Lattice-Based Signature from Key Consensus*

Leixiao Cheng[1], Boru Gong[2], and Yunlei Zhao[2]

[1] School of Mathematical Sciences, Fudan University, Shanghai, China
[2] School of Computer Science, Fudan University, Shanghai, China

Abstract. In this work, we present generalization and optimization of Dilithium, which is one of the promising lattice-based signature candidates for NIST post-quantum cryptography (PQC) standardization. This is enabled by new insights in interpreting the design of Dilithium, in terms of key consensus presented in the KCL key encapsulation mechanism (KEM) proposal submitted to NIST PQC standardization. Based on OKCN developed in KCL, we present a generic and modular construction of lattice-based signature, and make analysis as it is deployed in reality. We thoroughly search and test a large set of parameters in order to achieve better trade-offs among security, efficiency, and bandwidth. On the recommended parameters for about 128-bit quantum security, compared with Dilithium, our scheme is more efficient both in computation and in bandwidth. This work also further justifies and highlights the desirability of OKCN as the same routine can be used for both KEM and signatures, which is useful to simplify system complexity of lattice-based cryptography. Of independent interest is a new estimation of the security against key recovery attacks in reality.

1 Introduction

Given the current research status in lattice-based cryptography, it is commonly suggested that lattice-based signature could be subtler and harder to achieve. For instance, there are more than twenty lattice-based KEM proposals to NIST PQC standardization, but only five lattice-based signature proposals [NIST]. Among them, Dilithium [DLL+17, LDK+17] is one of the most promising lattice-based signature candidates, for its simplicity, efficiency, small public key size, and resistance against side channel attacks. Its design is based on a list of pioneering works (e.g., [Lyu09,Lyu12,BG14] and more), with careful and comprehensive optimizations in implementation and parameter selection. Whether better trade-offs on the already remarkable performance can be achieved is left in [CRYSTALS] as an interesting open question.

In this work, we present generalization and optimization of Dilithium. This is enabled by new insights in interpreting the design of Dilithium, in terms of key consensus presented in the KCL KEM proposal to NIST PQC standardization [KCL,JZ16]. Based on the reconciliation mechanism, named optimal key consensus with noise (OKCN), developed in KCL, we present a generic and modular construction of lattice-based signature. The construction is generic and versatile, in the sense that we could choose parameters in a much broader range. We generalize the security analysis of Dilithium into its realistic setting, where the public key is only a part of the module-LWE (MLWE)

* Corresponding author: Yunlei Zhao, ylzhao@fudan.edu.cn

sample. Of independent interest is a new estimation of the security against key recovery attacks in reality.

We made efforts to thoroughly search and test a large set of parameters in order to achieve better trade-offs among security, efficiency, and bandwidth. On the recommended parameters for about 128-bit quantum security, compared with Dilithium our scheme is more efficient both in computation and in bandwidth. This work further justifies and highlights the desirability of OKCN as the same routine can be used for both KEM and signatures, which is useful to simplify system complexity of lattice-based cryptography.

2 Preliminaries

Without loss of generality, every string in this work is a binary one; for a (binary) string $s \in \{0,1\}^*$, let $|s|$ denote its length by default. For any real number $x \in \mathbb{R}$, let $\lfloor x \rfloor$ denote the largest integer that is no more than x, and $\lceil x \rceil := \lfloor x + 1/2 \rfloor$. For any $i, j \in \mathbb{Z}$ such that $i < j$, denote by $[i, j]$ the set of integers $\{i, i+1, \cdots, j-1, j\}$. For the positive integers $r, \alpha > 0$, let $r \bmod \alpha$ denote the unique integer $r' \in [0, \alpha - 1]$ such that $\alpha \mid (r' - r)$, and let $r \bmod^{\pm} \alpha$ denote the unique integer $r'' \in [-\lfloor \frac{\alpha-1}{2} \rfloor, \lfloor \frac{\alpha}{2} \rfloor]$ such that $\alpha \mid (r'' - r)$. For an element $x \in \mathbb{Z}_q$, we write $\|x\|_\infty$ for $|x \bmod^{\pm} q|$.

For a finite set S, $|S|$ denotes its cardinality, and $x \leftarrow S$ denotes the operation of picking an element uniformly at random from the set S. We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. For an arbitrary probability distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes the operation of picking an element according to the pre-defined distribution $\mathcal{D}$. If α is neither an algorithm nor a set, then $x \leftarrow \alpha$ is a simple assignment statement. If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ represents the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and outputting $y := A(x_1, x_2, \cdots; r)$. By $\Pr[R_1; \cdots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$.

We say that a function $f(\lambda) > 0$ is *negligible* in λ, if for every $c > 0$ there exists a positive $\lambda_c > 0$ such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. Two distribution ensembles $\{X(\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^*}$ and $\{Y(\lambda, z)\}_{\lambda \in N, z \in \{0,1\}^*}$ are *computationally indistinguishable*, if for any probabilistic polynomial-time (PPT) algorithm D, and for sufficiently large λ and any $z \in \{0,1\}^*$, we have $|\Pr[D(X(\lambda, z)) = 1] - \Pr[D(Y(\lambda, z)) = 1]|$ is negligible in λ.

2.1 Digital Signature Scheme and Its Security

Definition 1. *A digital signature scheme Π consists of three probabilistic polynomial-time algorithms* (KeyGen, Sign, Verify)*:*

- KeyGen *is the key generation algorithm that, on input the security parameter 1^λ, outputs* (pk, sk).
- Sign *is the signing algorithm that, on input the message $\mu \in \{0,1\}^*$ to be signed as well the secret key* sk*, outputs the signature σ.*

- Verify *is the* deterministic *verification algorithm that, on input the public key* pk *and the message/signature pair* (μ, σ)*, outputs* $b \in \{0,1\}$*, indicating whether it accepts the incoming message/signature pair* (μ, σ) *as a* valid *one or not.*

We say a signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* correct*, if any sufficiently large* λ*, any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and any* $\mu \in \{0,1\}^*$*, it holds*

$$\Pr[\mathsf{Verify}(\mathsf{pk}, \mu, \mathsf{Sign}(\mathsf{sk}, \mu)) = 1] = 1.$$

Definition 2 ((Strong) Existential unforgeability under adaptive chosen message attack, (S)EU-CMA). *The security for a signature scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, is defined with respect to the following security game between a challenger and an adversary* A*.*

- *Setup. On the security parameter* λ*, the challenger runs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*. The public key* pk *is given to adversary* A *(while the secret key* sk *is kept in private).*
- *Challenge. Suppose* A *makes at most* s *signature queries. Each signature query consists of the following steps: (1)* A *adaptively chooses the message* $\mu_i \in \{0,1\}^*$*,* $1 \leq i \leq s$*, based upon its entire view, and sends* μ_i *to the signer; (2) Given the secret key* sk *as well as the message* μ_i *to be signed , the signer generates and sends back the associated signature, denoted* σ_i*, to* A*.*
- *Output. Finally,* A *outputs a pair of* (μ, σ)*, and wins if (1)* $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma) = 1$ *and (2)* $(\mu, \sigma) \notin \{(\mu_1, \sigma_1), \cdots, (\mu_{q_s}, \sigma_{q_s})\}$*.*

We define $\mathsf{AdvSig}^{\text{seu-cma}}_{\Pi,A}(1^\lambda)$ *to be the probability that* A *wins in the above game, taken over the coin tosses of* KeyGen*,* A *and of the challenger (as well as that of the random oracle). We say the signature scheme* Π *is* strongly existentially unforgeable under adaptive chosen-message attack*, if* $\mathsf{AdvSig}^{\text{seu-cma}}_{\Pi,A}(1^\lambda) = \mathsf{negl}(\lambda)$ *holds for any probabilistic polynomial-time adversary* A*.*

A slightly weaker definition could be defined in the similar manner, where the security game is almost the same as the foregoing one, except that it is only required that $\mu \notin \{\mu_1, \cdots, \mu_{q_s}\}$*. And the signature scheme is called* (standard) existentially unforgeable under adaptive chosen-message attack*, if no probabilistic polynomial-time adversary can win in this modified security game with non-negligible advantage.*

2.2 Module-LWE and Module-SIS

In this work, let $n \geq 8$ be a power-of-two integer, and let $q > 17$ denotes a positive rational prime such that $2n \mid (q-1)$. In our signature scheme, we always have $n = 256$ and $q = 4191233$. Also, let $\mathcal{R}$ and $\mathcal{R}_q$ denote the rings $\mathbb{Z}[x]/\langle x^n + 1\rangle$ and $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$, respectively. For the element $w = \sum_{i=0}^{n-1} w_i x^i \in \mathcal{R}$, its ℓ_∞-norm is defined as $\|w\|_\infty := \max_i \|w_i\|_\infty$. Likewise, for the element $\mathbf{w} = (w_1, \cdots, w_k) \in \mathcal{R}^k$, its ℓ_∞-norm is defined as $\|\mathbf{w}\|_\infty := \max_i \|w_i\|_\infty$. In particular, Let S_η denote all elements $w \in \mathcal{R}$ such that $\|w\|_\infty \leq \eta$.

The hard problems underlying the security of our signature scheme are Module-LWE (MLWE) and Module-SIS (MSIS). They were well studied in [LS15] and could be seen

as a generalization of the Ring-LWE [LPR10] and Ring-SIS problems [LM06, PR06], respectively.

Fix the parameter $\ell \in \mathbb{N}$. The Module-LWE distribution (induced by $\mathbf{s} \in \mathcal{R}_q^\ell$) is the distribution of the random pair $(\mathbf{a}_i, b_i)$ over the support $\mathcal{R}_q^\ell \times \mathcal{R}_q$, where $\mathbf{a}_i \leftarrow \mathcal{R}_q^\ell$ is uniform and $b_i := \mathbf{a}_i^T\mathbf{s} + e_i$ with $\mathbf{s} \leftarrow S_\eta^\ell$ common to all samples and $e_i \leftarrow S_\eta$ fresh for every sample. Given arbitrarily many samples drawn from the Module-LWE distribution induced by $\mathbf{s} \leftarrow S_\eta^\ell$, the (search) Module-LWE problem asks to recover $\mathbf{s}$. And the associated Module-LWE assumption states that given $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ and $\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e}$ where $k = \mathrm{poly}(\lambda)$ and $(\mathbf{s}, \mathbf{e}) \leftarrow S_\eta^\ell \times S_\eta^k$, no probabilistic polynomial-time algorithm can succeed in recovering $\mathbf{s}$ with non-negligible probability, provided that the parameters are appropriately chosen. Likewise, given $\mathbf{A} \leftarrow \mathcal{R}_q^{h\times\ell}$ and $\mathbf{t} \leftarrow \mathcal{R}_q^h$ where $h = \mathrm{poly}(\lambda)$, the Module-SIS problem parameterized by $\beta > 0$ asks to find a pre-image $\mathbf{x} \in \mathcal{R}_q^{h+\ell}$ such that $[\mathbf{A} \,|\, \mathbf{I}] \cdot \mathbf{x} = \mathbf{t}$ and $\|\mathbf{x}\| \leq \beta$. And the associated Module-SIS assumption states that no probabilistic polynomial-time algorithm can find a desired pre-image $\mathbf{x}$ with non-negligible probability, provided that the parameters are appropriately chosen.

2.3 Rejection Sampling

Lemma 1 (Rejection Sampling #1). *Let $f : \mathbb{Z}^n \to \mathbb{R}$ be a probability distribution. Given a subset $V \subseteq \mathbb{Z}^n$, let $h : V \to \mathbb{R}$ be a probability distribution defined on V. Let $g_{\boldsymbol{v}} : \mathbb{Z}^n \to \mathbb{R}$ be a family of probability distributions indexed by $\boldsymbol{v} \in V$ such that for almost all $\boldsymbol{v}$'s from h, there exists a universal upper bound $M \in \mathbb{R}$ such that*

$$\Pr_{z\leftarrow f}\left[M \cdot g_{\boldsymbol{v}}(z) < f(z)\right] = \mathsf{negl}(\lambda).$$

Then the output distribution of the following two algorithms have negligible statistical difference:

1: $\boldsymbol{v} \leftarrow h$;
2: $z \leftarrow g_{\boldsymbol{v}}$;
3: Output $(z, \boldsymbol{v})$ *with probability* $\min\left(1, \frac{f(z)}{M \cdot g_{\boldsymbol{v}}(z)}\right)$

1: $\boldsymbol{v} \leftarrow h$;
2: $z \leftarrow f$;
3: Output $(z, \boldsymbol{v})$ *with probability* $\frac{1}{M}$

The following corollary follows immediately from Lemma 1.

Corollary 1 (Rejection Sampling #2). *Let $f : \mathbb{Z}^n \to \mathbb{R}$ be a probability distribution. Given a subset $V \subseteq \mathbb{Z}^n$, let $h : V \to \mathbb{R}$ be a probability distribution defined on V. Let $g_{\boldsymbol{v}} : \mathbb{Z}^n \to \mathbb{R}$ be a family of probability distributions indexed by $\boldsymbol{v} \in V$ such that for almost all $\boldsymbol{v}$'s from h there exists a universal upper bound $M \in \mathbb{R}$ such that*

$$\Pr_{z\leftarrow f}\left[M \cdot g_{\boldsymbol{v}}(z) < f(z)\right] = \mathsf{negl}(\lambda).$$

Then the output distribution of the following two algorithms have negligible statistical difference:

1: $\boldsymbol{v} \leftarrow h$;
2: $z \leftarrow g_{\boldsymbol{v}}$;
3: Repeat with probability $1 - \min\left(1, \frac{f(z)}{M \cdot g_{\boldsymbol{v}}(z)}\right)$;
4: Output $(z, \boldsymbol{v})$;

1: $\boldsymbol{v} \leftarrow h$;
2: $z \leftarrow f$;
3: Repeat with probability $1 - \frac{1}{M}$;
4: Output $(z, \boldsymbol{v})$;

Corollary 2 (Rejection Sampling #2'). *Let $f : \mathbb{Z}^n \to \mathbb{R}$ be a probability distribution. Given a bounded subset $V \subseteq \mathbb{Z}^n$, let $h : V \to \mathbb{R}$ be a probability distribution defined on V. Define B_v to be a positive integer such that $\Pr_{\boldsymbol{v} \leftarrow h}[\|\boldsymbol{v}\|_\infty > B_v] < 2^{-\kappa}$, where κ is a parameter indicated the level of precision of B_v. After sampling $\boldsymbol{v} \leftarrow h$, we reject $\boldsymbol{v}$ if $\|\boldsymbol{v}\|_\infty > B_v$, suppose this process of tailoring only happens with negligible probability. Let $g_{\boldsymbol{v}} : \mathbb{Z}^n \to \mathbb{R}$ be a family of probability distributions indexed by $\boldsymbol{v} \in V$ such that for almost all tailored $\boldsymbol{v}$ there exists a universal upper bound $M \in \mathbb{R}$ such that*

$$\Pr_{z \leftarrow f}[M \cdot g_{\boldsymbol{v}}(\boldsymbol{z}) < f(\boldsymbol{z})] = \mathsf{negl}(\lambda).$$

Then the output distribution of the following two algorithms have negligible statistical difference:

1: $\boldsymbol{v} \leftarrow h$;
2: repeat if $\|\boldsymbol{v}\|_\infty > B_v$;
3: $\boldsymbol{z} \leftarrow g_{\boldsymbol{v}}$;
4: Repeat with probability $1\text{–}\min\left(1, \frac{f(\boldsymbol{z})}{M \cdot g_{\boldsymbol{v}}(\boldsymbol{z})}\right)$;
5: Output $(\boldsymbol{z}, \boldsymbol{v})$;

1: $\boldsymbol{v} \leftarrow h$;
2: repeat if $\|\boldsymbol{v}\|_\infty > B_v$;
3: $\boldsymbol{z} \leftarrow f$;
4: Repeat with probability $1 - \frac{1}{M}$;
5: Output $(\boldsymbol{z}, \boldsymbol{v})$;

2.4 Extendable Output Function

The notion of extendable output function follows that of [DLL$^+$17, LDK$^+$17]. An *extendable output function* Sam is a function on bit string in which the output can be extended to any desired length, and the notation $y \in S := \mathsf{Sam}(x)$ represents that the function Sam takes as input x and then produces a value y that is distributed according the distribution S (or uniformly over a set S). The whole procedure is *deterministic* in the sense that a given x will always output the same y. For simplicity we assume that the output distribution of Sam is perfect, whereas in practice it will be implemented using cryptographic hash functions (modelled as random oracle) and produce an output that is statistically close to the perfect distribution.

2.5 Hashing

Our signature scheme to be introduced can be proved secure in the random oracle model. Let $B_w \subsetneq \mathcal{R}_q$ denote the set of elements in $\mathcal{R}_q$ that have *exactly* w coefficients that are either -1 or 1 and the rest are 0. It is always the case that $w = 60$ in this paper, since the set $B_{60} \subseteq \mathcal{R}_q$ is of size $2^{60} \cdot \binom{n}{60} \approx 2^{256}$. Let $H : \{0,1\}^* \to B_{60}$ be a hash function that is modeled as a random oracle in this work. In practice, to pick a random element in B_{60}, we can use an inside-out version of Fisher-Yates shuffle.

3 Building Tools of Our Signature Scheme

In this section, we give the definition of deterministic symmetric key consensus (DKC). Then we construct and analyze a DKC called the rounded symmetric key

consensus with noise (RKCN) (Algorithm 1), which is a variant of the optimal key consensus with noise (OKCN) scheme presented in [JZ16]. Based on RKCN, we propose several algorithms and develop some of their properties. For space limitation, the proofs of the theorems and properties developed in this section are listed in Appendix **??**. Jumping ahead, our signature scheme to be introduced in Section 4 is built upon these algorithms. Note that these algorithms can be naturally generalized to vectors in the component-wise manner.

Definition 3. *A DKC scheme* $DKC = (\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$*, is specified as follows.*

- $\mathsf{params} = (q, k, g, d, aux)$ *denotes the system parameters, where* q, k, g, d *are positive integers satisfying* $2 \leq k, g \leq q, 0 \leq d \leq \lfloor \frac{q}{2} \rfloor$*, and* aux *denotes some auxiliary values that are usually determined by* (q, k, g, d) *and could be set to be a special symbol* $\emptyset$ *indicating "empty".*
- $(k_1, v) \leftarrow \mathsf{Con}(\sigma_1, \mathsf{params})$*: On input of* $(\sigma_1 \in \mathbb{Z}_q, \mathsf{params})$*, the deterministic conciliation algorithm* Con *outputs* (k_1, v)*, where* $k_1 \in \mathbb{Z}_k$ *is the shared-key, and* $v \in \mathbb{Z}_g$ *is a hint signal that will be publicly delivered to the communicating peer to help the two parties reach consensus.*
- $k_2 \leftarrow \mathsf{Rec}(\sigma_2, v, \mathsf{params})$*: On input of* $(\sigma_2 \in \mathbb{Z}_q, v, \mathsf{params})$*, the* deterministic *polynomial-time reconciliation algorithm* Rec *outputs* $k_2 \in \mathbb{Z}_k$*.*

Correctness: *A DKC scheme is* correct*, if it holds* $k_1 = k_2$ *for any* $\sigma_1, \sigma_2 \in \mathbb{Z}_q$ *such that* $|\sigma_1 - \sigma_2|_q \leq d$*.*

We give the construction and analysis of the rounded symmetric key consensus with noise (RKCN), the illustration diagram of which is given in Algorithm 1. We sometimes omit "params" for simplicity. Note that RKCN is DKC according to Theorem 1, if we set parameters properly.

Algorithm 1 RKCN: Rounded Symmetric KC with Noise

$\mathsf{params} = (q, k, g, d, aux)$, $aux = \{q' = kq, \alpha = k, \beta = q\}$, q is prime
procedure $\textsc{Con}(\sigma_1, \mathsf{params})$ ▷ $\sigma_1 \in [0, q-1]$
  $v = k\sigma_1 \bmod^{\pm} q$
  if $k\sigma_1 - v = kq$ **then**
    $k_1 = 0$
  else
    $k_1 = (k\sigma_1 - v)/q$
  end if
  return (k_1, v)
end procedure
procedure $\textsc{Rec}(\sigma_2, v, \mathsf{params})$ ▷ $\sigma_2 \in [0, q-1]$
  $k_2 = \lfloor (k\sigma_2 - v)/q \rceil \bmod k$
  return k_2
end procedure

Theorem 1. *Suppose that the system parameters satisfy* $2kd < q$ *where* $k \geq 2$ *and* $g \geq 2$*. Then, the* RKCN *scheme is* correct*.*

Before proving Theorem 1, we first review the following lemma proved in [JZ16].

Lemma 2 ([JZ16]). *For any $x, y, t, l \in \mathbb{Z}$ where $t \geq 1$ and $l \geq 0$, if $|x - y|_t \leq l$, then there exists $\theta \in \mathbb{Z}$ and $\delta \in [-l, l]$ such that $x = y + \theta t + \delta$.*

Proof. (of Theorem 1) Suppose $|\sigma_1 - \sigma_2|_q \leq d$, by Lemma 2, there exist $\theta \in \mathbb{Z}$ and $\delta \in [-d, d]$ such that $\sigma_2 = \sigma_1 + \theta q + \delta$. From Line 3 to 7 in Algorithm 1, we know that there exists $\theta' \in \mathbb{Z}$ such that $k\sigma_1 = (k_1 + k\theta') \cdot q + v$. Taking these into the formula of k_2 in Rec (Line 12), we have

$$\begin{aligned} k_2 &= \lfloor (k\sigma_2 - v)/q \rceil \bmod k && (1)\\ &= \lfloor k(\sigma_1 + \theta q + \delta)/q - v/q \rceil \bmod k && (2)\\ &= \lfloor k_1 + k\delta/q \rceil \bmod k && (3) \end{aligned}$$

From the assumed condition $2kd < q$, we get that $|k\delta/q| \leq kd/q < 1/2$, thus $k_2 = k_1$. □

Based on RKCN, we propose several algorithms as follows. $\mathsf{HighBits}_{q,k}$ is the routine that extracts r_1, from the output of $\mathsf{Con}(r)$ of RKCN. Given $r, z \in \mathbb{Z}_q$, in order to derive $\mathsf{HighBits}_{q,k}(r + z)$ from r, q, k, we propose a procedure $\mathsf{MakeHint}_{q,k}$ to produce a 1-bit hint h. The procedure $\mathsf{UseHint}_{q,k}$ shows how to use the hint h to recover $\mathsf{HighBits}_{q,k}(r + z)$.

```
1: procedure HIGHBITS_{q,k}(r)
2:     (r_1, r_0) ← Con(r)
3:     return r_1
4: end procedure
5:
6: procedure MAKEHINT_{q,k}(z, r)
7:     r_1 := HighBits_{q,k}(r)
8:     v_1 := HighBits_{q,k}(r + z)
9:     if r_1 = v_1 then
10:        return 0
11:    else
12:        return 1
13:    end if
14: end procedure
```

```
1: procedure USEHINT_{q,k}(h, r)
2:     (r_1, r_0) := Con(r)
3:     if h = 0 then
4:         return r_1
5:     else if h = 1 and r_0 > 0 then
6:         return (r_1 + 1) mod k
7:     else
8:         return (r_1 − 1) mod k
9:     end if
10: end procedure
```

Proposition 1. *For every $r, z \in \mathbb{Z}_q$ such that $\|z\|_\infty < \lfloor q/(2k) \rfloor$, we have*

$$\mathsf{UseHint}_{q,k}(\mathsf{MakeHint}_{q,k}(z, r), r) = \mathsf{HighBits}_{q,k}(r + z).$$

Proof. The outputs of $(r_1, r_0) \leftarrow \mathsf{Con}(r)$, $(r_1', r_0') \leftarrow \mathsf{Con}(r + z)$ satisfy $0 \leq r_1, r_1' < k$, and $\|r_0\|_\infty, \|r_0'\|_\infty \leq q/2$. Since $\|z\|_\infty < \lfloor q/(2k) \rfloor$, by Theorem 1, we have

$\mathsf{Rec}(r, r'_0) = r'_1 = \mathsf{HighBits}_{q,k}(r+z)$. Let $h \stackrel{\text{def}}{=} \mathsf{MakeHint}_{q,k}(z, r)$. Since $r'_1 = \mathsf{Rec}(r, r'_0) = \lfloor (kr - r'_0)/q \rceil \bmod k = \lfloor r_1 + (r_0 - r'_0)/q \rceil \bmod k \in \{r_1 - 1, r_1, r_1 + 1\}$. When $r_0 > 0$, we have $\mathsf{Rec}(r, r'_0) \in \{r_1, r_1 + 1\}$; when $r_0 < 0$, we have $\mathsf{Rec}(r, r'_0) \in \{r_1 - 1, r_1\}$. Recall that by definition, $h = 0$ if and only if $r_1 = r'_1$. The correctness of $\mathsf{HighBits}_{q,k}(r+z) = r'_1 = \mathsf{Rec}(r, r'_0) = \mathsf{UseHint}_{q,k}(h, r)$ is thus established. □

Proposition 2. *For $r'_1 \in \mathbb{Z}_k$, $r \in \mathbb{Z}_q$, $h \in \{0, 1\}$, if $r'_1 = \mathsf{UseHint}_{q,k}(h, r)$, then $\|r - \lfloor q \cdot r'_1/k \rceil\|_\infty \leq q/k + 1/2$.*

Proof. It is routine to see that for $(r_1, r_0) \leftarrow \mathsf{Con}(r)$, we have $r_1 \in \mathbb{Z}_k, r_0 \in (-q/2, q/2)$, and there exists $\theta \in \{0, 1\}$ such that $k \cdot r = (r_1 + k\theta) \cdot q + r_0$. If $h = 0$, then $r'_1 = r_1$, and hence $\|r - \lfloor q \cdot r'_1/k \rceil\|_\infty \leq q/(2k) + 1/2$. If $h = 1$ and $r_0 > 0$, then $r'_1 = r_1 + 1 \bmod k$, and hence $\|r - \lfloor q \cdot r'_1/k \rceil\|_\infty \leq q/k + 1/2$. Finally, if $h = 1$ and $r_0 < 0$, then $r'_1 = r_1 - 1 \bmod k$, and therefore $\|r - \lfloor q \cdot r'_1/k \rceil\|_\infty \leq q/k + 1/2$. □

Proposition 3. *For $r, z \in \mathbb{Z}$ such that $\|z\|_\infty < U$. If $\|r'_0\|_\infty < q/2 - kU$ where $(r_1, r_0) \leftarrow \mathsf{Con}(r), (r'_1, r'_0) \leftarrow \mathsf{Con}(r+z)$, then $r_1 = r'_1$.*

Proof. Since $k \cdot r = q \cdot (r_1 + k\theta) + r_0$ and $k \cdot (r+z) = q \cdot (r'_1 + k\theta') + r'_0$ for some integers θ, θ', it is easy to verify $r_1 = \lfloor kr/q \rceil \bmod k = \lfloor k(r+z-z)/q \rceil \bmod k = \lfloor r'_1 + (r'_0 - kz)/q \rceil \bmod k = r'_1$. □

4 Construction of Lattice-Based Signature Based on RKCN

In this work, we have $n = 256, q = 4191233, \mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$, and $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. Also, $S_\eta = \{a \in \mathcal{R}_q \mid \|a\|_\infty \leq \eta\}$, and $B_{60} \subseteq \mathcal{R}_q$ is the set of elements in $\mathcal{R}_q$ that have *exactly* 60 coefficients that are either -1 or 1 and the rest are 0. Let $H : \{0,1\}^* \to B_{60}$ be a hash function that is modeled as a random oracle in our signature scheme. Let $\mathsf{Sam}(\cdot)$ be an extendable output function (Appendix 2.4). Finally, for every $a = \sum_{i=0}^{n-1} a_i \cdot x^i \in \mathcal{R}_q$, $a_i \in \mathbb{Z}_q$, define $\mathsf{Power2Round}_{q,d}(a) \stackrel{\text{def}}{=} \sum a'_i \cdot x^i$, where $a'_i \stackrel{\text{def}}{=} \left(a_i - \left(a_i \bmod^{\pm} 2^d\right)\right)/2^d$. This definition can be naturally generalized in the component-wise manner.

The signature scheme is parameterized by $q, k, n, h, \ell, d, \omega, \eta, U$. The key generation, signing, and verification algorithms are described in Algorithms 2, 3 and 4, respectively.

Algorithm 2 Key Generation Algorithm

1: **Input**: 1^λ
2: **Output**: $(\mathrm{pk} = (\rho, \mathbf{t}_1), \mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t}))$
3: $\rho, \rho' \leftarrow \{0,1\}^{256}$
4: $\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$
5: $(\mathbf{s}, \mathbf{e}) \in S_\eta^\ell \times S_\eta^h := \mathsf{Sam}(\rho')$
6: $\mathbf{t} := \mathbf{As} + \mathbf{e}$
7: $\mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})$
8: **return** $(\mathrm{pk} = (\rho, \mathbf{t}_1), \mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t}))$

Algorithm 3 The Signing Algorithm

Input: $\mu \in \{0,1\}^*$, $\mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t})$
Output: $\sigma = (\mathbf{z}, c, \mathbf{h})$
$\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$
$\mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})$
$\mathbf{t}_0 := \mathbf{t} - \mathbf{t}_1 \cdot 2^d$
$r \leftarrow \{0,1\}^{256}$
$\mathbf{y} \in S^{\ell}_{\lfloor q/k \rfloor - 1} := \mathsf{Sam}(r)$
$\mathbf{w} := \mathbf{A}\mathbf{y}$
$\mathbf{w}_1 := \mathsf{HighBits}_{q,k}(\mathbf{w})$
$c \leftarrow H(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{w}_1 / k \rceil, \mu)$
$\mathbf{z} := \mathbf{y} + c\mathbf{s}$
$(\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Con}(\mathbf{w} - c\mathbf{e})$
Restart if $\|\mathbf{z}\|_\infty \geq \lfloor q/k \rfloor - U$ or $\|\mathbf{r}_0\|_\infty \geq q/2 - kU$ or $\mathbf{r}_1 \neq \mathbf{w}_1$
$\mathbf{h} := \mathsf{MakeHint}_{q,k}(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{e} + c\mathbf{t}_0)$
Restart if $\|c\mathbf{t}_0\|_\infty \geq \lfloor q/2k \rfloor$ or the number of 1's in $\mathbf{h}$ is greater than ω
return $(\mathbf{z}, c, \mathbf{h})$

Algorithm 4 Verification Algorithm

Input: $\mathrm{pk} = (\rho, \mathbf{t}_1), \mu \in \{0,1\}^*, (\mathbf{z}, c, \mathbf{h})$
Output: $b \in \{0,1\}$
$\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$
$\mathbf{w}'_1 := \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d)$
$c' \leftarrow H(\rho, \mathbf{t}_1, \lfloor q\mathbf{w}'_1 / k \rceil, \mu)$
if $c = c'$ and $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$ and the number of 1's in $\mathbf{h}$ is $\leq \omega$ **then**
    return 1
else
    return 0
end if

The key generation algorithm first chooses a random 256-bit seed ρ and expands it into a matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{h \times \ell}$ by an extendable output function $\mathsf{Sam}(\cdot)$ that is modeled as a random oracle. Conversely, the crucial component in the secret key is $(\mathbf{s}, \mathbf{e}) \in \mathcal{R}_q^h \times \mathcal{R}_q^\ell$, and each coefficient is drawn uniformly at random in the set $[-\eta, \eta]$. Finally, we compute $\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathcal{R}_q^h$. The public key is $\mathrm{pk} = (\rho, \mathbf{t}_1 = \mathsf{Power2Round}_{q,d}(\mathbf{t}))$, whereas the associated secret key is $\mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t})$.

Given the secret key $\mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t})$ as well as the message $\mu \in \{0,1\}^*$ to be signed, the signing algorithm first recovers the public matrix $\mathbf{A} \in \mathcal{R}_q^{h \times \ell}$ via the random seed ρ in the secret key. After that, the signing algorithm picks a short $\mathbf{y}$ from the set $S^{\ell}_{\lfloor q/k \rfloor - 1} \subseteq \mathcal{R}_q^\ell$ uniformly at random, and computes $\mathbf{w}_1 := \mathsf{HighBits}_{q,k}(\mathbf{w})$ where $\mathbf{w} := \mathbf{A}\mathbf{y}$. The random oracle $H(\cdot)$, upon input $(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{w}_1 / k \rceil, \mu)$, returns a uniform c from the set $B_{60} \subseteq \mathcal{R}_q$ (cf. Appendix 2.5). After obtaining c, the signing algorithm conducts a rejection sampling process (cf. Appendix 2.3) to check if every coefficient of $\mathbf{z} := \mathbf{y} + c\mathbf{s} \in \mathcal{R}_q^\ell$ is "small" enough, if every coefficient of $\mathbf{r}_0$ is "small" enough, and if $\mathbf{r}_1 = \mathbf{w}_1$, where $(\mathbf{r}_1, \mathbf{r}_0) \leftarrow \mathsf{Con}(\mathbf{w} - c\mathbf{e})$; otherwise, the signing algorithm restarts, until all the foregoing conditions are satisfied. We should point out that if $\|c\mathbf{e}\|_\infty < U$, then by Proposition 3, $\|\mathbf{r}_0\|_\infty < q/2 - kU$ implies $\mathbf{r}_1 = \mathbf{w}_1$. We want $\|c\mathbf{e}\|_\infty < U$ happen with negligible probability, such that the probability of using the last check (whether $\mathbf{r}_1 = \mathbf{w}_1$) is negligible as well. Furthermore, the function $\mathsf{MakeHint}_{q,k}(\cdot)$ is invoked on input $(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{e} + c\mathbf{t}_0)$ to generate the hint $\mathbf{h}$, *i.e.*, a binary vector in $\{0,1\}^{n \cdot h}$. The signing algorithm concludes by conducting the second reject sampling process, *i.e.*, if $\|c\mathbf{t}_0\|_\infty \geq \lfloor q/2k \rfloor$ and if the number of nonzero elements in $\mathbf{h} \in \{0,1\}^{n \cdot h}$ does not exceed the pre-defined threshold ω; otherwise restart is carried out again. Here, the hint

$\mathbf{h}$ corresponds to the fact that it is $\mathbf{t}_1$, not the whole $\mathbf{t} = \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$, that is contained in the public key. With the hint $\mathbf{h}$, we can still carry out the verification, even without $\mathbf{t}_0$.

Given the public key $\mathrm{pk} = (\rho, \mathbf{t}_1)$, the message $\mu \in \{0,1\}^*$ and the claimed signature $(\mathbf{z}, c, \mathbf{h})$, the verifying algorithm first recovers the public matrix $\mathbf{A} \in \mathcal{R}_q^{h \times \ell}$ via the random seed ρ. After that, it computes $\mathbf{w}_1' := \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d)$. If the given $(\mathbf{z}, c, \mathbf{h})$ is indeed a *valid* signature of the message μ, then it is routine to see that every coefficient of $\mathbf{z}$ is "small" enough, and the number of 1's in $\mathbf{h}$ is no greater than ω; more importantly, we have $\mathsf{HighBits}_{q,k}(\mathbf{Ay}) = \mathsf{HighBits}_{q,k}(\mathbf{Ay} - c\mathbf{e}) = \mathbf{w}_1'$ and therefore $c = c'$, where $c' \leftarrow H(\rho, \mathbf{t}_1, \lfloor q\mathbf{w}_1'/k \rceil, \mu)$. The verifying algorithm would accept the input tuple if and only if the foregoing conditions are all satisfied.

Next, we show that our signature scheme is always correct, *provided that the involving parameters are appropriately set*. Roughly speaking, the correctness relies heavily on Properties 1 - 3.

When the public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ is fixed, for a *valid* message/signature pair $(\mu, (\mathbf{z}, c, \mathbf{h}))$, first it is routine to see that $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$ and the number of nonzero entries in $\mathbf{h}$ is no more than ω. Moreover, since $\|c\mathbf{t}_0\|_\infty < q/2k$ and $\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d = \mathbf{Ay} - c\mathbf{e} + c\mathbf{t}_0$, it follows directly from Proposition 1 that

$$\mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d) = \mathsf{HighBits}_{q,k}(\mathbf{Ay} - c\mathbf{e}).$$

Given that the signing algorithm forces $\mathsf{HighBits}_{q,k}(\mathbf{Ay} - c\mathbf{e}) = \mathsf{HighBits}_{q,k}(\mathbf{Ay})$ by rejection sampling, we thus have

$$\mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d) = \mathsf{HighBits}_{q,k}(\mathbf{Ay} - c\mathbf{e}) = \mathsf{HighBits}_{q,k}(\mathbf{Ay}),$$

and hence $c = c'$. This finishes the correctness analysis of our signature scheme.

5 Security Proof

In this section, we analyze the security of the RKCN-based signature scheme, in terms of strongly existentially unforgeable under adaptive-chosen message attack (cf. Appendix 2.1). The proof can be viewed as a generalization of that of Dilithium [DLL+17, LDK+17], for the realistic setting where $\mathbf{t}_0$ is hidden to the attacker as well as to the simulator. Roughly speaking, our security proof consists of two phases: In Phase I, we show that the behavior of the signing oracle is indistinguishable from that of an efficient simulator; In Phase II, we show that the any efficient attacker cannot forge a valid message/signature pair, after interacting with the foregoing simulator polynomially many times.

5.1 Security Proof in Phase I: the Simulator

Recall that in [DLL+17], when the simulation is conducted, the public key is assumed to be $(\rho, \mathbf{t})$ so as to simplify the analysis. However, this is too conservative, since in [DLL+17] the *real* public key is $(\rho, \mathbf{t}_1)$, without $\mathbf{t}_0 = \mathbf{t} - \mathbf{t}_1 \cdot 2^d$ at all. As we shall see later, it is very difficult for the efficient adversary to obtain some information regarding $\mathbf{t}_0$.

For our signature scheme, the construction of the simulator in Phase I follows that of [DLL$^+$17], with a key modification or generalization: in our construction, the public key given to the simulator is the *real* public key, *i.e.*, $(\rho, \mathbf{t}_1)$, instead of $(\rho, \mathbf{t})$.

To do so, we define an oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ parameterized by $\mathbf{t}_0 = \mathbf{t} - \mathbf{t}_1 \cdot 2^d \in S^h_{2^{d-1}}$, $\mathsf{params} = (\mathbf{t}_1, q, n, k, h, \ell, d, U, \rho, \omega)$ as well as the random oracle $H(\cdot)$. Fix an efficient adversary A against the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$. For simplicity, we assume that A makes $q_s = \mathsf{poly}(\lambda)$ queries to $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$. Upon the ith query with message μ_i from the adversary A, where $1 \leq i \leq q_s$, the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ first chooses $c^{(i)} \leftarrow B_{60}$ and $\mathbf{z}^{(i)} \leftarrow S^{\ell}_{\lfloor q/k \rfloor - U - 1}$ uniformly at random. Then it computes $\left(\mathbf{r}_1^{(i)}, \mathbf{r}_0^{(i)}\right) \leftarrow \mathsf{Con}\left(\mathbf{A}\mathbf{z}^{(i)} - c^{(i)}\mathbf{t}\right)$. The whole process restarts until $\left\|\mathbf{r}_0^{(i)}\right\|_\infty < q/2 - kU$ and $\left\|c^{(i)}\mathbf{t}_0\right\|_\infty < q/2k$. Finally, the oracle programs $c^{(i)} = H(\rho, \mathbf{t}_1, \left\lfloor q \cdot \mathbf{r}_1^{(i)}/k \right\rceil, \mu_i)$, computes $\mathbf{h}^{(i)} = \mathsf{MakeHint}_{q,k}\left(-c^{(i)}\mathbf{t}_0, \mathbf{A}\mathbf{z}^{(i)} - c^{(i)}\mathbf{t}_1 \cdot 2^d\right)$, restarts if necessary, and finally outputs $(\mathbf{z}^{(i)}, c^{(i)}, \mathbf{h}^{(i)})$.

Similar to [DLL$^+$17], the resulting $\left(\mathbf{z}^{(i)}, c^{(i)}\right)$ output by the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ follows the same distribution as that of the real signing oracle in the standard security game, *provided that collision occurs with negligible probability*. It remains to show that for each query $\mu_i, 1 \leq i \leq q_s$, the probability that $H\left(\rho, \mathbf{t}_1, \left\lfloor q \cdot \mathbf{r}_1^{(i)}/k \right\rceil, \mu_i\right)$ was already programmed previously is negligible. In fact, this follows directly from the following lemma, whose proof is similar to that of [KLS18] and thus is omitted for simplicity.

Lemma 3.

$$\Pr_{\mathbf{A} \leftarrow \mathcal{R}_q^{h \times \ell}}\left[\forall \boldsymbol{w}_1^* : \Pr_{\mathbf{y} \leftarrow S^{\ell}_{\lfloor q/k \rfloor - 1}}\left[\mathsf{HighBits}_{q,k}(\boldsymbol{A}\boldsymbol{y}) = \boldsymbol{w}_1^*\right] \leq \left(\frac{q/k+1}{2 \cdot \lfloor q/k \rfloor - 1}\right)^n\right] > 1 - (n/q)^{h\ell}.$$

□

It should be stressed that, both $\left(\frac{q/k+1}{2\lfloor q/k \rfloor - 1}\right)^n \ll 2^{-128}$ and $(n/q)^{h\ell} \ll 2^{-128}$ holds, for our recommended parameters in Table 1. This justifies our correctness analysis regarding the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ as well as our choice of parameters.

Remark The proof of Phase-1 is a generalization of that for Dilithium [DLL$^+$17, LDK$^+$17], where the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ is not needed if $\mathbf{t}_0$ is given to the simulator.

5.2 Security Proof in Phase II

In this section, we finish the Phase II of our security proof. Specifically, we argue that it is very difficult for every efficient adversary to forge a signature, after interacting polynomially with the aforementioned simulator in the security game. To do so, we define two related problems, *i.e.*, the Oracle-SelfTargetMSIS problem and the Oracle-MSIS problem, which could be seen as variants of the SelfTargetMSIS problem and the MSIS problem proposed in [KLS18], and might be of independent interest. We remark that,

the security proof of Dilithium is also based on similar problems, if t_0 is hidden from the adversary in reality.

First comes the definition of Oracle-SelfTargetMSIS problem. Informally, this problem is the underlying hard problem upon which the new message forgery is based. Recall that $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$ is an oracle defined in Section 5.1, and $H : \{0,1\}^* \to B_{60}$ is a cryptographic hash function that is modeled as a random oracle in our security game. In the Oracle-SelfTargetMSIS problem, given a random matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{h' \times \ell}$ for positive integer h' as well as access to the random oracle $H(\cdot)$ and the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$, the solver is asked to output a pair $\left(\mathbf{y} := \begin{bmatrix}\mathbf{r}\\ c\end{bmatrix}, \mu\right)$ such that $0 \leq \|\mathbf{y}\|_\infty \leq \gamma$ and $c = H\left(\rho, \mathbf{t}_1, [\mathbf{I} \,|\, \mathbf{A}] \cdot \mathbf{y}, \mu\right)$. Equivalently, for an *efficient* adversary A, its advantage in solving the Oracle-SelfTargetMSIS problem is defined as follows:

$$\mathsf{Adv}^{\mathsf{O\text{-}SelfTargetMSIS}}_{H,O,\ell,h',\gamma}(A) \stackrel{\text{def}}{=} \Pr\left[\begin{array}{r} \mathbf{A} \leftarrow \mathcal{R}_q^{h' \times \ell};\quad \mathbf{t} \leftarrow \mathcal{R}_q^{h'};\\ \mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})\,;\, \mathbf{t}_0 := \mathbf{t} - \mathbf{t}_1 \cdot 2^d;\\ \left(\mathbf{y} := \begin{bmatrix}\mathbf{r}\\ c\end{bmatrix}, \mu\right) \leftarrow A^{H(\cdot), O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)}(\mathbf{A}) \end{array} : \begin{array}{c} 0 \leq \|\mathbf{y}\|_\infty \leq \gamma, \text{ and}\\ c = H\left(\rho, \mathbf{t}_1, [\mathbf{I} \,|\, \mathbf{A}] \cdot \mathbf{y}, \mu\right)\end{array}\right]$$

Likewise, the Oracle-MSIS problem is defined as follows: given a random matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{h' \times \ell}$ and $\mathbf{t} \leftarrow \mathcal{R}_q^{h'}$ as well as access to the random oracle $H(\cdot)$ and the oracle $O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)$, the problem asks to find a pre-image $\mathbf{x} \in \mathcal{R}_q^{h'+\ell}$ such that $\|\mathbf{x}\|_\infty \leq \beta$ and more importantly, $[\mathbf{A} \,|\, \mathbf{I}] \cdot \mathbf{x} = \mathbf{0}$.

To be precise, for an efficient adversary A, its advantage in solving the Oracle-MSIS problem is defined as follows:

$$\mathbf{Adv}^{\mathsf{o\text{-}msis}}_{h',h'+\ell,\beta}(A) \stackrel{\text{def}}{=} \Pr\left[\begin{array}{r} \mathbf{A} \leftarrow \mathcal{R}_q^{h' \times \ell};\quad \mathbf{t} \leftarrow \mathcal{R}_q^{h};\\ \mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})\,;\, \mathbf{t}_0 := \mathbf{t} - \mathbf{t}_1 \cdot 2^d;\\ \mathbf{x} \leftarrow A^{H(\cdot), O_{\mathbf{t}_0,\mathsf{params},H(\cdot)}(\cdot)}(\mathbf{A}) \end{array} : \begin{array}{c} \mathbf{x} \in \mathcal{R}^{h'+\ell}, \text{ and}\\ [\mathbf{A} \,|\, \mathbf{I}] \cdot \mathbf{x} = \mathbf{0}, \text{ and}\\ \|\mathbf{x}\|_\infty \leq \beta\end{array}\right].$$

The following analysis shows that for an efficient adversary, to forge a valid message/signature pair implies that she can solve the Oracle-SelfTargetMSIS problem.

In the standard security game of our signature scheme, to forge a signature implies that the adversary receives a random $(\mathbf{A}, \mathbf{t}_1)$ and outputs a valid message/signature pair $(\mu, (\mathbf{z}, c, \mathbf{h}))$ such that

- $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$; and
- the number of nonzero entries in $\mathbf{h}$ is no more than ω; and
- $c = H\left(\rho, \mathbf{t}_1, \left\lfloor q \cdot \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d)/k \right\rceil, \mu\right)$.

By Proposition 2, we have

$$\left\lfloor q \cdot \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d)/k \right\rceil = \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d + \mathbf{u},$$

where $\|\mathbf{u}\|_\infty \leq q/k + 1/2$. Given that $\mathbf{t} = \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$ where $\|\mathbf{t}_0\|_\infty \leq 2^{d-1}$, the foregoing equality can be rewritten as

$$\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d + \mathbf{u} = \mathbf{Az} - c\mathbf{t} + (c\mathbf{t}_0 + \mathbf{u}) = \mathbf{Az} - c\mathbf{t} + \mathbf{u}'.$$

Note that the worst-case upper-bound for $\mathbf{u}'$ is

$$\|\mathbf{u}'\|_\infty \leq \|c\mathbf{t}_0\|_\infty + \|\mathbf{u}\|_\infty \leq 60 \cdot 2^{d-1} + q/k + 1/2.$$

Thus an adversary who can forge a valid message/signature pair is able to find $\mathbf{z}$, c, $\mathbf{u}'$, μ such that $\|\mathbf{z}\|_\infty \leq \lfloor q/k \rfloor - U$, $\|c\|_\infty = 1$, $\|\mathbf{u}'\|_\infty \leq 60 \cdot 2^{d-1} + q/k + 1/2, \mu \in \{0,1\}^*$, and more importantly

$$H\left(\rho, \mathbf{t}_1, \left[\mathbf{A} \mid \mathbf{t} \mid \mathbf{I}\right] \cdot \begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{u}' \end{bmatrix}, \mu\right) = c.$$

This is exactly the Oracle-SelfTargetMSIS problem defined previously. By the standard forking lemma [BN06], it can be shown that an efficient adversary who can solve the foregoing Oracle-SelfTargetMSIS problem can be adapted to solve the Oracle-MSIS problem. Clearly the forking lemma suffers from its non-tightness. However, such loss is usually ignored in practice when setting the parameters.

Similar to [DLL$^+$17], the hardness of the Oracle-SelfTargetMSIS problem relies on the following intuition. Given that H is a cryptographic hash function whose structure is completely independent of the algebraic structure of its inputs, it seems the choice of μ does not help in solving the equation. Thus, the problem would be equally hard if the message μ was chosen and fixed. Furthermore, given the independence of H and the algebraic structure of its inputs, it seems that the only plausible forgery attack appears to first pick some $\mathbf{w}'$, compute $H(\rho, \mathbf{t}_1, \mathbf{w}', \mu) = c$, and then find $\mathbf{z}, \mathbf{u}'$ such that $\|\mathbf{z}\|_\infty \leq \lfloor q/k \rfloor - U$, $\|\mathbf{u}'\|_\infty \leq 60 \cdot 2^{d-1} + q/k + 1/2$, and furthermore, $\mathbf{A}\mathbf{z} + \mathbf{u}' = \mathbf{w}' + c\mathbf{t}$. The problem now can be reduced to finding $\mathbf{z}, \mathbf{u}'$ with ℓ_∞-norm less than $60 \cdot 2^{d-1} + q/k + 1/2$ such that

$$\mathbf{A}\mathbf{z} + \mathbf{u}' = \mathbf{t}'$$

for the given $\mathbf{A}, \mathbf{t}'$, which is exactly the Oracle-MSIS problem defined previously. Note that this is conservative because $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U < 60 \cdot 2^{d-1} + q/k + 1/2$. Furthermore, only ω coefficients of $\mathbf{u}'$ can be larger than $\lfloor q/k \rfloor$.

In sum, the hardness of our signature scheme is boiled down to the hardness of the Oracle-MSIS problem. Again, if we assume $\mathbf{t}_0$ is public as in [DLL$^+$17, LDK$^+$17], the security is reduced to the MSIS problem in the infinity Norm. The concrete hardness of the Oracle-MSIS problem is analyzed in detail in Appendix B.2.

6 Recommended Parameters

To choose parameters, the following requirements or goals should be taken into account simultaneously.

- First, the parameters should be appropriately chosen so as to ensure the correctness of our signature scheme.
- Second, the involved parameters should be chosen with the goal of achieving 128-bit quantum security.
- Moreover, the parameters should be chosen such that the expected number of repetitions in the signing algorithm should be as small as possible, so as to ensure the efficiency of the signing algorithm.
- Finally, the parameters should be chosen such that the sum of the public key size and the signature size should be as minimal as possible.

	q	n	(h,ℓ)	η	d	k	U	ω
Recomm. Para.	4191233	256	$(5,4)$	2	13	16	118	96

Table 1. The recommended parameter set for our signature scheme.

Under such considerations, we recommend the following set of parameters for our signature scheme in Table 1.

Note that in our signature scheme, the public key size is of $32+h\cdot 32\cdot(\lceil\log(q)\rceil-d)$ bytes, whereas the signature size is of $\ell\cdot 32\cdot\lceil\log(2\cdot\lfloor q/k\rfloor)\rceil+(32+8)+(\omega+h-1)$ bytes. In regard to the expected number of repetitions, it depends on the probabilities that the two rejection sampling steps happen. The probability that the first restart occurs is roughly

$$\left(\frac{2(\lfloor q/k\rfloor-U)-1}{2\lfloor q/k\rfloor-1}\right)^{\ell\cdot n}\cdot\left(\frac{2(\lfloor q/2\rfloor-kU)-1}{q}\right)^{h\cdot n}.$$

Here, the parameter U is carefully chosen such that $\Pr\left[\|c\mathbf{e}\|_\infty\geq U\right]<2^{-128}$.

In regard to the second restart, experiments are carried to estimate the expected number of repetitions, and parameters are chosen such that in the experiments, the second restarts are carried out with probability no more than 1%.

Table 2 is a brief comparison between our recommended signature scheme and (the recommended version of) Dilithium [LDK+17]. Compared with Dilithium, the signature size is reduced, and the efficiency improvement is due to the smaller modulus q and the smaller repetition times.

We have conducted computer experiments with our recommended implementation to verify its efficiency in practice. Our implementation benefits much from that of Dilithium [LDK+17]. The experiments are carried out on a computer equipped with an Intel Core i7-8700T CPU running at 2.40GHz, which runs Ubuntu system (Linux Kernel version 4.15.0, and gcc version 7.3.0). Briefly speaking, each invocation of our key generation, signing, and verification algorithms takes on average 317K, 1,771K, and 338K CPU cycles, respectively; this is roughly in consistency with our earlier estimations.

7 Concrete Security Estimation

For our signature schemes, two types of important attacks should be taken into consideration: the key-recovery attack which aims to recover the secret key, with the given associated public key; the forgery attack which tries to forge a signature in the security game of SEU-CMA (cf. Definition 1). We use the same method proposed in [DLL+17,LDK+17] for measuring the concrete security against forgery attack, and the details are given in Appendix A and B.

	Recomm. Para. of our scheme	Recomm. Para. of Dilithium
q	4191233	8380417
n	256	256
(h, ℓ)	$(5, 4)$	$(5, 4)$
η	2	5
pk size (in byte)	1472	1472
sig. size (in byte)	2572	2701
expected # of repetitions	5.03	6.6
quantum bit-cost against key recovery attack	$119 - 249.4$	128
quantum bit-cost against forgery attack	131.2	125

Table 2. Comparison between our signature and Dilithium.

7.1 Key-Recovery Attack and Module-LWE Problem with Hidden $\mathbf{t}_0$

The efficient adversary may also try to recover the crucial part of the secret key, *i.e.*, $(\mathbf{s}, \mathbf{e}) \in \mathcal{R}_q^h \times \mathcal{R}_q^\ell$, from the associated public key $(\mathbf{A}, \mathbf{t}_1 = \mathsf{Power2Round}_{q,d}(\mathbf{As} + \mathbf{e}))$. This is in essence the key-recovery attack.

Of course, we can follow *exactly* the paradigm of Dilithium by assuming that in addition to others, the adversary is given the $\mathbf{t}_0$ part as well; under this assumption, standard method shows that our signature scheme with the recommended parameter achieves 119-bit security quantumly. However, this assumption does not hold in reality: in the *actual* key-recovery problem, the value $\mathbf{t}_0$ is not given to the adversary *directly*, which makes the key-recovery problem intuitively harder than the standard Module-LWE problem.

From the theoretical analysis perspective, this key-recovery problem in reality, i.e., recovering $(\mathbf{s}, \mathbf{e})$ given $\mathbf{A}$ and $\mathbf{t}_1 = \mathsf{Power2Round}_{q,d}(\mathbf{As} + \mathbf{e})$, has not been well-studied in the literature. Actually, to our knowledge, we explicitly formulate this problem for the first time. Intuitively, from the observation $\mathbf{As} + (\mathbf{e} - \mathbf{t}_0) = \mathbf{t}_1 \cdot 2^d$ this problem looks more like a SIS problem in the infinity norm (rather than Module-LWE), and we could have applied the paradigm suggested by Dilithium [DLL+17, LDK+17] to estimate its hardness. However, computational experiments shows that this paradigm does not work here any longer, mostly because its estimation is roughly > 1000-bit security level quantumly,[3] and is clearly far from reasonable.

Instead, we seek to compare this key-recovery problem with other related problems. It seems to us that this key-recovery problem resembles more like the SVP problem in the infinity norm, and the hardness of this key-recovery problem is no more than

[3] For the SIS instance handled by Dilithium, the upper bound for the ℓ_∞-norm of the desired nonzero vector is roughly about 2^{19}. But the upper-bound for our SIS instance obtained from the equality $\mathbf{As} + (\mathbf{e} - \mathbf{t}_0) - \mathbf{t}_1 \cdot 2^d = \mathbf{0}$ is approximately 2^{12}, which is much smaller than 2^{19}.

the hardness of the associated SVP problem in the infinity norm. The latest theoretical progress [AM18] suggests that with our suggested parameter, the associated SVP instance (in the infinity norm) achieves 249.4-bit security level quantumly; for the full detail, see Section B.3. Thus, we argue that the concrete hardness of our key-recovery instance is between 119 and 249.4-bit security level quantumly. It could be seen as one of our contributions of this work that we relate the key-recovery problem to the SVP problem in its infinity norm.

From the practical perspective, clearly it is hard to recover the whole $\mathbf{t}_0$ intact. It remains unknown to us that in the worst case, how much information regarding $\mathbf{t}_0$ could be recovered by the efficient adversary during the signing process. The following analysis implies that the efficient adversary can obtain very little useful information regarding $\mathbf{t}_0$ in practice. First, the most useful signatures in identifying $\mathbf{t}_0$ are those tuples $(\mathbf{z}, c, \mathbf{h})$ such that $\|c\mathbf{t}_0\|_\infty \geq \lfloor q/2k \rfloor$; however, those tuples are never output by the signing oracle. In comparison, the other tuples, *i.e.*, the valid signatures output by the signing oracle, contain less information regarding $\mathbf{t}_0$, making the adversary very difficult to recover $\mathbf{t}_0$. Furthermore, generally speaking, to obtain more information regarding $\mathbf{t}_0$, more signing queries are needed. However, in practice the number of signing queries is actually bounded. For example, in the NIST's call for the post-quantum cryptosystems, a qualified digital signature scheme should ensure that no adversary can forge a successful forgery after making as many as 2^{64} signing queries to the signing oracle. As another instance, each honest signer in an blockchain-based system like the Bitcoin can sign at most 2^{30} messages in its life time. Consequently, when the efficient attacker tries to recover $\mathbf{t}_0$ by collecting enough c's, she cannot decide whether some specific c cannot be output (because of the check $\|c\mathbf{t}_0\|_\infty \geq \lfloor q/(2k) \rfloor$), or some specific c has not been output (but will be output in the future) due to the practical limit on the number of signing queries. It is thus reasonable to argue that very little information regarding $\mathbf{t}_0$ is leaked to the adversary *in practice*.

Thus, our signature scheme with the recommended parameter achieves at least 119-bit quantum security level (recall that Dilithium achieves 125-bit quantum security level as claimed in [DLL$^+$17, LDK$^+$17]). In practice, it may be well above 128-bit quantum security level. Since $\mathbf{t}_0$ is not given directly in the actual key-recovery attack, the forgery attack is more practical than the key-recovery attack, and deserves more attention when the recommended parameters are chosen. Note the 131.2 vs. 125 quantum bit security between our signature scheme and Dilithium against forgery attack. In this sense, we may suggest that our signature scheme could obtain a better trade-offs between security and performance in practice, compared with Dilithium.

References

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. In Proceedings of the 25th USENIX Security Symposium. USENIX Association, 327-343.

AG11. Sanjeev Arora and Rong Ge. New Algorithms for Learning in Presence of Errors. In ICALP 2011, Part I (LNCS), Luca Aceto, Monika Henzinger, and Jiri Sgall (Eds.), Vol. 6755. Springer, Heidelberg, 403 - 415.

AM18. Divesh Aggarwal and Priyanka Mukhopadhyay. Improved algorithms for the Shortest Vector Problem and the Closest Vector Problem in the infinity norm. Available at http://arxiv.org/abs/1801.02358v2

BG14. Shi Bai and Steven D. Galbraith. An Improved Compression Technique for Signatures Based on Learning with Errors. CT-RSA 2014, LNCS Vol. 8366, pages 28-47. Springer, 2014.

BKW03. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4): 506-519 (2003).

BN06. Mihir Bellare and Gregory Neven Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. Proceedings of the 13th Association for Computing Machinery (ACM) Conference on Computer and Communications Security (CCS), Alexandria, Virginia, 2006, pp. 390-399.

CRYSTALS. Presentation of CRYSTALS (Kyber and Dilithium) at 2018 NIST PQC standardizaiton conference. https://csrc.nist.gov/CSRC/media/Presentations/Crystals-Dilithium

DLL+17. Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Dilithium: Digital Signatures from Module Lattices. *IACR Cryptology ePrint Archive*, 2017/633, 2017.

JZ16. Zhengzhong Jin and Yunlei Zhao. Optimal Key Consensus in Presence of Noise. In CoRR, Vol. abs/1611.06150, 2016. Available at http://arxiv.org/abs/1611.06150

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III, pages 552-586.

LDK+17. Vadim Lyubashevsky, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals-dilithium. Technical report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized Compact Knapsacks Are Collision Resistant. In ICALP 2006, Part II (LNCS), Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Vol. 4052. Springer, Heidelberg, 144-155.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In EUROCRYPT 2010 (LNCS), Henri Gilbert (Ed.), Vol. 6110. Springer, Heidelberg, 1-23.

LS15. Adeline Langlois and Damien Stehlé Worst-case to average-case reductions for module lattices. Des. Codes Cryptography 75, 3 (2015), 565-599.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. ASIACRYPT 2009: 598-616.

Lyu12. Vadim Lyubashevsky. Lattice Signatures without Trapdoors. In EUROCRYPT 2012 (LNCS), David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, Heidelberg, 738-755.

NV08. Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. Journal of Mathematical Cryptology, 2(2):181 – 207, 2008.

NIST. NIST. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization

PR06. Chris Peikert and Alon Rosen. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In TCC 2006 (LNCS), Shai Halevi and Tal Rabin (Eds.), Vol. 3876. Springer, Heidelberg, 145-166.

KCL. Yunlei Zhao, Zhengzhong Jin, Boru Gong, and Guangye Sui. KCL. Technical report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

A Remarks about the Parameter Estimation

First, as mentioned previously, we estimate the hardness of our recommended SIS instances by following the general methodology proposed by [DLL$^+$17,LDK$^+$17]. Since the associated estimation script in [DLL$^+$17, LDK$^+$17] was not made public, we develop a simple estimation script by following the general methodology in [DLL$^+$17, LDK$^+$17].

However, computational experiments show that for the four sets of suggested parameters proposed in [DLL$^+$17, LDK$^+$17], the hardness estimation output by our script is *always slightly smaller* than the claimed hardness estimation in [DLL$^+$17, LDK$^+$17]; see Table 3 below for the detailed comparison. We suggest the (slight) difference may be reasonable. The *possible* reason for such difference could be that the underlying subroutine, used by us to calculate the cumulative distribution function of the (continuous) Gaussian distribution, is *different* from that of Dilithium. This, in turn, has no elementary antiderivative, and numerical means must be used to evaluate the involving integrals.

Nevertheless, this simple comparison in Table 3 implies that our estimation script always works in a (slightly) *more conservative* manner than that of [DLL$^+$17,LDK$^+$17] does; in particular, for those SIS instances in the infinity norm appeared in this work, our estimation results on their hardness should be *slightly smaller* than the hardness estimation results output by the estimation script appearing in [DLL$^+$17, LDK$^+$17]. This makes us feel more confident about our choice of parameters.

	$(h,\ell)=(3,2)$	$(h,\ell)=(4,3)$	$(h,\ell)=(5,4)$	$(h,\ell)=(6,5)$
Claimed quantum security in Dilithium	62	94	125	160
Our estimated quantum security	61.1	92.2	124.5	158.2

Table 3. Comparison of the SIS-hardness of those four sets of suggested parameters in Dilithium. Specifically, for those four parameter sets we have $q = 8380417, d = 14, k = 16$.

B Concrete Security Analysis

For our signature schemes, two types of important attacks should be taken into consideration: the key-recovery attack which aims to recover the secret key, with the given associated public key; the forgery attack which tries to forge a signature in the security game of SEU-CMA (cf. Definition 1). To our knowledge, the best known algorithms to implement these two attack both involve the lattice basis reduction as well as the Core-SVP problem.

B.1 Lattice Basis Reduction, BKZ Algorithm, and Core-SVP Problem

Given our recommended parameter, in practice the best known algorithm for finding a "short" nonzero vector in *Euclidean* lattices is the BKZ algorithm as well as its variants, which outperforms the combinatorial attacks (*e.g.*, the BKW attack [BKW03]) and algebraic attacks (*e.g.*, the Arora-Ge algorithm [AG11]).

In order to solve the SVP problem in ℓ_2-norm, the BKZ algorithm solves a related yet more generic problem, *i.e.*, the lattice basis reduction problem. Generally speaking, BKZ solves the lattice basis reduction problem by making polynomial calls to a SVP oracle with block-size b. The hardness of lattice basis reduction problem is implied by the following two facts: to obtain a "good" basis, the BKZ algorithm should be equipped with a SVP oracle with a *large* block-size b; the cost of implementing the underlying SVP oracle is exponential in the block-size b (in fact, the best known quantum SVP solver [ADPS16] runs in time $\approx 2^{c_Q \cdot b}$, where $c_Q = \log_2 \sqrt{13/9} \approx 0.265$).

Since it is relatively difficult to analyze the upper-bound on the number of SVP calls, the Core-SVP model is thus introduced [ADPS16], which identifies the cost of BKZ algorithm with the cost of one single call to an SVP oracle with block-size b. This pessimistic estimation implies that similar to [DLL$^+$17,LDK$^+$17], our security analysis is pretty conservative.

B.2 Forgery Attack and Module-SIS Problem

As indicated in Section 5.2, the forgery attack could be boiled down to solving the Oracle-SelfTargetMSIS problem. To solve the Oracle-SelfTargetMSIS problem, we need either to break the security of $H(\cdot)$, or to solve the Oracle-MSIS problem with input $(\mathbf{A}, \mathbf{t}', \beta)$. The hardness of breaking the security of $H(\cdot)$ is guaranteed by the assumption that $H(\cdot)$ is modeled as a random oracle in our security proof and by the fact that the range B_{60} of $H(\cdot)$ is roughly of size 2^{256}. Hence, we concentrate our analysis on the hardness of the Oracle-MSIS problem hereafter.

In essence, the Oracle-MSIS problem could be seen as a variant of the SIS problem *in the ℓ_∞-norm*, which is in sharp contrast to the *ordinary* SIS problem endowed with the ℓ_2-norm. Roughly speaking, a "short" vector in the ℓ_∞-norm is also a "short" solution in the ℓ_2-norm, but the converse may not hold. (In fact, for our signature scheme with our recommended parameter, the solution to our Oracle-SelfTargetMSIS problem has Euclidean length above q, whereas the trivial vector $(q, 0, \cdots, 0)^t$ has Euclidean length q, but its infinity norm is far from satisfactory.) This indicates that the problem we are confronting is intuitively much harder than the SIS problem in the ℓ_2-norm.

Since BKZ algorithm works only on the Euclidean lattice, we cannot *directly* turn the Oracle-MSIS instance into a Core-SVP instance. Nevertheless, we shall follow the general methodology proposed in [DLL$^+$17, LDK$^+$17] by sticking to using the BKZ algorithm to determine the solution in the infinity norm.

To be specific, we first narrow down the range of our analysis by choosing a subset of w columns, and zeroing the other dimensions of the targeted vector. Then we still seek to choose a desired "short" vector by invoking the BKZ algorithm, but no longer the first one in the output lattice basis, *e.g.*, the shortest one in the Euclidean norm. Instead, we aim to find a lattice vector whose projection in either direction is not very large, which is

consistent with our purpose of finding a nonzero vector with small ℓ_∞-norm. When the block-size in BKZ is determined, heuristic assumptions are made so as to estimate the probability that such a desired lattice vector exists: we assume that for those dimensions that are not affected by the BKZ algorithm, the associated coordinates follow the uniform distribution modulo q, whereas for those dimensions that are affected by the BKZ algorithm, the associated coordinates follows the Gaussian distribution with appropriate standard deviation. The cost estimate is the inverse of that probability multiplied by the run-time of our b-dimensional SVP-solver [DLL$^+$17, LDK$^+$17].

B.3 The SVP Problem in the Infinity Norm

Recall that the BKZ algorithm solves the lattice basis reduction problem with the aid of a b-dimensional SVP solver in Euclidean norm, which enables BKZ to solve the SIS problem in the Euclidean norm. In Dilithium [DLL$^+$17, LDK$^+$17], this methodology was generalized to estimate the hardness of the SIS problem in the infinity norm. In particular, in one extreme case of the key-recovery problem in this work, where the adversary is given $(\mathbf{A}, \mathbf{t}_1)$ and is asked to find $(\mathbf{s}, \mathbf{e}-\mathbf{t}_0)$ such that $\mathbf{As}+(\mathbf{e}-\mathbf{t}_0)-\mathbf{t}_1 \cdot 2^d = \mathbf{0}$, it could be seen as an SIS instance with bound $\beta = \eta + 2^{d-1}$. However, we fail to apply the methodology proposed to estimate the quantum hardness of the foregoing instance: the returned estimation hardness result is 1471.8-bit quantum security, which is far from reasonable.

Instead, we turn to the state-of-the-art hardness result [AM18] on the SVP problem in the infinity norm. Roughly speaking, in [AM18] a new sieving procedure is proposed, which runs in time linear in the number N of randomly chosen samples in the underlying lattice; thus, among other results, an improved algorithm for solving the SVP problem in the infinity norm is obtained. This new algorithm runs in time at least $(4/3)^n$, where n denotes the dimension of the incoming lattice. Also, it is implied in [AM18] that this lower-bound on the space complexity of the SVP problem in the infinity norm is almost optimal, in the sense that some theoretical breakthrough is necessary in order to improve it further. Returning to our problem, the hardness of the SVP problem in the infinity norm is estimated to be about $2^{249.4}$ based on the result in [AM18].