TACHYON: Fast Signatures from Compact Knapsack

Rouzbeh Behnia
Oregon State University
Corvallis, Oregon
behniar@oregonstate.edu

Muslum Ozgur Ozmen
Oregon State University
Corvallis, Oregon
ozmenmu@oregonstate.edu

Attila A. Yavuz*
University of South Florida
Tampa, Florida
attilaayavuz@usf.edu

Mike Rosulek
Oregon State University
Corvallis, Oregon
rosulekm@eecs.oregonstate.edu

ABSTRACT

We introduce a simple, yet efficient digital signature scheme which offers post-quantum security promise. Our scheme, named TACHYON, is based on a novel approach for extending one-time hash-based signatures to (polynomially bounded) many-time signatures, using the additively homomorphic properties of generalized compact knapsack functions. Our design permits TACHYON to achieve several key properties. First, its signing and verification algorithms are the fastest among its current counterparts with a higher level of security. This allows TACHYON to achieve the lowest end-to-end delay among its counterparts, while also making it suitable for resource-limited signers. Second, its private keys can be as small as κ bits, where κ is the desired security level. Third, unlike most of its lattice-based counterparts, TACHYON does not require any Gaussian sampling during signing, and therefore, is free from side-channel attacks targeting this process. We also explore various speed and storage trade-offs for TACHYON, thanks to its highly tunable parameters. Some of these trade-offs can speed up TACHYON signing in exchange for larger keys, thereby permitting TACHYON to further improve its end-to-end delay.

KEYWORDS

Digital signatures; post-quantum security; authentication

1 INTRODUCTION

Ever since Shor [50] published polynomial-time *quantum* algorithms for factoring and discrete logarithm, the threat of quantum computation has loomed ominously over public-key cryptography. Since traditional public-key cryptography is broken by quantum attacks, alternative schemes with *post-quantum (PQ) security* must be identified before quantum computers become practical.

Recently, the NSA has announced an advisory on the possibility of transitioning to PQ-secure cryptography in the near future [43]. To avoid a hasty transition from current conventional cryptosystems to PQ-secure systems, NIST has already initiated the first round of standardizations for PQ cryptography[1].

1.1 The State of the Art and Limitations

Lamport [33] proposed the first PQ-secure one-time signature scheme based on the idea of committing to secret keys via one-way functions. Later, Bos and Chaum [12] and Reyzin and Reyzin [48] proposed different variants of Lamport's signature with the aim of minimizing the public key and signature size, respectively. Today, digital signatures based on lattices, hash functions, codes, multivariates and symmetric primitives are the leading practical candidates with PQ security.

• *Lattice-based Signatures*: There are two main categories of lattice-based signature schemes. One is focusing on hardness of worst-case to average-case problems with standard lattices (e.g., [36, 46]). While they provide a strong security, they suffer from very large parameter sizes (in the orders of a few MBs). Another direction, with more focus on efficiency, is based on ring analogs of standard lattice problems (e.g., [1, 18, 19]). Most of these efficient schemes, however, suffer from costly sampling operations with high precision over some normal distribution (e.g., Gaussian sampling) during the signing. Relaxation of this requirement, by only sampling over integers, permitted more efficient constructions like BLISS [18], which is based on the Fiat-Shamir transform [22]. Later, Ducas et al. proposed a hash-and-sign signature scheme [20] that has a smaller signature and key size than BLISS, but with slower signing due to expensive discrete Gaussian sampling over a lattice.

Gaussian sampling not only incurs a performance penalty, but its implementation is also prone to side-channel attacks. For instance, BLISS [18] has been targeted with a number of side-channel attacks [21, 25]. At the moment, avoiding such side channels in implementation is considered to be highly challenging and error-prone [19].

Recently, Ducas et al. proposed a new Fiat-Shamir-based scheme called Dilithium [19], which avoids Gaussian sampling during signing. The security of Dilithium is based on the learning with errors (LWE) and short integer solution (SIS) problems in ideal lattices.

qTESLA [11] is another lattice-based signature scheme proposed to the first round of NIST standardization for PQ cryptography. qTESLA is based on the decisional ring learning with errors (R-LWE) problem. While similar to Dilithium [19] , qTESLA avoids using Gaussian Sampling during signature generation, but it suffers from a higher end-to-end delay.

pqNTRUSign [27] is an instantiation of modular lattice signature (over the NTRU lattice). Signatures can be generated using a (bimodal) Gaussian or a uniform sampler. Similar to Dilithium [19],

*Work done in part while Attila A. Yavuz was at Oregon State University, Corvallis, OR.

[1] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

pqNTRUSign employs rejection sampling to avoid the leakage of the private key components. However, with the current suggested parameters, the scheme suffers from a high signing time that is due to the high rejection rate.

While other lattice-based primitives, such as key-exchange protocols, have undergone some real-world testing and evaluations (e.g., [13]), the current precarious state of lattice-based approaches has hindered the development of PQ-secure signatures.

• *Hash-Based Signatures*: Hash-based signatures can be proven secure in the standard model under the very well-studied properties of hash functions such as pre-image resistance. The combination of Merkle trees [40] with early one-time hash-based signatures (e.g., Lamport [33]) results in very efficient stateful schemes which are secure for a number of signatures. Traditional hash-based schemes are *stateful*, to ensure that the signer does not reuse some of the private key materials. Recently, stateless signatures (e.g., SPHINCS [10]) have been proposed. SPHINCS has a tight security reduction to the security of its building blocks such as hash functions and PRNGs. Unfortunately, these schemes have large signatures ($\approx$ 41 KB) and very costly signature generation, especially on low-end devices [29].

• *Code-Based Signatures*: Code-based cryptography has been largely affected by the Syndrome Decoding Problem [9]. Since McEliece cryptosystem [39], which is based on binary Goppa codes, there have been a lot of efforts in balancing security and efficiency of such systems. The most well-studied and provably secure approach to obtain signature schemes is applying the Fiat-Shamir transform [22] on the identification scheme proposed by Véron [53] and Stern [51]. pqsigRM [34] is a new code-based signature scheme based on punctured Reed-Muller (RM) submitted to the first NIST post-quantum standardization conference. pqsigRM can be considered as a highly improved version of the scheme in [17], where most of the improvements are due to the replacement of Goppa Codes in [17] with punctured RM codes. While pqsigRM has significantly improved the overall parameters sizes in [17], the key sizes are still larger than its lattice-based and hash-based counterparts.

• *Multivariate-Based Signatures*: There are a number of multivariate-based signatures submitted to the NIST standardization of PQ cryptography. For instance, GeMSS [14] can be considered as an improvement of its predecessor QUARTZ [44], that is based on the Hidden Field Equations cryptosystems. GeMSS enjoys from an efficient verification algorithm and very compact signatures, however, the signing algorithm is significantly slower than its hash-based counterparts (e.g., SPHINCS+ [28]).

• *Symmetric Key Based Signatures*: PICNIC [15] is another novel construction which is based on the problems related to symmetric key cryptography. PICNIC is obtained by applying the Fiat-Shamir transform on an efficient zero-knowledge proof which results in very short public key and private key sizes. However, the scheme suffers from large signature sizes with relatively slow (as compared to lattice-based schemes) signing and verification algorithms.

1.2 Our Contribution

We propose a simple and efficient PQ-secure signature scheme, TACHYON, based on well-studied primitives. We outline a comparison between TACHYON and some of its other PQ-secure counterparts in Table 2 (see Section 5), and further elaborate on its desirable properties below:

• *New Algorithmic Design*: TACHYON can be viewed as a novel modification of the HORS construction [48], which is based on one-way functions. We harness the HORS approach with the **generalized compact knapsack (GCK)** of Micciancio [41]. The *additively homomorphic* property of GCK provides two benefits: It allows us to compress the signature size as compared to one-time signatures, and more importantly, it leads to a totally new paradigm for extending few-time hash-based signatures to *stateless* schemes supporting polynomially-bounded number of signatures.

The security of our scheme is based on the one-wayness of GCK function family. These properties reduce to the worst-case hardness of problems in cyclic lattices [37, 41].

• *Improved Side-Channel Resiliency*: It has been shown that Gaussian sampling is prone to side-channel attacks (e.g., [25, 47]). Since side channels are a property of an algorithm's implementation, they can be somewhat mitigated with suitable implementation techniques. However, the process of eliminating side channels in Gaussian sampling algorithms (e.g., in BLISS [18]) is known to be arduous and error-prone [19]. TACHYON does not require any variants of Gaussian sampling. Instead, it uses uniform sampling over a bounded domain, and rejection sampling to check for an outputted signature to be in a safe range.

• *Fast Verification*: The verification algorithm of TACHYON is very efficient, involving only two hash function calls, a GCK one-way function call, and vector additions. This makes TACHYON the most verifier computationally efficient alternative among its counterparts. For example, using TACHYON with 256-bit security, it is possible to verify 35,714 messages per second on commodity hardware (e.g., Intel 6th generation i7 processor), which is up to 3.7× faster than Dilithium [19], one of its fastest alternatives.

• *Fast Signing*: Signature generation of TACHYON does not require any costly operations (e.g., Gaussian sampling) but only a GCK function call (which is demonstrated to be fast [38]), along with a small constant number of pseudorandom function (PRF) calls and a small number of vector additions. This makes the signature generation of TACHYON the fastest as compared to its counterparts.

• *Small Private Key*: The private keys in TACHYON are as small as κ-bit, which is the smallest among existing PQ-secure schemes. Furthermore, unlike some other schemes (e.g., [18]), the signer does not need to store a pre-computed table to be used in the sampling process. Along with the signer computational efficiency, this property makes TACHYON a feasible alternative for low-end devices.

• *Tunable Parameters*: Our new algorithmic design allows us to offer various speed and storage trade-offs based on the parameter

choices. For instance, one can pre-compute and store some intermediate values at the signer's side in exchange for a faster signing, reduce the public key and/or signature size but with an increase in the end-to-end delay, or increase the signature size to offer lower rejection sampling rates for a faster signing. Some of these possible trade-offs are further elaborated in Subsection 5.2.

Limitations: All of these desirable properties of TACHYON come at the cost of a larger public key. For instance, the public key in TACHYON-256 is as large as 2976 KB, whereas it is only 1760 bytes in Dilithium[19]. Yet, we believe there are many use-cases where storing a larger public key is tolerable. For instance, a resourceful command center that verifies a large number of signatures from sensors can store such a public key. However, if the verifier is strictly memory-limited and cannot afford to store large public keys, then schemes with a smaller public key, such as Dilithium, should be considered.

2 PRELIMINARIES

Notation. We work over a ring $R = \mathbb{Z}_q[x]/(f)$ (in this paper $f(x) = (x^N + 1)$), where N is a power of two, and q is a prime such that $1 \equiv q \bmod 2N$. We denote vectors as bold letters (i.e., $\mathbf{v}$), while scalars are denoted as non-bold letters (i.e., u). $x \xleftarrow{\$} \mathcal{S}$ denotes that x is being randomly selected from set $\mathcal{S}$. $|x|$ denotes the bit length of a number x, i.e., $|x| = \log_2 x$. $\mathcal{A}^{O_1 \dots O_n}(\cdot)$ denotes algorithm $\mathcal{A}$ is provided with access to oracles $O_1 \dots O_n$. For a vector $\mathbf{w} = (w_1, \dots, w_N)$ we define $\|\mathbf{w}\|_\infty = \max\{|w_i| : i = 1, \dots, N\}$.

2.1 Digital Signatures

Definition 2.1. *A digital signature scheme is a tuple of three algorithms* SGN = (Kg, Sig, Ver) *defined as follows.*

- $(sk, PK) \leftarrow \mathsf{SGN.Kg}(1^\kappa)$: Given the security parameter κ, it outputs a private/public key pair (sk, PK).
- $\sigma \leftarrow \mathsf{SGN.Sig}(M, sk)$: Given a message M and private key sk, it outputs a signature σ.
- $\{0,1\} \leftarrow \mathsf{SGN.Ver}(M, \sigma, PK)$: Given a message-signature pair (M, σ), and PK, it outputs $b \in \{0,1\}$.

We say that SGN *is correct if for all* $(sk, PK) \leftarrow \mathsf{SGN.Kg}(1^\kappa)$, $\mathsf{SGN.Ver}(M, \mathsf{SGN.Sig}(M, sk), PK) = 1$ *holds.*

We define security using the code-based games methodology of Bellare & Rogaway [8]. A **game** $\mathcal{G}$ is a collection of stateful oracles/functions. Given an adversary $\mathcal{A}$, the interaction $\mathcal{G}^{\mathcal{A}}$ refers to the following: (1) the Initialize function of the game is run, and its output given as input to $\mathcal{A}$. (2) $\mathcal{A}$ may invoke any of the functions of $\mathcal{G}$. (3) When $\mathcal{A}$ terminates, its output is given to the Finalize function of $\mathcal{G}$. The output of Finalize is the output of the interaction $\mathcal{G}^{\mathcal{A}}$.

Definition 2.2. *Existential Unforgeability under Chosen Message Attack (EU-CMA) [30] (in the random oracle model [7]) is defined in terms of the game* $\mathcal{G}[\mathsf{SGN}]$ *in Algorithm 1. The EU-CMA advantage of* $\mathcal{A}$ *is defined as*

$$Adv^{EU\text{-}CMA}_{\mathsf{SGN},\mathcal{A}} = \Pr[\mathcal{G}[\mathsf{SGN}]^{\mathcal{A}} = 1]$$

We say that $\mathcal{A}$ $(t_{\mathcal{A}}, q_S, q_H, \epsilon_{\mathcal{A}})$*-breaks the EU-CMA of* SGN *if it makes at most* q_S *and* q_H *signature and hash queries (respectively) and runs in time at most* $t_{\mathcal{A}}$ *where* $Adv^{EU\text{-}CMA}_{\mathsf{SGN},\mathcal{A}} \geq \epsilon_{\mathcal{A}}$*, and we say that* SGN *is* $(t_{\mathcal{A}}, q_S, q_H, \epsilon_{\mathcal{A}})$*-secure if no algorithm* $\mathcal{A}$ $(t_{\mathcal{A}}, q_S, q_H, \epsilon_{\mathcal{A}})$*-breaks it.*

Algorithm 1 EU-CMA game $\mathcal{G}[\mathsf{SGN}]$ for a signature scheme SGN, in the random oracle model. Algorithms of SGN are allowed to query oracle H.

function Initialize
  $(sk, PK) \leftarrow \mathsf{SGN.Kg}(1^\kappa)$
  return PK
function $H(q)$
  if $\mathcal{L}[q]$ is not defined **then**
    $a \xleftarrow{\$} \{0,1\}^\kappa$
    $\mathcal{L}[q] \leftarrow a$
  return $\mathcal{L}[q]$
function Sig(M)
  add M to set $\mathcal{M}$
  return $\mathsf{SGN.Sig}(M, sk)$
function Finalize(M^*, σ^*)
  return $[M^* \notin \mathcal{M}] \wedge [\mathsf{SGN.Ver}(M^*, \sigma^*, PK) = 1]$

2.2 Forking Lemma

The security model of TACHYON is in Random Oracle Model (ROM) [7], and also it relies on Generalized Forking Lemma (GFL) [6]. GFL is a commonly used technique in the security proof of various well-studied digital signature schemes (e.g., Schnorr [49]). Intuitively, GFL states that if an adversary can successfully generate a forgery, then it is possible to rewind the adversary, choose new random oracle responses after a certain point, and the adversary will still be able to generate a forgery with polynomially-related probability.

Lemma 2.1. *(General Forking Lemma [6]) Fix an integer* $q_F \geq 1$ *and a set* H *of size* $h_F \geq 2$*. Let* A *be a randomized algorithm that returns a pair* (J, σ) *where* $J \in \{0, \dots, h_{q_F}\}$ *and* σ *is the side output, on the input of* $(x, h_1, \dots, h_{q_F})$ *. For* IG *as a randomized input generator, the accepting probability of* A *(ACC) is defined as the probability that* $J \geq 1$ *in* $x \xleftarrow{\$} \mathsf{IG}; (h_1, \dots, h_{q_F}) \xleftarrow{\$} H; (J, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, \dots, h_{q_F})$*.*

The forking algorithm $\mathsf{Fork}_{\mathsf{A}}$ *associated with* A *is a randomized algorithm that behaves as in Algorithm 2. For* $FRK = \Pr[b = 1 : x \xleftarrow{\$} \mathsf{IG}; (b, \sigma, \sigma') \xleftarrow{\$} \mathsf{Fork}_{\mathsf{A}}(x)]$*, then* $FRK \geq ACC \cdot (\frac{ACC}{q_F} - \frac{1}{h_F})$*.*

2.3 Generalized Compact Knapsack

Our scheme uses the generalized compact knapsack (GCK) function family, introduced by Micciancio [41].

Definition 2.3 ([41]). *For a ring* R*, and a small integer* $\mu > 1$*, the generalized compact knapsack function family is the set of functions of the form* $F_{\mathbf{A}} : R^\mu \to R$*, where:*

$$F_{\mathbf{A}}(\mathbf{b}_1, \dots, \mathbf{b}_\mu) = \sum_{i=1}^{\mu} \mathbf{b}_i \cdot \mathbf{a}_i$$

An instance of this family is specified by μ *fixed elements* $\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_\mu) \in R^\mu$*. These elements are to be chosen randomly and*

Algorithm 2 Forking algorithm $\mathsf{Fork}_{\mathsf{A}}$ for the forking lemma.

1: Pick coins ρ for A at random.
2: $(h_1, \ldots, h_{q_F}) \xleftarrow{\$} H$
3: $(I, \sigma) \leftarrow \mathsf{A}(x, h_1, \ldots, h_{q_F}; \rho)$
4: If $I = 0$ then return $(0, 0, 0)$
5: $(h'_1, \ldots, h'_{q_F}) \xleftarrow{\$} H$
6: $(I', \sigma') \leftarrow \mathsf{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_{q_F}; \rho)$
7: If ($I = I'$ and $h_I \neq h'_I$, return $(1, \sigma, \sigma')$
8: Else, return $(0, 0, 0)$

independently. The inputs $\mathbf{b}_1, \ldots, \mathbf{b}_\mu$ *are polynomials over* R *where* $\|\mathbf{b}_i\|_\infty \leq \beta$ *for* $i \in \{1, \ldots, \mu\}$ *and some positive integer* β.

For the detailed security analysis of GCK function, we refer an interested reader to [37, 41, 42, 45]. We give the required parameters to securely instantiate GCK function in TACHYON in Subsection 4.1.

2.4 Bos-Chaum signatures

Since TACHYON is inspired by the construction of Bos and Chaum (BC) signature scheme which uses a bijective function $S(\cdot)$ and a one-way function (OWF) $f(\cdot)$ [12], we briefly explain about a simple generalization of their construction in the following.

Definition 2.4. BC *signature scheme consists of three algorithms* BC = (Kg, Sig, Ver) defined as follow.

- $(sk, PK) \leftarrow \mathsf{BC.Kg}(1^\kappa)$ Given the security parameter 1^κ it sets t, k and l and generate t random $l-$bit values for the private key $(x_1, \ldots, x_k)$ and compute the public key components (y_i) as the image of the private key components x_i with respect to a one-way function $f(\cdot)$, i.e., $y_i \leftarrow f(x_i)$ where $i \in \{1, \ldots, t\}$. Finally set $sk \leftarrow (x_1, \ldots, x_t)$ and $PK \leftarrow (t, k, \langle y_1, \ldots, y_t \rangle)$.
- $\sigma \leftarrow \mathsf{BC.Sig}(M, sk)$: Given a b-bit message M and sk, interpret M as an integer between 0 and $2^b - 1$ and set $(i_1 \ldots, i_k)$ as the $M-$th $k-$element subset of set $(1, 2, \ldots, t)$, computed as $S(M)$. Output the signature as $\sigma \leftarrow (x_{i_1}, \ldots, x_{i_k})$.
- $\{0, 1\} \leftarrow \mathsf{BC.Ver}(M, \sigma, PK)$: Given a message-signature pair $(M, \sigma = \langle x'_1, x'_2, \ldots, x'_k \rangle)$, interpret M as an integer between 0 and $2^b - 1$ and set $(i_1 \ldots, i_k)$ as the $M-$th $k-$element subset of set $(1, 2, \ldots, t)$, computed as $S(M)$. It the checks if $\{y_{i_j} = f(x'_j)\}_{j=1}^{j=k}$ holds, it outputs 1, else it outputs 0.

3 PROPOSED SCHEME

3.1 TACHYON

Our conceptual starting point is the HORS construction [48], which itself is a variant of the Bos and Chaum scheme [12]. The private key consists of many random values $\mathbf{x}_i$, and the public key consists of corresponding images $\mathbf{y}_i = F(\mathbf{x}_i)$, where F is a one-way function. Of course, the $\mathbf{x}_i$ values can be derived from a small seed using a PRF (this feature is preserved by TACHYON, and leads to a minimal signing key). To sign a message M, the signer first computes $H_2(M)$ and interprets it as a sequence of indices $(i_1, \ldots, i_k)$. The signature then consists of $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$. To verify, one can simply compare $F(\mathbf{x}_j)$ to the public key value $\mathbf{y}_j$, for each relevant j.

Our novel departure from this paradigm is to use an **additively homomorphic** OWF F. Specifically, we choose the generalized compact knapsack (GCK) function family of Micciancio [41]. This allows the signature to be compressed, as follows. Instead of $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$, the signature can contain only $\mathbf{s} = \sum_j \mathbf{x}_{i_j}$. The verifier can then check that $F(\mathbf{s}) = \sum_j \mathbf{y}_{i_j}$.

However, this approach leaks a linear combination of the secret key material. After a moderate number of signatures, it would be possible to solve for the entire secret key via a system of linear equations. To thwart this, we add some "noise". Specifically, the signature consists of $\mathbf{s} = \sum_j \mathbf{x}_{i_j} + \mathbf{r}'$ for a suitably distributed $\mathbf{r}'$.

There are two challenges when adding this noise. First, we must make sure the verifier can still verify such a signature. This can be achieved by giving out $F(\mathbf{r}')$ in the signature. Since the output of F is long, we instead give out a short hash $H_1(F(\mathbf{r}'))$.

Second, the GCK-OWF is defined over some ring but can only accept inputs that are "short" — i.e., the inputs come from a subset of the ring that are not closed under the homomorphic operation. This makes it challenging to mask the sensitive sum $\sum_j \mathbf{x}_{i_j}$. We use the following rejection-sampling approach proposed by Lyubashevsky [35]. Sample the noise $\mathbf{r}'$ from a suitable uniform distribution, and restart the entire signing algorithm if the result $\sum_j \mathbf{x}_{i_j} + \mathbf{r}'$ is "too large" or "too small". More details about this rejection sampling process are given in Subsection 3.2.

Finally, instead of choosing indices $i_1, \ldots, i_k$ as $H_2(M)$ as in HORS, we choose them as $H_2(M \| h)$ where $h = H_1(F(\mathbf{r}'))$. Intuitively, this ensures that the value $\mathbf{r}'$ is "committed" before the rest of the signature is generated. This aspect of the scheme is used in the security proof, specifically in our use of the generalized forking lemma (Lemma 2.1). The rewinding argument of the forking lemma implies that any adversary generating a forgery in our scheme can be rewound to output two forgeries with the same h. From these two forgeries, we can break the one-wayness of F.

Details. The formal description of the TACHYON scheme is given in Algorithm 3.

$F_{\mathbf{A}}$ refers to the GCK one-way function discussed in Subsection 2.3. Its input is a vector from R^μ and its output is a vector in R, where R is a suitable ring and μ is a small integer. The GCK function is parameterized by a public value $\mathbf{A}$, which is to be chosen randomly. The random choice of $\mathbf{A}$ ensures the one-wayness of $F_{\mathbf{A}}$ [35, 41]. As such, it may be a global parameter (i.e., shared among all users).

$\mathsf{Samp}(\gamma)$ samples a uniform distribution over vectors in R^μ with all entries in the range $[-\gamma, \gamma]$. This function can easily be implemented with a PRF or PRG, similar to other lattice-based constructions that uses uniform sampling (e.g., Dilithium [19]).

PRF refers to a pseudorandom function whose output is interpreted as a binary (0/1) vector of R^μ (i.e., an input to $F_{\mathbf{A}}$).

ξ and ρ are parameters related to both the security of the GCK-OWF (controlling the weight of its inputs) as well as the probabilities surrounding rejection sampling (discussed further in Subsection 3.2).

H_1 is a random oracle with output length l_1, used to commit the signature to $\mathbf{r}'$ before choosing the HORS indices. H_2 is a random oracle with output length $l_2 = k|t|$ used to choose HORS indices.

We write $\langle i_1 \| \cdots \| i_k \rangle \leftarrow H_2(M\|h)$ to mean that the output of H_2 is interpreted as a sequence of k indices, each $|t|$ bits long.

Algorithm 3 TACHYON signature scheme

TACHYON.Kg(1^κ):

1: $sk \xleftarrow{\$} \{0,1\}^\kappa$
2: $\mathbf{x}_i \leftarrow \mathsf{PRF}(sk, i)$, **for** $i = 1, \ldots, t$
3: $\mathbf{y}_i \leftarrow F_{\mathbf{A}}(\mathbf{x}_i)$, **for** $i = 1, \ldots, t$
4: **return** sk, $PK \leftarrow (t, k, \langle \mathbf{y}_1, \ldots, \mathbf{y}_t \rangle)$

TACHYON.Sig(M, sk):

1: $\mathbf{r}' \xleftarrow{\$} \mathsf{Samp}(\xi - 1)$, $\mathbf{r} \leftarrow F_{\mathbf{A}}(\mathbf{r}')$
2: $h \leftarrow H_1(\mathbf{r})$
3: $\langle i_1 \| \cdots \| i_k \rangle \leftarrow H_2(M\|h)$
4: $\mathbf{x}_{i_j} \leftarrow \mathsf{PRF}(sk, i_j)$, **for** $j = 1, \ldots, k$
5: $\mathbf{s} \leftarrow (\sum_{j=1}^{k} \mathbf{x}_{i_j}) + \mathbf{r}'$
6: **if** $\|\mathbf{s}\|_\infty \geq (\xi - \rho)$ **then goto** step 1
7: **return** $\sigma \leftarrow (\mathbf{s}, h)$

TACHYON.Ver(M, σ, PK):

1: parse σ as $(\mathbf{s}, h)$, and PK as $(t, k, \langle \mathbf{y}_1, \ldots, \mathbf{y}_t \rangle)$
2: **if** $\|\mathbf{s}\|_\infty \geq (\xi - \rho)$ **then return** 0
3: $\langle i_1 \| \cdots \| i_k \rangle \leftarrow H_2(M\|h)$
4: $\tilde{\mathbf{r}} \leftarrow F_{\mathbf{A}}(\mathbf{s}) - \sum_{j=1}^{k} \mathbf{y}_{i_j}$
5: **if** $H_1(\tilde{\mathbf{r}}) = h$ **then return** 1 **else return** 0.

Correctness: TACHYON algorithm is correct in the sense that a signature generated via TACHYON.Sig($\cdot$) will always be verified by TACHYON.Ver($\cdot$). This can be shown as follows:

Given a message-signature pair $(M, \sigma = \langle \mathbf{s}, h \rangle)$, due to the deterministic property of the hash oracle $H_2(\cdot)$ the indexes created in TACHYON.Sig($\cdot$) by computing $\langle i_1 \| \cdots \| i_k \rangle \leftarrow H_2(M\|h)$ are identical to those created in TACHYON.Ver($\cdot$). Therefore, given the public key $PK \leftarrow (t, k, \langle \mathbf{y}_1, \ldots, \mathbf{y}_t \rangle)$,

$$\begin{aligned} F_{\mathbf{A}}(s) - \sum_{j=1}^{k} \mathbf{y}_{i_j} &= F_{\mathbf{A}}((\sum_{j=1}^{k} \mathbf{x}_{i_j}) + \mathbf{r}') - \sum_{j=1}^{k} \mathbf{y}_{i_j} \\ &= F_{\mathbf{A}}(\sum_{j=1}^{k} \mathbf{x}_{i_j}) + F_{\mathbf{A}}(\mathbf{r}') - \sum_{j=1}^{k} F_{\mathbf{A}}(\mathbf{x}_{i_j}) \\ &= F_{\mathbf{A}}(\mathbf{r}') \end{aligned}$$

Therefore, for a valid message-signature pair $(M, \sigma = \langle \mathbf{s}, h \rangle)$, Step 5 in Algorithm 3 will always return 1.

3.2 Rejection Sampling

The idea of rejection sampling in lattices was first proposed by Lyubashevsky [35] in the construction of identification schemes. In our scheme, we need to mask the summation of secret keys $(\sum_j \mathbf{x}_{i_j})$ with a random $\mathbf{r}'$. If $\mathbf{r}'$ is uniform over the entire ring (on which the summation is defined), then clearly all information about the summation is hidden. However, the verifier must use $\mathbf{s} = \sum_j \mathbf{x}_{i_j} + \mathbf{r}'$ as input to $F_{\mathbf{A}}$, which is only possible if $\mathbf{s}$ is small. Hence, $\mathbf{r}'$ must be chosen from some bounded distribution. We now discuss how that distribution is determined.

The $\mathbf{x}_i$ vectors are chosen with coefficients from $\{0, 1\}$. One can easily compute a bound ρ such that

$$\Pr\left[\text{for all subsets } S \text{ with } |S| \leq k\text{: } \|\textstyle\sum_{i \in S} \mathbf{x}_i\|_\infty < \rho\right]$$

is very high, over the choice of the $\mathbf{x}_i$ values. The rest of the analysis conditions on this highly likely event, and we assume that each coefficient a of $\sum_j \mathbf{x}_{i_j}$ is in the range $a \in [-(\rho - 1), \rho - 1]$.

Now we choose $\mathbf{r}'$ uniformly with each coefficient in the range $[-(\xi - 1), \xi - 1]$ and set $\mathbf{s} = \sum_j \mathbf{x}_{i_j} + \mathbf{r}'$. This causes each coefficient of $\mathbf{s}$ to be uniform in a range $[a - (\xi - 1), a + \xi - 1]$ for some $a \in [-(\rho - 1), \rho - 1]$, which depends on the signing key. No matter what a is, the range $[a-(\xi-1), a+\xi-1]$ always contains $[-(\xi-\rho-1), \xi-\rho-1]$ as a subrange. Therefore if we condition on all coefficients falling in this subrange, the resulting value is uniform and independent of the signing key. We can achieve this conditioning by rejection sampling, and simply retrying if $\|\mathbf{s}\| \geq \xi - \rho$.

The parameter ξ must be chosen carefully, since larger ξ leads to larger signatures, but smaller ξ leads to more failures/retries during rejection sampling. We can compute the probability of rejection by considering each component of $\mathbf{s}$ in isolation. The coefficient is chosen uniformly from some range $[a-(\xi-1), a+\xi-1]$, which has $2\xi-1$ values. The "permissible" outcomes are $[-(\xi-\rho-1), \xi-\rho-1]$, a range of $2(\xi-\rho)-1$ values. Hence the probability that this coefficient is permissible is $\frac{2(\xi-\rho)-1}{2\xi-1} = 1 - \frac{2\rho}{2\xi-1}$. With μN coefficients in $\mathbf{s}$, the sampling success probability is therefore

$$\left(1 - \frac{2\rho}{2\xi - 1}\right)^{\mu N} \approx \mathrm{e}^{-N\mu\rho/\xi}$$

4 SECURITY ANALYSIS

In the random oracle model [7], we prove that TACHYON is *EU-CMA* in Theorem 4.1 below. Note that in our proof, we ignore terms that are negligible in terms of our security parameter.

Theorem 4.1. *In the random oracle model, if there exists an adversary $\mathcal{A}$ that can $(t_{\mathcal{A}}, q_S, q_H, \epsilon_{\mathcal{A}})$-break the EU-CMA security of* TACHYON*, then one can build another algorithm $\mathcal{B}$, that can break the one-wayness of the GCK function family (as defined in Definition 2.3) with success probability of at least*

$$\frac{1}{t}\left[\left(\epsilon_{\mathcal{A}} - \frac{q_H(q_S + q_H)}{2^{l_1}}\right)\left(\frac{\epsilon_{\mathcal{A}}}{q_H} - \frac{q_S + q_H}{2^{l_1}} - \frac{1}{2^{l_2}}\right) - \frac{q_H k!}{2^{l_2}}\right]$$

and running in time at most

$$O\big(2t_{\mathcal{A}} + t(t_{RNG} + t_{F_{\mathbf{A}}}) + q_S(2t_{RNG} + t_{F_{\mathbf{A}}} + kt_{Add}) + q_H t_{RNG}\big)$$

where t_{RNG}, t_{Add} and $t_{F_{\mathbf{A}}}$ are the running time of a random number generator, vector addition and $F_{\mathbf{A}}$ function, respectively.

The intuition behind the reduction is as follows. The reduction algorithm receives a value $\mathbf{y}^*$ and attempts to find a preimage of $\mathbf{y}^*$ under $F_{\mathbf{A}}$. The reduction algorithm plays the role of the challenger (EU-CMA game) against $\mathcal{A}$, and uses $\mathbf{y}^*$ as one of the public-key components $\mathbf{y}_{j^*}$, for random index j^*. It chooses all other public-key components $\mathbf{y}_i$ honestly.

The reduction algorithm does not know the entire signing key (it does not know $\mathbf{x}_{j^*}$), so it uses its ability to program the random oracle to generate simulated signatures. Specifically, it chooses the

signature $(\mathbf{s}, h)$ uniformly at random, and then programs H_1 and H_2 so that the signature verifies.

Suppose $\mathcal{A}$ successfully constructs a forgery $(\mathbf{s}, h)$. Consider rewinding the adversary to the point where it made the query $H_2(M\|h)$, then continuing with independent randomness. The forking lemma states that, with good probability, the adversary will output a forgery $(\mathbf{s}', h)$ in this case as well. Importantly, the new forgery will include the same h, hence:

$$h = H_1\left(F_{\mathbf{A}}(\mathbf{s}) - \sum_{j\in I}\mathbf{y}_j\right) = H_1\left(F_{\mathbf{A}}(\mathbf{s}') - \sum_{j\in I'}\mathbf{y}_j\right)$$

Note that the two summations are over different multisets I, I' of indices.

Conditioning on the absence of a collision in H_1, we have

$$F_{\mathbf{A}}(\mathbf{s}) - \sum_{j\in I}\mathbf{y}_j = F_{\mathbf{A}}(\mathbf{s}') - \sum_{j\in I'}\mathbf{y}_j$$

Say that I and I' are **compatible** if there is some index that appears with multiplicity exactly once in $I \cup I'$. Our reduction conditions on the fact that I and I' are always compatible. With independent probability $1/t$, we have that I and I' are actually compatible with respect to our special index j^*. Compatibility implies that we can solve for $\mathbf{y}^*$. Let's say $j^* \in I \setminus I'$, then:

$$\mathbf{y}^* = F_{\mathbf{A}}(\mathbf{s}') - F_{\mathbf{A}}(\mathbf{s}) + \sum_{j\in I\setminus\{j^*\}}\mathbf{y}_j - \sum_{j\in I'}\mathbf{y}_j$$

The reduction algorithm knows the preimages to all $\mathbf{y}_j$ terms on the right-hand side. It is therefore possible to apply the homomorphic property of $F_{\mathbf{A}}$ and write the right-hand side as $F_{\mathbf{A}}$ applied to a value known to the reduction algorithm. In other words, the reduction can compute a preimage of $\mathbf{y}^*$.

Compatible index sets. Before describing the reduction in more detail, we clarify the properties of compatible index sets.

Definition 4.2. Let I, I' be strings which encode multisets in the natural way as $I = \langle i_1\| \cdots \|i_k\rangle$, etc. We say that I and I' are **compatible with respect to** i if i appears with multiplicity 1 in I and multiplicity 0 in I' (or vice-versa). We say that I and I' are **compatible** if they are compatible for some value i.

Each I encodes k indices. In the worst case there are at most $k!$ other strings that encode a multiset that is *incompatible* with I. If we have one fixed string I^* and q other uniformly chosen strings $I_1, \ldots, I_q$ (all strings with l_2 bits)

$$\Pr[I^* \text{ is compatible with all } I_1, \ldots, I_q] \geq \left(1 - \frac{k!}{2^{l_2}}\right)^q \geq 1 - \frac{q\cdot k!}{2^{l_2}}$$

And hence:

$$\Pr[I^* \text{ is } \textbf{not} \text{ compatible with all } I_1, \ldots, I_q] \leq \frac{q\cdot k!}{2^{l_2}}$$

We abbreviate the latter probability as $\Pr[\overline{\mathsf{Compat}(q,k,l_2)}]$.

Reduction algorithm. Given an adversary $\mathcal{A}$, we define the reduction algorithm/game $\mathcal{B}$ in Algorithm 4. $\mathcal{B}$ takes $\mathbf{y}^*$ (an $F_{\mathbf{A}}$-output) as input, as well as a list $\mathcal{H}$ of random oracle responses that it will use to program H_2. This interface is necessary for our usage of the forking lemma.

Algorithm 4 Reduction algorithm $\mathcal{B}$.

function INITIALIZE($\mathbf{y}^*, \mathcal{H}$)
  $j^* \xleftarrow{\$} \{1, \ldots, t\}$
  $\mathbf{y}_{j^*} \leftarrow \mathbf{y}^*$
  $\mathbf{x}_i \xleftarrow{\$} \mathsf{Samp}(1)$, **for** $i \in \{1, \ldots, t\} \setminus \{j^*\}$
  $\mathbf{y}_i \leftarrow F_{\mathbf{A}}(\mathbf{x}_i)$, **for** $i \in \{1, \ldots, t\} \setminus \{j^*\}$
  return $PK \leftarrow (t, k, \langle \mathbf{y}_1, \ldots, \mathbf{y}_t\rangle)$
function $H_1(q)$
  if $\mathcal{L}_1[q]$ is not defined **then**
    $\mathcal{L}_1[q] \xleftarrow{\$} \{0,1\}^{l_1}$
  return $\mathcal{L}_1[q]$
function $H_2(q)$
  if $\mathcal{L}_2[q]$ is not defined **then**
    $\mathcal{L}_2[q] \leftarrow$ next unused value from $\mathcal{H}$
  return $\mathcal{L}_2[q]$
function SIG(M)
  add M to set $\mathcal{M}$
  $\mathbf{s} \xleftarrow{\$} \mathsf{Samp}(\xi - \rho - 1)$
  $h \xleftarrow{\$} \{0,1\}^{l_1}$
  $I = \langle i_1\| \ldots \|i_k\rangle \leftarrow$ next unused value from $\mathcal{H}$
  $\tilde{\mathbf{r}} \leftarrow F_{\mathbf{A}}(\mathbf{s}) - \sum_{j=1}^{k}\mathbf{y}_{i_j}$
  if $\mathcal{L}_1[\tilde{\mathbf{r}}]$ or $\mathcal{L}_2[M\|h]$ are defined **then** BAD1 $\leftarrow 1$; **abort**
  $\mathcal{L}_1[\tilde{\mathbf{r}}] \leftarrow h$
  $\mathcal{L}_2[M\|h] \leftarrow I$
  return $(\mathbf{s}, h)$
function FINALIZE($M^*, \sigma^* = (\mathbf{s}^*, h^*)$)
  if there is a duplicate value in $\mathcal{L}_1$ **then** BAD2 $\leftarrow 1$; **abort**
  if $[M^* \notin \mathcal{M}] \wedge [\mathtt{SGN.Ver}(M^*, \sigma^*, PK) = 1]$ **then**
    FORGERY $\leftarrow 1$
    let v be the index such that $\mathcal{L}_2[M^*\|h^*] = \mathcal{H}[v]$
    return (v, σ^*)
  else
    return $(0, 0)$

$\mathcal{B}$ proceeds to simulate the EU-CMA game against $\mathcal{A}$, implanting $\mathbf{y}^*$ within the public key and generating simulated signatures as described above.

If $\mathcal{A}$ is successful in generating a forgery, then $\mathcal{B}$ outputs it, as well as the index of the hash call corresponding to $H_2(M^*\|h^*)$. This indicates to the forking lemma that we wish to rewind to this query and resume with fresh randomness.

CLAIM 1. $\Pr[\textit{FORGERY}] \geq \epsilon_{\mathcal{A}} - \frac{q_H q_S + q_H^2}{2^{l_1}} + \mathit{negl}(\kappa)$, *where the negligible quantity is from the security of* PRF.

PROOF. First, we compare the view of $\mathcal{A}$ in the reduction to its view in the standard EU-CMA game. The only differences are:

(1) The $\mathbf{x}_i$ values are chosen uniformly rather than pseudorandomly. This changes the adversary's view by a negligible amount.

(2) The signature is generated in "reverse order". From the discussion in Subsection 3.2, real signatures are distributed uniformly, hence this difference has no effect on the adversary's view.

Overall, we see that the adversary's view is indistinguishable.

The only other difference between the reduction and EU-CMA game is that the reduction may abort in the event of BAD1 or BAD2. BAD1 happens when the reduction needs to program the random oracles but they have already been queried on the desired point. On line 21, the values $\tilde{\mathbf{r}}$ and h are uniform, each with at least l_1 bits of entropy. Hence the probability that such a prior query has been made is at most $q_H/2^{l_1}$. Taking a union bound over all q_S calls to Sig, the overall probability of BAD1 is bounded by $q_S q_H/2^{l_1}$.

BAD2 happens when a collision is found in H_1. This probability is bounded by $q_H^2/2^{l_1}$. □

Forking lemma. Now, we can consider invoking the forking lemma (Lemma 2.1) with $\mathcal{B}^{\mathcal{A}}$. The result is an algorithm $\mathsf{Fork}_{\mathcal{B}}$ that has probability at least

$$\Pr[\text{FORGERY}]\left(\frac{\Pr[\text{FORGERY}]}{q_H} - \frac{1}{2^{l_2}}\right)$$

of producing *two forgeries*. Note that these forgeries must be with respect to the same M^* and h^* values because of the way that $\mathcal{B}$ computes the index v of the "special" oracle query, and the fact that the forking lemma ensures that this index is the same in both "forks." Each forgery verifies with respect to a different value of $H_2(M^*\|h^*)$.

Claim 2. *Let $\sigma_1^* = (\mathbf{s}_1^*, h^*)$ and $\sigma_2^* = (\mathbf{s}_2^*, h^*)$ be the two forgeries output by $\mathsf{Fork}_{\mathcal{B}}$, for message M^*. Let I_1 be the value of $H_2(M^*\|h^*)$ in the first "fork" and I_2 be its value in the second "fork." When I_1 and I_2 are **compatible with respect to** j^*, a preimage of $\mathbf{y}^*$ can be computed efficiently.*

Proof. Following the high-level discussion, we can solve for a preimage of $\mathbf{y}^*$. Write $I_1 = \langle i_1^{(1)}\| \cdots \|i_k^{(1)}\rangle$ and $I_2 = \langle i_1^{(2)}\| \cdots \|i_k^{(2)}\rangle$.

By symmetry, suppose j^* appears in I_1 but not I_2. From the verification equation for these signatures we have:

$$h^* = H_1\left(F_{\mathbf{A}}(\mathbf{s}_1^*) - \sum_{j=1}^{k} \mathbf{y}_{i_j^{(1)}}\right) = H_1\left(F_{\mathbf{A}}(\mathbf{s}_2^*) - \sum_{j=1}^{k} \mathbf{y}_{i_j^{(2)}}\right)$$

Since $\mathcal{B}$ aborts if a collision was found in H_1 (BAD2 event), we have

$$F_{\mathbf{A}}(\mathbf{s}_1^*) - \sum_{j=1}^{k} \mathbf{y}_{i_j^{(1)}} = F_{\mathbf{A}}(\mathbf{s}_2^*) - \sum_{j=1}^{k} \mathbf{y}_{i_j^{(2)}}$$

Isolating $\mathbf{y}_{j^*} = \mathbf{y}^*$ (which appears in the left summation but not the right one) and using the homomorphic property of $F_{\mathbf{A}}$ gives:

$$\begin{aligned}
\mathbf{y}^* &= F_{\mathbf{A}}(\mathbf{s}_1^*) - F_{\mathbf{A}}(\mathbf{s}_2^*) - \sum_{\substack{j=1 \\ i_j^{(1)} \neq j^*}}^{k} \mathbf{y}_{i_j^{(1)}} + \sum_{j=1}^{k} \mathbf{y}_{i_j^{(2)}} \\
&= F_{\mathbf{A}}(\mathbf{s}_1^*) - F_{\mathbf{A}}(\mathbf{s}_2^*) - \sum_{\substack{j=1 \\ i_j^{(1)} \neq j^*}}^{k} F_{\mathbf{A}}(\mathbf{x}_{i_j^{(1)}}) + \sum_{j=1}^{k} F_{\mathbf{A}}(\mathbf{x}_{i_j^{(2)}}) \\
&= F_{\mathbf{A}}\left(\mathbf{s}_1^* - \mathbf{s}_2^* - \sum_{\substack{j=1 \\ i_j^{(1)} \neq j^*}}^{k} \mathbf{x}_{i_j^{(1)}} + \sum_{j=1}^{k} \mathbf{x}_{i_j^{(2)}}\right)
\end{aligned}$$

The final argument to $F_{\mathbf{A}}$ is a value that can be computed from known values, and it is a preimage of $\mathbf{y}^*$. □

Proof of Theorem 4.1. Given an adversary $\mathcal{A}$ breaking EU-CMA security as stated, we first construct the reduction algorithm/game $\mathcal{B}$ (Algorithm 4). From Claim 1, the game produces a forgery with probability (ignoring negligible terms related to the PRF):

$$\Pr[\text{FORGERY}] \geq \epsilon_{\mathcal{A}} - \frac{q_H q_S + q_H^2}{2^{l_1}}$$

We then apply the forking lemma (Lemma 2.1) to $\mathcal{B}^{\mathcal{A}}$. The result is an algorithm $\mathsf{Fork}_{\mathcal{B}}$ that generates two forgeries with probability at least:

$$\Pr[\text{FORGERY}]\left(\frac{\Pr[\text{FORGERY}]}{q_H} - \frac{1}{2^{l_2}}\right)$$

In the event that $\mathsf{Fork}_{\mathcal{B}}$ outputs two forgeries, define I_1 to be the value of $H_2(M^*\|h^*)$ in the first "fork" and I_2 to be its value in the second "fork." Looking ahead, we would like to bound the probability that I_1 and I_2 are compatible. However, we run into a problem because I_2 is not distributed independently of $\mathsf{Fork}_{\mathcal{B}}$'s success. Intuitively, the adversary gets to "choose" whether the second fork succeeds after seeing I_2.

On the other hand, let $\mathcal{H}'$ be the set of oracle responses that are re-sampled uniformly during the second "fork." Importantly, $\mathcal{H}'$ is distributed independently of I_1, so we can bound the probability that I_1 is compatible with **all** elements of $\mathcal{H}'$. Since I_2 (if it exists) is guaranteed to be an element of $\mathcal{H}'$, this allows us to reason about the compatibility of I_1 and I_2.

From these observations, we obtain:

$$
\begin{aligned}
&\Pr[\text{preimage of } \mathbf{y}^* \text{ is found}] \\
&= \Pr[\mathsf{Fork}_{\mathcal{B}} \text{ outputs 2 forgeries and } I_1, I_2 \text{ are compatible wrt } j^*] \\
&= \frac{1}{t}\Pr[\mathsf{Fork}_{\mathcal{B}} \text{ outputs 2 forgeries and } I_1, I_2 \text{ are compatible}] \\
&\geq \frac{1}{t}\Pr[\mathsf{Fork}_{\mathcal{B}} \text{ outputs 2 forgeries and } I_1, \mathcal{H}' \text{ are compatible}] \\
&\geq \frac{1}{t}\Big(\Pr[\mathsf{Fork}_{\mathcal{B}} \text{ outputs 2 forgeries}] - \Pr[\overline{\mathsf{Compat}(q_H, k, l_2)}]\Big) \\
&\geq \frac{1}{t}\left[\Pr[\text{FORGERY}]\left(\frac{\Pr[\text{FORGERY}]}{q_H} - \frac{1}{2^{l_2}}\right) - \frac{q_H k!}{2^{l_2}}\right] \\
&= \frac{1}{t}\left[\left(\epsilon_{\mathcal{A}} - \frac{q_H(q_S+q_H)}{2^{l_1}}\right)\left(\frac{\epsilon_{\mathcal{A}}}{q_H} - \frac{q_S+q_H}{2^{l_1}} - \frac{1}{2^{l_2}}\right) - \frac{q_H k!}{2^{l_2}}\right]
\end{aligned}
$$

Note that the third line follows from the fact that the adversary's view in $\mathcal{B}^{\mathcal{A}}$ is independent of j^*.

The running time of $\mathcal{B}$ is that of $\mathcal{A}$ to output two forgery signatures with an overwhelming probability plus the time it takes for the simulation processes. For the sake of convenience, we do not consider the negligible processes. The setup process takes $t \cdot (t_{RNG} + t_{F_A})$, where t is the HORS parameter, for generating private keys and the corresponding public keys. Each signing process would require $2t_{RNG}$ to generate $\mathbf{r}'$ and $I = (i_1, \ldots, i_k)$ one t_{F_A} and $k \cdot t_{Add}$. Each hash query would require a t_{RNG}. Therefore, the total running time of $\mathcal{B}$ is upper-bounded by

$$O\big(2t_{\mathcal{A}} + t(t_{RNG} + t_{F_A}) + q_S(2t_{RNG} + t_{F_A} + kt_{Add}) + q_H t_{RNG}\big)$$

This completes the proof. □

4.1 Parameters

In this section, we discuss parameter choices for our construction as shown in Table 1.

4.1.1 Collision-freeness of GCK function. For TACHYON, N and μ are 256 and 8, respectively. As it has been shown in [37, 45], for the family of GCK functions to admit a strong security reduction, one needs to ensure that $\mu > \frac{|q|}{2|d|}$, $q > 4d\mu N^{1.5}|N|$ for domain $D = \{\mathbf{g} \in R : \|\mathbf{g}\|_\infty \leq d\}$ for some value d. Specifically, based on the analysis in [37, 42, 45], with these parameters, finding collision on average (when $a_{i,j} \in \mathbb{Z}_q$) with any non-negligible probability is at least as hard as solving the underlying problem (i.e., $SPP_\gamma(I)$ [37]) on certain kinds of point lattices, in the worst-case. We note that our concrete parameter selection, as provided in the following, meets the requirements stated above to allow for a strong security reduction.

4.1.2 Lattice Attacks. Given a uniformly random vector $\mathbf{a} = (a_1, \ldots, a_\mu) \in R^\mu$, the SIS problem over a ring asks to find a non-zero vector $\mathbf{x} = (x_1, \ldots, x_\mu) \in \mathbb{Z}[x]/(x^N+1)$ such that

$$\sum_{i=1}^{m} a_i x_i \equiv 0 \mod q, \text{ where } \|\mathbf{x}\| \leq \beta$$

An approach to estimate the hardness of this problem is by measuring the run-time of lattice basis reduction algorithms. These reduction algorithms aim to find the nice bases which consist of reasonably short and (nearly) orthogonal vectors. Gama and Nguyen [23] show that such reduction algorithms for a lattice $\mathcal{L}$ with dimension N can find vectors of length $\leq \delta^N \cdot \det(\mathcal{L})^{\frac{1}{N}}$ where δ is the *Hermite delta*. The BKZ algorithm [49] is the best known algorithm for finding short (non-zero) vectors in lattices. The BKZ algorithm starts by reducing a lattice basis using a *Shortest Vector Problem* (SVP) oracle in a smaller dimension. As shown in [26], the number of calls to the SVP oracle remains polynomial, however, precisely computing the number of calls is an arduous task and therefore, subject to heuristic approaches (e.g., BKZ 2.0 [16]). BKZ 2.0 requires solving the SVP problem in lattices with dimension at most $b < N$, where b is called the *block size*. Therefore, BKZ 2.0 runs for multiple rounds to find the final output. Given the norm bound β of an SIS instance, the corresponding δ can be computed as $\beta = \delta \det(\mathcal{L})^{1/N}$, then an estimate of the run time of BKZ 2.0 to attain δ is computed. Following [2, 24, 54], we use the following relation to determine the smallest block size b to achieve δ.

$$\delta = \left(\frac{b \cdot (\pi b)^{\frac{1}{b}}}{2\pi e}\right)^{\frac{1}{2(b-1)}}$$

The most recent classical solver for SVP [5] runs in time $\approx 2^{0.292b}$ and the best known quantum solver for SVP [32] runs in time $\approx 2^{0.265b}$.

In the following we discuss our estimation based on the works in [2, 3, 19, 24].

We consider two types of adversary powers, namely, the classical and post-quantum. For TACHYON, we proffer three parameter sets (for three security levels) and analyze the security level of each for the adversarial types mentioned above. In the classical model, for our medium security instantiation, we set $q = 2^{27} - 2^{11} + 1$ and $\beta = 2^{16}$ to achieve $\delta \approx 1.00339$ with $b = 502$. We set $q = 2^{30} - 2^{18} + 1$ and $\beta = 2^{17}$ for recommended instantiation which achieves $\delta \approx 1.00271$ with $b = 682$. We set $q = 2^{31} - 2^9 + 1$ and $\beta = 2^{17}$ for the high security instantiation with $\delta \approx 1.00203$ with $b = 1007$. Therefore, based on the analysis in [3, 19], we achieve 146, 199 and 294 classical bit security for the medium, recommended and high security instantiations of TACHYON against lattice attacks, respectively. For post-quantum security against lattice attacks, we achieve 133, 180 and 266 bit security for the medium, recommended and high security instantiations, respectively. Similar to Dilithium [19], our parameter choices are conservative.

4.1.3 k-Element Combinatorial Problem. As captured in our security proof, k, t parameters must be selected such that the probability $\frac{q_H \cdot k!}{2^{l_2}}$ is negligible. Considering that $l_2 = k|t|$ (since k indexes that are $|t|$-bit long are selected with the hash output), this gives us $\frac{q_H \cdot k!}{2^{k|t|}}$. We further elaborate on some choices of (k, t) along with their security/performance implications in Section 5.

4.1.4 Quantum Random Oracle Model (QROM). QROM considers the scenario where the adversary has classical access to the signing oracle and quantum access to the hash function oracle. TACHYON is proven to be secure in the random oracle model and we do not provide the proof for the security of TACHYON in QROM. This trend is true for a wide range of "efficient" schemes (e.g., [19]), which are mostly based on Fiat-Shamir framework, since their ROM is

Table 1: Parameter Selection of TACHYON

Parameter	TACHYON-128	TACHYON-192	TACHYON-256
N	256	256	256
μ	8	8	8
q	134215681	1073479681	2147483137
t	1024	2048	3072
k	18	25	32
l_1	256 bits	384 bits	512 bits
l_2	180 bits	275 bits	384 bits
RS Rate†	3.08	2.18	2.72

† RS Rate denotes Rejection Sampling Rate.

not "history free" due to the forking lemma in the reduction step. Initial approaches (e.g., [52]) to obtain QROM security for schemes based on Fiat-Shamir transformation resulted in considerably less efficient signatures since they needed multiple execution of the underlying identification scheme. However, recently, in line of providing QROM security for Dilithium [19], Kiltz et al. [31] provide a tight reduction in the QROM which incurs less performance/storage penalty as compared to directly applying the method in [52]. This generic framework [31] can be applied to the identification schemes that admit lossy public keys. We believe it is possible to prove the security of TACHYON in QROM and therefore, in the line of Dilithium [19] and its QROM secure instantiation [31], we will investigate the QROM security of TACHYON in our future work.

5 PERFORMANCE EVALUATION

We first present analytical performance analysis and some of the potential performance/speed trade-offs for TACHYON. We then provide our evaluation metrics and experimental setup followed by a detailed experimental comparison of TACHYON with the state-of-the-art PQ-secure digital signature schemes.

5.1 Analytical Performance Analysis

We now describe the analytical performance of our scheme based on the parameters. In the computational overhead analysis, we present our runtime in terms of the total number of PRF, GCK function, and vector addition calls. We omit the overhead of small-constant number of hash calls.

• *Signer Computation and Storage Overhead:* TACHYON only requires storing a κ-bit random seed number as the private key, which is used to deterministically generate the required $\mathbf{x_i}$ components via PRF calls, where each $\mathbf{x_i}$ is $\mu \cdot N$ bits.

The signature generation cost is significantly affected by the derivation and summation of k number of $\mathbf{x_i}$. This requires $k \cdot$ PRF calls, extracting the binary vectors from the PRF outputs and vector additions (whose computational overhead is negligible). For each PRF call, a κ-bit input is extended to a $\mu \cdot N$ bit output. In addition, a $\mathsf{Samp}(\xi - 1)$ function is required. $\mathsf{Samp}(\xi - 1)$ generates a vector of length $\mu \cdot N$ with components of length $|\xi|$ bits. Therefore, $\mathsf{Samp}(\xi - 1)$ can be implemented with a PRF that extends a κ-bit input to a $|\xi| \cdot \mu \cdot N$ bit output. In total, these correspond to the generation of $(|\xi| + k) \cdot \mu \cdot N$ pseudorandom bits via a PRF. Another significant cost for signature generation is the GCK function call that is made to compute the image of the randomness $\mathbf{r'}$. A GCK call is basically composed of two operations: Number Theoretic Transform (NTT) calculation and a linear combination. In order to compute a GCK call, μ number of NTT calls and a single linear combination is necessary, where both of these operations are based on simple multiplications and additions under $\bmod q$. Therefore, in total, TACHYON signature generation requires storing κ-bit of private key, k PRF invocations, k vector additions, a single $\mathsf{Samp}(\xi - 1)$ and a GCK function call to compute a signature.

• *Signature Size*: The signature σ is comprised of the vector $\mathbf{s}$ and a hash output h, where $|h| = l_1$. Rejection sampling enforces $\mathbf{s}$ to satisfy $\|\mathbf{s}\|_\infty < \xi - \rho$. Since $\mathbf{s}$ consists of $\mu \cdot N$ components, this vector can be represented with $|\xi - \rho| \cdot \mu \cdot N$ bits. The total size of a signature is $|\xi - \rho| \cdot \mu \cdot N + l_1$ bits.

• *Verifier Computation and Storage Overhead*: The signature verification requires only a single GCK call and k vector additions, which makes it the most verifier computationally efficient scheme among its current counterparts. On the other hand, the size of public key is $|q| \cdot \mu \cdot N \cdot t$ bits (i.e., t vectors of length $\mu \cdot N$), which is relatively larger than its counterparts.

• *Improved Side-Channel Resiliency*: TACHYON only requires a uniform sampling $\mathsf{Samp}(\xi - 1)$ in its signature generation. Since it does not require Gaussian sampling, it has an improved side-channel resiliency as compared to some of its lattice-based counterparts (e.g., BLISS [18]). Moreover, the rejection sampling in BLISS is based on iterated Bernoulli trials, that is prone to some attacks. As it is shown in [21], this efficient rejection sampling technique has been exposed to some side channel attacks. Although, TACHYON requires rejection sampling to make sure the statistical distribution of the signatures does not leak information about the private key components, similar to [19], since our rejection sampling does not require any Bernoulli trials, the attack does not apply to our rejection sampling step.

5.2 Performance-vs-Storage Trade-offs

Our design allows several trade-offs between performance and storage that may be suitable for different use-cases.

• *Signer Pre-computation*: With a basic implementation trick, one can store the $\mathbf{x_i}$'s instead of deterministically generating them at the signature generation. This enables the signer to avoid the cost of generating these values ($k \cdot$ PRF calls, and extracting the binary vectors) during the signature generation. Since the signer must store these $\mathbf{x_i}$ vectors, this adds up to a private key of at least $t \cdot \mu \cdot N$ bits, that is larger than that of TACHYON. However, this caching strategy offers a faster signature generation and therefore can be preferred when the signer is able to store such vectors. Signature generation speed advantages and required private key size are further explained in Subsection 5.4.

• *Selection of t, k*: The parameter t linearly impacts the size of public key of TACHYON. The parameter k determines the number of PRF calls, binary vectors to be extracted and vector additions in TACHYON signing, and also the number of vector additions in

TACHYON signature verification. Note that decreasing t requires an increase in k (or vice versa) to preserve the desired security level. We selected ($t = 1024$, $k = 18$), ($t = 2048$, $k = 25$), and ($t = 3072$, $k = 32$) to provide $\kappa = 128$-bit, $\kappa = 192$-bit, and $\kappa = 256$-bit security, respectively. However, different parameters for the same security levels are also possible. For instance $t = 256$, $k = 26$ would also offer $\kappa = 128$-bit security level and could be preferred (over $t = 1024$, $k = 18$) for TACHYON medium level security instantiation. This would provide a 4× smaller public key, where the signature generation time would be increased.

• *Rejection Sampling Parameters*: Rejection sampling rate implies how many times (on average) the signature generation should be executed to output an "acceptable" signature. Therefore, the increment of the acceptance probability has a linear effect on the signature generation time. We discuss two parameters that can be tuned to increase the acceptance probability of the outputted signatures, (i) increasing $\xi - 1$ (where $\xi - 1 = \|\mathbf{r}'\|_\infty$), and (ii) decreasing k. While tuning these parameters can result in significantly decreasing the average signing time, there are trade-offs to consider. Increasing $\xi - 1$ causes an increase on the signature size. Additionally, this increase incurs a security loss as it directly affects the hardness of the lattice attacks discussed in Subsection 4.1. On the other hand, as discussed above, decreasing k would require increasing t to compensate for the security loss, that increases the public key size.

5.3 Experimental Evaluation and Setup

We describe our experimental evaluation metrics and setup, wherein our scheme and their counterparts are compared with each other.

• *Evaluation Metrics*: We have evaluated and compared TACHYON with its counterparts in terms of signature generation and verification times, private key, public key and signature sizes and end-to-end cryptographic delay (i.e., the sum of signature generation and verification times, excluding the signature transmission time, as it is network depended).

• *Hardware Configurations*: We used a laptop equipped with an Intel i7 6th generation (Skylake) 2.6GHz processor and 12 GB of RAM for our experiments.

• *Implementation Details* : Our parameter selection which is based on [37] - i.e., N is a power-of-two and $1 \equiv q \mod (2N)$ - allows us to use NTT to accelerate the GCK function computations. Similar approach has been done in [19]. Then, to finalize the GCK function, we computed the linear combination under $\mod q$ of input with random and public matrix $\mathbf{A}$. Since highest $|q|$ selected is just 31, we did not use any libraries for these calculations. We would like to note that this operation can be performed very fast with some assembly level optimizations. However, in this paper, we used a conservative implementation.

We instantiated H_1 and H_2 random oracles using BLAKE2b due to its optimization for commodity hardware, in terms of speed and security [4]. We used Intel intrinsics to implement our PRF function and $\mathsf{Samp}(\xi - 1)$ (with AES in counter mode). Our implementation is open-sourced in the following link.

https://github.com/ozgurozmen/TACHYON

For our counterparts, we used the optimized codes (if available, otherwise the reference codes) that are submitted to the NIST competition and ran them on our processor. Note that among all the schemes presented in Table 2, only BLISS is not a NIST competitor. For this scheme, we used the open-sourced implementation provided by the authors.

5.4 Performance Analysis and Comparison

Table 2 shows the experimental performances of TACHYON and its state-of-the-art counterparts. We selected various schemes that are submitted to the first NIST post-quantum cryptography standardization conference (except BLISS [18], that is selected since it is one of the fastest lattice-based signatures). These schemes include lattice-based constructions (qTESLA [11], pqNTRUsign [27], and Dilithium [19]), a hash-based construction (SPHINCS+ [28]), a code-based construction (pqsigRM [34]), a symmetric key cryptography based construction (PICNIC [15]) and a multivariate-based scheme (GeMSS [14]).

Table 2 shows that TACHYON has the lowest end-to-end delay and both its signature generation and verification are the fastest among its counterparts, for every security level. For instance TACHYON-192 has the fastest signature generation and the lowest end-to-end delay among all the schemes with *any* security level. Moreover, TACHYON offers the lowest possible private key size (that is the same with symmetric key based PICNIC). TACHYON has a signature of slightly more than 4 KB, that is comparable to its lattice-based counterparts but larger than multivariate and code-based constructions. TACHYON public key is significantly larger than most of their counterparts (only smaller than GeMSS in high security levels). Considering the overall efficiency of TACHYON, we believe it can be preferred when the verifier can tolerate such a storage.

As discussed in Subsection 5.2, one can consider caching the $\mathbf{x_i}$ vectors as the private key instead of deterministically deriving them with a κ-bit seed. When this optimization is considered, it provides a signature generation that is significantly faster than that of TACHYON. With the verification being unchanged, this variant can further improve the end-to-end delay (which is currently the fastest). On the other hand, when these vectors are cached, the private key size increases significantly (e.g., 256 − 768 KB), that is only smaller than pqsigRM, for certain security levels. This can make caching impractical for some applications where the signer is memory-limited. In these cases, TACHYON without any caching should be preferred.

We also dissected the cost of TACHYON, for future optimizations. GCK function computation corresponds to the ≈ 40% of the total cost for TACHYON-128 signature generation, that slightly decreases on higher security levels. The highest cost is identified as the PRF calls and the extraction of the binary vectors from this PRF output, made to deterministically generate the vectors ($\mathbf{x_i}$'s). This can be further confirmed with the improvements observed by caching the $\mathbf{x_i}$ vectors, where this cost is eliminated and replaced with only vector additions. For the signature verification, over 80% of the total cost is due to the GCK function.

Discussions. The GCK function calculations can be further accelerated with assembly instructions on NTT function as in Dilithium

Table 2: Experimental Performance Comparison of TACHYON with Its Counterparts

Scheme	Security Level (bit)	Signature Gen Time† (μs)	Private Key (Byte)	Signature Size (Byte)	Signature Ver Time (μs)	Public Key (Byte)	End-to-End Delay (μs)	Gaussian Sampling‡
SPHINCS+ [28]	128	14625	64	16976	617	32	15242	N/A
	192	18580	96	35664	974	48	19554	
	256	42898	128	49216	1015	64	43913	
pqsigRM [34]	128	3960	1382118	260	21	336804	3981	N/A
	192	20260	334006	516	30	501176	20290	
	256	406	2105344	1028	138	2144166	544	
GeMSS [14]	128	252844	14208	48	39	417408	252883	N/A
	192	776330	39440	88	109	1304192	776439	
	256	1118542	82056	104	326	3603792	1118868	
PICNIC [15]	128	1966	16	34000	1335	32	3301	N/A
	192	6951	24	76740	4804	48	11755	
	256	13963	32	132824	9639	64	23602	
BLISS [18]	128	141	256	717	28	896	169	✓
	160*	211	384	768	28	896	239	
	192*	392	384	813	31	896	423	
qTESLA [11]	128	650	1856	2720	133	2976	783	×
	192	2524	4160	5664	272	6176	2796	
	256	6793	4128	5920	334	6432	7127	
pqNTRUsign [27]	128	14516	1024	576	304	1024	14820	×
Dilithium[19]	100*	166	2800	2044	53	1184	219	×
	138*	272	3504	2701	76	1472	348	
	176*	219	3856	3366	103	1760	322	
TACHYON	128	**138**	**16**	4416	**18**	884736	**156**	×
	192	**124**	**24**	4672	**21**	1966080	**145**	
	256	**198**	**32**	4672	**28**	3047424	**226**	

† TACHYON requires rejection sampling in its signature generation (similar to BLISS [18], Dilithium [19]). The number of required signature generation repetitions due to rejection sampling are 3.08, 2.18 and 2.72 for medium, recommended and high security levels, respectively.
‡ Gaussian sampling requirement is same for the *all security levels*, and therefore, it is represented with a single value.
✓Denotes the scheme requires Gaussian sampling, that can be considered unfavorable due to the side-channel attacks.
∗ Denotes security level other than standard 128, 192, 256 bits.

[19]. In this paper, we presented our benchmark results with a reference implementation, without any assembly level instructions. Therefore, we believe that there is still a significant room for performance improvement for our scheme, especially in the verification algorithm, where the dominative cost is the GCK function. On the other hand, since we implemented the PRF functions of our scheme using Intel intrinsics, TACHYON might face a performance penalty on other platforms. Therefore, light-weight symmetric ciphers or hash functions should be preferred to implement the PRF calls in TACHYON on other platforms.

6 CONCLUSION

In this paper, we proposed a new digital signature scheme with a post-quantum promise, which we refer to as TACHYON. Our unique algorithmic design leverages the well-known HORS construction and additively homomorphic GCK functions to extend one-time signatures to (polynomially bounded) many-time signatures. TACHYON offers several desirable properties: (i) It achieves the lowest end-to-end delay with the fastest signature generation and verification among its counterparts in every security level. (ii) TACHYON has the smallest private key size (i.e., κ-bit) among its counterparts. (iii) TACHYON has highly tunable parameters, which offer various speed and storage trade-offs. (iv) TACHYON does not require any Gaussian sampling, and therefore it is immune to the side-channel attacks targeting this function. All these desirable properties of TACHYON come with a larger public key than most of its counterparts.

ACKNOWLEDGMENTS

The authors want to thank Chris Peikert, Vadim Lyubashevsky and Daniele Micciancio for their comments and valuable suggestions. The authors also thank Peter Rindal for his suggestions on the implementation of the scheme. This work is supported by NSF awards #1652389 and #1617197.

REFERENCES

[1] Sedat Akleylek, Nina Bindel, Johannes Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. 2016. An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation. In *Progress in Cryptology – AFRICACRYPT 2016*, David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi (Eds.). Springer International Publishing, 44–60.
[2] Nabil Alkeilani Alkadri, Johannes Buchmann, Rachid El Bansarkhani, and Juliane Krämer. 2017. A Framework to Select Parameters for Lattice-Based Cryptography.

Cryptology ePrint Archive, Report 2017/615. (2017). https://eprint.iacr.org/2017/615.

[3] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum Key Exchange-A New Hope.. In *USENIX Security Symposium*. 327–343.

[4] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. 2010. SHA-3 proposal BLAKE. Submission to NIST (Round 3). (2010). http://131002.net/blake/blake.pdf

[5] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. 2016. New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 10–24. http://dl.acm.org/citation.cfm?id=2884435.2884437

[6] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the Plain public-Key Model and a General Forking Lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. ACM, New York, NY, USA, 390–399.

[7] M. Bellare and P. Rogaway. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93)*. ACM, NY, USA, 62–73.

[8] Mihir Bellare and Phillip Rogaway. 2006. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology - EUROCRYPT 2006*, Serge Vaudenay (Ed.). Springer Berlin Heidelberg, 409–426.

[9] E. Berlekamp, R. McEliece, and H. van Tilborg. 1978. On the inherent intractability of certain coding problems (Corresp.). *IEEE Transactions on Information Theory* 24, 3 (1978), 384–386.

[10] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. 2015. SPHINCS: Practical Stateless Hash-Based Signatures. In *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer Berlin Heidelberg, 368–397.

[11] Nina Bindel, Sedat Akeylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Julaine Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. 2018. qTESLA. Submission to the NIST's post-quantum cryptography standardization process. (2018). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/qTESLA.zip.

[12] Jurjen N. E. Bos and David Chaum. 1993. Provably Unforgeable Signatures. In *Advances in Cryptology — CRYPTO' 92*, Ernest F. Brickell (Ed.). Springer Berlin Heidelberg, 1–14.

[13] Matt Braithwaite. 2016. Experimenting with Post-Quantum Cryptography. (2016). https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

[14] A. Casanova, J.-C. Faugere, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. 2018. GeMSS. Submission to the NIST's post-quantum cryptography standardization process. (2018). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/GeMSS.zip.

[15] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. 2017. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 1825–1842. https://doi.org/10.1145/3133956.3133997

[16] Yuanmi Chen and Phong Q. Nguyen. 2011. BKZ 2.0: Better Lattice Security Estimates. In *Advances in Cryptology – ASIACRYPT 2011*, Dong Hoon Lee and Xiaoyun Wang (Eds.). Springer Berlin Heidelberg, 1–20.

[17] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. 2001. How to Achieve a McEliece-Based Digital Signature Scheme. In *Advances in Cryptology — ASIACRYPT 2001*, Colin Boyd (Ed.). Springer Berlin Heidelberg, 157–174.

[18] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. 2013. Lattice Signatures and Bimodal Gaussians. In *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, Ran Canetti and Juan A. Garay (Eds.). Springer Berlin Heidelberg, 40–56.

[19] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. 2017. CRYSTALS – Dilithium: Digital Signatures from Module Lattices. Cryptology ePrint Archive, Report 2017/633. (2017). http://eprint.iacr.org/2017/633.

[20] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. 2014. Efficient Identity-Based Encryption over NTRU Lattices. In *Advances in Cryptology – ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, Palash Sarkar and Tetsu Iwata (Eds.). Springer Berlin Heidelberg, 22–41.

[21] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. 2017. Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing against strongSwan and Electromagnetic Emanations in Microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*. 1857–1874.

[22] Amos Fiat and Adi Shamir. 1987. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, Andrew M. Odlyzko (Ed.). Springer Berlin Heidelberg.

[23] Nicolas Gama and Phong Q. Nguyen. 2008. Predicting Lattice Reduction. In *Advances in Cryptology – EUROCRYPT 2008*, Nigel Smart (Ed.). Springer Berlin Heidelberg, 31–51.

[24] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. 2017. A Hybrid Lattice Basis Reduction and Quantum Search Attack on LWE. In *Post-Quantum Cryptography*, Tanja Lange and Tsuyoshi Takagi (Eds.). Springer International Publishing, Cham, 184–202.

[25] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. 2016. Flush, Gauss, and Reload – A Cache Attack on the BLISS Lattice-Based Signature Scheme. In *Cryptographic Hardware and Embedded Systems – CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, Benedikt Gierlichs and Axel Y. Poschmann (Eds.). Springer Berlin Heidelberg, 323–345.

[26] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. 2011. Terminating BKZ. Cryptology ePrint Archive, Report 2011/198. (2011). https://eprint.iacr.org/2011/198.

[27] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. 2017. A signature scheme from Learning with Truncation. Cryptology ePrint Archive, Report 2017/995. (2017). https://eprint.iacr.org/2017/995.

[28] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. 2018. SPHINCS+. Submission to the NIST's post-quantum cryptography standardization process. (2018). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/SPHINCS_Plus.zip.

[29] Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. 2016. ARMed SPHINCS - Computing a 41 KB Signature in 16 KB of RAM. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography*. 446–470.

[30] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC.

[31] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. 2018. A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model. In *Advances in Cryptology – EUROCRYPT 2018*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 552–586.

[32] Thijs Laarhoven. 2015. *Search problems in cryptography from fingerprinting to lattice sieving*. Ph.D. Dissertation. Gildeprint Drukkerijen, Enschede, The Netherlands.

[33] Leslie Lamport. 1979. *Constructing digital signatures from a one-way function*. Technical Report. Technical Report CSL-98, SRI International Palo Alto.

[34] Wijik Lee, Young-Sik Kim, Yong-Woo Lee, and Jong-Seon No. 2018. pqsigRM. Submission to the NIST's post-quantum cryptography standardization process. (2018). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/pqsigRM.zip.

[35] Vadim Lyubashevsky. 2008. Lattice-Based Identification Schemes Secure Under Active Attacks. In *Public Key Cryptography – PKC 2008: 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, Ronald Cramer (Ed.). Springer Berlin Heidelberg, 162–179.

[36] Vadim Lyubashevsky. 2012. Lattice Signatures Without Trapdoors. In *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'12)*. Springer-Verlag, 738–755.

[37] Vadim Lyubashevsky and Daniele Micciancio. 2006. Generalized Compact Knapsacks Are Collision Resistant. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, 144–155.

[38] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. 2008. SWIFFT: A Modest Proposal for FFT Hashing. In *Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*. Springer Berlin Heidelberg, 54–72.

[39] Robert J Mceliece. 1978. A public-key cryptosystem based on algebraic. *Coding Thv* 4244 (1978), 114–116.

[40] Ralph C. Merkle. 1989. A certified digital signature. In *Proceedings on Advances in cryptology (CRYPTO '89)*. Springer-Verlag, New York, NY, USA, 218–238.

[41] D. Micciancio. 2002. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 356–365.

[42] Daniele Micciancio. 2007. Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions. *computational complexity* 16, 4 (2007), 365–411. https://doi.org/10.1007/s00037-007-0234-9

[43] Committee on National Security Systems. 2015. Use of Public Standards for the Secure Sharing of Information Among National Security Systems. (2015).

[44] Jacques Patarin, Nicolas Courtois, and Louis Goubin. 2001. QUARTZ, 128-Bit Long Digital Signatures. In *Topics in Cryptology — CT-RSA 2001*, David Naccache (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 282–297.

[45] Chris Peikert. 2010. An Efficient and Parallel Gaussian Sampler for Lattices. In *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, Tal Rabin (Ed.). Springer Berlin Heidelberg, 80–97.

[46] A. Perrig, R. Canetti, D. Song, and D. Tygar. 2000. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proceedings of the IEEE Symposium on Security and Privacy*.

[47] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. 2017. To BLISS-B or Not to Be: Attacking strongSwan's Implementation of Post-Quantum Signatures. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 1843–1855.

[48] L. Reyzin and N. Reyzin. 2002. Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02)*. Springer-Verlag, 144–153.

[49] C. P. Schnorr and M. Euchner. 1994. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming* 66, 1 (01 Aug 1994), 181–199.

[50] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332.

[51] Jacques Stern. 1994. A new identification scheme based on syndrome decoding. In *Advances in Cryptology — CRYPTO' 93*, Douglas R. Stinson (Ed.). Springer Berlin Heidelberg, 13–21.

[52] Dominique Unruh. 2015. Non-Interactive Zero-Knowledge Proofs in the Quantum Random Oracle Model. In *Advances in Cryptology - EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 755–784.

[53] Pascal Véron. 1997. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing* 8, 1 (01 Jan 1997), 57–69.

[54] Thomas Wunderer. 2016. Revisiting the Hybrid Attack: Improved Analysis and Refined Security Estimates. Cryptology ePrint Archive, Report 2016/733. (2016). https://eprint.iacr.org/2016/733.