}}^{O_c}(h, \mu) := O_c(h, \mu)$.

$\underline{\mathsf{Sign}}^{O_c}$: When $\mathscr{A}$ asks the signature oracle on message μ, algorithm $\mathscr{B}$ proceeds as follows. First, $\mathscr{B}$ samples values $\mathbf{v}' \leftarrow_\$ \mathbb{Z}_q^m, \mathbf{r}' \leftarrow_\$ \mathscr{D}_{\mathsf{R_{ek}}}$ and $\mathbf{z} \leftarrow_\$ [-B+U, B-U]^n$ uniformly at random. Afterwards, $\mathscr{B}$ queries its internal oracle O_c on $(\mathsf{CH}(\lfloor \mathbf{v}' \rceil_{d,q}, r'), \mu)$ to obtain $c = O_c(\mathsf{CH}(\lfloor \mathbf{v}' \rceil_{d,q}, r'), \mu)$. Let $\mathbf{w} = \mathbf{Az} - \mathbf{Tc} \pmod q$ where $\mathbf{c} = F(c)$. Algorithm $\mathscr{B}$ responds with signature $\underline{\mathsf{Sign}}^{O_c}(\mu) := (\mathbf{z}, c, r)$ where $r = \mathsf{CH}^{-1}(\mathsf{td}, \lfloor \mathbf{v}' \rceil_{d,q}, \lfloor \mathbf{w} \rceil_{d,q}, r)$.

$\underline{\mathsf{Finish}}^{O_c}$: If $\mathscr{A}$ outputs a forgery $(\mu^*, \sigma^*) = (\mu^*, (\mathbf{z}^*, c^*, r^*))$ in time $t_{\mathscr{A}}$, then $\mathscr{B}$ outputs $\underline{\mathsf{Finish}}^{O_c}(\mu^*, \sigma^*) = 1$, i.e., $\mathscr{B}$ knows with overwhelming probability that $(\mathbf{A}, \mathbf{T})$ is an LWE tuple. Otherwise, $\underline{\mathsf{Finish}}^{O_c}$ outputs 0.

$\underline{\mathsf{Instance}}$: We define $\underline{\mathsf{Instance}}(\mathbf{A}, \mathbf{T}, \mathsf{ek}) := (\mathbf{A}, \mathbf{T})$, with $\underline{\mathsf{Start}}(\underline{\mathsf{Instance}}(\mathbf{A}, \mathbf{T})) = ((\mathbf{A}, \mathbf{T}, \mathsf{ek}), (\mathbf{A}, \mathbf{T}, \mathsf{ek}, \mathsf{td}))$, thus $\underline{\mathsf{Instance}}$ and $\underline{\mathsf{Start}}$ satisfy the required property of Definition 12(b).

We now show that the remaining conditions for history-freeness are also satisfied.

Since $\underline{\mathsf{Rand}}^{O_c}$ returns a value given by the classical random oracle O_c, the output of $\underline{\mathsf{Rand}}^{O_c}$ is completely random and independent distributed, which shows property 12(c). The simulated responses to sign queries do not abort in any case. In fact, the distribution of the simulated signature are identical to genuine signatures by randomness indistinguishability of chameleon hash functions. Note that in the simulation we have $\mathsf{CH}(\lfloor \mathbf{v}' \rceil_{d,q}, r') = \mathsf{CH}(\lfloor \mathbf{Az} - \mathbf{Tc} \rceil_{d,q}, r)$ such that $c = O(\mathsf{CH}(\lfloor \mathbf{Az} - \mathbf{Tc} \rceil_{d,q}, r), \mu) = O(\mathsf{CH}(\lfloor \mathbf{v}' \rceil_{d,q}, r'), \mu) = O_c(\mathsf{CH}(\lfloor \mathbf{v}' \rceil_{d,q}, r'), \mu)$ holds for any message μ. Thus, property 12(d) is satisfied.

Assume $\mathscr{A}$ outputs a valid signature forgery (σ^*, μ^*), where $\mathscr{A}$ did not ask μ^* to her signing oracle. Then, by the reduction proof of Theorem 1, we know that with overwhelming probability, the tuple $(\mathbf{A}, \mathbf{T})$ given as the challenge is an LWE tuple. If no signature is output by $\mathscr{A}$ in time $t_{\mathscr{A}}$, similarly with overwhelming probability it must have been two uniformly sampled matrices $(\mathbf{A}, \mathbf{T})$. Thus, $\mathscr{B}$ solves the LWE problem which shows 12(e). □

Using Theorem 13 and Lemma 15 we know that the signature scheme TESLA_Q is secure in the quantum-accessible random oracle model. We note that the security reduction for TESLA_Q in Theorem 16 is as tight as shown for TESLA.

Theorem 16 *Let* $\mathsf{TESLA}_Q = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be the signature scheme defined in A.2. If* $\mathsf{LWE}_{n,m,q,\sigma}$ *is hard against quantum adversaries, then the signature scheme* TESLA_Q *is unforgeable against adaptively chosen-message attacks in the quantum-accessible random oracle model.*