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. Given the current research status in lattice-based cryptography, it is commonly suggested that lattice-based signature could be subtler and harder to achieve. Among them, Dilithium [DLL+17, LDK+17] is one of the most promising signature candidates for the post-quantum era, for its simplicity, efficiency, small public key size, and resistance against side channel attacks. The design of Dilithium is based on a list of pioneering works (e.g., [Lyu09, Lyu12, BG14]), and has very remarkable performance by very careful and comprehensive optimizations in implementation and parameter selection. Whether better trade-offs on the already remarkable performance of Dilithium can be made is left in [CRYSTALS] as an interesting open question.

In this work, we provide new insights in interpreting the design of Dilithium, in terms of key consensus previously proposed in the literature for key encapsulation mechanisms (KEM) and key exchange (KEX). Based on the deterministic version of the optimal key consensus with noise (OKCN) mechanism, originally developed in [JZ16] for KEM/KEX, we present *signature from key consensus with noise* (SKCN), which could be viewed as generalization and optimization of Dilithium. The construction of SKCN is generic, modular and flexible, which in particular allows a much broader range of parameters for searching better tradeoffs among security, computational efficiency, and bandwidth. For example, on the recommended parameters, compared with Dilithium our SKCN scheme is more efficient both in computation and in bandwidth, while preserving the same level of post-quantum security. In addition, using the same routine of OKCN for both KEM/KEX and digital signature eases (hardware) implementation and deployment in practice, and is useful to simplify the system complexity of lattice-based cryptography in general.

Keywords: Digital signature · Module-LWE · Module-SIS · post-quantum cryptography

1 Introduction

Over the last decades, lattice have emerged as a very attractive foundation for cryptography. Ever since the seminal work of Ajtai [Ajt96] connecting the average-case complexity of lattice problems to their complexity in the worst case, there has been an intriguing and fruitful efforts to base cryptographic schemes on worst-case lattice assumptions. In addition to their unique theoretical niche, lattice-based schemes enjoy many potential advantages: their asymptotic efficiency and conceptual simplicity (usually requiring only linear operations on small integers); their resistance so far to cryptanalysis by *quantum* algorithms; and the guarantee that their random instances are "as hard as possible" [Reg09, BLP+13].

Given the importance of digital signature scheme in modern cryptography, it is natural to consider building practical and provably secure digital signature schemes based on lattice assumptions. Generally speaking, lattice-based signature schemes are designed by following either of the following paradigms: hash-and-sign paradigm [DH76, BR93, GPV08], and Fiat-Shamir heuristic [FS87, Lyu09, DDLL13]. Nevertheless, 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 submissions of lattice-based key encapsulation mechanisms to NIST post-quantum cryptography (NIST-PQC), but only five lattice-based signature submissions [NIST]. Among them, Falcon [PFH+17], and pqNTRUSign [ZCHW17] follow the hash-and-sign paradigm; Dilithium [DLL+17] and qTESLA [BAA+17] follow the Fiat-Shamir heuristic. Now, Dilithium [DLL+17], qTESLA [BAA+17] and Falcon [PFH+17] are in the second round submissions of NIST-PQC.

In this work, we focus on the study of Dilithium [DLL+17, LDK+17]. Dilithium is one of the best lattice-based signature schemes that follow the Fiat-Shamir paradigm, and is one of the most promising lattice-based signature candidates. Some salient features of Dilithium include: simplicity (both for the algorithmic design and for the algebraic structure of the underlying lattice), 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 very careful and comprehensive optimizations in implementation and parameter selection. Whether better trade-offs on the already remarkable performance of Dilithium can be made is left in [CRYSTALS] as an interesting open question.

1.1 Our contributions

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 symmetric key consensus previously proposed in the literature for achieving key encapsulation mechanisms (KEM) and key exchange (KEX) [Reg09, LPR10, LP11, DXL12, Pei14, BCD+16, JZ16]. Based on the deterministic version of the optimal key consensus with noise (OKCN) mechanism, originally developed in [JZ16] for highly practical KEM/KEX schemes, we present *signature from key consensus with noise* (SKCN). The construction of SKCN is generic, modular and flexible, which in particular allows a much broader range of parameters.

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, compared with Dilithium our SKCN scheme is more efficient both in computation and in bandwidth, while preserving the same level of post-quantum security. This work also further justifies and highlights the desirability of OKCN, originally developed in [JZ16] for highly practical KEM/KEX, as the same routine can be used for both KEM/KEX and digital signature, which eases (hardware) implementation and deployment in practice, and is useful to simplify the system complexity of lattice-based cryptography in general.

2 Preliminaries

For any real number $x \in \mathbb{R}$, let $\lfloor x \rfloor$ denote the largest integer that is no more than x, and $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$. 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 a positive integer q and an element $x \in \mathbb{Z}_q$, we write $\|x\|_\infty$ for $|x \bmod^{\pm} q|$, and let $|x|_q$ denote the absolute value of $x \bmod^{\pm} q$. 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 $a_i' \stackrel{\text{def}}{=} \left(a_i - \left(a_i \bmod^{\pm} 2^d\right)\right)/2^d$.

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 $\mathcal{D}$, the notation $x \leftarrow \mathcal{D}$ denotes the operation of picking an element according to the pre-defined distribution $\mathcal{D}$. We say that a positive 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 $\lambda > \lambda_c$.

2.0.1 Digital signature scheme

A digital signature scheme Π consists of three probabilistic polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. KeyGen is the key generation algorithm that, on input the security parameter 1^λ, outputs $(\mathsf{pk}, \mathsf{sk})$. Sign is the signing algorithm that, on input the secret key sk as well as the message $\mu \in \{0,1\}^*$ to be signed, outputs the signature σ. Verify is the *deterministic* verification algorithm that, on input the public key pk as well as the message/signature pair (μ, σ), outputs $b \in \{0,1\}$, indicating whether it accepts the incoming (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

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

2.0.2 (S)EU-CMA

The security for a signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, is defined in the following security game between a challenger and an adversary A.

- Setup. Given λ, the challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The public key pk is given to adversary A, whereas the secret key sk is kept in private.

- Challenge. Suppose A makes at most q_s signature queries. Each signature query consists of the following steps: (1) A adaptively chooses the message $\mu_i \in \{0,1\}^*$, $1 \le i \le q_s$, 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 challenger 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 say the signature scheme Π is *strongly existentially unforgeable under adaptive chosen-message attack*, if the probability that every p.p.t. attacker A wins in the foregoing game is negligible. A weaker model, *i.e.*, the EU-CMA model, could be define by requiring that A wins if and only if (1) $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma) = 1$ and (2) $\mu \notin \{\mu_1, \mu_2, \cdots, \mu_{q_s}\}$. Then Π is called *(standard) existentially unforgeable under adaptive chosen-message attack*, if no efficient adversary can win in this weaker game with non-negligible probability.

2.0.3 Module-LWE and Module-SIS

In this work, we always have $n = 256$ and $q = 1952257$. 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 $\|w\|_\infty := \max \|w_i\|_\infty$. 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, when the other parameters are clear from the context, let $S_\eta \subseteq \mathcal{R}$ denote the set of elements $w \in \mathcal{R}$ such that $\|w\|_\infty \le \eta$.

The hard problems underlying the security of our signature scheme are Module-LWE (MLWE), Module-SIS (MSIS) (as well as a variant of MSIS problem). They were well studied in [LS15] and could be seen as a natural 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 taken uniformly at random, and $b_i := \mathbf{a}_i^T \mathbf{s} + e_i$ with $e_i \leftarrow S_\eta$ 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 efficient algorithm can succeed in recovering **s** with non-negligible probability, provided that the parameters are appropriately chosen.

Fix $p \in [1, \infty]$. Given $\mathbf{A} \leftarrow \mathcal{R}_q^{h \times \ell}$ where $h = \mathrm{poly}(\lambda)$, the Module-SIS problem (in ℓ_p-norm) parameterized by $\beta > 0$ asks to find a "'short" yet nonzero pre-image $\mathbf{x} \in \mathcal{R}_q^{\ell}$ in the lattice determined by $\mathbf{A}$, *i.e.*, $\mathbf{x} \neq \mathbf{0}, \mathbf{A} \cdot \mathbf{x} = \mathbf{0}$ and $\|\mathbf{x}\| \leq \beta$. And the associated Module-SIS assumption (in ℓ_p-norm) states that no probabilistic polynomial-time algorithm can find a feasible pre-image **x** with non-negligible probability, provided that the parameters are appropriately chosen. In the literature, the module-SIS problem in *Euclidean* norm, *i.e.*, $p = 2$, is well-studied; nevertheless, in this work, we are mostly interested in the Module-SIS problem/assumption *in ℓ_∞-norm*, *i.e.*, $p = \infty$.

2.0.4 Hashing

As is in [DLL+17, LDK+17], when the other related parameters are clear from the context, for every positive integer $w > 0$, let $B_w := \{x \in \mathcal{R} \mid \|x\|_\infty = 1, \|x\|_1 = w\} \subseteq \mathcal{R}$. In this work, we always have $w = 60$, since the set $B_{60} \subseteq \mathcal{R}$ is of size $2^{60} \cdot \binom{n}{60} \approx 2^{256}$ (recall that $n = 256$ by default in this work). 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.

2.0.5 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 to the pre-defined distribution S (or according to the uniform distribution over the pre-defined set S). The whole procedure is *deterministic* in the sense that for a given x will always output the same y, *i.e.*, the map $x \mapsto y$ is well-defined. For simplicity we always assume that the output distribution of Sam is perfect, whereas in practice it will be implemented by using some cryptographic hash functions (which are modelled as random oracle in this work) and produce an output that is *statistically close* to the perfect distribution.

3 Building Tools of SKCN

In this section, we first propose the notion of deterministic symmetric key consensus (DKC); then we construct and analyze a concrete DKC instance, *i.e.*, the deterministic symmetric key consensus with noise (DKCN), which is a variant of the optimal key consensus with noise (OKCN) scheme presented in [JZ16]. Based on DKCN, we then define several algorithms/tools, and develop some of their properties. These algorithms will serve as the building tools for our signature scheme to be introduced in Section 4.

Note that although all these algorithms/tools proposed in this section are defined with respect to the finite field $\mathbb{Z}_q$ for some positive rational prime q, they could be naturally generalized to vectors (as well as the ring $\mathcal{R}_q$) in the component-wise manner.

Definition 1. 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 $(\sigma_1 \in \mathbb{Z}_q, \mathsf{params})$, the *deterministic* polynomial-time *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 $(\sigma_2 \in \mathbb{Z}_q, v, \mathsf{params})$, the *deterministic* polynomial-time *reconciliation algorithm* Rec outputs $k_2 \in \mathbb{Z}_k$.

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

Next, we develop a concrete instance of DKC, *i.e.*, the rounded symmetric key consensus with noise (DKCN) depicted in Algorithm 1. Note that by Theorem 1, as a concrete DKC, DKCN itself is *correct*, provided that parameters are appropriately set.

Algorithm 1 DKCN: Deterministic Symmetric KC with Noise

1: $\mathsf{params} := (q, k, g, d, aux = \emptyset)$
2: **procedure** $\mathrm{CON}(\sigma_1, \mathsf{params})$ $\quad \triangleright\ \sigma_1 \in \mathbb{Z}_q$
3: $\quad v := k\sigma_1 \bmod^{\pm} q$
4: $\quad$ **if** $k\sigma_1 - v = kq$ **then**
5: $\quad\quad k_1 := 0$
6: $\quad$ **else**
7: $\quad\quad k_1 := (k\sigma_1 - v)/q$
8: $\quad$ **end if**
9: $\quad$ **return** (k_1, v)
10: **end procedure**
11: **procedure** $\mathrm{REC}(\sigma_2, v, \mathsf{params})$ $\quad \triangleright\ \sigma_2 \in \mathbb{Z}_q$
12: $\quad k_2 := \lfloor (k\sigma_2 - v)/q \rceil \bmod k$
13: $\quad$ **return** k_2
14: **end procedure**

Theorem 1. *When $k \geq 2, g \geq 2$ and $2kd < q$, the* DKCN *scheme* $(\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ *depicted in Algorithm 1 is* correct.

Before proving theorem 1, we introduce a lemma that was proposed in [JZ16].

Lemma 1 ([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 $|\sigma_1 - \sigma_2|_q \leq d$. Then by Lemma 1, 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

$$k_2 = \lfloor (k\sigma_2 - v)/q \rceil \bmod k = \lfloor k(\sigma_1 + \theta q + \delta)/q - v/q \rceil \bmod k == \lfloor k_1 + k\delta/q \rceil \bmod k.$$

It follows from $2kd < q$ that $|k\delta/q| \leq kd/q < 1/2$, making $k_2 = k_1$. □ □

Based on DKCN, we then present several algorithms and some of their properties.

procedure $\textsc{HighBits}_{q,k}(r)$
  $(r_1, r_0) \leftarrow \mathsf{Con}(r)$
  return r_1
end procedure
procedure $\textsc{MakeHint}_{q,k}(z, r)$
  $r_1 := \mathsf{HighBits}_{q,k}(r)$
  $v_1 := \mathsf{HighBits}_{q,k}(r + z)$
  if $r_1 = v_1$ **then**
    return 0
  else
    return 1
  end if
end procedure

procedure $\textsc{LowBits}_{q,k}(r)$
  $(r_1, r_0) \leftarrow \mathsf{Con}(r)$
  return r_0
end procedure
procedure $\textsc{UseHint}_{q,k}(h, r)$
  $(r_1, r_0) := \mathsf{Con}(r)$
  if $h = 0$ **then**
    return r_1
  else if $h = 1$ and $r_0 > 0$ **then**
    return $(r_1 + 1) \bmod k$
  else
    return $(r_1 - 1) \bmod k$
  end if
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 $0 \le r_1, r_1' < k$, and $\|r_0\|_\infty, \|r_0'\|_\infty \le q/2$. 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 $h \overset{\text{def}}{=} \mathsf{MakeHint}_{q,k}(z, r)$. 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 $\|r - \lfloor q \cdot r_1'/k \rceil\|_\infty \le q/k + 1/2$.*

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 $\|r - \lfloor q \cdot r_1'/k \rceil\|_\infty \le q/(2k) + 1/2$. If $h = 1$ and $r_0 > 0$, then $r_1' = (r_1 + 1) \bmod k$, and hence $\|r - \lfloor q \cdot r_1'/k \rceil\|_\infty \le q/k + 1/2$. Finally, if $h = 1$ and $r_0 < 0$, then $r_1' = (r_1 - 1) \bmod k$, and therefore $\|r - \lfloor q \cdot r_1'/k \rceil\|_\infty \le q/k + 1/2$. □ □

Proposition 3. *For $r, z \in \mathbb{Z}$ such that $\|z\|_\infty \le U$. 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 $k \cdot r = q \cdot (r_1 + k\theta) + r_0$ ($\|r_0\|_\infty < q/2$) and $k \cdot (r + z) = q \cdot (r_1' + k\theta') + r_0'$ ($\|r_0'\|_\infty < q/2$) 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 SKCN: Signature from Key Consensus with Noise

In this section, we propose our signature scheme SKCN, which is defined in the module lattice, and can be proven to be strongly existentially unforgeable under adaptive chosen-message attacks in the quantum random oracle model.

SKCN could be seen as a generalization and optimization of Dilithium with the aid of DKCN, and its correctness and security roughly follows from that of Dilithium as well.

4.1 Description of SKCN

Our key generation, signing, and verification algorithms are fully described in Algorithms 2, 4 and 3, respectively.

Algorithm 2 Key Generation Algorithm

Input: 1^λ

Output: $(\mathrm{pk} = (\rho, \mathbf{t}_1), \mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t}))$

$\rho, \rho' \leftarrow \{0,1\}^{256}$

$\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$

$(\mathbf{s}, \mathbf{e}) \in S_\eta^\ell \times S_{\eta'}^h := \mathsf{Sam}(\rho')$

$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$

$\mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})$

return $(\mathrm{pk} = (\rho, \mathbf{t}_1), \mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t}))$

Algorithm 3 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

Algorithm 4 The Signing Algorithm

Input: $\mu \in \{0,1\}^*$, $\mathrm{sk} = (\rho, \mathbf{s}, \mathbf{e}, \mathbf{t})$

Output: $\sigma = (\mathbf{z}, c, \mathbf{h})$

$\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$

$\mathbf{t}_1 := \mathsf{Power2Round}_{q,d}(\mathbf{t})$

$\mathbf{t}_0 := \mathbf{t} - \mathbf{t}_1 \cdot 2^d$

$r \leftarrow \{0,1\}^{256}$

$\mathbf{y} \in S_{\lfloor q/k \rfloor - 1}^\ell := \mathsf{Sam}(r)$

$\mathbf{w} := \mathbf{A}\mathbf{y}$

$\mathbf{w}_1 := \mathsf{HighBits}_{q,k}(\mathbf{w})$

$c \leftarrow H(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{w}_1/k \rceil, \mu)$

$\mathbf{z} := \mathbf{y} + c\mathbf{s}$

$(\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Con}(\mathbf{w} - c\mathbf{e})$

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 5 The Simulator

Input: $\mu \in \{0,1\}^*$, $\rho, \mathbf{t}_1, \mathbf{t}_0$

Output: $\sigma = (\mathbf{z}, c, \mathbf{h})$

$\mathbf{A} \in \mathcal{R}_q^{h \times \ell} := \mathsf{Sam}(\rho)$

$(\mathbf{z}, c) \leftarrow S_{\lfloor q,k \rfloor - U} \times B_{60}$

$(\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Con}(\mathbf{A}\mathbf{z} - c\mathbf{t})$

Restart if $\|\mathbf{r}_0\|_\infty \geq q/2 - kU$

if H has already been defined on $(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{r}_1/k \rceil, \mu)$ **then**

  Abort

else

  Program $H(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{r}_1/k \rceil, \mu) = c$

end if

$\mathbf{h} := \mathsf{MakeHint}_{q,k}(-c\mathbf{t}_0, \mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0)$

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 $(\mathbf{z}, c, \mathbf{h})$

4.1.1 Practical Implementation

When we implement SKCN with our recommended parameter set (cf. Table 1), several improvements that are similar to [DLL+17, LDK+17] are made, so as to improve its efficiency. Specifically, the sign algorithm in our implementation is *deterministic* in nature which is similar to that of Dilithium [LDK+17]. This is achieved by adding some new seeds $(\mathsf{tr}, \mathsf{key})$ into the secret key sk; thus, the random nonce $\mathbf{y}$ in the sign algorithm could be obtained via a pseudorandom string, which is obtained by extending the hash value of $(\mathsf{tr}, \mathsf{key})$, the message to be signed, and a counter. As a result, the $\mathbf{t}_1$ in sk is no longer necessary, making $\mathsf{sk} = (\rho, \mathsf{tr}, \mathsf{key}, \mathbf{s}, \mathbf{e}, \mathbf{t}_0)$. This minor modification can improve the efficiency of the sign algorithm significantly, and shorten the size of sk. Our implementation code is available at https://github.com/mulansig.

4.2 Correctness Analysis

In SKCN, 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 $\mathsf{Sam}(\cdot)$. 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 of $\mathbf{s}$ (resp., $\mathbf{e}$) is drawn uniformly at random from the set $[-\eta, \eta]$ (resp., $[-\eta', \eta']$). Finally, we compute $\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathcal{R}_q^h$. The public key is $\mathrm{pk} = (\rho, \mathbf{t}_1)$ where $\mathbf{t}_1 = \mathsf{Power2Round}_{q,d}(\mathbf{t})$, and 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 $\mathbf{w} := \mathbf{A}\mathbf{y}$. Upon input $(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{w}_1/k\rceil, \mu)$, the random oracle $H(\cdot)$ returns a uniform $c \leftarrow B_{60}$. After obtaining c, the signing algorithm conducts a rejection sampling process 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 requirements are satisfied. We should point out that if $\|c\mathbf{e}\|_\infty \le U'$, then by Proposition 3, the requirement $\|\mathbf{r}_0\|_\infty < q/2 - kU'$ forces $\mathbf{r}_1 = \mathbf{w}_1$. We hope $\|c\mathbf{e}\|_\infty > U'$ occurs with negligible probability, such that the probability that the check $\mathbf{r}_1 = \mathbf{w}_1$ fails is negligible as well. In addition, U is chosen such that $\|c\mathbf{s}\|_\infty \le U$ holds with overwhelming probability. 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 remaining two checks, *i.e.*, if $\|c\mathbf{t}_0\|_\infty < \lfloor q/2k \rfloor$ 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 $\mathbf{A} \in \mathcal{R}_q^{h\times\ell}$ via the random seed ρ. After that, it computes $\mathbf{w}_1' := \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d)$. If the given $(\mathbf{z}, c, \mathbf{h})$ is indeed a *honestly generated* signature of the incoming 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 $\mathsf{HighBits}_{q,k}(\mathbf{A}\mathbf{y}) = \mathsf{HighBits}_{q,k}(\mathbf{A}\mathbf{y} - c\mathbf{e}) = \mathbf{w}_1'$ 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 SKCN signature scheme is always *correct*, *provided that the involving parameters are appropriately set*. Roughly speaking, the correctness relies heavily on Proposition 1. When the public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ is fixed, for a *valid* message/signature pair $(\mu, (\mathbf{z}, c, \mathbf{h}))$, it suffices to show that $c = c'$. Since $\|c\mathbf{t}_0\|_\infty < q/2k$ and $\mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d = \mathbf{A}\mathbf{y} - c\mathbf{e} + c\mathbf{t}_0$, it follows directly from Proposition 1 that

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

Given that the signing algorithm forces $\mathsf{HighBits}_{q,k}(\mathbf{A}\mathbf{y} - c\mathbf{e}) = \mathsf{HighBits}_{q,k}(\mathbf{A}\mathbf{y})$ by rejection sampling, it follows from the following equality that $c = c'$:

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

4.3 Recommended Parameters, and Comparison

To improve the time/space efficiency, our SKCN signature scheme could be set *asymmetrically*, in the sense that as long as the resulting scheme can resist the key-recovery attack, η may not equal to η'. Moreover, the parameter U and U' are carefully chosen such that $\Pr[\|c\mathbf{s}\|_\infty \ge U]$ and $\Pr[\|c\mathbf{e}\|_\infty \ge U']$ are sufficiently small, say they are both small than 2^{-128}. By default we have $\eta = \eta'$ (and hence $U = U'$).

Table 2: Comparison between SKCN/Dilithium.

Parameter	Value
q	1952257
n	256
(h, ℓ)	$(5, 4)$
d	13
k	8
U	118
U'	118
ω	96

	SKCN	Dilithium
q	1952257	8380417
n	256	256
(h, ℓ)	$(5, 4)$	$(5, 4)$
(η, η')	$(2, 2)$	$(5, 5)$
pk size (in byte)	1312	1472
sk size (in byte)	3056	3504
sig. size (in byte)	2573	2701
expected # of repetitions	5.67	6.6
quantum bit-cost against key recovery attack	128	128
quantum bit-cost against forgery attack	125	125

The efficiency of the signing algorithm is firmly connected to the expected number of repetitions, which depends on the probabilities that the two rejection sampling steps occur. When some assumption is made with respect to the distribution of $\mathbf{w} = \mathbf{A}\mathbf{y} \in \mathcal{R}_q^h$, the probability that the first restart occurs is

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

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%. In sum, the average number of repetitions is dominated by the probability that the first restart occurs.

To choose the set of recommended parameters for SKCN, 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.

Under such considerations, we choose the set of recommended parameters for SKCN which is depicted in Table 1. And the quantitative comparison between recommended-SKCN and recommended-Dilithium is summarized in Table 2. Note that for the security issue, the (quantum) security of our recommended parameter set is estimated by following exactly the methodology proposed in [DLL+17, LDK+17].

The strength of SKCN is best described by the foregoing quantitative measures. Roughly speaking, compared with Dilithium, SKCN is *more efficient*: while preserving the same (quantum) security level as Dilithium does, SKCN has shorter public/secret key, has shorter signature, and runs fasters.

4.4 Implementation Details

In this subsection, we describe how our SKCN equipped with the set of recommended parameters (cf. Table 1) is implemented in practice. Our implementation can be best described by Algorithms 6 - 8 presented in Appendix A. Roughly speaking, compared with our theoretical design depicted in Algorithms 2 - 4, several changes similar to [LDK+17] are made so as to improve the efficiency of SKCN.

4.4.1 Deterministic signing procedure

First and foremost, the sign procedure in SKCN is made determinist by adding a seed to the secret key and using this seed key as well as the message μ to be signed to produce the nonce $\mathbf{y} \in \mathcal{R}_q^4$ via SHAKE-256. Given that repetition is almost necessary to generate a valid signature, we append a counter κ into the secret key, which makes the SHAKE-256 output differ with each signing attempt of the same message.

4.4.2 NTT operation

As in most lattice-based schemes that are based on operations over algebraic lattice, the underlying ring in SKCN is well chosen so that the multiplication operation could be carried out efficiently via the Number Theoretic Transform (NTT) over the finite field $\mathbb{Z}_q$. To enable the NTT, we need to choose a prime q so that the cyclic group $\mathbb{Z}_q^*$ has an element $\alpha \in \mathbb{Z}_q^*$ of order $2n = 512$, or equivalently, $q \equiv 1 \pmod{512}$. In our recommended parameter,

$$x^n + 1 = x^{256} + 1 \equiv (x - \alpha) \cdot (x - \alpha^3) \cdots (x - \alpha^{511}) \pmod{q},$$

and each polynomial $f \in \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ could be represented in its NTT form as

$$\hat{f} \stackrel{\text{def}}{=} (f(\alpha), f(\alpha^3), \cdots, f(\alpha^{511})).$$

The homomorphic property of this conversion enables us to carry out the polynomial multiplication efficiently.

4.4.3 Bit packing

Bit-packing is applied in the implementation of SKCN.

Take the binary representation of $\mathbf{t}_0 \in \mathcal{R}_q^5$ as an example. Recall that $\mathbf{t}_1$ contains 5 polynomials in $\mathbb{Z}_q[x]/\langle x^{256} + 1 \rangle$, and every coefficient belongs to $\left\{-2^{12} + 1, \cdots, 2^{12}\right\}$. Therefore, each polynomial coefficient could be represented by 13 bits via a simple translation, and $\mathbf{t}_0$ could be represented by $5 \cdot 256 \cdot 13$ bits; in our implementation, they are bit-packed in little-endian byte-order. Similar techniques apply to the polynomial vectors $\mathbf{s}, \mathbf{e}, \mathbf{t}_1, \mathbf{z}$.

Also, the public key $(\rho, \mathbf{t}_1)$ of SKCN is stored as the concatenation of the bit-packed representations of ρ and $\mathbf{t}_1$ in consecutive order. Similar considerations apply to the secret key and the signature.

4.4.4 Collision resistant hash

Similar to [LDK+17], the function CRH in Appendix A is a collision resistant hash function. In our implementation, it is instantiated by using the first 384 bits of the SHAKE-256 output.

4.4.5 Generation of $\mathbf{A}, \mathbf{s}, \mathbf{e}$

To keep the public key as small as possible, the matrix $\mathbf{A}$ is shared in terms of the seed ρ: in each procedure, the NTT representation of the matrix $\mathbf{A}$, *i.e.*, $\hat{\mathbf{A}}$, is extracted from the seed ρ by squeezing SHAKE-256 to obtain a stream of random bytes of arbitrary length. Similarly, part of the secret key $(\mathbf{s}, \mathbf{e})$ is obtained from a random seed in the key generation procedure, via the use of SHAKE-256.

Table 3: The concrete harware/software details for the implementation comparison.

Hardware / Software	Details
Operating System	Ubuntu 18.04.3 LTS system
Computer	Lenovo ThinkPad T480S
CPU	Intel(R) Core(TM) i7-8550U
Memory	16G
Implementation of RO	SHA-3
Compiler	GCC
Hyperthreading option	On

Table 4: Comparison between SKCN and Dilithium.

	SKCN	Dilithium
q	1952257	8380417
n	256	256
(h, ℓ)	$(5, 4)$	$(5, 4)$
(η, η')	$(2, 2)$	$(5, 5)$
pk size (in byte)	1312	1472
sk size (in byte)	3056	3504
sig. size (in byte)	2573	2701
average # of repetitions	5.7	6.6
KeyGen cycles	177707	198167
Sign cycles	859774	1056305
Verification cycles	191645	201511

4.4.6 Performance comparison

We test both the implementations of SKCN and that of Dilithium under the same software/hardware environment depicted in Table 3. And the quantitative comparison between recommended-SKCN and recommended-Dilithium is summarized in Table 4.

5 Security Analysis of SKCN

In this section, we analyze the security of the SKCN signature scheme. Roughly speaking, the security proof consists of two phases: In Phase I, the behavior of the signing oracle is proven to be statistically indistinguishable from that of an efficient simulator; In Phase II, we show that when the underlying hardness assumption holds, no efficient attacker can forge a valid message/signature pair with non-negligible probability, after interacting with the foregoing simulator polynomially many times.

In the following security proof, we will assume that the public key of SKCN is $(\rho, \mathbf{t}_1, \mathbf{t}_0)$ instead of $(\rho, \mathbf{t}_1)$, which is similar to that of Dilithium [DLL+17, LDK+17].

5.1 Security Proof in Phase I: the Simulator

The simulation of the signature follows from that of [DLL+17]. The associated simulator for SKCN is depicted in Algorithm 5. It should be stressed that we assume the public key is $\mathbf{t}$ instead of $\mathbf{t}_1$ as well. It suffices to show that the output of the signing oracle is indistinguishable from that of the simulator. The following two facts plays an essential role for the indistinguishability proof. First, in the real signing algorithm, we have $\Pr[\mathbf{z}, c] = \Pr[c] \cdot \Pr[\mathbf{y} = \mathbf{z} - c\mathbf{s} \mid c]$. Since $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$ and $\|c\mathbf{s}\|_\infty \le U$ (with overwhelming probability), we know that $\|\mathbf{z} - c\mathbf{s}\|_\infty < \lfloor q/k \rfloor$, then $\Pr[\mathbf{z}, c]$ is exactly the same for every such tuple $(\mathbf{z}, c)$. Second, when $\mathbf{z}$ does satisfy $\|\mathsf{LowBits}_{q,k}(\mathbf{w} - c\mathbf{e})\|_\infty < q/2 - kU'$, then as long as $\|c\mathbf{e}\|_\infty < U'$, we have

$$\mathbf{r}_1 = \mathrm{HighBits}_{q,k}(\mathbf{w} - c\mathbf{e}) = \mathrm{HighBits}_{q,k}(\mathbf{w}) = \mathbf{w}_1$$

by Proposition 3. Thus the simulator does not need to perform the check whether $\mathbf{r}_1 = \mathbf{w}_1$ or not, and can always assume that it passes.

With the foregoing facts, it is routine to see that the distribution of the pair $(\mathbf{z}, c)$ generated by the simulator is statistically indistinguishable from that of the pair $(\mathbf{z}, c)$ generated by the signing oracle.

After that, the simulator computes $\mathbf{r}_1$ and programs $H(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{r}_1/k \rceil, \mu) = c$. The resulting $(\mathbf{z}, c)$ output by the simulator is indistinguishable from that of the real signing oracle in the security game, *provided that collision occurs with negligible probability*.

It remains to show that for each query μ, the probability that $H(\rho, \mathbf{t}_1, \lfloor q \cdot \mathbf{r}_1/k \rceil, \mu)$ was programmed previously is negligible. This follows directly from the following lemma, whose proof is similar to that of [KLS18].

Lemma 2. *For every $\boldsymbol{A} \leftarrow \mathcal{R}_q^{h\times\ell}$, we have*

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

Proof. Since the polynomial $x^n + 1$ splits into n linear factors modulo q, the probability that for a uniform $a \leftarrow \mathcal{R}_q$, the probability that a is invertible in $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ is $(1 - 1/q)^n > 1 - n/q$. Thus the probability that at least one of $h\ell$ polynomials in $\mathbf{A} \leftarrow \mathcal{R}_q^{h\times\ell}$ is invertible is greater than $1 - (n/q)^{h\ell}$.

We shall now prove that for all $\mathbf{A}$ that contain at least one invertible polynomial, we will have that for all $\mathbf{w}_1^*$,

$$\Pr_{\boldsymbol{y}\leftarrow S^\ell_{\lfloor q/k \rfloor -1}}\left[\mathsf{HighBits}_{q,k}(\mathbf{A}\mathbf{y}) = \mathbf{w}_1^*\right] \le \left(\frac{q/k+1}{2\cdot\lfloor q/k \rfloor - 1}\right)^n,$$

which establishes the correctness of this lemma.

First, let us only consider the row of $\mathbf{A}$ which contains the invertible polynomial. Call the elements in this row $[a_1, \cdots, a_\ell]$ and without loss of generality assume that a_1 is invertible. We want to prove that for all w_1^*,

$$\Pr_{\boldsymbol{y}\leftarrow S^\ell_{\lfloor q/k \rfloor -1}}\left[\mathsf{HighBits}_{q,k}(\sum a_i y_i) = w_1^*\right] \le \left(\frac{q/k+1}{2\cdot\lfloor q/k \rfloor - 1}\right)^n.$$

Define T to be the set containing all the elements w such that $\mathsf{HighBits}_{q,k}(w) = w_1^*$. By the definition of Con in Algorithm 1, the size of T is upper-bounded by $(q/k + 1)^n$. Therefore, we can rewrite the above probability as

$$\Pr_{\boldsymbol{y}\leftarrow S^\ell_{\lfloor q/k \rfloor -1}}\left[\sum_{i=1}^{\ell} a_i y_i \in T\right] = \Pr_{\boldsymbol{y}\leftarrow S^\ell_{\lfloor q/k \rfloor -1}}\left[y_1 \in a_1^{-1}(T - \sum_{i=2}^{\ell} a_i y_i)\right] \le \left(\frac{q/k+1}{2\cdot\lfloor q/k \rfloor - 1}\right)^n,$$

where the last inequality follows due to the fact that the size of the set $a_1^{-1}\left(T - \sum_{i=2}^{\ell} a_i y_i\right)$ is the same as that of T, and the size of the set $S^\ell_{\lfloor q/k \rfloor -1}$ is exactly $(2\cdot\lfloor q/k \rfloor - 1)^n$. □ □

It should be stressed that, both inequalities $\left(\frac{q/k+1}{2\lfloor q/k \rfloor - 1}\right)^n \ll 2^{-128}$ and $(n/q)^{h\ell} \ll 2^{-128}$ holds for our set of recommended parameters depicted in Table 1.

5.2 Security Proof in Phase II

By applying forking lemma, we can show SKCN is strongly existentially unforgeable under adaptive chosen-message attacks in the random oracle model, provided that the parameters are appropriately chosen and the underlying MLWE and MSIS (in ℓ_∞ norm) assumptions hold. However, this proof is not tight, and cannot be directly applied into the quantum setting. In contrast, in this section, we shall develop a quantum reduction of SKCN that is tight in nature by introducing another new underlying hardness assumption for SKCN, as is done in [DLL+17, LDK+17].

As is observed in [DLL+17, LDK+17], no counter-examples of schemes whose security is actually affected by the non-tightness of the reduction has been proposed. The main reason for this absence of counter-examples lies in that there is an intermediate problem which is tightly equivalent, to the UF-CMA security of the signature scheme. What is more, this equivalence still holds even under in quantum settings. Compared with classical hardness problems, this problem is essentially a *convolution* of the underlying mathematical problem with a cryptographic hash function $H(\cdot)$. As is justified in [LDK+17], as long as there is no relationship between the structure of the math problem and the hash function $H(\cdot)$, solving this intermediate problem is not easier than solving the mathematical problem alone. In our setting, this intermediate problem is called the SelfTargetMSIS problem, which is to be defined later.

5.3 The SelfTargetMSIS Problem, and Quantum Security of SKCN

We follow the definition in [LDK+17]. Assume $H : \{0,1\}^* \to B_{60}$ is a cryptographic hash function. For a given adversary A, it is given a random $\mathbf{A} \leftarrow \mathcal{R}_q^{h\times\ell}$ and access to the quantum random oracle $H(\cdot)$, and is asked to output a pair $\left(\mathbf{y} = ([\mathbf{r}, c]^T, \mu)\right)$ such that $0 \le \|\mathbf{y}\|_\infty \le \gamma$, $H(\mu, [\mathbf{I}, \mathbf{A}] \cdot \mathbf{y}) = c$. In other words, the adversary A is asked to solve the *SelfTargetMSIS* problem. In this work, let $\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{H,h,\ell,\gamma}(A)$ denote the probability that A solves the given SelfTargetMSIS problem successfully.

Similar to results in [LDK+17], given the similarity between SKCN and Dilithium, it follows from [KLS18] that when $H(\cdot)$ is modeled as quantum random oracle, the probability a given efficient adversary A breaks the SEU-CMA security of SKCN is

$$\mathsf{Adv}^{\mathsf{SUF-CMA}}_{\mathsf{SKCN}}(A) \quad \le \quad \mathsf{Adv}^{\mathsf{MLWE}}_{h,\ell,D}(B) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{H,h,\ell+1,\zeta}(C) + \mathsf{Adv}^{\mathsf{MSIS}}_{h,\ell,\zeta'}(D) + 2^{-254},$$

where D denotes the uniform distribution over S_η, and

$$\zeta \quad = \quad \max(\lfloor q/k \rfloor - U, \lfloor q/k \rfloor + 1 + 60 \cdot 2^{d-1}), \zeta' = \max(2 \cdot (\lfloor q/k \rfloor - U), 2\lfloor q/k \rfloor + 2).$$

Similar to Dilithium, SKCN is built upon three underlying hardness assumptions: intuitively, the MLWE assumption is needed to protect against key-recovery attack, the SelfTargetMSIS is the assumption upon which new message forgery is based, and the MSIS assumption is needed for *strong* unforgeability instead of standard unforgeability.

Note that the simulation proof in Section 5.1 holds even in quantum setting; equivalently, if an adversary having quantum access to $H(\cdot)$ and classical access to a signing oracle can produce a forgery of a new message, then there is also an adversary who can produce a forgery after interacting with the simulator defined in Section 5.1. When MLWE assumption holds with the distribution D, it remains for us to analyze the following experiment: for an efficient adversary A, it is given a random $(\mathbf{A}, \mathbf{t})$, and is asked to output a valid message/signature pair $(\mu, (\mathbf{z}, c, \mathbf{h}))$ such that $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U, H(\mu, \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d)) = c, \|\mathbf{h}\|_1 \le \omega$.

It follows from the properties presented in Section 3 that

$$\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d + \mathbf{u} = \lfloor q/k \rfloor \cdot \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d) = \mathbf{Az} - c\mathbf{t} + \mathbf{u}',$$

where $\|\mathbf{u}'\|_\infty \le \|\mathbf{u}\|_\infty + \|c \cdot \mathbf{t}_0\|_\infty \le 60 \cdot 2^{d-1} + \lfloor q/k \rfloor$. In sum, to forge a valid message/signature pair means to find $\mathbf{z}, c, \mathbf{u}', \mu$ such that

- $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$; and
- $\|c\|_\infty = 1$; and
- $\|\mathbf{u}'\|_\infty < 2\lfloor q/k \rfloor$; and
- $c = H(\mu, \mathbf{Az} + c\mathbf{t} + \mathbf{u}' \cdot \frac{1}{\lfloor q/k \rfloor})$.

This is the SelfTargetMSIS problem defined previously.

As is analyzed in [LDK+17], the only way to solving the SelfTargetMSIS problem appears to be picking some $\mathbf{w} \in \mathcal{R}_q^h$, computing $H(\mu, \mathbf{w}) = c$, and then finding the feasible $\mathbf{z}, \mathbf{u}'$ such that $\mathbf{Az} + \mathbf{u}' = \mathbf{w} + c\mathbf{t}$. And forging a valid forgery in the UF-CMA security of SKCN is finding some $\mathbf{z}, \mathbf{u}'$ such that $\|\mathbf{z}\|_\infty \le \lfloor q/k \rfloor - U$, $\|\mathbf{u}'\|_\infty \le 60 \cdot 2^{d-1} + \lfloor q/k \rfloor$, and $\mathbf{Az} + \mathbf{u}' = \mathbf{t}'$ for some pre-defined $\mathbf{t}'$.

Finally comes to the strong unforgeability of SKCN. In quantum setting, the analysis is similar to that of [DLL+17, LDK+17]. In addition to the foregoing possible forgery, an extra one needs considering for the strong unforgeability, *i.e.*, when the adversary sees a valid message/signature pair $(\mu, (\mathbf{z}, c, \mathbf{h}))$ and then aims to forge another valid pair $(\mu, (\mathbf{z}', c, \mathbf{h}'))$. In this special case, a successful forgery means the adversary obtains two valid signatures such that

$$\mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d) = \mathbf{w}_1 = \mathsf{UseHint}_{q,k}(\mathbf{h}', \mathbf{Az}' - c\mathbf{t}_1 \cdot 2^d).$$

It is routine to see $\mathbf{z} \neq \mathbf{z}'$. Therefore,

$$\left\|\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d - \mathbf{w}_1 \cdot \lfloor q/k \rfloor\right\|_\infty, \left\|\mathbf{Az}' - c\mathbf{t}_1 \cdot 2^d - \mathbf{w}_1 \cdot \lfloor q/k \rfloor\right\|_\infty \le \lfloor q/k \rfloor + 1.$$

It follows from the triangular inequality that $\|\mathbf{A} \cdot \Delta\mathbf{z}\|_\infty \le 2\lfloor q/k \rfloor + 2$, where $\Delta\mathbf{z} \stackrel{\text{def}}{=} (\mathbf{z} - \mathbf{z}')$ satisfies $\|\Delta\mathbf{z}\|_\infty \le 2(\lfloor q/k \rfloor - U)$. The hardness of this reduced problem is thus guaranteed by the hardness of the Module-SIS problem (in the ℓ_∞-norm).

6 Concrete Security Analysis

In this section, we analyze how to conduct the concrete security analysis of SKCN. As mentioned previously, we estimate the concrete hardness of SKCN by following the same methodology proposed in [LDK+17], and the result shows our recommended-SKCN achieves the same (quantum) security level as that of recommended-Dilithium.

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. 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.

6.1 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.

6.2 Forgery Attack and Module-SIS Problem

As indicated in Section 5.2, the forgery attack could be boiled down to solving the $\mathsf{SelfTargetMSIS}$ problem. To solve the $\mathsf{SelfTargetMSIS}$ problem, we need either to break the security of $H(\cdot)$, or to solve the 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 MSIS problem hereafter.

In essence, the 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 $\mathsf{SelfTargetMSIS}$ 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 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. Finally, the cost estimate is the inverse of that probability multiplied by the run-time of our b-dimensional SVP-solver [DLL+17, LDK+17].

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.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. Quaderni di Matematica, 13:1-32, 2004. Preliminary version in STOC 1996.

[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.

[BCD+16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. *ACM CCS 2016*: 1006-1018.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, Gauss, and Reload - A Cache Attack on the BLISS Lattice-Based Signature Scheme. In CHES 2016 (LNCS), Benedikt Gierlichs and Axel Y. Poschmann (Eds.), Vol. 9813. Springer, Heidelberg, 323ąV345.

[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).

[BAA+17] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In STOC 2013, pages 575-584.

[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.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In 1st ACM Conference on Computer and Communications Security, pages 62-73. ACM Press, 1993.

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

[DXL12] J. Ding, X. Xie and X. Lin. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. *Cryptology ePrint Archive*, Report 2012/688, 2012.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice Signatures and Bimodal Gaussians. In CRYPTO 2013, Part I (LNCS), Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, Heidelberg, 40-56.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, 22(6):644-654, 1976.

[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.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, Advances in Cryptology CRYPTO'86, pages 186-194, Berlin, Heidelberg, 1987. Springer.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In STOC, pages 197ÍC206. 2008.

[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 Extended abstract in ACNS 2019 (Best Student Paper), pages 302-322, springer 2019.

[KCL17] 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

[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.

[LP11] R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. *CT-RSA 2011*: 319-339.

[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

[Pei14] C. Peikert. Lattice Cryptography for the Internet. *PQCrypto 2014*: 197-219.

[PFH+17] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang Falcon. Technical report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

[Pes16] Peter Pessl. Analyzing the Shuffling Side-Channel Countermeasure for Lattice-Based Signatures. In INDOCRYPT 2016 (LNCS), Orr Dunkelman and Somitra Kumar Sanadhya (Eds.), Vol. 10095.

[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.

[Reg09] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, Volume 56, Issue 6, pages 34, 2009.

[ZCHW17] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte. pqNTRUSign Technical report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

A The Implementation Procedures

Algorithm 6 The key generation algorithm in our implementation

Input: 1^λ
Output: $(\mathsf{pk}, \mathsf{sk})$
$\rho \leftarrow \{0,1\}^{256}$
$\mathsf{key} \leftarrow \{0,1\}^{256}$
$(\mathbf{s}, \mathbf{e}) \leftarrow S_\eta^\ell \times S_{\eta'}^h$
$\mathbf{A} \in \mathcal{R}_q^{h\times\ell} := \mathsf{Sam}(\rho)$
$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$
$(\mathbf{t}_1, \mathbf{t}_0) := \mathsf{Power2Round}_{q,d}(\mathbf{t})$
$\mathsf{tr} \in \{0,1\}^{384} := \mathsf{CRH}(\rho, \mathbf{t}_1)$
$\mathsf{pk} := (\rho, \mathbf{t}_1)$
$\mathsf{sk} := (\rho, \mathsf{key}, \mathsf{tr}, \mathbf{s}, \mathbf{e}, \mathbf{t}_0)$
return $(\mathsf{pk}, \mathsf{sk})$

Algorithm 7 The Verification algorithm in our implementation

Input: $\mathsf{pk}, \mu, (\mathbf{z}, c, \mathbf{h})$
Output: $b \in \{0,1\}$
$\mathbf{A} \in \mathcal{R}_q^{h\times\ell} := \mathsf{Sam}(\rho)$
$\mathsf{tr} \in \{0,1\}^{384} := \mathsf{CRH}(\rho, \mathbf{t}_1)$
$\mu \in \{0,1\}^{384} := \mathsf{CRH}(\mathsf{tr}, \mu)$
$\mathbf{w}'_1 := \mathsf{UseHint}_{q,k}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d)$
$c' \leftarrow H(\rho, \mathbf{t}_1, \mathbf{w}'_1, \mu)$
if $c = c'$ **AND** $\|\mathbf{z}\|_\infty < \lfloor q/k \rfloor - U$ **AND** $\|\mathbf{h}\|_1 \geq \omega$ **then**
    $b := 1$
else
    $b := 0$
end if
return b

Algorithm 8 The Sign algorithm in our implementation

Input: sk, μ
Output: $(\mathbf{z}, c, \mathbf{h})$
$\mathbf{A} \in \mathcal{R}_q^{h\times\ell} := \mathsf{Sam}(\rho)$
$\mu \in \{0,1\}^{384} := \mathsf{CRH}(\mathsf{tr}, \mu)$
$\kappa := 0$
while true do
    $\mathbf{y} \in S_{\lfloor q/k \rfloor - 1}^\ell := \mathsf{ExpandMask}(\mathsf{key}, \mu, \kappa)$
    $\kappa := \kappa + 1$
    $\mathbf{w} := \mathbf{A}\mathbf{y}$
    $\mathbf{w}_1 := \mathsf{HighBits}_{q,k}(\mathbf{w})$
    $c \in B_{60} := H(\mu, \mathbf{w}_1)$
    $\mathbf{z} := \mathbf{y} + c\mathbf{s}$
    $\mathbf{u} := \mathbf{w} - c\mathbf{e}$
    $(\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Con}(\mathbf{u})$
    if $\|\mathbf{z}\|_\infty \geq \lfloor q/k \rfloor - U$ **OR** $\|\mathbf{r}_0\|_\infty \geq \lfloor q/2 \rfloor - k \cdot U'$ **OR** $\mathbf{r}_1 \neq \mathbf{w}_1$ **then**
        continue
    end if
    $\mathbf{v} := c\mathbf{t}_0$
    if $\|\mathbf{v}\|_\infty \geq \lfloor q/(2k) \rfloor$ **then**
        continue
    end if
    $\mathbf{h} := \mathsf{MakeHint}_{q,k}(-\mathbf{v}, \mathbf{u} + \mathbf{v})$
    if $\|\mathbf{h}\|_1 \geq \omega$ **then**
        continue
    end if
    break
end while
return $(\mathbf{z}, c, \mathbf{h})$