Sharper Ring-LWE Signatures

Paulo S. L. M. Barreto[1,3⋆], Patrick Longa[2], Michael Naehrig[2], Jefferson E. Ricardini[3⋆⋆], and Gustavo Zanon[3]

[1] University of Washington Tacoma
[2] Microsoft Research
[3] University of São Paulo.
E-mails: pbarreto@{uw.edu,usp.br},{plonga,mnaehrig}@microsoft.com, {jricardini,gzanon}@larc.usp.br

Abstract. We present TESLA♯ (pronounced "Tesla Sharp"), a digital signature scheme based on the R-LWE assumption that continues a recent line of proposals of lattice-based digital signature schemes originating in work by Lyubashevsky as well as by Bai and Galbraith. It improves upon all of its predecessors in that it attains much faster key pair generation, signing, and verification, outperforming most (conventional or lattice-based) signature schemes on modern processors. We propose a selection of concrete parameter sets, including a high-security instance that aims at achieving post-quantum security. Based on these parameters, we present a full-fledged software implementation protected against timing and cache attacks that supports two scheme variants: one providing 128 bits of classical security and another providing 128 bits of post-quantum security.

1 Introduction

One of the simplest, most distinctive and, perhaps, most critical public-key cryptographic constructions is that of digital signatures. They are a fundamental ingredient in every public-key infrastructure (PKI), web of trust or related method of identity certification. Therefore, any system intended to offer large-scale security services (including key agreement, plain or authenticated encryption, and more complex protocols) will have to rely ultimately upon the basic functionality provided by digital signatures.

Given the existence of efficient quantum algorithms for the integer factorization problem (IFP) and for the discrete logarithm problem (DLP) in Abelian groups [36], digital signatures and other cryptosystems whose security stems from the presumed intractability of those problems cannot be expected to remain useful for practical deployment in the medium-to-long term. Coupled with the ever-growing evidence that quantum computers will become technologically

⋆ Supported by CNPq research productivity grant 306935/2012-0.
⋆⋆ Supported by the joint São Paulo Research Foundation (FAPESP)/Intel Research grant 2015/50520-6 "Efficient Post-Quantum Cryptography for Building Advanced Security Applications."

feasible in the foreseeable future [8, 9, 17], this means that cryptographic systems like the widespread RSA [31] and ECC [24, 28] will have to be replaced by alternatives that withstand attacks mounted with the help of quantum computers.

This has led to the announcement of multiple efforts [15, 23] to develop and standardize so-called quantum-resistant or post-quantum cryptosystems. Among these, cryptosystems based on the hardness of certain computational problems on lattices have gained significant importance due to their high flexibility to support a large variety of cryptographic primitives, including encryption, signatures and key agreement protocols. Also, lattice based cryptosystems allow for strong security reductions and, when defined over ideal lattices, offer high performance coupled with relatively small key sizes (e.g., in comparison with cryptosystems based on standard lattices).

Our contributions: In the light of the central role played by digital signatures as a necessary framework underlying the deployment of essentially all large-scale cryptographic services, and the need to obtain and standardize efficient quantum-resistant constructions for that purpose, in this paper we propose a high-performance, high-security digital signature scheme dubbed TESLA♯ (pronounced "Tesla Sharp"). Our proposal is an improvement of the Ring-TESLA scheme [2], a ring variant of the earlier TESLA [4] scheme, which in turn improves upon the schemes by Dağdelen et al. [16] and by Bai and Galbraith [7].

As for Ring-TESLA [2], the security of TESLA♯ stems from the hardness of the Ring Learning with Errors (R-LWE) problem. Our scheme corrects a flaw that was present in all of its predecessors except for the earliest scheme by Lyubashevsky [26] from which all of the above constructions derive, and attains much faster signing and verification. The problem has been fixed in a recent update to TESLA [4] at the cost of an additional rejection condition during signature generation.

Moreover, previous TESLA variants ([2, 4]) make use of global parameters that force the use of a fixed and unique lattice for all the signatures. An attacker can potentially exploit this either by forcing the use of a "weak" lattice instance, which would reduce the security of the whole signature scheme, or by exploiting "all-for-the-price-of-one" precomputation attacks (e.g., see [1]). We propose a variant of TESLA♯ that minimizes the risk of these potential vulnerabilities.

Most lattice-based signature schemes in the literature (including all of the aforementioned constructions as well as GLP [22]) provide parameters only for classical pre-quantum security. In contrast, we target not only the classical setting but, arguably, proper post-quantum security as well. In the classical setting, TESLA♯ could be adopted as a plug-in replacement even for RSA and ECC digital signatures at lower security levels and surpass, in many cases, their efficiency. For the post-quantum parameters, we present experimental evidence that such a long-term secure instantiation of TESLA♯ is still competitive and even beats in some cases the performance of other schemes that do not offer proper post-quantum security.

Interestingly, the above is possible despite the proposal being fully and easily implementable in an isochronous[4] fashion, thereby being arguably far more resistant against side-channel attacks than signature schemes that need to perform Gaussian sampling for signing, particularly the scheme by Lyubashevsky [26] as well as BLISS and its variants [19, 18].

To showcase the performance of TESLA♯, we wrote a reference software implementation in C targeting x64 platforms and supporting *two* different security levels: one variant that offers 128 bits of classical security and another variant that offers 128 bits of post-quantum security (with 256 bits of classical security). Our implementations are secure against timing and cache attacks, and are shown to achieve high-performance even though they do not exploit modern 256-bit wide vector instructions and are only assisted by a few lines of inline assembly. Our software will be made publicly available.

As an additional contribution that may have independent interest, we improve upon the Gaussian sampler that was proposed by Ducas *et al.* [19, Section 6] and propose a simpler, more efficient and isochronous Gaussian sampler. The design is not only easier to protect against timing attacks but also reduces significantly the amount of entropy required to generate each sample. We use this improved Gaussian sampler to speed up the computation of TESLA♯'s key generation.

Organization: The remainder of this document is organized as follows. Section 2 introduces the essential notation and recaps basic notions on rings, lattices, signatures and their security. In Section 3, we describe our proposed signature scheme TESLA♯. This is also where we explain the flaw present in some TESLA♯ predecessor schemes and show how it is avoided in our proposal. A formal security proof in the random oracle model under the Ring Learning with Errors (R-LWE) assumption is presented in Appendix A. In Section 4, we propose two parameter sets that favor secure and efficient implementations of TESLA♯ at both classical (pre-quantum) and post-quantum scenarios. In Section 5, we describe an efficient isochronous Gaussian sampler that is considerably simpler than related constructions in the literature, and we also review our guidelines to obtain a concrete and efficient implementation. We summarize and compare our experimental results in Section 6, and we conclude in Section 7.

[4] We mostly avoid the expression 'constant time' that is often abused in the literature, and use the term 'isochronous' instead. This is to stress that an implementation need not (and, at times, cannot possibly) be properly "constant time," and that what is really required to protect against timing attacks is that execution takes the "same time" (hence the Greek roots, *iso+chronos*) regardless of the private data. By the same token, we adopt the expression "isotopic" rather than "uniform access" to refer to table access that does not depend on private data (but need not be "uniform" in any sense).

2 Preliminaries

The following notation is adhered to throughout the text, except where otherwise indicated. To make this paper more self-contained, we repeat here several concepts and closely follow the notation from [2, 7].

2.1 Notation and definitions

Rounding operators: For $c \in \mathbb{Z}$ and $d \in \mathbb{N}$, $[c]_{2^d}$ stands for the unique integer $c \bmod 2^d$ in the range $(-2^{d-1}, 2^{d-1}]$, and $[\mathbf{v}]_{2^d}$ extends that to every component (or coefficient) of a vector (or polynomial) $\mathbf{v}$. The d-bit modular rounding of c is denoted by $\lfloor c \rceil_d := (c - [c]_{2^d})/2^d$ and the notation is extended to vectors and polynomials in the same manner.

Rings: For any integer q, we write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ for the sake of brevity.

For an integer $k > 0$, let $n = 2^k$. The ring $\mathcal{R}_q := \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ is isomorphic to $\mathbb{Z}_q^n$ as a $\mathbb{Z}$-module by associating the corresponding polynomial coefficients to vector components and vice versa. Therefore, we can identify the ring element uniquely represented by $a_0 + \cdots + a_{n-1}x^{n-1}$ with the coefficient vector $\big(a_0, \ldots, a_{n-1}\big)^T$, and write $\mathbf{a}$ for either representation. As in [2], we also write $\mathcal{R}_{q,[B]} := \{\sum_{i=0}^{n-1} a_i x^i \in \mathcal{R}_q \mid a_i \in [-B, B]\}$ for any non-negative $B \in \mathbb{R}$.

Given a vector $\mathbf{a} = (a_0, \ldots, a_{n-1})^T \in \mathbb{Z}_q^n$, its negacyclic rotation is the vector $\mathrm{rot}(\mathbf{a}) := (-a_{n-1}, a_0, \ldots, a_{n-2})^T \in \mathbb{Z}_q^n$, which is the sequence of coefficients of the ring element $(a_0 + \cdots + a_{n-1}x^{n-1}) \cdot x \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, and its associated negacyclic matrix is $\mathrm{Rot}(\mathbf{a}) := \big(\mathbf{a}, \mathrm{rot}(\mathbf{a}), \mathrm{rot}^2(\mathbf{a}), \ldots, \mathrm{rot}^{n-1}(\mathbf{a})\big)^T \in \mathbb{Z}_q^{n\times n}$.

We denote the product of two ring elements $\mathbf{u}, \mathbf{v} \in \mathcal{R}_q$ as $\mathbf{u} * \mathbf{v}$, and the component-wise product of two vectors $\hat{\mathbf{u}} = (u_0, \ldots, u_{n-1}), \hat{\mathbf{v}} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_q^n$ as $\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = (u_0v_0, \ldots, u_{n-1}v_{n-1})$. This component-wise product typically appears after applying the number theoretic transform (NTT) in $\mathcal{R}_q$ to diagonalize ring elements $\mathbf{u}$ and $\mathbf{v}$, obtaining their so called 'frequency domain' representation (i.e. $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ are typically the eigenvalues of $\mathrm{Rot}(\mathbf{u})$ and $\mathrm{Rot}(\mathbf{v})$, respectively).

Fixed weight encoding of bit strings. Let $\mathbb{B}_\omega^n$ be the set of binary vectors of length n with Hamming weight ω, i.e. $\mathbb{B}_\omega^n = \{\mathbf{v} \in \{0,1\}^n \mid ||\mathbf{v}||^2 = \omega\}$. Analogously, denote by $\mathbb{T}_\omega^n$ the set of signed binary vectors of length n with Hanming weight ω, i.e. $\mathbb{T}_\omega^n = \{\mathbf{v} \in \{-1,0,1\}^n \mid ||\mathbf{v}||^2 = \omega\}$. Let κ be a positive integer. We denote by F a function that maps bit strings of length κ to fixed weight ω vectors in $\mathbb{B}_\omega^n$ or $\mathbb{T}_\omega^n$, e.g. $F : \{0,1\}^\kappa \to \mathbb{T}_\omega^n$. In order, for F to be a useful encoding, which means that F is injective (or almost injective), it is necessary that $2^\kappa \leq |\mathbb{B}_\omega^n| = \binom{n}{\omega}$, or $2^\kappa \leq |\mathbb{B}_\omega^n| = \binom{n}{\omega}2^\omega$, respectively.

Algorithms, distributions and oracles: Let $\mathcal{A}$ be a randomized algorithm. Then, $y \leftarrow \mathcal{A}(x)$ denotes the process of running $\mathcal{A}$ on input x and internally

chosen randomness and assigning the output to y. Further, if $\mathcal{O}$ is an oracle, we write $\mathcal{A}^{\mathcal{O}}$ to indicate that $\mathcal{A}$ has access to $\mathcal{O}$.

For a finite set S the notation $s \leftarrow U(S)$ or $s \xleftarrow{\$} S$, means that an element is sampled uniformly at random from S and assigned to s.

Because of their importance, we particularly define the (discrete) Gaussian distribution and the R-LWE distribution as follows.

Definition 1 (Discrete Gaussian Distribution). *Let $\sigma \in \mathbb{R}$ be positive. The* (centered) discrete Gaussian distribution D_σ *over $\mathbb{Z}$ with standard deviation σ is the unique distribution such that the probability of any $z \in \mathbb{Z}$ is $\rho_\sigma(z)/\rho_\sigma(\mathbb{Z})$, where $\rho_\sigma(z) := e^{-\frac{z^2}{2\sigma^2}}$ and $\rho_\sigma(\mathbb{Z}) := 1 + 2\sum_{z=1}^{\infty} \rho_\sigma(z)$.*

The operation of sampling an integer d with discrete Gaussian distribution D_σ is denoted by $d \leftarrow D_\sigma$. Likewise, the coefficient-wise extension of this operation to vectors, i.e. the sampling of a vector $\mathbf{d} \in \mathbb{Z}^n$ whose components are all distributed according to D_σ is denoted by $\mathbf{d} \leftarrow D_\sigma^n$. For the sake of simplicity, via the identification of polynomials with their coefficient vectors, we also denote by $\mathbf{a} \leftarrow D_\sigma^n$ the sampling of a ring element $\mathbf{a} \in \mathcal{R}$ whose polynomial coefficients are all distributed according to D_σ.

Definition 2 (R-LWE Distribution). *Let $k, q \in \mathbb{N}$ be positive integers, $n = 2^k$, $\mathbf{s} \in \mathcal{R}_q$, and let χ be a distribution over $\mathcal{R}$. The* ring learning with errors (R-LWE) distribution $D_{\mathbf{s},\chi}$ *is defined as the distribution that samples $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q$ and $\mathbf{e} \leftarrow \chi$ and outputs $(\mathbf{a}, \mathbf{a} * \mathbf{s} + \mathbf{e}) \in \mathcal{R}_q \times \mathcal{R}_q$.*

2.2 Security notions

In this section, we define the R-LWE problem, digital signatures and the security model for signatures.

Definition 3 (R-LWE Problem). *Let $k, q \in \mathbb{N}$ be positive integers, $n = 2^k$, and let χ be a probability distribution over $\mathcal{R}$. For a uniform choice $\mathbf{s} \leftarrow U(\mathcal{R}_q)$, the* ring learning with errors (R-LWE) problem *is to distinguish the R-LWE distribution $D_{\mathbf{s},\chi}$ from the uniform distribution $U(\mathcal{R}_q \times \mathcal{R}_q)$. Given $\mathbf{s} \in \mathcal{R}_q$ define the following two oracles:*

- *$\mathcal{O}_{\mathbf{s},\chi}$: return a sample $(\mathbf{a}, \mathbf{b} = \mathbf{a} * \mathbf{s} + \mathbf{e}) \leftarrow D_{\mathbf{s},\chi}$ from the R-LWE distribution,*
- *$\mathcal{U}_{\mathcal{R}_q \times \mathcal{R}_q}$: return a uniformly random sample $(\mathbf{a}, \mathbf{u}) \xleftarrow{\$} \mathcal{R}_q \times \mathcal{R}_q$.*

We write R-LWE$_{n,m,q,\chi}$ for the R-LWE problem with parameters n, q and χ given at most m samples. We say that it is (τ, ε)-hard, if for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ that runs in time τ and makes at most m queries to its oracle, it holds that

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

where the probabilities are taken over the random choice of $\mathbf{s} \xleftarrow{\$} \mathcal{R}_q$, the random choices of the oracles $\mathcal{O}_{\mathbf{s},\chi}$, $\mathcal{U}_{\mathcal{R}_q \times \mathcal{R}_q}$, and the random coins of $\mathcal{A}$.

The R-LWE problem was introduced in [27] together with a (quantum) worst-case to average-case reduction to certain problems over ideal lattices. The assumption that it is hard is called the R-LWE assumption. It has been shown [5] that the problem does not become easier if $\mathbf{s}$ is sampled from the distribution χ. If $\chi = D_\sigma^n$, we write $\mathsf{R\text{-}LWE}_{n,m,q,\sigma}$.

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

- KeyGen: Given a security parameter λ, the probabilistic key generation algorithm creates a key pair $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$, denoted by $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. We call sk the secret signing key and pk the public verification key.
- Sign: On input of a signing key sk and a message $\mu \in \mathcal{M}$, the probabilistic signing algorithm outputs a signature $\sigma \in \mathcal{S}$, denoted by $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$.
- Verify: On input of a verification key pk, a message $\mu \in \mathcal{M}$, and a purported signature $\sigma \in \mathcal{S}$, the verification algorithm returns a bit $b \in \{0, 1\}$, denoted by $b \leftarrow \mathsf{Verify}(\mathsf{pk}, \mu, \sigma)$. We say that the algorithm *accepts* the signature if $b = 1$, otherwise we say that it *rejects* the signature.

Definition 4. *A signature scheme is said to be* (perfectly) correct *if, for every security parameter λ, every key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, every message $\mu \in \mathcal{M}$, every signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$, and every choice of the randomness of KeyGen and Sign, it holds that $\mathsf{Verify}(pk, \mu, \sigma) = 1$.*

We require signature schemes to be (perfectly) correct according to Definition 4. The standard security requirement for signature schemes is existential unforgeability under chosen-message attack (EUF-CMA). Figure 1 shows the corresponding security game for an adversary $\mathcal{A}$ against a signature scheme Σ. The algorithm $\mathcal{A}$ has access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$ that, on input of a message $\mu \in \mathcal{M}$, returns a valid signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$. We prove security of TESLA♯ in the random oracle model [11], which means that in the EUF-CMA game, $\mathcal{A}$ has access to a random oracle $\mathcal{H} : \mathcal{R}_q \times \mathcal{R}_q \times \mathcal{M} \to \{0, 1\}^\kappa$.

Definition 5. *Let Σ be a signature scheme. We say that Σ is $(\tau, \varepsilon, q_\mathcal{H}, q_\mathcal{S})$-unforgeable under a chosen-message attack if every adversary $\mathcal{A}$ that runs in time τ and poses at most $q_\mathcal{S}$ queries to the signing oracle and $q_\mathcal{H}$ queries to the random oracle has advantage*

$$\mathrm{Adv}_\Sigma^{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr\left[\mathrm{Game}_{\Sigma,\mathcal{A}}^{\mathsf{EUF-CMA}} = 1\right] \leqslant \varepsilon.$$

Definition 6 (EUF-CMA security [21]). *Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and let $\mathcal{A}$ be a probabilistic algorithm. We say that it $(\tau, \varepsilon, q_\mathcal{H}, q_\mathcal{S})$-breaks Σ if it runs for at most τ steps, makes at most $q_\mathcal{H}$ adaptive queries to a hash function oracle, and at most $q_\mathcal{S}$ queries to a signing oracle for the signature of messages $\mu_i \in \mathcal{M}$, $1 \leq i \leqslant q_\mathcal{S}$, of its choice, and then*

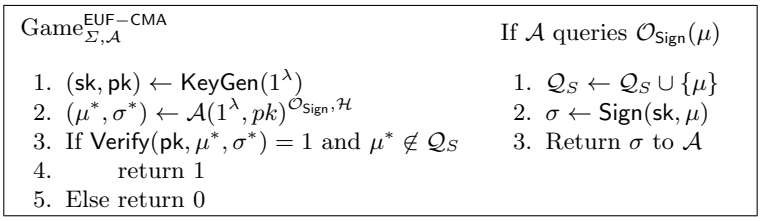

Fig. 1. The EUF-CMA game for an adversary $\mathcal{A}$ against a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ in the random oracle model (i.e. all parties including $\mathcal{A}$ have access to a public function $\mathcal{H}$ with uniformly distributed output).

outputs a forged signature σ^* on some other message $\mu^* \in \mathcal{M}$, $\mu^* \neq \mu_i$ for all $1 \leq i \leqslant q_{\mathcal{S}}$, with probability at least ε. The probability is taken over the coins $\mathcal{A}$ tosses, the KeyGen and Sign algorithms of the signature scheme, and the hash function oracle $\mathcal{H}$.

A signature scheme is said to be $(\tau, \varepsilon, q_{\mathcal{H}}, q_{\mathcal{S}})$-EUF-CMA-secure if no adversary $\mathcal{A}$ can $(\tau, \varepsilon, q_{\mathcal{H}}, q_{\mathcal{S}})$-break it.

3 The proposed TESLA♯ scheme

Our proposal is the R-LWE-based digital signature scheme specified by Algorithms 1, 2, and 3. These algorithms continue the line of research originating in Lyubashevsky's work [26] with various modifications and improvements by Bai and Galbraith [7], Dağdelen et al. [16], Alkim et al. [4], and Akleylek et al. [2]. In particular, our algorithms are direct modifications of the Ring-TESLA algorithms from [2]. In this section, we comment on the specific changes and their consequences.

Algorithm 1 TESLA♯ Key Pairs

INPUT: Parameters n, q, σ, and two uniform, invertible ring elements $\mathbf{a_1}$ and $\mathbf{a_2}$.
OUTPUT: A private key $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2})$ and a public key $(\mathbf{t_1}, \mathbf{t_2})$.

1: $\mathbf{s} \xleftarrow{\$} D^n_\sigma$
2: $\mathbf{e_1} \xleftarrow{\$} D^n_\sigma, \quad \mathbf{e_2} \xleftarrow{\$} D^n_\sigma$
3: $\mathbf{t_1} \leftarrow \mathbf{a_1} * \mathbf{s} + \mathbf{e_1}, \quad \mathbf{t_2} \leftarrow \mathbf{a_2} * \mathbf{s} + \mathbf{e_2}$
4: **return** $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2})$, $(\mathbf{t_1}, \mathbf{t_2})$

Simplification to KeyGen. The key generation algorithm is simplified to the pure generation of two samples from the R-LWE distribution. Specifically, the components $\mathbf{e_1}$ and $\mathbf{e_2}$ of the private key (generated in line 2 of Algorithm 1) are simple Gaussian samples. In contrast with the corresponding components in the TESLA♯ predecessor schemes [2, 4, 7], these ring elements are not filtered to

ensure that the sum of their ω largest absolute coefficients is upper-bounded by a certain constant L. This not only speeds up the key generation substantially, but also eliminates the need to maintain a priority queue, which is difficult to implement in an isochronous, isotopic fashion.

Algorithm 2 TESLA♯ Signing

INPUT: public parameters n, q, ω, d, B, U, κ; base $(\mathbf{a_1}, \mathbf{a_2}) \in \mathcal{R}_q^2$; message $\mu \in \{0,1\}^*$; private key $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2}) \in \mathcal{R}_q^3$; hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$, nearly injective mapping $F : \{0,1\}^\kappa \to \mathbb{T}_\omega^n$.
OUTPUT: A signature $(\mathbf{z}, c)$.

repeat
  $\mathbf{r} \xleftarrow{\$} [-B, B]^n$
  $\mathbf{v_1} \leftarrow \mathbf{a_1} * \mathbf{r}, \quad \mathbf{v_2} \leftarrow \mathbf{a_2} * \mathbf{r}$
  $c \leftarrow H(\lfloor \mathbf{v_1} \rceil_d, \lfloor \mathbf{v_2} \rceil_d, \mu)$
  $\mathbf{c} \leftarrow F(c)$
  $\mathbf{z} \leftarrow \mathbf{r} + \mathbf{s} * \mathbf{c}$
  $\mathbf{w_1} \leftarrow \mathbf{v_1} - \mathbf{e_1} * \mathbf{c}, \quad \mathbf{w_2} \leftarrow \mathbf{v_2} - \mathbf{e_2} * \mathbf{c}$
until $\|\mathbf{z}\|_\infty \leqslant B - U$
  and $\lfloor \mathbf{w_1} \rceil_d = \lfloor \mathbf{v_1} \rceil_d$ **and** $\lfloor \mathbf{w_2} \rceil_d = \lfloor \mathbf{v_2} \rceil_d$
return $(\mathbf{z}, c)$

Algorithm 3 TESLA♯ Verification

INPUT: Public parameters n, q, ω, d, κ; base $(\mathbf{a_1}, \mathbf{a_2}) \in \mathcal{R}_q^2$; message $\mu \in \{0,1\}^*$, public key $(\mathbf{t_1}, \mathbf{t_2}) \in \mathcal{R}_q^2$, signature $(\mathbf{z}, c)$, hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$, nearly injective mapping $F : \{0,1\}^\kappa \to \mathbb{T}_\omega^n$.
OUTPUT: $\{0,1\}$ ▷ reject, accept

$\mathbf{c} \leftarrow F(c)$
$\mathbf{w_1} \leftarrow \mathbf{a_1} * \mathbf{z} - \mathbf{t_1} * \mathbf{c}, \quad \mathbf{w_2} \leftarrow \mathbf{a_2} * \mathbf{z} - \mathbf{t_2} * \mathbf{c}$
$c' \leftarrow H(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$
return (**if** $c' = c$ **and** $\|\mathbf{z}\|_\infty \leqslant B - U$ **then** 1 **else** 0)

Verification failures with previous rejection conditions in Sign. Similarly, during signature generation, our proposal makes no attempt at ensuring that the absolute coefficients $|[\mathbf{w_i}]_{2^d}|$ of the "rounded modular nonces" are upper-bounded by $2^{d-1} - L$, as the predecessor schemes did. Apparently, the purpose of this test was to cause the rounded nonces computed during verification to match the corresponding rounded nonces computed during signing, which is necessary since those rounded values are part of the input to the hash function.

It turns out that the above test does *not* prevent some genuine signatures from failing to verify. In other words, it is possible (and observable in practice)

that the rounded $\mathbf{w_i}$ nonces computed by the verifier satisfy $|[\mathbf{w_i}]_{2^d}| < 2^{d-1} - L$, yet they do *not* match the rounded $\mathbf{v_i}$ nonces computed by the signer, i.e. $\lfloor \mathbf{w_i} \rceil_d \neq \lfloor \mathbf{v_i} \rceil_d$. This points to the fact that this test alone is not suitable to ensure that those schemes are (perfectly) correct as required by Definition 4. Appendix B describes concrete instances for which the test fails and gives more details about the nature of the problem.

Simplification of the rejection condition in Sign. Instead of the above, we resort to the exact condition (line 9 of Algorithm 2) that the actual input to the hash function is the same at signing and at verification, i.e. $\lfloor \mathbf{w_i} \rceil_d = \lfloor \mathbf{v_i} \rceil_d$. As a bonus, the new signature consistency test (whose running time is dominated by the check on $\mathbf{z}$) attains a success rate of $\approx e^{-1/\beta}$ where β is a 'speed' parameter, typically close to $1 + e$ that leads to signing success rates above ≈ 0.75. This is much better than the predecessor schemes, where signing success rates of ≈ 0.4 or less are common. These modifications make the parameter L of the predecessor schemes useless, and indeed it does not occur anywhere in our scheme.

Remark 1. Dağdelen et al. note in Section 3.2 of [16] the possibility of simplifying the rejection condition during signing by checking the modular rounded values of the nonces for equality. However, they conclude that such a modification would be incompatible with the security proof by Bai and Galbraith [7] and discard it.

As far as we can see, the security proof for TESLA♯ given in Appendix A does not incur the problem mentioned by Dağdelen et al. Indeed, the extra condition on the absolute values of the modular rounded nonces is essential to the Bai-Galbraith sequence of games because it seeks a reduction to the search-SIS problem [7, Lemma 6], but becomes irrelevant in the corresponding TESLA♯ sequence of games, which establishes a reduction to the R-LWE problem instead.

Remark 2. We note that the most recent update to the TESLA paper [4] has added an additional condition to the signing algorithm. Namely, it is now tested that $|w| < \lfloor q/2 \rfloor - L$ for all coefficients w of $\mathbf{w}_1$ and $\mathbf{w}_2$. If this is not the case, the signature algorithm restarts. The discussion in Appendix B.1 shows that this fixes the flaw described here and thus achieves perfect correctness at the price of an additional check. This solution increases the complexity of the scheme, it decreases the signing success rate, and also makes it necessary to keep the filtering of Gaussian samples during key generation, which is a lot more difficult to implement in a side-channel resistant fashion than our simpler solution, which speeds up key generation considerably.

Fixed weight encoding of hash values. The Sign algorithm computes a hash value c of length κ in Line 4 of Algorithm 2. For encoding c as a fixed weight vector, we use an encoding function $F : \{0,1\}^\kappa \to \mathbb{T}_\omega^n$ using signed binary vectors of fixed weight ω (see Section 5.3).

The role of Gaussian sampling. TESLA♯ inherits the property from Ring-TESLA that Gaussian sampling is only needed for key pair generation. The signing process itself only needs uniform sampling. This favors high-performance isochronous signing. Although isochronous key generation incurs a heavier computational cost, it is also carried out far less frequently than signing. Yet, neither is such a sampler difficult to implement nor is its cost prohibitively high, as we will show in Section 5.1.

Sampling fresh public constants $(\mathbf{a_1}, \mathbf{a_2})$ per key pair. TESLA♯ makes use of a pair $(\mathbf{a_1}, \mathbf{a_2})$ of public, invertible ring elements, which in our description are independent from the public keys. However, they can be easily generated already in the frequency domain: hash a seed (which may be e.g. the user identifier and a timestamp) into the sequence of their eigenvalues rather than their coefficients and make sure none of them is zero, so that the ring elements are themselves invertible.

In this light, a fresh pair $(\mathbf{a_1}, \mathbf{a_2})$ could be generated for each key pair from information that is kept as part of the public key. The security benefits of doing this, namely, forcing adversaries to face a distinct lattice for each new key pair rather than optimizing an attack against a single lattice that is shared by several users, more than make up for the relatively small loss in performance. In Appendix C, we present a modified version of TESLA♯ with this additional security measure. In our experiments (see Table 3), regenerating the pair $(\mathbf{a_1}, \mathbf{a_2})$ using a 32-byte seed and the ChaCha20 pseudo-random function incurs a slowdown below 15% and 20% during signing and verification, respectively. Therefore, we recommend that users of TESLA♯ benefit from this distinctive, advantageous feature whenever possible.

Security proof. Finally, TESLA♯ is EUF-CMA secure under the R-LWE assumption in the random oracle model. The proof is in Appendix A and closely follows the corresponding proofs of the Ring-TESLA and Bai-Galbraith schemes [2, 7].

4 Choosing parameters

In this section, we provide *two* parameter sets for instantiating TESLA♯ signatures: one pre-quantum parameter set at the 128-bit classical security level, and one post-quantum set at (presumably) the 128-bit post-quantum security level, with 256-bit classical security.

We note that the parameters adopted in the predecessors to TESLA♯ appear to be either largely generic or not suitable for implementations that are intended to offer strong security against quantum attacks. Next, we lay out our reasoning for choosing the TESLA♯ parameters.

Dimension n, modulus q and Gaussian standard deviation σ. In order to use the standard techniques for the number theoretic transform, the dimension n is restricted to being a power of 2, i.e. $n = 2^k$ for some $k > 0$.

We choose the modulus q to be a prime of the form $q = 2^r - 2^s + 1$ for some r and s, so as to enable efficient modular reduction on constrained (8-bit and 16-bit) platforms. This also helps to speed up uniform sampling modulo q via rejection sampling from the uniform distribution modulo 2^r, as needed to generate the lattice base $(\mathbf{a_1}, \mathbf{a_2})$ directly in the frequency domain (as mentioned before, by specifying their eigenvalues rather than their coefficients, thereby speeding up the inspection of those ring elements for invertibility). Also, we require $q \equiv 1 \pmod{2n}$ in order to have the right roots of unity modulo q for the number theoretic transform.

The standard deviation σ for the Gaussian distribution is an integer multiple $\sigma = \xi\sigma_2$ of the constant $\sigma_2 := 1/\sqrt{2\ln 2}$ to enable efficient Gaussian sampling. This condition is used in the Gaussian sampling technique by Ducas et al. [19], which is the basis for our Gaussian sampler described in Section 5.1 below.

In order to estimate the concrete security level of the above parameters, we followed the thorough methodology by Albrecht *et al.* [3]. Although their work focuses on the general LWE problem, no asymptotically faster attack is presently known against R-LWE (only linear factor speedups are currently obtainable due to the ring structure through sieving algorithms). Besides, the analysis can be easily reproduced by using the public script the authors made available online[5].

Hash output size. Finally, the hash size κ is taken to be the classical security level λ itself. This is consistent with the use of the hash in a Fiat-Shamir-style scheme where preimage resistance is relevant while collision resistance is much less so: in a pre-quantum scenario a λ-bit hash is expected to have preimage resistance $\approx 2^\lambda$, but only $\approx 2^{\lambda/2}$ in a post-quantum setting.

Parameters ω, B and U and rounding parameter d. The dimension n and hash length κ are chosen according to the desired security level. Once they are fixed, the weight parameter ω can be chosen such that the function $F : \{0,1\}^\kappa \to \mathbb{T}_\omega^n$ can be injective. This means that $2^\kappa \leq 2^\omega \binom{n}{\omega}$.

The parameter U is defined as $U = \lfloor \eta\sqrt{\omega}\sigma \rceil$ where η is chosen so that the probability of n independent $\eta\sigma$ events is $\approx 2^{-\lambda}$ for the (classical) security level λ. This ensures that the coefficients of the secret key $\mathbf{s}$ will not leak through the signature component $\mathbf{z}$ beyond the target security level.

The minimum restriction between B and U for the security reduction to hold is $B \geqslant nU$. To facilitate uniform sampling modulo B (as needed for signing) via rejection sampling, we want it to be as close as possible to the next power of 2. Because the security proof also requires $q \geqslant 4B$, we specifically take $B = \lfloor q/4 \rfloor \approx \beta nU$, where β is a small "speed" parameter: the probability that $\|\mathbf{z}\|_\infty < B - U$ is $\left(\frac{2(B-U)+1}{2B+1}\right)^n \approx e^{-1/\beta}$. Therefore, the larger the chosen β, the higher the signing success rate and hence the faster the signing (with diminishing returns), and also the larger the signatures. The best trade-off (the 'elbow' of the curve $e^{-1/\beta}$ for $\beta \geqslant 1$) is around $\beta \approx e$, so in practice we keep β close to that value.

[5] <https://bitbucket.org/malb/lwe-estimator/src>.

The parameter d is simply $\lceil \lg B \rceil = \lceil \lg q \rceil - 2$.

Parameter sets. Overall, taking into account the above choices and relations between the different parameters, we suggest the following two parameter sets.

1. **TESLA♯–I:** 128-bit classical security, $n = 512$, $q = 2^{26} - 2^{16} + 1$, $\sigma = 158\sigma_2$, $\omega = 19$, $\eta \approx 13.91$ and $\beta = 4$ (and hence $d = 24$, $B = 2^{24} - 2^{14} = 16760832$, $U = 8137$): signatures are 1616 bytes long, private keys are 2112 bytes long, public keys are 3328 bytes long.
2. **TESLA♯–II:** 128-bit post-quantum security, 256-bit classical security, $n = 1024$, $q = 2^{28} - 2^{16} + 1$, $\sigma = 164\sigma_2$, $\omega = 37$, $\eta = 19.30$ and $\beta = 4$ (and hence $d = 26$, $B = 2^{26} - 2^{14} = 67092480$, $U = 16349$): signatures are 3488 bytes long, private keys are 4224 bytes long, public keys are 7168 bytes long.

5 Implementation aspects

This section details several implementation aspects of TESLA♯. We discuss the improved Gaussian sampler, polynomial multiplication via the number theoretic transform (NTT), and implementation choices like hash and extendable-output functions as well as the integration of the NTT into the signature scheme.

5.1 An isochronous Gaussian sampler

In this section, we improve upon the Gaussian sampler that was proposed by Ducas *et al.* [19, Section 6]. We propose a simpler, more efficient, and isochronous Gaussian sampler that offers protection against timing attacks and requires less entropy per sample.

The basic idea of Ducas *et al.* [19, Algorithms 10–12] is to start from a distribution that approximates the desired Gaussian much more closely than a plain uniform distribution, namely, a stepwise uniform distribution where the steps have width ξ and height distributed according to an efficiently implementable, ξ-scaled binary Gaussian. From there, a high-quality Gaussian is obtained by rejection sampling guided by Bernoulli distributions $\mathcal{B}_c$ with parameters c related to the σ parameter of the desired Gaussian, as well as to the particular segment of the stepwise approximation. Ducas *et al.* implement those Bernoulli distributions by decomposing them into ℓ certain base distributions $(\mathcal{B}_{c_0}, \mathcal{B}_{c_1}, \ldots, \mathcal{B}_{c_{\ell-1}})$ where the c_k constants are precomputed to the desired accuracy, and then sampling from those base distributions to that accuracy. This Bernoulli decomposition is rather involved as it requires heavy algebraic manipulation of distributions, and although the result is reportedly quite efficient, its running time is highly dependent on the private bits, which are sampled one at a time and individually influence the sampling termination. Besides, each $\mathcal{B}_{c_k}$ must be sampled to the same precision as the target distribution, so the total amount of entropy needed to obtain one Gaussian sample is far higher than theoretically necessary, roughly $O(\ell\lambda)$ bits rather than $O(\lambda)$ for security level λ.

However, because we only need a basic Gaussian sampler for key generation, we are able to obtain a much simpler construction. Specifically, only one kind of Bernoulli distribution is ever needed in our case[6], namely, a distribution $\mathcal{B}_{\exp(-t/2\sigma^2)}$ and t is an ℓ-bit integer. This means that we could simply compute the bias $c = \exp(-t/2\sigma^2)$ directly using well known isochronous exponentiation techniques that are commonplace for (say) DLP-based cryptographic schemes. The value c is an approximation of a real number between 0 and 1 to the desired precision. Because floating point support in hardware at the desired precision may be unavailable for the higher security levels, (isochronous, isotopic) integer approximations may be necessary according to the underlying processor. Yet, the required precision for the λ security level is arguably about $\lambda/2$ bits [32], so this can be done with available floating point arithmetic at pre-quantum security levels on most processors.

Having that bias, a single uniform sample to the desired precision is necessary, plus one uniform sample modulo ξ and one binary Gaussian sample that can be obtained by cumulative distribution table (CDT) lookup by always scanning over the whole table (to ensure the process is isochronous and isotopic). The total entropy consumption is thus $O(\lambda)$ bits.

5.2 NTT computation

Efficient instantiations in the context of R-LWE exploit the number theoretic transform (NTT) to speed up polynomial multiplication of ring elements in $\mathcal{R}_q$. In the case of TESLA♯, key generation (Algorithm 1), signing (Algorithm 2), and verification (Algorithm 3) require one NTT computation and two inverse number theoretic transform (NTT^{-1}) computations each. Hence, improvements to the NTT should have a noticeable impact on the proposed signature scheme.

There is plenty of literature focused on the optimization and efficient implementation of the NTT. Below, we describe an efficient yet compact implementation approach.

The plain number theoretic transform is a map NTT on $\mathcal{R}_q := \mathbb{Z}_q[x]/\langle x^n - 1\rangle$, mapping to vectors in $\mathbb{Z}_q^n$ such that $\text{NTT}(\mathbf{a} * \mathbf{b}) = \text{NTT}(\mathbf{a}) \cdot \text{NTT}(\mathbf{b})$ and $\text{NTT}(\mathbf{c}^{-1}) = \text{NTT}(\mathbf{c})^{-1}$ for any $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q$ and any invertible $\mathbf{c} \in \mathcal{R}_q$, with component-wise products and inversions on the NTT outputs. The NTT requires the existence of a primitive n-th root of unity $\omega \in \mathbb{Z}_q$.

The usual technique to obtain a map NTT' on $\mathcal{R}'_q := \mathbb{Z}_q[x]/\langle x^n+1\rangle$ satisfying the same properties as the plain NTT is to choose a primitive $2n$-th root of unity $\zeta \in \mathbb{Z}_q$, set a primitive n-th root of unity as required by the plain NTT as $\omega := \zeta^2$, then define an auxiliary map $\psi(\mathbf{a}) = \psi(a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1}) := (a_0 + a_1 x\zeta + a_2 x^2\zeta^2 + \cdots + a_{n-1}x^{n-1}\zeta^{n-1})$, and finally set $\text{NTT}' := \text{NTT} \circ \psi$.

At first sight, implementing NTT' would require storing a table for the powers of ω (to compute NTT) and another for their inverses (to compute NTT^{-1}), plus

[6] The BLISS scheme, for which that sampler was designed, has the additional complication that Bernoulli distributions with inverse hyperbolic cosine biases $\mathcal{B}_{1/\cosh(t/f)}$ are needed as well.

an extra table for the powers of ζ (to compute ψ) and another for their inverses (to compute ψ^{-1}). It turns out one can implement NTT' while storing only one table, namely the powers of ζ, without any substantial performance loss. This is the result of the following observations.

First, the NTT is not far from being a scaled involution. More precisely, let ρ denote the map:

$$\rho(a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}) := a_0 + a_{n-1} x + a_{n-2} x^2 + \cdots + a_1 x^{n-1},$$

and notice that $\rho^{-1} = \rho$. Then $\mathrm{NTT}^2 = n \circ \rho$, or $\mathrm{NTT}^{-1} = n^{-1} \circ \rho \circ \mathrm{NTT}$, so that a multiplication by $n^{-1} \bmod q$ and a cyclic coefficient permutation enable implementing NTT^{-1} using the NTT implementation itself and very simple extra computations.

Second, a related property holds for ψ. Specifically, let ϕ denote the map:

$$\phi(a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}) := (a_0 - a_1 x - a_2 x^2 - \cdots - a_{n-1} x^{n-1}),$$

and notice that $\phi^{-1} = \phi$. Then $(\rho \circ \psi)^2 = \phi$, or $\psi^{-1} = \phi \circ \rho \circ \psi \circ \rho$, so that simple sign changes and a cyclic coefficient permutation enable implementing ψ^{-1} using the ψ implementation itself and very simple extra computations.

Combining both observations above one gets the following relation: $\mathrm{NTT}'^{-1} = (\mathrm{NTT} \circ \psi)^{-1} = \psi^{-1} \circ \mathrm{NTT}^{-1} = (\phi \circ \rho \circ \psi \circ \rho) \circ (n^{-1} \circ \rho \circ \mathrm{NTT}) = n^{-1} \circ \phi \circ \rho \circ \psi \circ \mathrm{NTT}$. In other words, one can compute NTT'^{-1} using existing implementations of NTT and ψ.

Finally, we remark that the j-th power of ω coincides with the $2j$-th power of ζ, so if one defines a table $\texttt{phi\_tab}[j] := \zeta^j$, then $\texttt{phi\_tab}[2j] = \omega^j$, i.e., the powers of ω are contained in the even-indexed entries of a table of powers of ζ.

Yet, one may argue that, although the computation of $\phi \circ \rho$ is lightweight, multiplying by $n^{-1} \bmod q$ is not, so it would still be necessary to keep an extra table for (say) $n^{-1} \circ \phi \circ \psi$, noticing that ρ commutes with ϕ and n^{-1} so $\mathrm{NTT}'^{-1} = \rho \circ (n^{-1} \circ \phi \circ \psi) \circ \mathrm{NTT}$. However, that multiplication can be combined with other operations in several circumstances. For instance, in a convolution $\mathbf{a} * \mathbf{b} = \mathrm{NTT}^{-1}(\mathrm{NTT}(\mathbf{a}) \cdot \mathrm{NTT}(\mathbf{b}))$ it could be combined with Barrett reduction in the component-wise multiplications involved in computing $\mathrm{NTT}(\mathbf{a}) \cdot \mathrm{NTT}(\mathbf{b})$.

5.3 Implementation choices

We have implemented TESLA♯ in the C language with very scarce use of inline assembly and intrinsics, which were limited to implementing Barrett reduction [10] and vectorized ring additions. This was done in order to facilitate the portability of the implementation to other platforms and to offer a first-order evaluation of TESLA♯'s performance.

We now review the actual choices that accompany our reference implementation. Because many primitives involved are not part of the signature scheme itself, many other settings would be equally possible, e.g., different hash functions could substitute for the ones we report herein.

We adopted ChaCha20 [12, 30, 29, 25] as the underlying pseudo-random bit generator due to its widespread availability and efficiency. We also adopted BLAKE2 [6, 34] and the SHAKE extendable-output functions derived from SHA-3 [20] for the hash functions needed by the signature scheme.

For the encoding of hash values as sparse ring elements of constant weight, we chose Sendrier's method [35]. Although the more elaborate Biswas-Sendrier method [13] is reported to outperform Sendrier's method in certain scenarios, its performance impact within TESLA♯ would be limited: our experiments show that the overall performance improvement is very small, even when completely eliminating the cost of encoding (e.g. by setting the sparse polynomial $\mathbf{c}$ to a constant). Since this encoding never involves private data (specifically, it merely changes the representation of the signature component c, which is entirely public), it does not need to be isochronous nor isotopic.

Ring products are usually carried out using the number theoretic transform (NTT). This requires keeping the base elements $(\mathbf{a_1}, \mathbf{a_2})$ in the form of their eigenvalues, as they are never used in the protocol except as part of products.

However, the multiplications by $\mathbf{c}$ (lines 6 and 7 of Algorithm 2, and line 2 of Algorithm 3) benefit from sparse product techniques. In particular, one can resort to key-scheduling the private key $\mathbf{s}$ into a precomputed table of form $(-\mathbf{s}, \mathbf{s}, -\mathbf{s})$ and length $3n$, so that the negacyclic structure of the underlying ring maps the multiplication by $\pm x^j$ (all nonzero terms of $\mathbf{c}$ have this form) to an addition/subtraction of n table entries starting at offset j on that table. A similar observation holds for $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{t_1}$, $\mathbf{t_2}$ with a change in sign, so that the tables have the form $(\mathbf{e_1}, -\mathbf{e_1}, \mathbf{e_1})$, $(\mathbf{e_2}, -\mathbf{e_2}, \mathbf{e_2})$, $(\mathbf{t_1}, -\mathbf{t_1}, \mathbf{t_1})$, $(\mathbf{t_2}, -\mathbf{t_2}, \mathbf{t_2})$. It is possible to reduce the tables sizes down to $2n$ at the cost of a modest decrease in performance by reducing indices modulo $2n$ when accessing the middle part of the table. On the other hand, at the higher security levels the sparse products needed at verification become less competitive than the NTT.

Exact modular reductions and centerings within loops are typically deferred as much as the machine word size permits. Meanwhile, representatives stemming from incomplete reduction are used, e.g. when mapping to the frequency domain via the NTT or processing the corresponding values.

Loop unrolling has only a limited impact on performance (no more than 5% or so) and was therefore applied sparingly.

To reduce the performance impact of the underlying hash function, the coefficients of modular-rounded polynomial coefficients are packed together before being hashed, keeping only the $32 - d$ most significant bits of each coefficient rather than the entire 32-bit word where they are stored, since all the other bits are forcibly discarded anyway.

6 Experimental results

To evaluate the performance of TESLA♯, we benchmarked our reference implementation with the GNU GCC compiler v4.9.2 on a 3.4GHz Intel Core i7-4770 Haswell processor running Ubuntu 14.04 LTS and with TurboBoost disabled. To

restrict the influence of the adopted hash function on the measurements to the signature nonces that accompany the message, we sign only the empty string. Timings were averaged over 10^6 random tests for key generation, signing and verification.

6.1 TESLA♯ performance results

Table 1 summarizes the observed results at the classical 128-bit security level using parameter set TESLA♯–I and at the post-quantum 128-bit security level (roughly 256-bit classical security level) using parameter set TESLA♯–II. For comparison, we also list published results for other signature schemes based on ideal lattices: the GLP scheme [22, 16] (benchmarked on a 3.4GHz Intel Core i5-3210M Ivy Bridge processor with TurboBoost disabled, and using an unknown hash function), Ring-TESLA-II [2] (benchmarked on a 3.3GHz Intel Core i7-5820K Haswell CPU with TurboBoost disabled, and using the SHA-256 hash function) and for BLISS-BI [18] (benchmarked on an unspecified processor of the Intel Core family at 3.4 GHz, and using an unknown hash function; the use of TurboBoost is not specified). From all the implementations listed in Table 1, only the ones corresponding to GLP and ring-Tesla-II have been optimized using AVX2 instructions. Public key (PK), secret key (SK) and signature sizes are given in bytes. Timings have been rounded to the closest 10^3 clock cycles.

Table 1. Performance of various signature schemes over ideal lattices.

Scheme	GLP	BLISS-BI	Ring-TESLA-II	TESLA♯–I	TESLA♯–II
Security (Pre-Q)	100	128	118	128	256
Security (Post-Q)	$\leqslant 80^{\dagger}$	$\leqslant 66^{\dagger}$	64	64	128
PK size	1536	896	3328	3328	7168
SK size	256	≈ 256	1920	2112	4608
Signature size	1184	≈ 700	1568	1616	3488
Isochronous?	✓	✗	✗✓‡	✓	✓
Base	NA	NA	NA	18000	37000
KeyGen	NA	NA	NA	340000	710000
Sign	452000	422000	511000	114000	268000
Verify	34000	102000	168000	81000	202000

† Our conservative approach would put this at no more than half the classical security.
‡ Key generation: ✗ Signing/verification: ✓

As we can see, TESLA♯–I achieves significantly faster signing and verification times in comparison with other alternatives at the 128-bit classical level. Note that this is achieved with full protection against timing and cache attacks and without exploiting AVX2 instructions. The TESLA♯–II instantiation of the

implementation achieves 128 bits of post-quantum security[7]. Nonetheless, it still achieves very competitive performance in comparison with pre-quantum schemes (e.g., signing is faster than any other competing signature scheme).

Recently, Saarinen [33] proposed a quantum-secure signature scheme called BLZZRD, and reported performance results for a side-channel secure implementation on a 2.5GHz Core-i7 Haswell processor. Although verification cost of BLZZRD is quite competitive, signing is significantly more costly. For example, BLZZRD-I, a variant that offers 128-bit security, computes a signature in about 1.25 million cycles, whereas verification takes only 160 Kcycles. On the other hand, we note that Saarinen's implementation has not been optimized yet, and that BLZZRD offers shorter signature sizes (e.g., BLZZRD-I's signatures are roughly 731 bytes long).

Table 2 compares TESLA♯–I and TESLA♯–II against several other signature schemes as benchmarked in [19] using OpenSSL v1.0.1c. Security is stated against both classical and quantum attacks (in the format 'classical:quantum'); we indicate the fact that RSA and ECDSA are breakable in quantum polynomial time by assigning them a zero quantum security level. The timings are those observed on an Intel Core i7 processor at 3.4 GHz.

We can see that TESLA♯–I, which offers 128-bit security, beats the signing and verification times of all signature schemes on this list, except for the verification times of RSA-1024, GLP (which offer 80-bit classical security) and BLISS-0 (which offers no more than 60-bit classical or 33-bit quantum security). We point out that the security estimates for the BLISS family follow Saarinen [33, Section 3], who shows how those schemes are more susceptible to Grover-style quantum attacks and, hence, are less secure than anticipated by the authors.

6.2 TESLA♯ with fresh, per key pair generation of $(\mathbf{a_1}, \mathbf{a_2})$

As discussed in Section 3, TESLA♯ requires the use of a public pair $(\mathbf{a_1}, \mathbf{a_2}) \in \mathcal{R}_q^2$. Arguably, there is a potential risk associated to the generation of these global values: an adversary could fix these parameters based on a "weak" lattice instance and thus reduce the security of the whole signature scheme. Moreover, the use of a unique lattice for all the generated signatures can facilitate "all-for-the-price-of-one" precomputation attacks, as was exemplified by the Logjam attack in the case of finite field Diffie-Hellman [1]. Note that similar issues also appear in previous TESLA variants ([2, 4]).

In order to avoid these vulnerabilities, we present a slightly modified version of TESLA♯ that requires the generation of a fresh pair $(\mathbf{a_1}, \mathbf{a_2})$ for every new key pair using a 32-byte random seed. Algorithms for the modified scheme are shown in Algorithms 4, 5 and 6 in Appendix C.

[7] GLP has a purported high security version where the underlying lattice problem would attain 256-bit classical security. However, as per [22, Section 2.3], all versions of that scheme use a 160-bit hash, thus offering no more than 160-bit classical security while also creating an 80-bit quantum security bottleneck since a quantum algorithm is known [14] that can find preimages of κ-bit hashes in $O(2^{\kappa/2})$ time.

Table 2. General benchmarks

Scheme	Sec(bits)	Sign/s	Verify/s
TESLA♯–I	128:64	29.9k	42.1k
TESLA♯–II	256:128	12.7k	16.8k
GLP	80:80	7.5k	100.0k
BLISS-0	60:33	4.2k	58.8k
BLISS-I	128:66	8.1k	33.3k
BLISS-II	128:66	2.1k	33.3k
BLISS-III	160:80	4.9k	32.3k
BLISS-IV	192:97	2.7k	31.3k
RSA-1024	80:0	6.0k	90.9k
RSA-2048	112:0	0.85k	26.3k
RSA-4096	144:0	0.11k	7.3k
ECDSA-160	80:0	17.24k	4.9k
ECDSA-256	128:0	9.43k	2.6k
ECDSA-384	192:0	5.13k	1.2k

In Table 3, we show the performance of TESLA♯ with the aforementioned modifications. In our implementation, we used ChaCha20 for pseudo-random generation of the pair $(\mathbf{a_1}, \mathbf{a_2})$. As can be seen, the increase in computing cost is very small (below 15% cost increase in most cases) as well as the increase in the public key size (only 32 bytes). In addition, the modified scheme potentially saves space and bandwidth by avoiding the storage or transmission of a full pair $(\mathbf{a_1}, \mathbf{a_2}) \in \mathcal{R}_q^2$, which is replaced by a 32-byte seed.

7 Concluding remarks

We have described the high-performance, high-security digital signature scheme TESLA♯ and proved its security on the random oracle model under the R-LWE assumption. Our proposal achieves fast key generation, signing, and verification on modern processors, partly as a bonus result of correcting a design flaw that was present in its predecessors. We proposed specific parameter sets for both pre-quantum and post-quantum applications, and showed that our proposal typically outperforms alternative conventional or lattice-based signature schemes even when our scheme targets a higher (and hence more computationally expensive) security level. Finally, we described a variant of our signature scheme that offers improved security against backdoors and "all-for-the-price-of-one" precomputation attacks.

References

1. D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella Béguelin, and P. Zimmermann. Imperfect forward secrecy: how Diffie-Hellman fails in practice. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of*

Table 3. Performance of the modified TESLA♯ scheme in comparison with other signature schemes over ideal lattices. In the case of TESLA♯, each new key pair uses a different underlying lattice and a fresh pseudo-randomly generated pair $(\mathbf{a_1}, \mathbf{a_2})$, providing protection against potential backdoors and "all-for-the-price-of-one" precomputation attacks.

Scheme	GLP	BLISS-BI	Ring-TESLA-II	TESLA♯–I	TESLA♯–II
Security (Pre-Q)	100	128	118	128	256
Security (Post-Q)	$\leqslant 80^{\dagger}$	$\leqslant 66^{\dagger}$	64	64	128
PK size	1536	896	3328	3360	7200
SK size	256	≈ 256	1920	2112	4608
Signature size	1184	≈ 700	1568	1616	3488
Isochronous?	✓	✗	✗✓‡	✓	✓
KeyGen	NA	NA	NA	358000	740000
Sign	452000	422000	511000	129000	294000
Verify	34000	102000	168000	96000	226000

† Our conservative approach would put this at no more than half the classical security.
‡ Key generation: ✗ Signing/verification: ✓

the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 5–17, Denver (CO), USA, 2015. ACM.

2. S. Akleylek, N. Bindel, J. Buchmann, J. Krämer, and G. A. Marson. An efficient lattice-based signature scheme with provably secure instantiation. In D. Pointcheval, A. Nitaj, and T. Rachidi, editors, *Progress in Cryptology – Africacrypt 2016*, volume 9646 of *Lecture Notes in Computer Science*, pages 44–60, Fes, Morocco, 2016. Springer.
3. 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.
4. E. Alkim, N. Bindel, J. Buchmann, Ö. Dağdelen, and P. Schwabe. TESLA: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/755, 2015.
5. 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 *Lecture Notes in Computer Science*, pages 595–618, Santa Barbara, CA, USA, 2009. Springer.
6. J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein. BLAKE2: simpler, smaller, fast as MD5. `https://blake2.net/blake2.pdf`, 2013.
7. S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In J. Benaloh, editor, *RSA Conference – Cryptographer's Track – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47, San Francisco, CA, USA, 2014. Springer.
8. C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Phys. Rev. Lett.*, 117(6):060504, 2016.
9. R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. L. Heras, R. Babbush, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J.

O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and J. M. Martinis. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, 2016.
10. P. D. Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323, Santa Barbara (CA), USA, 1986. Springer.
11. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security – CCS 1993*, volume 263 of *CCS*, pages 62–73, Fairfax (VA), USA, 1993. ACM.
12. D. J. Bernstein. ChaCha, a variant of Salsa20. Technical report, University of Illinois at Chicago, 2008.
13. B. Biswas and N. Sendrier. McEliece cryptosystem implementation: Theory and practice. In J. Buchmann and J. Ding, editors, *International Conference on Post-Quantum Cryptography – PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 47–62, Cincinnati (OH), USA, 2008. Springer.
14. G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology Column)*, 28:4–19, 1997.
15. L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on Post-Quantum Cryptography (NIST IR 8105 draft). Technical report, National Institute of Standards and Technology (NIST), Gaithersburg (MD), USA, 2 2016.
16. Ö. Dağdelen, 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, *International Conference on Cryptology and Information Security in Latin America – Latincrypt 2014*, volume 8895 of *Lecture Notes in Computer Science*, pages 84–103, Florianópolis, Brazil, 2014. Springer.
17. S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63–66, 2016.
18. L. Ducas. Accelerating BLISS: the geometry of ternary polynomials. Cryptology ePrint Archive, Report 2014/874, 2014.
19. 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 *Lecture Notes in Computer Science*, pages 40–56, Santa Barbara (CA), USA, 2013. Springer.
20. M. J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology (NIST), Gaithersburg (MD), USA, 8 2015.
21. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, 1988.
22. T. Güneysu, V. Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *Conference on Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547, Leuven, Belgium, 2012. Springer.
23. National Security Agency (NSA) Information Assurance Directorate (IAD). Commercial national security algorithm suite. Technical report, National Security Agency (NSA), Fort George G. Meade (MD), USA, 8 2015.

24. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.
25. A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson. ChaCha20–Poly1305 cipher suites for transport layer security (TLS). RFC 7905, Internet Research Task Force (IRTF), June 2016.
26. V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – Eurocrypt 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, 2012. Springer.
27. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *Advances in Cryptology – Eurocrypt 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, France, 2010. Springer.
28. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426, Santa Barbara (CA), USA, 1985. Springer.
29. Y. Nir. ChaCha20, Poly1305 and their use in the internet key exchange protocol (IKE) and IPsec. RFC 7634, Internet Research Task Force (IRTF), August 2015.
30. Y. Nir and A. Langley. ChaCha20 and Poly1305 for IETF protocols. RFC 7539, Internet Research Task Force (IRTF), May 2015.
31. R. L. Rivest, A Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
32. M.-J. Saarinen. Gaussian sampling precision in lattice cryptography. Cryptology ePrint Archive, Report 2015/953, 2015.
33. M.-J. Saarinen. Arithmetic coding and blinding countermeasures for lattice signatures: Engineering a side-channel resistant post-quantum signature scheme with compact signatures. Cryptology ePrint Archive, Report 2016/276, 2016.
34. M.-J. Saarinen and J.-P. Aumasson. The BLAKE2 cryptographic hash and message authentication code (MAC). RFC 7693, Internet Research Task Force (IRTF), November 2015.
35. N. Sendrier. Encoding information into constant weight words. In *IEEE International Symposium on Information Theory – ISIT 2005*, pages 435–438. IEEE, 2005.
36. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing*, 26(5):1484–1509, 1997.

A Formal security proof

In this section, we provide the formal proof that the TESLA♯ scheme is EUF-CMA-secure under the R-LWE assumption. The proof follows the proof of Theorem 1 in the original paper introducing the ring version of TESLA by Akleylek et al. [2] very closely. Hence, we state some of their results almost word for word. The following theorem is the main result of the security reduction, which is the analog of [2, Theorem 1].

Theorem 1. *Let n, ω, d, B, q, U, σ, κ be the TESLA♯ parameters satisfying the conditions described in Sections 3 and 4. We assume that the Gaussian heuristic holds for lattice instances defined by the parameters above. Let $\mathcal{A}$ be a probabilistic*

algorithm that breaks EUF-CMA *security, i.e.* $\mathcal{A}$ *runs in time* $\tau_{\mathcal{A}}$*, asks at most* $q_{\mathcal{S}}$ *queries to the signing oracle and at most* $q_{\mathcal{H}}$ *queries to the hash oracle, and forges a valid TESLA♯ signature with probability* $\varepsilon_{\mathcal{A}}$*.*

Then there exists a distinguisher $\mathcal{D}$ *that runs in time* $\tau_{\mathcal{D}} = \tau_{\mathcal{A}} + O(q_{\mathcal{S}}\kappa^2 + q_{\mathcal{H}})$ *and solves the* R-LWE$_{n,2,q,\sigma}$ *problem (in the random oracle model) with success probability*

$$\varepsilon_{\mathcal{D}} \geqslant \varepsilon_{\mathcal{A}} \left(1 - \frac{q_{\mathcal{S}}(q_{\mathcal{H}} + q_{\mathcal{S}})2^{(d+1)2n}}{(2B+1)^n q^n}\right) - \frac{q_{\mathcal{H}} 2^{dn}(2(B-U)+1)^n + (2\eta\sigma+1)^{3n}}{q^{2n}}.$$

Proof. We first describe the distinguisher $\mathcal{D}$ that solves the R-LWE$_{n,2,q,\sigma}$ problem by running the EUF-CMA adversary $\mathcal{A}$ internally.

The distinguisher $\mathcal{D}$ receives as input two challenge samples $(\mathbf{a_1}, \mathbf{t_1})$ and $(\mathbf{a_2}, \mathbf{t_2}) \in \mathcal{R}_q \times \mathcal{R}_q$ and outputs a bit guessing whether the two samples came from the R-LWE distribution $D_{\mathbf{s},\chi}$ for some given secret $\mathbf{s}$ or from the uniform distribution on $\mathcal{R}_q \times \mathcal{R}_q$. In order to use the algorithm $\mathcal{A}$, $\mathcal{D}$ has to simulate the EUF-CMA-game for $\mathcal{A}$. The first step for $\mathcal{D}$ is to define $\mathsf{pk} = (\mathbf{a_1}, \mathbf{a_2}, \mathbf{t_1}, \mathbf{t_2})$ and hand pk over to $\mathcal{A}$ as a public signing key. To simulate the EUF-CMA game, the distinguisher $\mathcal{D}$ answers the hash and signing queries by $\mathcal{A}$ as follows.

- *Hash queries:* The distinguisher $\mathcal{D}$ keeps a list of queries to the hash oracle $\mathcal{H}$ and their hash values. When $\mathcal{A}$ asks a query that has been asked before, $\mathcal{D}$ returns the corresponding answer. When $\mathcal{A}$ asks a query that has not been asked before, $\mathcal{D}$ samples a fresh, uniformly random $c' \in \{0,1\}^\kappa$, records this value together with its input in the list and returns c' to $\mathcal{A}$.
- *Signing queries:* When $\mathcal{A}$ asks to obtain a signature on a message μ, $\mathcal{D}$ samples a uniformly random bit string $c' \xleftarrow{\$} \{0,1\}^\kappa$ and a uniformly random polynomial $\mathbf{z} \xleftarrow{\$} \mathcal{R}_{q,[B-U]}$. It then computes $\mathbf{c} \leftarrow F(c')$ and $\mathbf{w_i} \leftarrow \mathbf{a_i z} - \mathbf{t_i c} \pmod q$ for $\mathbf{i} = 1, 2$. If $\mathcal{A}$ has already asked the hash oracle $\mathcal{H}$ a query with input $(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$ before, $\mathcal{D}$ aborts the simulation. If this input is used for the first time, $\mathcal{D}$ sets $\mathcal{H}(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu) = c'$ and returns $(\mathbf{z}, c')$ as its signature on μ with probability $1/M$, where $M = (2B+1)^n/(2(B-U)+1)^n$.

The simulation ends, when $\mathcal{A}$ outputs a forgery $(\tilde{\mathbf{z}}, \tilde{c}')$ on a message $\tilde{\mu}$ for which it has not yet asked to obtain a signature. The distinguisher $\mathcal{D}$ verifies the signature $(\tilde{\mathbf{z}}, \tilde{c}')$: if $\mathsf{Verify}(\mathsf{pk}, \mu, (\tilde{\mathbf{z}}, \tilde{c}')) = 1$ it returns 1, else it returns 0 as its guess for the solution of the R-LWE$_{n,2,q,\sigma}$ problem.

We first show that responses provided by the distinguisher $\mathcal{D}$ to $\mathcal{A}$'s hash and signing queries are both indistinguishable from responses provided by the random oracle and a true signing oracle.

In the random oracle model, hash query responses by $\mathcal{D}$ return uniformly random values as long as queries do not repeat on the same input. If during the response to a signing query a hash query on a previously used input occurs, the simulation aborts. The probability for aborts in this case is analyzed below. We need to show that if the forger $\mathcal{A}$ plays the game against an instantiation of the real signature scheme, hash values are indistinguishable from uniformly

random values. As argued by Bai and Galbraith, we need to ensure that the set $\{(\lfloor \mathbf{a_1} * \mathbf{y} \rceil_d, \lfloor \mathbf{a_2} * \mathbf{y} \rceil_d) \mid \mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,B}\}$ carries enough entropy. The following lemma shows that this is the case. It is based on [7, Lemma 3] and essentially the same as [2, Lemma 2]. For an integer a the notation $\lfloor a \rceil_{d,q}$ means $\lfloor a \bmod q \rceil_d$ and extends to vectors, matrices and polynomials coefficientwise.

Lemma 1. *Let parameters be as in Theorem 1 and let $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q^*$. Define the lattice $\Lambda = \{v \in \mathbb{Z}^{2n} \mid v \equiv \mathbf{Aw} \pmod q$ for some $\mathbf{w} \in \mathbb{Z}^n\}$, where $\mathbf{A} = \big(\mathrm{Rot}(\mathbf{a_1})\ \mathrm{Rot}(\mathbf{a_2})\big)^T$. Assume that the Gaussian heuristic holds for Λ, i.e. $\Lambda \cap \big[-2^d, 2^d\big] \leqslant 2^{(d+1)2n}/q^n$. Then for all $\mathbf{y_1} \in \mathcal{R}_{q,[B]}$ we have*

$$\Pr_{\mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_{q,[B]}} [\lfloor \mathbf{A y_1} \rceil_{d,q} = \lfloor \mathbf{A y_2} \rceil_{d,q}] \leqslant \frac{2^{(d+1)2n}}{(2B+1)^n q^n}.$$

In particular, $\big|\{\lfloor \mathbf{Ay} \rceil_{d,q} \;\big|\; \mathbf{y} \in \mathcal{R}_{q,[B]}\}\big| \geqslant \dfrac{(2B+1)^n q^n}{2^{(d+2)2n}}$.

Proof (Proof of Lemma 1). The proof is the same as that of [2, Lemma 2], which closely follows the proof of [7, Lemma 3]. We repeat the main arguments here for clarity.

For $\mathbf{y_1} \in \mathcal{R}_{q,[B]}$ consider the set $S = \big\{y_2 \in \mathcal{R}_{q,[B]} \;\big|\; \lfloor \mathbf{A y_1} \rceil_{d,q} = \lfloor \mathbf{A y_2} \rceil_{d,q}\big\}$. The lemma follows by showing that $|S| \leqslant 2^{(d+1)2n}/q^n$ since $|\mathcal{R}_{q,[B]}| = (2B+1)^n$. The set $M = \{\mathbf{y_1} - \mathbf{x} \in \mathbb{Z}^n \mid \mathbf{x} \in S\}$ satisfies $|M| = |S|$. Let $N := \{\mathbf{Ax} \bmod q \mid \mathbf{x} \in M\}$. Since both $\mathbf{a}_1, \mathbf{a}_2$ are invertible in $\mathcal{R}_q$, $M \subseteq \mathcal{R}_{q,[2B]}$, and $q/2 > 2B$, it follows that $|N| = |M|$.

Let $\mathbf{x} = \mathbf{A}(\mathbf{y_1} - \mathbf{z}) \pmod q$ with $\mathbf{z} \in S$ be an arbitrary element in N. Then, $\lfloor \mathbf{A y_1} \rceil_{d,q} = \lfloor \mathbf{Az} \rceil_{d,q}$ and thus $\lfloor \mathbf{x} \rceil_{d+1,q} = \lfloor \mathbf{A y_2} - \mathbf{Az} \rceil_{d+1,q} = \mathbf{0}$, which means $\mathbf{x} \in \big[-2^d, 2^d\big]^{2n}$. Therefore, $N \subseteq \Lambda \cap \big[-2^d, 2^d\big]^{2n}$ and by the Gaussian heuristic, we obtain $|N| \leqslant \Big|\Lambda \cap \big[-2^d, 2^d\big]^{2n}\Big| \leqslant 2^{(d+1)2n}/q^n$. □

Next, for signing queries, we need to show that the distribution of polynomials z returned by $\mathcal{D}$ in the simulation is indistinguishable from the distribution of polynomials z produced by the signing algorithm. This means, it needs to be shown that the latter is statistically close to the uniform distribution on $\mathcal{R}_{q,[B-U]}$. Again, this can be shown exactly as in [2] by deploying the rejection sampling lemma by Lyubashevsky [26, Lemma 4.7 of the full version]. We do not state the general lemma again, but directly apply it to the situation needed for this proof.

Lemma 2. *For the parameters as in Theorem 1, let X be the uniform distribution on $\mathcal{R}_{q,[B-U]}$, i.e. the probability for a polynomial $\mathbf{z} \xleftarrow{\$} \mathcal{R}_{q,[B-U]}$ to be sampled according to X is $f_X(\mathbf{z}) = 1/(2(B-U)+1)^n$. We extend the distribution to $\mathcal{R}$ and assign the probability $f_X(\mathbf{z}) = 0$ for all $\mathbf{z} \notin \mathcal{R}_{q,[B-U]}$. For a fixed secret $\mathbf{s} \xleftarrow{\$} D_\sigma^n$, define $V = \{\mathbf{s} * \mathbf{c} : \mathbf{c} = F(c), c \in \{0,1\}^\kappa\}$ and let Y be the distribution on V that samples $c \in \{0,1\}^\kappa$ uniformly at random, then computes*

$\mathbf{c} = F(c)$ *and returns* $\mathbf{v} = \mathbf{s} * \mathbf{c}$. *Finally, for* $\mathbf{v} \in V$, *let* Z_v *be the uniform distribution on polynomials with coefficient vectors* $\mathbf{v} + [-B, B]^n$, *i.e. the probability for sampling* $\mathbf{z} = \mathbf{v} + \mathbf{y}$, $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$ *is* $f_{Z_\mathbf{v}}(\mathbf{z}) = 1/(2B+1)^n$, *and* $f_{Z_\mathbf{v}}(\mathbf{z}) = 0$ *for* $\mathbf{z} \notin \mathbf{v} + \mathcal{R}_{q,[B]}$.

Then for $M = (2B+1)^n/(2(B-U)+1)^n$, *we have for any* $\mathbf{v} \in V$

$$\Pr_{\mathbf{z} \xleftarrow{\$} X}[M \cdot f_{Z_\mathbf{v}}(\mathbf{z}) \geqslant f_X(\mathbf{z})] \geqslant 1 - 2^{-\nu}, \tag{1}$$

where $\nu \geqslant \log(e) \cdot \eta^2/2 + \log(2\eta(\beta n - 1))$ *and the output distributions of the following two algorithms are within statistical distance* $2^{-\nu}/M$:

1. $\mathbf{v} \xleftarrow{\$} Y$, $\mathbf{z} \xleftarrow{\$} Z_v$, *output* $(\mathbf{z}, \mathbf{v})$ *only if* $\mathbf{z} \in \mathcal{R}_{q,[B-U]}$.
2. $\mathbf{v} \leftarrow Y$, $\mathbf{z} \leftarrow X$, *output* $(\mathbf{z}, \mathbf{v})$ *with probability* $1/M$.

The probability that the first algorithm produces an output is $(1 - 2^{-\nu})/M$. *Note that with our concrete parameter choice,* $1/M \approx e^{-1/\beta}$.

Proof (Proof of Lemma 2). We first show that the inequality in (1) holds for all $\mathbf{v} \in V$. Note that if $|v| < U$ for any coefficient v of $\mathbf{v}$, then any $\mathbf{z}$ sampled from X lies in $\mathbf{v} + \mathcal{R}_{q,[B]}$, i.e. in the support of $Z_\mathbf{v}$. In this case, $M \cdot f_{Z_\mathbf{v}}(\mathbf{z}) = f_X(\mathbf{z})$ and the probability in (1) is equal to 1.

The condition $M \cdot f_{Z_\mathbf{v}}(\mathbf{z}) \geqslant f_X(\mathbf{z})$ is not satisfied if there exists a coefficient v of $\mathbf{v}$ such that $|v| > U$ and if the corresponding coefficient z of $\mathbf{z}$ is sampled from $[B - v, B - U]$ or $[-B + U, -B - v]$, depending on the sign of v. The probability for this to happen is $2\sum_{j=U+1}^{\infty} \Pr[v = j] \cdot \frac{j-U}{2(B-U)+1}$. The fraction is the probability that z is sampled from one of the above intervals, the factor 2 takes into account both cases, positive and negative v. Using the fact that v is the sum of exactly ω coefficients of $\mathbf{s}$, i.e. is distributed as a discrete Gaussian with standard deviation $\sigma_\omega = \sqrt{\omega}\sigma$, we have that $\Pr[v = j] = e^{-j^2/2\omega\sigma^2}/\sqrt{2\pi\omega}\sigma$. Replacing the sum by an integral and integrating, we obtain an upper bound on the probability for one coefficient of $\mathbf{z}$ to fall outside the valid interval. Summing up over all n coefficients and taking the complement, we obtain a lower bound for the probability in (1) as $1 - e^{-k^2/2}/2k(\beta n - 1)$, which directly yields the bound on ν.

At this point, we note that the assumptions in the rejection sampling lemma [26, Lemma 4.7 of the full version] are satisfied and the remainder of the proof is simply an application of that lemma. Note that the condition on the probabilities in the first algorithm translates into the check $\mathbf{z} \in \mathcal{R}_{q,[B-U]}$. □

The above lemma shows that the forger $\mathcal{A}$ cannot distinguish between the distribution of z presented by the distinguisher $\mathcal{D}$ and that presented by the actual signing algorithm. We now turn to analyzing the abort probability during the signing simulation.

Probability of abort during simulation. As described above, the simulation in the EUF-CMA game aborts, when, during a sign query for a message μ, the

simulator notices that the hash function oracle has already been queried on $(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$. The following argument is the same as in [2].

Assume that the signing simulation samples an additional value $\mathbf{y}$ uniformly at random from $\mathcal{R}_{q,[B]}$. Furthermore, the simulation programs not only $c' = \mathcal{H}(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$ but also $c' = \mathcal{H}(\lfloor \mathbf{a_1 y} \rceil_d, \lfloor \mathbf{a_2 y} \rceil_d, \mu)$. Sampling $\mathbf{y}$ does not influence $(\mathbf{z}, c')$. The probability of aborting during the original signing simulation is at most the abort probability during the signing simulation with the changes just described. The latter probability is the same as finding a collision for the rounding function. Hence, by Lemma 1, the probability that $\mathcal{D}$ aborts during the simulation of $\mathcal{A}$'s environment is less than $q_{\mathcal{S}}(q_{\mathcal{H}} + q_{\mathcal{S}})\frac{2^{(d+1)2n}}{(2B+1)^n q^n}$.

Next we derive the bound on $\mathcal{D}$'s distinguishing advantage depending on the probability ϵ_A for $\mathcal{A}$ to forge a valid signature. As is done in the proof of security in the appendix of [2], we also distinguish between the two possible cases for $(\mathbf{a_1}, \mathbf{t_1})$ and $(\mathbf{a_2}, \mathbf{t_2}) \in \mathcal{R}_q \times \mathcal{R}_q$ that can occur in the security game. Namely, either, $(\mathbf{a_1}, \mathbf{t_1})$ and $(\mathbf{a_2}, \mathbf{t_2})$ are both chosen according to the R-LWE distribution or both according to the uniform distribution on $\mathcal{R}_q \times \mathcal{R}_q$.

Case 1: (R-LWE samples) We assume that $(\mathbf{a}_1, \mathbf{t}_1) \leftarrow D_\sigma^n$ and $(\mathbf{a}_2, \mathbf{t}_2) \leftarrow D_\sigma^n$, which means that $\mathbf{t}_1 = \mathbf{a}_1 * \mathbf{s} + \mathbf{e}_1$ and $\mathbf{t}_2 = \mathbf{a}_2 * \mathbf{s} + \mathbf{e}_2$ for $\mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma^n$. We need to determine a bound for the probability that $\mathcal{D}$ returns 1 correctly.

There are two possibilities for $\mathcal{D}$ to falsely return 0, namely first, if $\mathcal{D}$ aborts while answering a signing query; or, second, if the algorithm $\mathcal{A}$ does not output a valid forgery. As described above, the probability that $\mathcal{D}$ aborts during the simulation is at most $q_{\mathcal{S}}(q_{\mathcal{S}} + q_{\mathcal{H}})\frac{2^{(d+1)2n}}{(2B+1)^n q^n}$. For $\mathcal{D}$ to return 1, both situations above must not occur, which means that $\mathcal{A}$ must output a valid forgery, which happens with probability $\varepsilon_{\mathcal{A}}$ and the simulation must not abort. Thus, $\mathcal{D}$ returns 1 with probability at least $\varepsilon_{\mathcal{A}} \left(1 - q_{\mathcal{S}}(q_{\mathcal{H}} + q_{\mathcal{S}})\frac{2^{(d+1)2n}}{(2B+1)^n q^n}\right)$.

Case 2: (Uniformly random samples) The second case assumes that $(\mathbf{a}_1, \mathbf{t}_1) \xleftarrow{\$} \mathcal{R}_q^* \times \mathcal{R}_q$ and $(\mathbf{a}_2, \mathbf{t}_2) \xleftarrow{\$} \mathcal{R}_q^* \times \mathcal{R}_n$ are sampled uniformly at random. First, we argue that it is highly unlikely that the t_i have the form as in the first case, where $\mathbf{t}_i = \mathbf{a}_i * \mathbf{s} + \mathbf{e}_i$ for polynomials $\mathbf{s}, \mathbf{e}_i$ with small coefficients.

We start by repeating [2, Remark 1], which bounds the size of coefficients sampled from a discrete Gaussian distribution by $\eta\sigma$ with overwhelming probability. We repeat it here, since we differentiate between different cases that apply to different parameter sets.

Remark 3. Lemma 4.4 in [26] states that for any $\eta > 0$ and a single coefficient $x \leftarrow D_\sigma$, we have $\Pr[|x| > \eta\sigma] \leqslant 2e^{-\eta^2/2}$. It follows that the probability that all coefficients of $\mathbf{s} \leftarrow D_\sigma^n$, $\mathbf{e_1} \leftarrow D_\sigma^n$, and $\mathbf{e_2} \leftarrow D_\sigma^n$ are in $[-\eta\sigma, \eta\sigma]$ is at least $\left(1 - 2e^{-\eta 2/2}\right)^{3n}$. In other words, the probability that KeyGen will fail to yield a suitable private key does not exceed $1 - \left(1 - 2e^{-\eta^2/2}\right)^{3n}$. For instance, for

$n = 512$, any value $\eta \gtrsim 13.91$ limits the probability of failure to less than 2^{-128}. Similarly, for $n = 1024$, any value $\eta \gtrsim 19.30$ limits the probability of failure to less than 2^{-256}.

With the above bound, we may use the following lemma, also resembling the analogous result in [2, Appendix B], to obtain a bound on the probability that $\mathbf{t}_i = \mathbf{a}_i * \mathbf{s} + \mathbf{e}_i$, where $\mathbf{s}, \mathbf{e}_i$ have small coefficients.

Lemma 3. *Let parameters be as in Theorem 1. Furthermore, let* $\mathbf{a_1}, \mathbf{a_2} \xleftarrow{\$} \mathcal{R}_q^*$ *and* $\mathbf{t_1}, \mathbf{t_2} \xleftarrow{\$} \mathcal{R}_q$*. Then it holds that*

$$\Pr\left[\exists\ \mathbf{s}, \mathbf{e_1}, \mathbf{e_2} \in \mathcal{R}_{q,[\eta\sigma]} \;\middle|\; \mathbf{t_i} = \mathbf{a_i} * \mathbf{s} + \mathbf{e_i} \,(\mathrm{mod}\, q), i \in \{1,2\}\right] \leqslant \frac{(2\eta\sigma+1)^{3n}}{q^{2n}},$$

where the probability is taken over random choices of $\mathbf{a_1}$*,* $\mathbf{a_2}$*,* $\mathbf{t_1}$*, and* $\mathbf{t_2}$*.*

Proof. The polynomials $\mathbf{t_1}$, and $\mathbf{t_2}$ are uniformly random, therefore, the probability to sample two specific polynomials $\mathbf{t}_1, \mathbf{t}_2$ is $1/\left|\{(\mathbf{t}_1, \mathbf{t}_2) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n\}\right| = 1/q^{2n}$. Next, we take a look at the set $S_{\mathbf{a}_1,\mathbf{a}_2} = \{(\mathbf{a}_1 * \mathbf{s} + \mathbf{e}_1, \mathbf{a}_2 * \mathbf{s} + \mathbf{e}_2) \mid \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \in [-\eta\sigma, \eta\sigma]^n\}$. Its cardinality can be bounded by $|S_{\mathbf{a}_1,\mathbf{a}_2}| \leqslant |\{(\mathbf{a}_1 * \mathbf{s}, \mathbf{a}_2 * \mathbf{s}) \mid \mathbf{s} \in [-\eta\sigma, \eta\sigma]^n\}| \cdot |\{(\mathbf{e}_1, \mathbf{e}_2) \mid \mathbf{e}_1, \mathbf{e}_2 \in [-\eta\sigma, \eta\sigma]^n\}| = (2\eta\sigma+1)^n \cdot (2\eta\sigma+1)^{2n}$ since $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q^*$. □

This lemma shows that the probability for the $\mathbf{t}_i$ being sampled uniformly at random and at the same time having the R-LWE shape is very low. From now on, we assume that there do not exist $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \in [-\eta\sigma, \eta\sigma]^n$ such that $\mathbf{t}_i = \mathbf{a}_i * \mathbf{s} + \mathbf{e}_i$.

We continue to follow the proofs in [7] and [2]. At the end of the simulation, the forger $\mathcal{A}$ will output a forged signature $(\tilde{\mathbf{z}}, \tilde{c}')$ on a message $\tilde{\mu}$. Therefore, we conclude that $\mathcal{A}$ has made a hash query on $(\lfloor \mathbf{a_1}\tilde{\mathbf{z}} - \mathbf{t_1}\tilde{\mathbf{c}} \rceil_{d,q}, \lfloor \mathbf{a_2}\tilde{\mathbf{z}} - \mathbf{t_2}\tilde{\mathbf{c}} \rceil_{d,q}, \tilde{\mu})$ and recieved $\tilde{c}'$ in order for the verification equation to hold. Note that $\tilde{\mathbf{c}}' = F(\tilde{c}')$. The forger $\mathcal{A}$ can only produce a valid signature on some message μ, if for some $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{R}_q$ with $c' = \mathcal{H}(\mathbf{v}_1, \mathbf{v}_2, \mu)$, there exists a $\mathbf{z} \in \mathcal{R}_{q,[B-U]}$ such that $(\mathbf{z}, c')$ satisfies the verification condition. With the next lemma, we give an upper bound on the probability for this to happen.

Lemma 4. *Let the parameters be as in Theorem 1, and let* $\delta \in \mathbb{Q} > 0$*. For* $\mathbf{u} \in \mathcal{R}_q$*, let* $\Lambda_{\mathbf{u}} = \{\bar{\mathbf{z}} \in \mathbb{Z}^n \mid \mathbf{A}\bar{\mathbf{z}} = \mathbf{u} \,(\mathrm{mod}\, q)\}$*, where* $\mathbf{A} = \left(\mathrm{Rot}(\mathbf{a_1})\ \mathrm{Rot}(\mathbf{a_2})\right)^T$ *for* $\mathbf{a_1}, \mathbf{a_2} \in \mathcal{R}_q^*$*. Assume that the Gaussian heuristic holds for* $\Lambda_{\mathbf{u}}$*, i.e.* $|\Lambda_{\mathbf{u}} \cap [-\delta, \delta]^n| = (2\delta+1)^n/q^n$*.*

Then for all $\mathbf{v_1}, \mathbf{v_2} \in \mathcal{R}_q$*,* $c' \xleftarrow{\$} \{0,1\}^\kappa$*,* $\mathbf{c} = F(c')$*,* $\mathbf{a_1}, \mathbf{a_2} \xleftarrow{\$} \mathcal{R}_q^*$*, and* $\mathbf{t_1}, \mathbf{t_2} \xleftarrow{\$} \mathcal{R}_q$ *it holds that*

$$\Pr\left[\exists\ \mathbf{z} \in \mathcal{R}_{q,[\delta]} \;\middle|\; \mathbf{v_i} = \lfloor \mathbf{a_i} * \mathbf{z} - \mathbf{t_i} * \mathbf{c} \rceil_{d,q} \,(\mathrm{mod}\, q), i \in \{1,2\}\right] \leqslant \frac{2^{dn}(2\delta+1)^n}{q^{2n}},$$

where the probability is taken over random choices of c'*,* $\mathbf{a_1}$*,* $\mathbf{a_2}$*,* $\mathbf{t_1}$*, and* $\mathbf{t_2}$*.*

Proof. This statement follows from the proof of [2, Lemma 4]. □

With this result, it follows that the probability for $\mathcal{A}$ to forge a valid signature, i.e. the probability that for $\mathbf{v_1}, \mathbf{v_2} \in \mathcal{R}_q$, and $c' \xleftarrow{\$} \{0,1\}^\kappa$, there exists $\mathbf{z} \in \mathcal{R}_q$ such that $\|z\|_\infty \leqslant B - U$ and $\mathbf{v_i} = \lfloor \mathbf{a_i z} - \mathbf{t_i c} \rceil_{d,q} \pmod q$, $\mathbf{i} \in \{1,2\}$ is smaller than $2^{dn}(2(B-U)+1)^n/q^{2n}$. Given that $\mathcal{A}$ asks at most $q_\mathcal{H}$ hash queries, the probability that $\mathcal{A}$ forges a valid signature can be bound by $q_\mathcal{H} 2^{dn}(2(B-U)+1)^n/q^{2n}$. Overall, the probability that $\mathcal{D}$ returns 1 is smaller than $q_\mathcal{H} 2^{dn}(2(B-U)+1)^n/q^{2n} + (2\eta\sigma+1)^{3n}/q^{2n}$.

The distinguisher's advantage. Putting the results from the two cases together, we observe that the distinguisher $\mathcal{D}$ has the advantage $\mathrm{Adv}^{\mathsf{R\text{-}LWE}}_{n,q,\chi}(\mathcal{A}) = \varepsilon_\mathcal{D}$, which is bounded from below by

$$\varepsilon_\mathcal{D} \geqslant \varepsilon_\mathcal{A} \left(1 - \frac{q_\mathcal{S}(q_\mathcal{H}+q_\mathcal{S})2^{(d+1)2n}}{(2B+1)^n q^n}\right) - \frac{q_\mathcal{H} 2^{dn}(2(B-U)+1)^n + (2\eta\sigma+1)^{3n}}{q^{2n}}.$$

The distinguisher's running time. The relation between $\mathcal{A}$'s running time $\tau_\mathcal{A}$ and $\mathcal{D}$'s running time $\tau_\mathcal{D}$ is deduced by the arguments presented in [2, Appendix B], which we recall below.

Because $\mathcal{A}$ is a subroutine of $\mathcal{D}$, it follows that $\tau_\mathcal{D} \geqslant \tau_\mathcal{A}$. The running time overhead for $\mathcal{D}$ is generated by the extra steps needed to emulate the $\mathsf{EUF\text{-}CMA}$ game for $\mathcal{A}$. These consist of two queries to an $\mathsf{R\text{-}LWE}$-oracle to obtain the challenge tuples $(\mathbf{a_1}, \mathbf{t_1})$ and $(\mathbf{a_2}, \mathbf{t_2})$, and the answers to hash and signing oracle queries. Lemma 2 shows that the distributions of the signatures simulated by $\mathcal{D}$ and those output by the actual signing algorithm are statistically close. Thus, when emulating the signing procedure, $\mathcal{D}$ rejects a pair $(\mathbf{z}, c')$ with the same probability as Algorithm 2 (Sign) restarts the loop on line 2 due to an unsatisfied condition on lines 8 or 9. The answer to each signing query requires a constant number of polynomial multiplications, which cost $\mathcal{O}(\kappa^2)$. Overall, we obtain the approximate bound $\tau_\mathcal{D} \approx \tau_\mathcal{A} + \mathcal{O}(q_\mathcal{S}\kappa^2 + q_\mathcal{H})$. This concludes the proof of Theorem 1. □

B Verification failures in predecessor schemes

The Bai-Galbraith scheme [7], the original TESLA[8] [4], and the Ring-TESLA scheme [2] all try to ensure that a genuine signature $(\mathbf{z}, c)$ generated with $c \leftarrow H(\lfloor \mathbf{v_1} \rceil_d, \lfloor \mathbf{v_2} \rceil_d, \mu)$ (as computed in line 4 of the TESLA♯ signing algorithm) will verify by matching a corresponding $c' = H(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$ (as computed in line 3 of the TESLA♯ verification algorithm) by restricting the $\mathbf{e_1}, \mathbf{e_2}$ components of the private key so that the sum of their ω largest coefficients in absolute value does not exceed a certain bound L, and by discarding attempted signatures unless $|[\mathbf{w_1}]_{2^d}|$ and $|[\mathbf{w_2}]_{2^d}|$ are both bounded by $2^{d-1} - L$. The probability of meeting this condition is thus approximately $\left(1 - 2L/2^d\right)^{2n}$. Each scheme

[8] As mentioned in Remark 2, the authors of TESLA have recently updated their scheme and added an additional check that fixes the problem.

then chooses the parameter L on different but largely arbitrary grounds: Bai-Galbraith sets $L = 7\omega\sigma$, TESLA takes L between $2.65\omega\sigma$ and $3\omega\sigma$, and Ring-TESLA sets L so that this probability is at least $\left(1 - 2L/2^d\right)^{2n} \leqslant 0.4$.

We show that these conditions are not sufficient to ensure that the inputs to the hash function match. To avoid wraparound errors, it is necessary that the prime modulus q and the rounding parameter d satisfy the restriction $\lfloor q/2 \rfloor \geqslant 2^d$, and indeed this condition is satisfied by all parameter sets proposed for TESLA♯ and its predecessors. Hence, the overall relations between the parameters are $0 < L \ll 2^{d-1}$ and $2^{d+1} < 2\lfloor q/2 \rfloor < q$.

As a concrete example we now consider the algorithms for Ring-TESLA. We focus on a single coefficient w of either $\mathbf{w}_1$ or $\mathbf{w}_2$. Let v be the corresponding coefficient of $\mathbf{v}_1$ or $\mathbf{v}_2$, respectively. Then it holds for the corresponding coefficient e of $-\mathbf{e}_1 * \mathbf{c}$ or $-\mathbf{e}_2 * \mathbf{c}$ that $w = v + e \pmod q$. Since every coefficient of $\mathbf{e}_1 * \mathbf{c}$ and $\mathbf{e}_2 * \mathbf{c}$ is a sum of exactly ω values that are equal to plus or minus the coefficients of $\mathbf{e}_1$ or $\mathbf{e}_2$, the check on those during key generation ensures that $|e| \leqslant L$.

We have $v, w \in [-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ and $|e| \leqslant L$ and therefore $w = v + e - cq$, where $c \in \{-1, 0, 1\}$. Since the absolute value of e is so much smaller than $\lfloor q/2 \rfloor$, in most cases, there is no reduction modulo q, i.e. $c = 0$.

B.1 Concrete instances leading to verification failures

We want to see whether $\lfloor w \rceil_d \neq \lfloor v \rceil_d$. It holds

$$\lfloor w \rceil_d - \lfloor v \rceil_d = \left(e - cq - [w]_{2^d} + [v]_{2^d}\right) / 2^d.$$

Thus, $\lfloor w \rceil_d = \lfloor v \rceil_d$ if, and only if $e - [w]_{2^d} + [v]_{2^d} = cq$. Since $|e - [w]_{2^d} + [v]_{2^d}| \leqslant L + 2^{d-1} + 2^{d-1} < 2^{d+1} < q$, it follows that $c \neq 0$ implies $\lfloor w \rceil_d \neq \lfloor v \rceil_d$.

If $c = 0$, then the conditions enforced by the checks in the original signature generation are indeed sufficient to ensure that $\lfloor w \rceil_d = \lfloor v \rceil_d$. Namely, using the above equation, we see that $\lfloor w \rceil_d = \lfloor v \rceil_d$ if, and only if $[w]_{2^d} - e = [v]_{2^d}$. Because $|[w]_{2^d}| < 2^{d-1} - L$ and $|e| \leqslant L$, it follows that $|[w]_{2^d} - e| < 2^{d-1}$ and therefore $[w]_{2^d} - e = [w - e]_{2^d} = [v]_{2^d}$. This means that $\lfloor w \rceil_d = \lfloor v \rceil_d$.

Under the above assumptions in the context of a genuinely generated signature, we have shown that $\lfloor w \rceil_d \neq \lfloor v \rceil_d$ occurs if and only if $c \neq 0$ holds[9]. Indeed one can now easily write down an instance of v and e such that a reduction modulo q occurs in the computation of w.

Fix $0 < e \leqslant L$ and set $v = \lfloor q/2 \rfloor - \delta$, where $\delta < e$. Then $v + e > \lfloor q/2 \rfloor$ and $w = v + e - q$, leading to $\lfloor w \rceil_d \neq \lfloor v \rceil_d$. Analogously, let $0 > e \geqslant -L$, then setting $v = -\lfloor q/2 \rfloor + \varepsilon$ for $0 \leqslant \varepsilon < -e$ means that $v + e < -\lfloor q/2 \rfloor$ and $w = v + e + q$ such that, again, $\lfloor w \rceil_d \neq \lfloor v \rceil_d$. If $e = 0$, then always $\lfloor w \rceil_d = \lfloor v \rceil_d$.

When choosing $e \neq 0$ and v as above, w will be within distance L to $-\lfloor q/2 \rfloor$ or $\lfloor q/2 \rfloor$. Write $\lfloor q/2 \rfloor = m \cdot 2^d + r$, where $r = \lfloor q/2 \rfloor \mod 2^d$ such that $-2^{d-1} \leqslant r < 2^{d-1}$ and $m > 0$ since $\lfloor q/2 \rfloor \geqslant 2^d$. Then $[\lfloor q/2 \rfloor]_{2^d} = r$ and if $-2^{d-1} + 2L <$

[9] The additional check in the updated TESLA scheme prevents the case $c \neq 0$ and thus prevents such verification failures.

$r < 2^{d-1} - 2L$, it follows that $|[w]_{2^d}| < 2^{d-1} - L$. This is highly likely to occur in practice and is indeed true for both parameter sets proposed in the Ring-TESLA paper [2], namely, $(q = 8399873, d = 21, L = 814 < r = 5632 < 2^{d-1} = 1048576)$ and $(q = 39960577, d = 23, L = 2766 < r = 3203072 < 2^{d-1} = 4194304)$. Hence all conditions in the Ring-TESLA signing are satisfied and still a signature generated from these values for e and v will fail to verify. This shows that there exist genuine signatures that will be rejected by the verification algorithm. For the second Ring-TESLA parameter set above, we have observed this behavior in practice.

B.2 The probability of verification failures

In this subsection, we show that the probability for a legitimate signing algorithm to generate non-verifying signatures is non-negligible. We can assume that $v \in [-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ is generated uniformly at random during the signing operation. The coefficient e is generated as the sum of ω coefficients that are distributed according to a discrete Gaussian with standard deviation σ. This means that e is distributed according to a discrete Gaussian with standard deviation $\sqrt{\omega}\sigma$. We can also assume that $|e| \leqslant L$. Since these coefficients are selected according to the output of a hash function, we assume that e is independent of v, although v determines part of the input to the hash function.

Then, in order to get a failure $\lfloor w \rceil_d \neq \lfloor v \rceil_d$, v must fall into the interval $[\lfloor q/2 \rfloor - e + 1, \lfloor q/2 \rfloor]$ if $e > 0$ and into the interval $[-\lfloor q/2 \rfloor, -\lfloor q/2 \rfloor - e - 1]$ if $e < 0$. The probabilty for this to happen is $|e|/q$ in either case.

Overall, averaging over the possibilities for e, the probability that either case is satisfied is

$$\Pr(\lfloor w \rceil_d \neq \lfloor v \rceil_d) = 2 \sum_{x=1}^{L} \Pr(e = x) \frac{x}{q},$$

where $\Pr(e = x) = \rho_{\sqrt{\omega}\sigma}(x)/\rho_L = \exp(-x^2/(2\omega\sigma^2))/\rho_L$ with $\rho_L = \rho_{\sqrt{\omega}\sigma}([-L, L]) = 1 + 2\sum_{x=1}^{L} \exp(-x^2/(2\omega\sigma^2)$. Thus, we obtain

$$\Pr(\lfloor \mathbf{w}_i \rceil_d \neq \lfloor \mathbf{v}_i \rceil_d \mid i \in \{1, 2\}) = \frac{4n}{q\rho_L} \sum_{x=1}^{L} x e^{-x^2/(2\omega\sigma^2)}.$$

Evaluating this probability for the first set of Ring-TESLA parameters above yields approximately 0.0097, for the second, one obtains 0.0046. For the latter, we have observed this failure rate exactly in our experiments.

C TESLA♯ algorithms with fresh, per key pair generation of $(\mathbf{a_1}, \mathbf{a_2})$

Algorithm 4 TESLA♯ Key Pairs

INPUT: Parameters n, q, σ, and ω; pseudo-random generator PRG : $\{0,1\}^{256} \to \mathcal{R}_q^2$.
OUTPUT: A private key $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2})$ and a public key $(\mathbf{t_1}, \mathbf{t_2}, \text{seed})$.

seed $\xleftarrow{\$} \{0,1\}^{256}$
$(\mathbf{a_1}, \mathbf{a_2}) \leftarrow \text{PRG}(\text{seed})$
$\mathbf{s} \xleftarrow{\$} D_\sigma^n$
$\mathbf{e_1} \xleftarrow{\$} D_\sigma^n, \quad \mathbf{e_2} \xleftarrow{\$} D_\sigma^n$
$\mathbf{t_1} \leftarrow \mathbf{a_1} * \mathbf{s} + \mathbf{e_1}, \quad \mathbf{t_2} \leftarrow \mathbf{a_2} * \mathbf{s} + \mathbf{e_2}$
return $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2})$, $(\mathbf{t_1}, \mathbf{t_2}, \text{seed})$

Algorithm 5 TESLA♯ Signing

INPUT: public parameters n, q, ω, d, B, U, κ; message $\mu \in \{0,1\}^*$; private key $(\mathbf{s}, \mathbf{e_1}, \mathbf{e_2}) \in \mathcal{R}_q^3$; public key $(\mathbf{t_1}, \mathbf{t_2}, \text{seed})$, where $(\mathbf{t_1}, \mathbf{t_2}) \in \mathcal{R}_q^2$ and seed is a 32-byte string; pseudo-random generator PRG : $\{0,1\}^{256} \to \mathcal{R}_q^2$; hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$, nearly injective mapping $F : \{0,1\}^\kappa \to \mathbb{B}_\omega^n$.
OUTPUT: A signature $(\mathbf{z}, c)$.

$(\mathbf{a_1}, \mathbf{a_2}) \leftarrow \text{PRG}(\text{seed})$
repeat
    $\mathbf{r} \xleftarrow{\$} [-B, B]^n$
    $\mathbf{v_1} \leftarrow \mathbf{a_1} * \mathbf{r}, \quad \mathbf{v_2} \leftarrow \mathbf{a_2} * \mathbf{r}$
    $c \leftarrow H(\lfloor \mathbf{v_1} \rceil_d, \lfloor \mathbf{v_2} \rceil_d, \mu)$
    $\mathbf{c} \leftarrow F(c)$
    $\mathbf{z} \leftarrow \mathbf{r} + \mathbf{s} * \mathbf{c}$
    $\mathbf{w_1} \leftarrow \mathbf{v_1} - \mathbf{e_1} * \mathbf{c}, \quad \mathbf{w_2} \leftarrow \mathbf{v_2} - \mathbf{e_2} * \mathbf{c}$
until $\|\mathbf{z}\|_\infty \leqslant B - U$
    and $\lfloor \mathbf{w_1} \rceil_d = \lfloor \mathbf{v_1} \rceil_d$ **and** $\lfloor \mathbf{w_2} \rceil_d = \lfloor \mathbf{v_2} \rceil_d$
return $(\mathbf{z}, c)$

Algorithm 6 TESLA♯ Verification

INPUT: Public parameters n, q, ω, d, κ; base $(\mathbf{a_1}, \mathbf{a_2}) \in \mathcal{R}_q^2$; message $\mu \in \{0,1\}^*$; public key $(\mathbf{t_1}, \mathbf{t_2}, \text{seed})$, where $(\mathbf{t_1}, \mathbf{t_2}) \in \mathcal{R}_q^2$ and seed is a 32-byte string; signature $(\mathbf{z}, c)$; pseudo-random generator PRG : $\{0,1\}^{256} \to \mathcal{R}_q^2$; hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$; nearly injective mapping $F : \{0,1\}^\kappa \to \mathbb{B}_\omega^n$.
OUTPUT: $\{0,1\}$ ▷ reject, accept

$(\mathbf{a_1}, \mathbf{a_2}) \leftarrow \text{PRG}(\text{seed})$
$\mathbf{c} \leftarrow F(c)$
$\mathbf{w_1} \leftarrow \mathbf{a_1} * \mathbf{z} - \mathbf{t_1} * \mathbf{c}, \quad \mathbf{w_2} \leftarrow \mathbf{a_2} * \mathbf{z} - \mathbf{t_2} * \mathbf{c}$
$c' \leftarrow H(\lfloor \mathbf{w_1} \rceil_d, \lfloor \mathbf{w_2} \rceil_d, \mu)$
return if $c' = c$ **and** $\|\mathbf{z}\|_\infty \leqslant B - U$ **then** 1 **else** 0