Revisiting the Concrete Hardness of SelfTargetMSIS in CRYSTALS-Dilithium

Geng Wang[1], Wenwen Xia[2], Gongyu Shi[1], Ming Wan[1], Yuncong Zhang[1] and Dawu Gu[1,2]

[1]School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, 200240, P.R.China
[2]School of Cyber Engineering
Xidian University, 710126, P.R.China

Abstract. In this paper, we reconsider the security for CRYSTALS-Dilithium, a lattice-based post-quantum signature scheme standardized by NIST. In their documentation, the authors proved that the security of the signature scheme can be based on the hardness of the following three assumptions: MLWE, MSIS and SelfTargetMSIS. While the first two are standard lattice assumptions with hardness well studied, the authors claimed that the third assumption SelfTargetMSIS can be estimated by the hardness of MSIS (and further into SIS). However, we point out that this is in fact not the case. We give a new algorithm for solving SelfTargetMSIS, by both experimental results and asymptotic complexities, we prove that under specific parameters, solving SelfTargetMSIS might be faster than MSIS. Although our algorithm does not propose a real threat to parameters used in Dilithium, we successfully show that solving SelfTargetMSIS cannot be turned into solving MSIS or MISIS. Furthermore, we define a new variant of MISIS, called sel-MISIS, and show that solving SelfTargetMSIS can only be turned into solving sel-MISIS. We believe that in order to fully understand the concrete hardness of SelfTargetMSIS and prevent potential attacks to Dilithium, the hardness of this new problem needs to be further studied.

Keywords: Lattice-based cryptography, short integer solution problem, concrete hardness, digital signature

1 Introduction

Since the security of classical asymmetric cryptographic algorithms has been challenged by the fast development of quantum computing, nowadays, many cryptographic researchers focus on constructing post-quantum encryption/signature schemes which can resist the attack of quantum algorithms. In 2016, NIST has risen a competition on post-quantum cryptography, which aim at standardizing quantum-resistant cryptographic algorithms, and the final results were announced in July 2022.

CRYSTALS-Dilithium [12] is one of the three digital signature schemes for standardization, which security based on the hardness of lattice assumptions.

Since lattice assumptions have been widely studied recent years, lattice-based schemes are usually believed to be as reliable as others. As the authors claimed, comparing with other candidates, Dilithium is more efficient, and simple to implement especially on low-energy devices. There are already a few implementation results on Dilithium [1, 16].

Dilithium follows a paradigm called Fiat-Shamir with abort [24, 25, 11], which is an extension of the most famous Fiat-Shamir heuristic. Fiat-Shamir heuristic can be used to transform a canonical identification scheme into a signature scheme in the random oracle model (ROM), while the unforgeability of the signature was guaranteed by the security of the identification scheme using the forking lemma. From Fiat-Shamir heuristic, the unforgeability of Dilithium can be (non-tightly) reduced to the hardness of MLWE and MSIS (which can be considered as LWE and SIS problem in modular lattices [22]). However, if we allow quantum access to a random oracle (such a model is called QROM, the quantum RO model), using forking lemma becomes more difficult. So while considering quantum adversaries, one must find new methods to prove the unforgeability of the signature. In [19], a new hardness assumption called SelfTargetMSIS was presented, so that the security of Dilithium can be based on this new assumption. We first informally state the hardness assumption SelfTargetMSIS before further discussions.

SelfTargetMSIS: Let R be a polynomial ring. Given a hash function H which hashes into small norm polynomials in R and a random matrix $\mathbf{A} \leftarrow R_q^{m \times k}$, a random vector $\mathbf{t} \leftarrow R_q^m$, find a message μ, small norm pair $\mathbf{y} \in R^{k+m}, c \in R$ such that $H(\mu \| [\mathbf{I}|\mathbf{A}]\mathbf{y} + c\mathbf{t}) = c$.

Although challenging the hardness of SelfTargetMSIS seems to be difficult, the concrete security of this assumption has not yet been fully studied, compared with other hardness assumptions used in Dilithium (MLWE and MSIS). In the document of Dilithium, the authors claimed that in order to solve the SelfTargetMSIS problem, one must either invert the hash function H, or take the following approach: First, pick a message μ and a vector $\mathbf{w}$, calculate $c = H(\mu \| \mathbf{w})$ and then find a small norm solution $\mathbf{y}$ for $[\mathbf{I}|\mathbf{A}]\mathbf{y} = \mathbf{w} - c\mathbf{t}$. The authors further set $\mathbf{t}' = \mathbf{w} - c\mathbf{t}$, so the equation has a same form with the lattice hard problem MISIS. Furthermore, in order to be more conservative in the security estimation, the authors set $\mathbf{A}' = [\mathbf{A}|\mathbf{t}']$, and use the hardness of MSIS problem $[\mathbf{I}|\mathbf{A}']\mathbf{y}' = \mathbf{0}$ to estimate the hardness of SelfTargetMSIS.

In this paper, we point out that the authors have made the wrong claim by presenting an algorithm for solving SelfTargetMSIS. We show by both experiments and complexity analysis, that under certain parameters, solving SelfTargetMSIS is faster than solving the MSIS problem $[\mathbf{I}|\mathbf{A}']\mathbf{y}' = \mathbf{0}$ using standard lattice-solving algorithms, so it turns out to be problematic for the designers to use MSIS/SIS for estimating the concrete hardness of SelfTargetMSIS.

We further give some theoretical analysis on the relationship between SelfTargetMSIS and MSIS. We show that the claim by the designers is incorrect at two points: (1) since any adversary can choose multiple pairs of $(\mu, \mathbf{w})$, it also gets multiple values for $\mathbf{t}'$, say $\mathbf{t}'_1, ..., \mathbf{t}'_Q$, which means that instead of solving

a fixed equation $[\mathbf{I}|\mathbf{A}|\mathbf{t}']\mathbf{y}' = \mathbf{0}$ (resp. $[\mathbf{I}|\mathbf{A}']\mathbf{y} = \mathbf{t}'$), the adversary can solve one of the following Q equations: $[\mathbf{I}|\mathbf{A}|\mathbf{t}'_i]\mathbf{y}' = \mathbf{0}$ (resp. $[\mathbf{I}|\mathbf{A}']\mathbf{y} = \mathbf{t}'_i$), $i = 1, ..., Q$; (2) the adversary can choose $\mathbf{w}$ after seeing $\mathbf{A}, \mathbf{t}$, thus $\mathbf{t}' = \mathbf{w} - c\mathbf{t}$ is not independent with $\mathbf{A}$, hence $[\mathbf{A}|\mathbf{t}']$ is not a uniform random matrix, which is necessary for the hardness of MSIS or MISIS problem.

We note that in fact, the problem correlated to SelfTargetMSIS is a new assumption different from the original MSIS or MISIS, and we call it sel-MISIS. We also introduced another problem called mul-MISIS which is a special case of sel-MISIS. The figure below shows the relationship between different hardness assumptions.

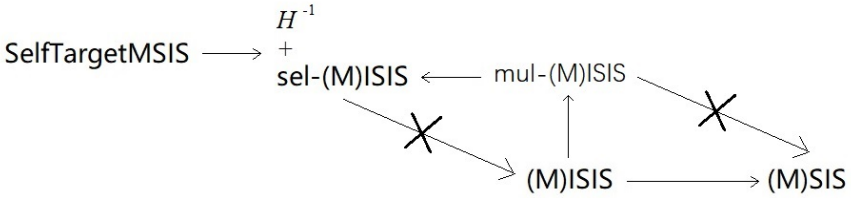

Fig. 1: Relationship between hardness assumptions; $A \to B$ means that the hardness of A is equal or harder than B.

We analyse this new problem sel-MISIS from both theoretical reduction and concrete hardness, and show that is is nearly impossible to say that sel-MISIS is equal or harder than MSIS. So in determining the security level of Dilithium, one must use sel-MISIS instead of MSIS to estimate the hardness of SelfTargetMSIS. However, since the complexity in solving sel-MISIS (or even MISIS) has not yet been well studied, we leave the open question that whether there exists a solid lower bound for the hardness in solving sel-MISIS.

The paper is organized as follows:

In Section 2, we give some basic knowledge on lattices. In Section 3, we introduce basic lattice solving algorithms. In Section 4, we give our algorithm for solving SelfTargetMSIS. In Section 5, we give some new assumptions which are related to SelfTargetMSIS, and give some analysis. In Section 6, we draw the conclusion.

2 Preliminaries

Notations. $x \leftarrow \chi$ for a distribution χ means that x is sampled from χ. $x \leftarrow X$ for a set X means that x is uniformly random chosen from X. For any odd modulus q, $\mathbb{Z}_q$ and the operation $\bmod q$ takes value from $[-\frac{q-1}{2}, \frac{q-1}{2}]$. We say that ϵ is negligible in λ, if $\epsilon < 1/\Omega(\lambda^c)$ for any $c > 0$. The polynomial ring $R := \mathbb{Z}[X]/(X^n + 1)$ and $R_q := \mathbb{Z}_q[X]/(X^n + 1)$ ($n = 256$ in Dilithium).

Infinity norm, 1-norm and 2-norm (Euclidean norm): For a polynomial $x = c_{n-1}X^{n-1} + ... + c_1X + c_0$, the infinity norm $\|x\|_\infty = \max_{i=0}^{n-1} |c_i|$, 1-norm $\|x\|_1 = \sum_{i=0}^{n-1} |c_i|$ and the Euclidean norm $\|x\| = \sqrt{\sum_{i=0}^{n-1} c_i^2}$.

For a vector $\mathbf{x} = (x_1, ..., x_m)$ of polynomials, the infinity norm $\|\mathbf{x}\|_\infty = \max_{j=1}^{m} \|x_j\|_\infty$, the 1-norm $\|x\|_1 = \sum_{j=1}^{m} \|x_j\|_1$, and the Euclidean norm $\|x\| = \sqrt{\sum_{j=1}^{m} \|x_j\|^2}$.

For a vector $\mathbf{x} = (x_1, ..., x_m)$ of scalars, the infinity norm $\|\mathbf{x}\|_\infty = \max_{j=1}^{m} |x_j|$, the 1-norm $\|x\|_1 = \sum_{j=1}^{m} |x_j|$ and the Euclidean norm $\|\mathbf{x}\| = \sqrt{\sum_{j=1}^{m} x_j^2}$.

The "target ball" in Dilithium which H hashes into: $B_\tau := \{c \in R : \|c\|_1 = \tau \wedge \|c\|_\infty = 1\}$.

2.1 Lattice Background

Definition 2.1. *Given a set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_d\} \subset \mathbb{R}^n$, lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is defined as:*

$$\mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^n = \{\sum_{i=1}^{d} z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}.$$

$\mathbf{B}$ is called a basis of the lattice $\mathcal{L}$, and d is called the dimension of $\mathcal{L}$. We write $\lambda_1(\mathcal{L})$ as the length of the shortest non-zero vector in $\mathcal{L}$.

Definition 2.2. *The Shortest Vector Problem (SVP) is defined as: given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, finding the shortest non-zero vector $\boldsymbol{v}$ in $\mathcal{L}$.*

The γ-approximate Shortest Vector Problem (γ-SVP) is defined as: given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, finding a non-zero vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma\lambda_1(\mathcal{L})$.

Definition 2.3. *The Closest Vector Problem (CVP) is: given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, and a target vector $\mathbf{t}$, finding the closest vector $\mathbf{v}$ in $\mathcal{L}$ to $\mathbf{t}$, i.e. finding a vector $\mathbf{v}$ such that $\|\mathbf{v} - \mathbf{t}\| = \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$.*

The γ-approximate Closest Vector Problem (γ-CVP) is: given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, and a target vector $\mathbf{t}$, finding a vector $\mathbf{v}$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$.

2.2 Hardness Assumptions in Dilithium

The main results in this paper do not require the detailed description of Dilithium, only its hardness assumptions. Those who are interested may refer to the document of Dilithium [12]. Below, we state the hardness assumptions used in the security proof of Dilithium [12, 19].

Definition 2.4. *For integers m, k, and a probability distribution $D : R_q \to [0, 1]$, we say that the advantage of algorithm $\mathcal{A}$ in solving the decisional $\mathsf{MLWE}_{m,k,D}$ problem over the ring R_q is*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{MLWE}}_{m,k,D}(\mathcal{A}) := |\Pr[b = 1 | \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{t} \leftarrow R_q^m; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})] \\ - \Pr[b = 1 | \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{s}_1 \leftarrow D^k; \mathbf{s}_2 \leftarrow D^m; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)]|.\end{aligned}$$

Definition 2.5. *To an algorithm $\mathcal{A}$ we associate the advantage function* $\mathrm{Adv}^{\mathsf{MSIS}}_{m,k,\gamma}$*, to solve the (Hermite Normal Form)* $\mathsf{MSIS}_{m,k,\gamma}$ *problem over the ring* R_q *as*

$$\mathrm{Adv}^{\mathsf{MSIS}}_{m,k,\gamma}(\mathcal{A}) := \Pr[0 < \|\mathbf{y}\|_\infty \leq \gamma \wedge [\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} = \mathbf{0} | \mathbf{A} \leftarrow R_q^{m \times k}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})].$$

If we set $R = \mathbb{Z}$*, the problem degenerates into the* $\mathsf{SIS}_{m,k,\gamma}$ *problem.*

Definition 2.6. *Suppose that* $H : \{0,1\}^* \to B_\tau$ *is a cryptographic hash function. To an algorithm* $\mathcal{A}$ *we associate the advantage function*

$$\mathrm{Adv}^{\mathsf{SelfTargetMSIS}}_{H,m,k,\gamma}(\mathcal{A}) :=$$
$$\Pr\left[\begin{array}{c} 0 \leq \|\mathbf{y}\|_\infty \leq \gamma \\ \wedge H(\mu \| [\mathbf{I}|\mathbf{A}|\mathbf{t}] \cdot \mathbf{r}) = c \end{array} \middle| \mathbf{A} \leftarrow R_q^{m \times k}; \left(\mathbf{r} := \begin{bmatrix} \mathbf{y} \\ c \end{bmatrix}, \mu \right) \leftarrow \mathcal{A}^{|H(.)\rangle}(\mathbf{A}, \mathbf{t}) \right].$$

to solve the hardness problem $\mathsf{SelfTargetMSIS}_{H,m,k,\gamma}$*.*

It was shown in [19] that the hardness of SelfTargetMSIS problem can be reduced to MSIS under the random oracle model (but not quantum random oracle model). However, it is not a tight reduction. We write the result below:

Lemma 2.1. *[19] Assume that* H *is a random oracle, and* $\mathcal{A}$ *makes at most* Q_H *queries to* H*. Then there exists an adversary* $\mathcal{A}'$ *such that* $\mathrm{Adv}^{\mathsf{SelfTargetMSIS}}_{H,m,k,\gamma}(\mathcal{A}) \approx \sqrt{\mathrm{Adv}^{\mathsf{MSIS}}_{m,k,2\gamma}(\mathcal{A}')}$.

Next, we define the inhomogeneous SIS (ISIS) problem, which we will also use in this paper.

Definition 2.7. *To an algorithm* $\mathcal{A}$ *we associate the advantage function* $\mathrm{Adv}^{\mathsf{MISIS}}_{m,k,\gamma}$*, to solve the (Hermite Normal Form)* $\mathsf{MISIS}_{m,k,\gamma}$ *problem over the ring* R_q *as*

$$\mathrm{Adv}^{\mathsf{MISIS}}_{m,k,\gamma}(\mathcal{A}) := \Pr[0 < \|\mathbf{y}\|_\infty \leq \gamma \wedge [\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} = \mathbf{t} | \mathbf{A} \leftarrow R_q^{m \times k}; \mathbf{t} \leftarrow R_q^m; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})].$$

If we set $R = \mathbb{Z}$*, the problem degenerates into* $\mathsf{ISIS}_{m,k,\gamma}$ *problem.*

2.3 Gaussian Heuristics and Voronoi Cells

The definitions below are useful for the security estimation in this paper. Here we consider only full-rank lattices, where the dimension of the lattice d is equal to the dimension of lattice vectors n.

Definition 2.8 (Fundamental Parallelpiped). *A fundamental parallelpiped* $\mathcal{P}(\mathbf{B})$ *for a lattice* $\mathcal{L}(\mathbf{B})$ *is defined as:* $\mathcal{P}(\mathbf{B}) := \mathbf{B} \cdot [-1/2, 1/2)^n$*.*

Since the basis of a lattice is not unique, fundamental parallelpiped is not unique either. However, each fundamental parallelpiped has the same volume, which is equal to $\det(\mathbf{B})$. We also write $\det(\mathcal{L}) = \det(\mathbf{B})$.

Proposition 2.1 (Gaussian Heuristic). *Gaussian heuristic claims that, for a (random) region $\Omega \subset \mathbb{R}^d$, for a random lattice $\mathcal{L}$, the number of lattice points contained in this region is roughly equal to $vol(\Omega)/\det(\mathcal{L})$.*

Moreover, $\det(\mathcal{L})$ is approximately equal to the volume of $\mathcal{B}(\lambda_1(\mathcal{L}))$, where $\mathcal{B}(r)$ is an n-dimension ball with radius r (as the latter contains exactly one lattice point). If we define

$$gh(\mathcal{L}) = \sqrt{\frac{d}{2\pi e}} \det(\mathcal{L})^{1/d},$$

then $\lambda_1(\mathcal{L}) \approx gh(\mathcal{L})$.

We note that, since the exact value of $\lambda_1(\mathcal{L})$ is usually hard to calculate, in most cases, solving γ-SVP means to find a short vector $\mathbf{x}$ which $\|\mathbf{x}\| \leq \gamma \cdot gh(\mathcal{L})$.

Definition 2.9 (Voronoi Cells). *[9] We write $\mathcal{H}(\mathbf{x})$ for half-spaces whose boundaries are orthogonal to $\mathbf{x}$ and pass through $\frac{1}{2}\mathbf{x}$. For a lattice $\mathcal{L}$ and a list $L \subset \mathcal{L}$, the approximate Voronoi cell generated by L is defined as:*

$$\mathcal{V}_L := \bigcap\nolimits_{\mathbf{r} \in L} \mathcal{H}(\mathbf{r}).$$

Voronoi cells are highly related to CVP. In fact, we have the following property:

Lemma 2.2. *[21] Let $\mathbf{v}$ be the solution of $CVP(\mathcal{L}, \mathbf{t})$, and L be the 2^d shortest vectors of $\mathcal{L}$. Then $\mathbf{t} - \mathbf{v} \in \mathcal{V}_L$.*

We also write $\mathcal{V} = \mathcal{V}_L$ if L contains the 2^d shortest vectors of $\mathcal{L}$. For a random target $\mathbf{t}$ and its CVP solution $\mathbf{v}$, we can heuristically assume that $\mathbf{t} - \mathbf{v}$ is uniformly distributed in $\mathcal{V}$.

For the volume of a Voronoi cell, we have the following result:

Lemma 2.3. *[21] If L contains 2^d shortest vectors in $\mathcal{L}$, then the volume of $\mathcal{V}_L$ is equal to $\det(\mathcal{L})$. Moreover, if L contains more than $2^{d/2}$ shortest vectors in $\mathcal{L}$, the result holds approximately.*

3 Algorithms for Solving Lattice Problems

From the definition, we can view SelfTargetMSIS as a variant of MSIS/MISIS. Just like other researchers in this field, we simplify the problem from two aspects:

1. If we expand each polynomial into $n \times n$ matrix, solving $\mathsf{MSIS}_{m,k,\gamma}$ and $\mathsf{MISIS}_{m,k,\gamma}$ can be turned into solving a special case of $\mathsf{SIS}_{nm,nk,\gamma}$ and $\mathsf{ISIS}_{nm,nk,\gamma}$. Although these new problems have some special algebraic structures, there is currently no methods for exploiting these structures, so we simply discuss SIS/ISIS solver instead of MSIS/MISIS solver.

2. Since currently all lattice solving algorithms consider the Euclidean norm, we use Euclidean norm instead of infinity norm for the length of the short solution, which is the same as Dilithium itself. Note that there is a high probability

where a vector with small Euclidean norm also has infinity norm small enough. For a more detailed discussion, we refer the readers to [12].

It is well known that solving SIS is equivalent to solve (approximate)-SVP with the following lattice basis: $\begin{bmatrix} q\mathbf{I} & \mathbf{A} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix}$, and solving ISIS is equivalent to solve (approximate)-CVP with the same lattice and the target vector $\binom{\mathbf{t}}{\mathbf{0}}$.

For correctness, we can easily see that any (short) vector in this lattice could be written by $\mathbf{y} = \begin{bmatrix} q\mathbf{u} + \mathbf{A}\mathbf{v} \\ -\mathbf{v} \end{bmatrix}$. So $[\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} \mod q = q\mathbf{u} + \mathbf{A}\mathbf{v} - \mathbf{A}\mathbf{v} \mod q = \mathbf{0}$ and $\mathbf{y}$ is a solution to the SIS problem, and any solution to the SIS problem is a short vector in this lattice. The correctness of ISIS is similar and we omit the details here.

So before we move on to the problem SelfTargetMSIS itself, we first briefly introduce the algorithms in solving SVP and CVP problems.

3.1 Lattice Sieving

Lattice sieving [2] is considered the most efficient algorithm in solving SVP or γ-SVP when γ is small. Briefly speaking, a sieving algorithm takes two random lattice vectors $\mathbf{x}, \mathbf{y}$ each step from a list, and if the length of $\mathbf{x} \pm \mathbf{y}$ is small, $\mathbf{x} \pm \mathbf{y}$ is added to the list, repeat such procedure until a short enough lattice vector is found. The two most commonly used basic lattice sieving algorithms are Guass sieve by Micciancio and Voulgaris [26], and NV sieve by Ngyuen and Vidick [28]. In [21, 7, 6], the authors introduced nearest neighbour data structures to store the lattice vectors, so that it is more easier to find a pair of close vectors in the algorithm. Other improvements include dimension-for-free [10] which lowers the lattice dimension and triple sieve [18] which saves space complexity. Most of these improvements can be used on both sieve.

We first give a simplified description for Gauss sieve.

Table 1: Description of Gauss sieve

Input: a set of lattice vectors $L \subset \mathcal{L}$, $\lvert L\rvert = (4/3)^{d/2}$, approximation factor γ.
Output: a set of short lattice vectors $L \subset \mathcal{L}$.
for each pair $\mathbf{x}, \mathbf{y} \in L$ **and** $\lVert\mathbf{x}\rVert \leq \lVert\mathbf{y}\rVert$ **do**:
if $\lVert\mathbf{x} - \mathbf{y}\rVert < \lVert\mathbf{y}\rVert$ **then**:
$L := L \cup \{\mathbf{x} - \mathbf{y}\} - \{\mathbf{y}\}$;
if $\lVert\mathbf{x} + \mathbf{y}\rVert < \lVert\mathbf{y}\rVert$ **then**:
$L := L \cup \{\mathbf{x} + \mathbf{y}\} - \{\mathbf{y}\}$;
$\mathbf{w}$:= the shortest vector in L;
if $\lVert\mathbf{w}\rVert \leq \gamma \cdot gh(\mathcal{L})$ **then**:
break;
Output $\mathbf{w}$.

We see that the sieving algorithm not only outputs a short vector, but also a set of (relatively) short vectors L. This property is used in our algorithm below.

It was proven by [20], that sieving algorithms has a lower bound of $O(2^{0.292n})$, which is achieved by the BDGL sieve [6] (in theory only). The most efficient sieving algorithm in practice is the BGJ1 sieve [7], which can achieve a time complexity of $O(2^{0.311n})$.

3.2 BKZ

When the lattice dimension is large, the best algorithm in solving approximate SVP is the blockwize Korkin-Zolotarev (BKZ) algorithm [30, 8]. BKZ algorithm has a block size β as its parameter, uses SVP oracle (either enumeration or lattice sieving) each time on a β-dimension sub-lattice of the original lattice to generate short lattice vectors, and LLL [23] on the original lattice to further reduce the lattice basis. The complexity of BKZ is determined by its block size, so BKZ with larger β costs more time, but the output vector is shorter.

We give the description of plain BKZ.

Table 2: Description of BKZ

Input: a basis $\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n) \in \mathbb{Z}^{n \times n}$ for lattice $\mathcal{L}$, the block size β.
Output: the β-BKZ reduced basis of $\mathcal{L}$.

LLL($\mathbf{B}$);
$z \leftarrow 0; j \leftarrow 0;$
while $z < n - 1$ **do**:
  $j \leftarrow (j \mod (n-1)) + 1$; $k \leftarrow \min(j + \beta - 1, n)$; $h \leftarrow \min(k + 1, n)$;
  Find $\mathbf{v} = (v_j, ..., v_k) \in \mathbb{Z}^{k-j+1}$, such that $\|\pi_j(\sum_{i=j}^{k} v_i \mathbf{b}_i)\| = \lambda_1(\mathcal{L}_{[j,k]})$;
  if $\mathbf{v} \neq (1, 0, ..., 0)$ **then**:
    $z \leftarrow 0$; LLL($\mathbf{b}_1, ..., \mathbf{b}_{j-1}, \sum_{i=j}^{k} v_i \mathbf{b}_i, \mathbf{b}_j, ..., \mathbf{b}_h$);
      % start from the j-th loop
  else:
    $z \leftarrow z + 1$; LLL($\mathbf{b}_1, ..., \mathbf{b}_h$);% start from the $h - 1$-th loop
Output $(\mathbf{b}_1, ..., \mathbf{b}_n)$.

Since LLL is a polynomial time algorithm, the time cost of BKZ is dominated by the time cost of SVP-oracle, i.e. enumeration or sieving. In [4], the authors used the time of one call to SVP-oracle $2^{0.292\beta}$ to estimate the time cost of BKZ, which is called core-SVP model. We note that Dilithium also takes the same model for estimating the security levels.

There are many other improvements to BKZ, such as progressive BKZ [5], and pump-and-jump BKZ in G6K [3].

3.3 Randomized Slicer and Solving CVP

Above, we introduce algorithms for solving SVP and approximate SVP, which can also be used to solve the (M)SIS problem. However, by the definition, Self-TargetMSIS is more like a variant of (M)ISIS than a variant of (M)SIS. While (M)SIS can be easily turned into solving (approximate)-SVP, (M)ISIS can only be turned into (approximate)-CVP. We first simply introduce the current results on solving CVP.

As it was pointed out in [9, 13], the most efficient algorithm in solving CVP is using a preprocessing method: first output an intermediate result L which is only related with $\mathcal{L}$ but not the target $\mathbf{t}$; and then use L and $\mathbf{t}$ to solve the CVP problem. Such approach is called CVP with preprocessing, or CVPP for short.

The first algorithm for solving CVPP is presented by Laarhoven [21], which uses a variant of the Voronoi cells approach by Micciancio and Walter [27]. Doulgerakis et al [9] improved the method along with a randomized version of the iterative slicing algorithm [31], gaining a $2^{0.292d+o(d)}$ space and $2^{0.220d+o(d)}$ time complexity. Ducas et al [13] provided a sharp bound for the success rate of randomized slicing algorithm, which improved the result of [9].

Solving CVPP using approximate Voronoi cells contains two steps: first using enumeration or lattice sieving to find a set of shortest lattice vectors (preprocessing step), and then using *The Randomized Heuristic slicer for finding closest vectors* to find the closest lattice vector. The randomized slicing algorithm is described as follows:

Table 3: The Randomized Heuristic slicer for finding closest vectors

Input: a list $L \subset \mathcal{L}$ (sorted by norm from small to large) and a target $\mathbf{t} \in \mathbb{R}^d$
Output: closest lattice vector $\mathbf{s} \in \mathcal{L}$ to $\mathbf{t}$

$\mathbf{s} := \mathbf{0}$
while $\mathbf{s}$ is not a closest vector to $\mathbf{t}$:
  Sample $\mathbf{t}' \in \mathbf{t} + \mathcal{L}$; %Sample from discrete Gaussian
  for $\mathbf{x} \in L$:
    if $\|\mathbf{t}' - \mathbf{x}\| < \|\mathbf{t}'\|$:
      $\mathbf{t}' := \mathbf{t}' - \mathbf{x}$ and restart the for- loop;
  if $\|\mathbf{t}'\| < \|\mathbf{t} - \mathbf{s}\|$:
    $\mathbf{s} := \mathbf{t} - \mathbf{t}'$
return $\mathbf{s}$

From what we know, this algorithm is the fastest algorithm for solving CVP up to now.

In [9], if L contains α^d shortest vectors, the running time of randomized slicer can be bounded by the following inequations:

$$\mathcal{V}_L \leq \left(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16}\right)^{d^2+o(d)} \cdot \det(\mathcal{L}).$$

In practice, randomized slicer runs much faster than the sieving algorithm. In their experiment [9], for a 50-dimension lattice, sieving takes 4 seconds, while slicer takes only 2 milliseconds.

4 Constructing SelfTargetMISIS Solver

4.1 Algorithm Description

First we briefly introduce the main idea of our algorithm. In SelfTargetMSIS, the hash function H hashes into B_τ which contains more than 2^λ different elements, λ is the security parameter, so it is impossible to invert the hash function H. However, finding a collision for H is much simpler. After fixing a vector $\mathbf{w}$, we generate two different sets, and try to find a collision between them: (1) $C_1 \subset \{c := H(\mu\|\mathbf{w})|\mu \in \{0,1\}^*\}$ with different message μ; (2) $C_2 \subset \{c \in B_\tau|[\mathbf{I}|\mathbf{A}]\mathbf{y} = \mathbf{w} - c\mathbf{t}, \|\mathbf{y}\| \leq \beta\}$ where we can find a short solution for the MISIS problem. If we can find $c^* \in C_1 \cap C_2$, which means that $c^* = H(\mathbf{w}|\mu^*)$ and $[\mathbf{I}|\mathbf{A}]\mathbf{y}^* = \mathbf{w} + c^*\mathbf{t}$, then $\mu^*, c^*, \mathbf{y}^*$ form a solution to SelfTargetMSIS.

At a first glimpse, it seems that for generating each element in C_2, we still need to solve an MISIS problem which is not easier than the MSIS problem. However, if we are able to generate many elements of C_2 in one run of randomized slicer, then it might be possible to generate the set C_2 in a reasonable time to solve SelfTargetMSIS problem. So we modify the randomized slicer to handle this problem.

We noticed that, for $c' \in R$ which coefficients contain only 2 $\pm$1s and others are all 0, there is a high probability that for $c \in B_\tau$, $c - c' \in B_\tau$ (for the parameters in Dilithium where $n = 256, \tau = 60$, the probability is $\approx 18\%$).

So if we expand the slicing set L to include short vectors $\mathbf{y}'$ such that $[\mathbf{I}|\mathbf{A}]\mathbf{y}' = c'\mathbf{t}$, $\mathbf{y}'$ can be used to "slice" the target vector to change c into another element in B_τ. Also, as we include more vectors in the slicing set L, it is more possible for the slicer to find a close vector, hence a solution to SelfTargetMSIS.

Now we give the pseudo-code for our SelfTargetMSIS solver. Below we let $\theta \cdot |B_\tau|$ be the size of C_1, $\phi \cdot \gamma$ with $\phi > 1$ is the length of the shortest vector in the original slicing set, and $|L'| = \psi \cdot |L|$ is the size of additional slicing set. Let $M(\mathbf{A})$ be the $nm \times nk$ matrix which expands each polynomial in $\mathbf{A}$ into a $n \times n$ matrix. For simplicity reason, we do not distinguish between vector of polynomials and vector of polynomial coefficients.

We note that in this algorithm, we still choose a random $\mathbf{w}$, while $\mathbf{w}$ in fact can be chosen arbitrarily by the adversary. If we find a better strategy for the adversary to choose $\mathbf{w}$, the algorithm might further be improved.

Table 4: SelfTargetMSIS-Solver

Input: the parameter $\mathbf{A}, \mathbf{t}, \gamma$ for SelfTargetMSIS and θ, ϕ, ψ
Output: a short integer solution $\mathbf{y}$ such that $\|\mathbf{y}\| \leq \gamma$

function SelfTargetMISIS-Solver($\mathbf{A}, \mathbf{t}, H, B_\tau, \gamma, \theta, \phi, \psi$):
$C_1 := \emptyset$; $\mathbf{w} \leftarrow_\$ R_q^m$;
for $i = 1$ **to** $\theta \cdot |B_\tau|$ **do**:
$\mu \leftarrow_\$ \{0,1\}^*$;
$C_1 := C_1 \cup \{(\mu, H(\mu \| \mathbf{w}))\}$;
$\mathbf{B} := \begin{bmatrix} q\mathbf{I} & M(\mathbf{A}) \\ \mathbf{0} & -\mathbf{I} \end{bmatrix}$;
$L := \text{Sieving}(\mathbf{B}, \phi \cdot \gamma)$; %Use the sieving algorithm to generate a set of short vectors;
$L' := \emptyset$;
for $i = 1$ **to** $\psi \cdot |L|$ **do**:
$c' \leftarrow B_2 \cup \{0\}$; $\mathbf{y}' := \begin{bmatrix} c'\mathbf{t} \\ \mathbf{0} \end{bmatrix}$;
$L' := L' \cup \{(\text{Slicer}(L, \mathbf{y}'), c')\}$; %Use a (non-randomized) slicer once for each vector;
$\mathbf{s} := \mathbf{0}$;
do:
$c \leftarrow B_\tau$; $\mathbf{u} := \mathbf{w} - c\mathbf{t}$;
$\mathbf{v} := \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix}$;
Sample $\mathbf{v}' \in \mathbf{v} + \mathcal{L}(\mathbf{B})$; %Sample from discrete Gaussian;
for $\mathbf{x} \in L$:
if $\|\mathbf{v}' - \mathbf{x}\| < \|\mathbf{v}'\|$:
$\mathbf{v}' := \mathbf{v}' - \mathbf{x}$; and restart the for- loop;
for $(\mathbf{x}, c') \in L'$:
if $\|\mathbf{v}' - \mathbf{x}\| < \|\mathbf{v}'\|$ **and** $c - c' \in B_\tau$:
$\mathbf{v}' := \mathbf{v}' - \mathbf{x}$; $c := c - c'$ and restart the for- loop;
if $\|\mathbf{v}'\| < \|\mathbf{v} - \mathbf{s}\|$:
$\mathbf{s} := \mathbf{v} - \mathbf{v}'$;
loop while $\|\mathbf{s} - \mathbf{v}\| > \gamma$ **or** $\nexists \mu$, $(\mu, c) \in C_1$;
find μ s.t. $(\mu, c) \in C_1$;
return $\mu, c, \mathbf{s} - \mathbf{v}$;

4.2 Experimental Results

We compare the efficiency of solving SelfTargetMSIS using our algorithm with solving the corresponding MSIS problem using lattice sieving. Here, we note that for many parameter selections, such as parameters in Dilithium and the parameters we choose below, solving $\text{MSIS}_{m,k+1,\gamma}$ by SVP on $\mathcal{L}' = \mathcal{L}(\begin{bmatrix} q\mathbf{I} & M(\mathbf{A}|\mathbf{t}') \\ \mathbf{0} & -\mathbf{I} \end{bmatrix})$ in fact takes more time than solving $\text{MSIS}_{m,k,\gamma}$ by SVP on $\mathcal{L} = \mathcal{L}(\begin{bmatrix} q\mathbf{I} & M(\mathbf{A}) \\ \mathbf{0} & -\mathbf{I} \end{bmatrix})$ (we give a theoretical analysis in the appendix), so we only compare solving $\text{SelfTargetMSIS}_{H,m,k,\gamma}$ with solving $\text{MSIS}_{m,k,\gamma}$.

For parameter selection, we set $n \in \{20, 22, 24\}, \tau = n/2$, $k = 3, m = 1$, $q = 32768$, and the norm of short vector $\mathbf{y}$ is $1.2 \cdot gh(\mathcal{L})$. We do not fix θ, ϕ, ψ but test different values to find an optimal selection.

For solving SVP, we use Gauss sieve as the SVP-solver to compare with the SelfTargetMSIS solver described above, and both algorithms have a same level of optimization to show the comparison between the running time. We implement the hash function H such that inverting H takes approximately the same time as solving $\text{MSIS}_{m,k,\gamma}$.

Let $T_{H^{-1}}$ be the average time for inverting H, T_{MSIS} be the time for solving MSIS, $T = T_1 + T_2 + T_3$ be the time for solving SelfTargetMSIS, where T_1 is the time of generating C_1, T_2 is the time for generating the slicing set $L \cup L'$, and T_3 is the time for randomized slicer. Our experimental results are list in the table below:

n	$T_{H^{-1}}$	T_{MSIS}	T	T_1	T_2	T_3
20	≈462s	446.38s	368.86s	12.83s	343.96s	12.07s
22	≈3527s	3700.15s	3220.50s	141.08s	2956.92s	122.50s
24	≈20281s	20953.01s	19980.28s	1267.56s	17769.44s	943.28s

Table 5: Comparison between solving MSIS and SelfTargetMSIS

We can see that T is smaller than both $T_{H^{-1}}$ and T_{MSIS} in all parameter selections.

Discussion on Countermeasures. We can see that there is an easy way to fix this problem by enlarging the size of B_τ into $2^{2\lambda}$, so that even finding a collision for H has complexity 2^λ. However, for $n = 256$, $|B_\tau| < 2^{2\lambda}$ for any possible $\tau = [0, .., 256]$ with $\lambda = 256$, hence one must change n to reach a security for 256-bit, which cannot easily be done under the framework of Dilithium.

4.3 Asymptotic Complexity

To further show the efficiency of our new algorithm, we give an asymptotic analysis of the complexity of our new algorithm. We suppose that the target vector length of SelfTargetMSIS is $\gamma = \gamma' \cdot \lambda_1(\mathcal{L})$.

Our SelfTargetMSIS solver can be approximately divided into four steps: Step 1, generating C_1; Step 2, generating L; Step 3, generating L'; Step 4, using randomized slicer to solve the required short vector. Now we analyse the time cost of each steps.

In Step 1, we suppose that each call to hash function H costs t, so the time cost is approximately $T_1 = t \cdot |C_1| = t\theta \cdot |B_\tau|$.

In Step 2, we call a sieving algorithm to generate a set of short vectors, which shortest vector is $\phi\gamma' \cdot \lambda_1(\mathcal{L})$. By the theoretical analysis of sieving in [28], the relationship between running time and the length of vectors is determined by a tunable parameter $1 < \delta < 1.5$. With a total number of vectors for no more than

$|L| \approx 2^{\delta \cdot 0.207d+o(d)}$, each iteration reduces the length of vectors to $1/\delta$ times and costs no more than $2^{\delta \cdot cd+o(d)}$, where $c \approx 0.292$ for the most efficient sieving algorithm [6]. We do not have an analytical solution for δ in minimizing the total cost, but for $d \geq 100$, $\delta < 1.05$ which is close to 1.

Suppose that we begin with a LLL-reduced basis which shortest vector length is about $2^{(d-1)/2}\lambda_1(\mathcal{L})$, the cost should be $\approx ((\frac{d-1}{2} - \log(\phi\gamma'))/\log\delta) \cdot 2^{\delta \cdot 0.292d+o(d)}$.

In Step 3, each vector is sliced by all vectors in L before adding to the set L'. As pointed out in [9], by using nearest neighbour search, adding a vector costs only $\approx 2^{0.085n+o(n)}$, and the total cost is $|L'| \cdot 2^{0.085d+o(d)}$. Since the set L is large enough, we heuristically assume that the average length of vectors in L' is approximate to that in L. We suppose that $|L'| = \psi \cdot |L|$, then the total cost of Step 2 and Step 3 is $T_2 = ((\frac{d-1}{2} - \log(\phi\gamma'))/\log\delta + \psi) \cdot 2^{\delta \cdot 0.292d+o(d)}$.

In Step 4, we calculate the cost of randomized slicer. We note that, not all vectors in L' can be used to slice the target vector $c\mathbf{t}$, but only those vector $c'\mathbf{t}$ such that $c - c' \in B_\tau$, which fraction is $\rho = \frac{\tau(n-\tau)}{n(n-1)}$. We suppose that L^* is the actual slicing set, where $|L^*| \approx |L| + \rho \cdot |L'| = (1 + \rho\psi) \cdot 2^{0.207d+o(d)}$.

Let $\alpha = |L^*|^{1/d}$, we heuristically assume that vectors in L^* have the lengths of approximately $\phi\gamma'$ times of the lengths of the $|L^*|$ shortest vectors in $\mathcal{L}$. Adopting the heuristic assumption in [9], we have:

$$\mathcal{V}_{L^*} \leq (\phi\gamma')^d \left(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16}\right)^{d/2+o(d)} \cdot \det(\mathcal{L}).$$

By Gaussian Heuristic, a d-dimension ball with radius $\lambda_1(\mathcal{L})$ is approximately $\det(\mathcal{L})$, which means that a d-dimension ball with radius $\gamma'\lambda_1(\mathcal{L})$ is approximately $\gamma'^d \det(\mathcal{L})$. Since each time finding a short vector which length within $\gamma' \cdot \lambda_1(\mathcal{L})$, there is a probability of $\frac{|C_1|}{|B_\tau|} = \theta$ for solving SelfTargetMSIS, we can see that the running time of randomized slicer can be estimated by $T_3 = \frac{1}{\theta}\phi^d \left(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16}\right)^{d/2+o(d)}$.

Since $|L^*| \approx (1 + \rho\psi) \cdot 2^{0.207d+o(d)}$, we have $\alpha \approx (1 + \rho\psi)^{1/d} \cdot 2^{0.207} > 2^{0.207}$. We can calculate that $T_3 < \frac{1}{\theta}\phi^d \cdot 2^{0.150d+o(d)}$.

The time cost of SelfTargetMSIS solver is $T = T_1 + T_2 + T_3$. Similar to the analysis of Step 2, we see that the time cost of MSIS solver using sieving algorithms can be estimated by $T_{\mathsf{MSIS}} = ((\frac{d-1}{2} - \log\gamma')/\log\delta) \cdot 2^{\delta \cdot 0.292d+o(d)}$. Also, for a secure hash H, the average cost of inverting H is about $T_{H^{-1}} = t \cdot (|B_\tau|/2)$, for simplicity reason, we suppose that $T_{H^{-1}} = T_{\mathsf{MSIS}}$, thus $T_1 = 2\theta \cdot T_{\mathsf{MSIS}}$.

Now we compare between the costs of SelfTargetMSIS solver T and MSIS solver T_{MSIS}. We omit all the $o(d)$ terms, and we have that:

$$\begin{aligned} T_{\mathsf{MSIS}} - T \geq & ((1-2\theta)(\frac{d-1}{2} - \log\gamma')/\log\delta - (\frac{d-1}{2} - \log(\phi\gamma'))/\log\delta - \psi) \cdot 2^{0.292d} \\ & - \frac{1}{\theta}\phi^d \cdot 2^{0.150d}. \end{aligned}$$

Consider the expression above as a substraction of two terms, we see that for $\phi < 2^{0.292-0.150} \approx 1.1$, the former term grows faster than the latter term with d,

as long as $(1-2\theta)(\frac{d-1}{2} - \log\gamma')/\log\delta - (\frac{d-1}{2} - \log(\phi\gamma'))/\log\delta - \psi > 0$, which is not hard to obtain by a careful choice of ψ and θ. So $T_{\mathsf{MSIS}} - T > 0$ holds asymptotically.

By the theoretical result, we show that our SelfTargetMSIS solver is more efficient than an MSIS solver with sieving algorithm. But for larger approximate factor γ, solving MSIS by BKZ proves to be more efficient than a full-dimensional sieving, thus an MSIS solver with BKZ may take less time than our SelfTargetMSIS solver. We note that our SelfTargetMSIS solver cannot be accelerated by BKZ, since randomized slicer requires a large slicing set, which can only be generated from a full-dimensional sieving.

For the parameter choices of Dilithium, the length of target vector $\approx q/16$ is relatively large, hence our SelfTargetMSIS solver does not propose a real threat to Dilithium in practice. However, we show that changing the parameters of Dilithium must be more careful, since the security analysis of Dilithium cannot be extended to all possible parameters. Also, we cannot rule out the possibility that our algorithm can further be improved to attack Dilithium with real parameter choices.

5 Revisiting the Relationship between SelfTargetMSIS and MSIS

We already show that it is incorrect for the designers of Dilithium to say that SelfTargetMSIS is harder to solve than MSIS. Now we shall give a theoretical analysis on why this problem occurs.

In [12], the authors claimed that the hardness of SelfTargetMSIS can be estimated by the hardness of MSIS (further into SIS) by a standard method in estimating the security of Fiat-Shamir type signatures. However, we point out the main difference between Dilithium and earlier Fiat-Shamir signature schemes (such as in [17, 29]). In [17, 29], in order to forge a signature, one must either break the hash function, or solve a standard hardness assumption: RSA or discrete log. However, it is not the case for Dilithium.

We simply write down the Fiat-Shamir type signature scheme in [29] (with slight modification) with possible attack, and discuss the differences between breaking [29] and breaking SelfTargetMSIS.

- Setup: Let p be a prime and α be elements in $\mathbb{Z}_p^*$ with prime order q ($\alpha^q \mod p = 1$). Pick $sk := s \leftarrow \mathbb{Z}_q$, $pk := (\alpha, v = \alpha^s)$. Let $H : \mathbb{Z}_p^* \times \{0,1\}^* \to \mathbb{Z}_p^*$ be a hash function.
- Sign: For message m, pick $r \leftarrow \mathbb{Z}_q$, set $x := \alpha^r \mod p$, $e := H(x, m)$, and $y := r + se \mod q$. Return (e, y).
- Verify: For message m and public key $pk = v$, compute $\bar{x} = \alpha^y v^e \mod p$, and check that $e = H(\bar{x}, m)$.

In order to break the signature scheme, the adversary may first guess x, m, calculate $e := H(x, m)$, and try to find the discrete log y of x/v^e. Since there

are only two elements in the public key, the best strategy for the adversary to choose x is to choose $c, d \in \mathbb{Z}_q$ and let $x := v^c\alpha^d$, thus $v^c\alpha^d/v^e = \alpha^y$. If H is secure, the probability of $e = c$ is negligible, then we have $v = \alpha^{(y-d)/(c-e)}$, hence $s = (y-d)/(c-e)$. So breaking [29] (using such approach) is equivalent in either breaking discrete log or breaking the hash function H.

The discussion above holds partly due to the fact that the "single-target" and "multi-target" versions of discrete log problem have the same hardness. We can see that no matter for how many successful guesses, it finally results in finding the same secret key s. However, it is not the case for breaking SelfTargetMSIS. In Dilithium, in order to solve the SelfTargetMSIS problem using a similar method, one must either break the hash function H, or take the following approach: First, pick a message μ and a vector $\mathbf{w}$, calculate $c = H(\mu\|\mathbf{w})$ (or query the random oracle H, while considering RO model) and then find a small norm solution $\mathbf{y}$ for the problem $[\mathbf{I}|\mathbf{A}]\mathbf{y} = \mathbf{w} - c\mathbf{t}$. Different successful guesses for $\mathbf{w}, c$ lead to different target vectors $\mathbf{t}' = \mathbf{w} - c\mathbf{t}$, and forging a signature only results in breaking one of them. This is in fact a multi-target version of MISIS, which we call it mul-MISIS, and its hardness cannot be tightly reduced to the single-target version.

There is also another main difference. In breaking [29], we aim to find the discrete log of x/v^e. No matter how the adversary chooses $x := v^c\alpha^d$, since e is uniformly random, x/v^e is uniformly distributed in $\{\alpha^i | i \in \mathbb{Z}_q\}$. But for breaking SelfTargetMSIS, $\mathbf{t}$ is a fixed vector and the value of c is chosen form a set B_τ with only about 2^{256} elements, while $\mathbf{w}$ is chosen from R_q^m which contains more than 2^{20000} elements for the parameter choice in Dilithium. It is clear that $c\mathbf{t}$ cannot be used to randomize $\mathbf{w}$ (which is chosen by the adversary), which means that the problem corresponding with SelfTargetMSIS is not the standard MISIS (or even the multi-target version), as the randomness of the target vector(s) are required in MISIS.

We can see that the target vectors are determined by three parameters which are chosen at three different timings: $\mathbf{t}$ is chosen at the beginning when generating the public key; $\mathbf{w}$ is chosen by the adversary; c is chosen by the random oracle after given $\mathbf{w}$. It is impossible to describe the concrete distribution of target vectors, since we do not know how the adversary chooses $\mathbf{w}$. Instead, we use sets of distributions to describe the target vectors, and we call them "selection sets". The modified version of MISIS where target vectors are chosen from selection sets is called selective MISIS (sel-MISIS for short), the term "selective" shows the fact that an adversary can choose $\mathbf{w}$, hence partly choose target vectors in this problem. Below, we give the formal definition of this assumption.

5.1 New Hardness Assumptions: mul-(M)ISIS and sel-(M)ISIS

First, we formally define the multi-target version of (M)ISIS.

Definition 5.1. *To an algorithm $\mathcal{A}$ we associate the advantage function* $\mathrm{Adv}^{\mathsf{mul-MISIS}}_{m,k,l,\gamma}$, *to solve the (Hermite Normal Form)* $\mathsf{mul-MISIS}_{m,k,l,\gamma}$ *problem over the ring*

R_q *as*

$$\mathrm{Adv}^{\mathsf{mul-MISIS}}_{m,k,l,\gamma}(\mathcal{A}) := \Pr[0 < \|\mathbf{y}\|_\infty \leq \gamma \wedge [\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} = \mathbf{t}_i, i \in [l]| \\ \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{t} = \{\mathbf{t}_1, ..., \mathbf{t}_l\} \leftarrow R_q^{m\times l}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})].$$

If we set $R = \mathbb{Z}$, the problem degenerates into $\mathsf{mul} - \mathsf{ISIS}_{m,k,l,\gamma}$ *problem.*

We can see that $\mathsf{MISIS}_{m,k,l,\gamma}$ is at least as hard as $\mathsf{mul} - \mathsf{MISIS}_{m,k,\gamma}$ as it can be viewed as a special case which $l = 1$. We also prove the fact that $\mathsf{mul} - \mathsf{MISIS}$ can be reduced to MISIS.

Lemma 5.1. *If there is an adversary $\mathcal{A}$ which solves* $\mathsf{mul} - \mathsf{MISIS}_{m,k,l,\gamma}$ *with probability ϵ and time t, then there is an adversary $\mathcal{A}'$ which solves* $\mathsf{MISIS}_{m,k,\gamma+\Omega(q^{m/(k+m)})}$ *(or* $\mathsf{MSIS}_{m,k,\gamma+\Omega(q^{k/(k+m)})}$*) with probability ϵ and time $t+O(l)\mathrm{poly}(n, m, k, \log q)$.*

Proof. Given a MISIS input $\mathbf{A}, \mathbf{t}$ (or MSIS input $\mathbf{A}, \mathbf{t} = \mathbf{0}$), the adversary $\mathcal{A}'$ does the following: let $\mathbf{y}_1, ..., \mathbf{y}_l$ be random strings such that $\|\mathbf{y}_i\|_\infty \leq \beta$, $\beta = \Omega(q^{m/(k+m)})$, such that $[\mathbf{I}|\mathbf{A}] \cdot \mathbf{y}_i$, $i \in [l]$ is uniformly random in R_q^m, β exists by leftover hash lemma. Let $\mathbf{t}_i = \mathbf{t} + [\mathbf{I}|\mathbf{A}] \cdot \mathbf{y}_i$, $i \in [l]$, and pass $\mathbf{A}, \mathbf{t}_1, ..., \mathbf{t}_l$ to $\mathcal{A}$. If $\mathcal{A}$ returns $\mathbf{y}$ such that $[\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} = \mathbf{t}_i$, return $\mathbf{y} - \mathbf{y}_i$ as the short solution. It is easy to see that if $\mathcal{A}$ successes, $\mathcal{A}'$ also successes, with computational overhead linear in l and polynomial in other parameters. Thus we have our result. □

We note that this proof only works for infinity norm. While considering Euclidean norm, we also have a similar result, but the reduction is looser. We omit the details here.

Below, we omit l as long as l is polynomial in other parameters $(n, m, k, \log q)$, since l only has a small effect on the running time, as it was shown in the reduction above.

However in mul-(M)ISIS, the target vectors are still chosen randomly, which cannot cover the fact that the adversary can partly determine the target vector in solving SelfTargetMSIS. We give another hardness assumption which considers such ability of an adversary.

Definition 5.2. *Given a family of sets of distributions $\mathcal{D} = \{D_1, ..., D_l\}$, $D_i = \{P_1, ..., P_s\}$ such that for each $P_j \in D_i \in \mathcal{D}$, $P_j : R_q^m \to [0, 1]$ is a distribution. For $D = \{P_1, ..., P_s\} \in \mathcal{D}$, let $\mathcal{V}_D(j)$ output a random vector $\mathbf{t} \leftarrow P_j$. The advantage in solving* $\mathsf{sel} - \mathsf{MISIS}_{\mathcal{D},m,k,\gamma}$ *is defined as:*

$$\mathrm{Adv}^{\mathsf{sel-MISIS}}_{\mathcal{D},m,k,\gamma}(\mathcal{A}) := \Pr[0 < \|\mathbf{y}\|_\infty \leq \gamma \wedge [\mathbf{I}|\mathbf{A}] \cdot \mathbf{y} = \mathbf{t}'| \\ \mathbf{A} \leftarrow R_q^{m\times k}; D \leftarrow \mathcal{D}; \mathbf{y} \leftarrow \mathcal{A}^{\mathcal{V}_D(.)}(\mathbf{A}, D); \mathbf{t}' \textit{ is an output of } \mathcal{V}_D].$$

Each $D \in \mathcal{D}$ is called a selection set.
For $R = \mathbb{Z}$, the problem is called $\mathsf{sel} - \mathsf{ISIS}_{\mathcal{D},m,k,\gamma}$.

We can see that the hardness of sel-MISIS is related to $\mathcal{D}$. For example, if all distributions P_i in a selection set D are γ-bounded, the sel-MISIS problem

is easy, as $\binom{\mathbf{t}'}{\mathbf{0}}$ is an obvious solution for any $\mathbf{t}'$ returned by $\mathcal{V}_D$. However, when every distribution P_i is a uniform distribution on R_q^m, sel-MISIS degenerates into mul-MISIS which can be reduced to the standard MISIS.

For the sel-MISIS problem corresponding to SelfTargetMSIS, each selection set $D_{\mathbf{t}} = \{P_{\mathbf{w}} | \mathbf{w} \in R_q^m\}$ where $P_{\mathbf{w}} : R_q^m \to [0, 1]$ is defined as:

$$P_{\mathbf{w}}(\mathbf{r}) = \frac{|\{c \in B_\tau | \mathbf{r} = \mathbf{w} - c\mathbf{t})|}{|B_\tau|}.$$

Below, we name the selection sets as $\mathcal{ST}$.

By Lemma 5.1, the reduction between mul-(M)ISIS and (M)ISIS/(M)SIS is tight only when $k >> m \log q$. However, in the choice of Dilithium parameters, $k \approx m$ in all security levels, so even if we can reduce sel-(M)ISIS into (M)SIS by the same way, sel-(M)ISIS it is still simpler than (M)SIS with a same parameter. Next, we show that even that is impossible: the same reduction cannot be used to reduce sel-(M)ISIS$_{\mathcal{ST},m,k,r}$ into (M)ISIS$_{m,k,r'}$ or (M)SIS$_{m,k,r'}$ for selection set $\mathcal{ST}$ and r' such that $r' - r = \Omega(q^{m/(k+m)})$.

Let $S = \{\mathbf{x} \in R^m | \|\mathbf{x}\|_\infty \leq r' - r\}$. First, we suppose that in each selection set $D = \{P_1, ..., P_s\}$, $(|\{\mathbf{t}|P_i(\mathbf{t}) > 0\}| \cdot |S|)/|R_q^m|$ is negligible for each $i = 1, ..., s$. Since in the context of Dilithium, $|\{\mathbf{t}|P_i(\mathbf{t}) > 0\}| \leq |B_\tau| \leq 2^{258}$ and $|R_q^m| = q^{256m} \approx 2^{5888m}$, this assumption is reasonable if we let $r' - r = \Omega(q^{m/k+m})$.

In order to reduce sel-MISIS to MISIS/MSIS using a same way from Lemma 5.1, given an MISIS/MSIS sample $(\mathbf{A}, \mathbf{t})$, the challenger $\mathcal{C}$ must find a short integer solution for $[\mathbf{I}|\mathbf{A}]\mathbf{y} = \mathbf{t}$ (either $\mathbf{t} = \mathbf{0}$ or $\mathbf{t}$ is uniform) by challenging an adversary $\mathcal{A}$ which can solve the sel-MISIS problem. $\mathcal{C}$ can choose polynomially many short integer vectors $\mathbf{y}_1, ..., \mathbf{y}_t$ such that $\|\mathbf{y}_j\|_\infty \leq r' - r$, and asks $\mathcal{A}$ to solve one of the problems $[\mathbf{I}|\mathbf{A}]\mathbf{y}' = \mathbf{t}'$, $j = 1, ..., t$ such that $\|\mathbf{t}'_j - (\mathbf{t} + \mathbf{A}\mathbf{y}_j)\|_\infty \leq r' - r$, then return the solution $\mathbf{y} = \mathbf{y}' + \binom{\mathbf{t}'_j - (\mathbf{t} + \mathbf{A}\mathbf{y}_j)}{-\mathbf{y}_j}$, which means that $\mathcal{C}$ can choose the target $\mathbf{t}'$ from a set with $t|S|$ vectors, while the set is independent with any distribution $P_i \in D \in \mathcal{D}$.

However, since $\mathcal{A}$ can arbitrarily choose from a selection set, we can let $\mathcal{A}$ queries $\mathcal{V}$ with a same input, that is, $\mathcal{A}$ always samples from a distribution P_i. Since $(|\{\mathbf{t}|P_i(\mathbf{t}) > 0\}| \cdot |S|)/|R_q^m|$ is negligible, the probability that $\mathcal{C}$ can find a target $\mathbf{t}'$ such that $P_i(\mathbf{t}') > 0$ is also negligible, which means $\mathcal{C}$ cannot use $\mathcal{A}$ to solve MISIS/MSIS.

Gentry et.al. [15] gave a reduction from ISIS to so called IncIVD problem which does not require the target vector to be random. However, the reduction requires that $\mathbf{t}$ be chosen before given $\mathbf{A}$. Since in the definition of sel-ISIS, the adversary is allowed to choose the target vector from a distribution after seeing $\mathbf{A}$, the same reduction cannot be applied to sel-ISIS. We recommend the readers to [15] for more details.

By the discussions above, we have strong confidence that sel-(M)ISIS cannot be reduced to (M)ISIS or (M)SIS.

5.2 Concrete Hardness between mul-(M)ISIS and (M)SIS

Above, we discussed the reduction between mul/sel-(M)ISIS and (M)SIS, and now we move on to the concrete hardness of these new assumptions. Unfortunately, we have not found a proper way to estimate the hardness of sel-ISIS, mainly because that we do not know how to maximally exploit the structure of the selection set D. In this section, we only discuss the hardness estimation for mul-ISIS, and compare it with SIS.

The difference between solving SIS and ISIS is much like between solving SVP and CVP. While SVP always returns the shortest lattice vector which length is $\lambda_1(\mathcal{L})$, the distance between the target vector $\mathbf{t}$ and the CVP solution $\mathbf{v}$ is determined by the choice of $\mathbf{t}$. It is possible that $\mathbf{t}$ is extremely close to a lattice point, so that $\|\mathbf{v}-\mathbf{t}\| \ll \lambda_1(\mathcal{L})$. Although $\|\mathbf{v}-\mathbf{t}\| \approx \lambda_1(\mathcal{L})$ by average (from Gaussian heuristic), if we choose many different target vectors $\mathbf{t}_1, ..., \mathbf{t}_l$, we may get at least one lattice point $\mathbf{v}_i$ such that $\|\mathbf{v}_i - \mathbf{t}_i\| \ll \lambda_1(\mathcal{L})$.

The discussion above also holds for the comparison between solving SIS and mul-ISIS, which means that solving mul-ISIS using the same block size in BKZ may result in a shorter solution than solving SIS. However, we note that randomized slicer is not designed to solve approximate-CVP, especially when the lattice dimension is large, it is impossible to list all the shortest vectors for slicing. Here we modify the algorithm into solving γ-CVP with $\gamma > 1$.

The algorithm runs in the following steps:

- Run BKZ-β on the lattice to generate a reduced lattice basis.
- Using sieving to generate short lattice vectors on each β-dimension sublattice of $\mathcal{L}$, and lift them into vectors of $\mathcal{L}$, the vector set is denoted by L, and we suppose that $|L| = l = \alpha^d$.
- Use randomized slicer on the target vector $\mathbf{t}$ with L to generate a close lattice vector $\mathbf{u}$.

We suppose that BKZ-β outputs a vector which length is $\gamma'\lambda_1(\mathcal{L})$, and we heuristically assume that vectors in L have the lengths of approximately γ' times of the lengths of the l shortest vectors. So the approximate Voronoi cell expanded by L has the size of γ'^d times of the Voronoi cell expanded by the l shortest vectors. Adopting the heuristic assumption in [9], we have:

$$\mathcal{V}_L \leq \gamma'^d \left(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16}\right)^{d/2+o(d)} \cdot \det(\mathcal{L}).$$

We assume that it is a good estimation, such that $\leq$ is replaced by $\approx$.

By Gaussian Heuristic, a d-dimension ball with radius $\lambda_1(\mathcal{L})$ is approximately $\det(\mathcal{L})$, which means that a d-dimension ball with radius $\gamma\lambda_1(\mathcal{L})$ is approximately $\gamma^d \det(\mathcal{L})$. So the running time of randomized slicer can be estimated by $(\gamma'/\gamma)^d(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16})^{d/2+o(d)}$.

So the time complexity of solving γ-CVP using randomized slicer is $O(2^{0.292\beta})+O(\alpha^d)+(\gamma'/\gamma)^d(\frac{16\alpha^4(\alpha^2-1)}{-9\alpha^8+64\alpha^6-104\alpha^4+64\alpha^2-16})^{d/2+o(d)}$, where γ' is determined by β (which can be deduced from a BKZ simulator, see [8] for details).

We consider a specific case, where $\gamma = 1$, and we choose $\alpha = (4/3)^{0.95/2}$. We turn SIS into solving SVP and mul-ISIS into solving approximate-CVP, and we give the estimated time cost by the approximation above in the figure below. We can see that solving mul-ISIS takes less time than solving SIS when $n > 50$. However, for $\gamma \gg 1$, the randomized slicer may takes much more time than BKZ, since α is therefore small.

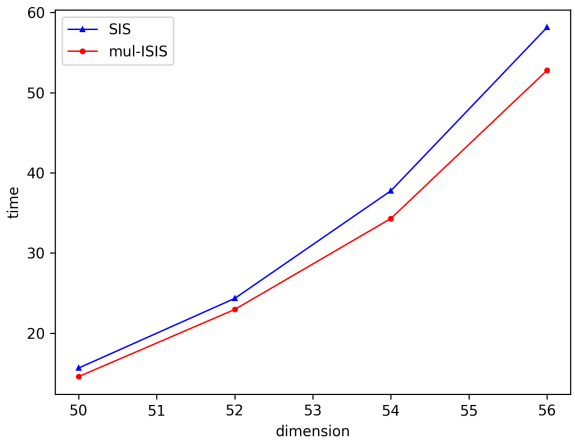

Fig. 2: Estimation for running time of solving SIS and mul-ISIS.

In our experiments, lattice sieving can only output a vector of length $\approx 1.05gh(\mathcal{L})$, while randomized slicer can easily find a close vector of length $< gh(\mathcal{L})$, so for $\gamma < 1.05$, mul-ISIS is obviously easier to solve. However, if we choose $\gamma > 5$, even the running time of randomized slicer is 10 times than that of BKZ, the distance between $\mathbf{t}$ and the output lattice vector $\mathbf{u}$ is still larger than the average length of vectors in L, so solving approximate-CVP with large approximation factor γ using randomized slicer turns out to be much slower than solving approximate-SVP. We leave an open problem here to design more efficient approximate-CVP and mul-ISIS solving algorithms.

6 Conclusion and Future Work

In this paper, we revisited the concrete hardness for the hardness assumption SelfTargetMSIS in the NIST post-quantum standard CRYSTALS-Dilithium. We pointed out that, although the authors used MSIS to estimate the hardness of SelfTargetMSIS, it is in fact not the case, as we introduced an algorithm which solves SelfTargetMSIS faster than MSIS under specific parameters. We also analyse this problem in a theoretical way, and introduced new assumptions

called mul-(M)ISIS and sel-(M)ISIS, where sel-(M)ISIS is a hardness assumption related to SelfTargetMSIS. We give the reduction from mul-(M)ISIS to (M)ISIS, but show that it is hard to reduce sel-(M)ISIS to standard (M)SIS in a same way. We also discuss the concrete hardness for mul-(M)ISIS using an estimation of approximate-CVP solver.

Although we solved some of the questions, we rise more questions in this paper: (1) What is the most efficient algorithm in solving approximate-CVP or ISIS? (2) How to construct a more efficient algorithm in solving sel-ISIS than ISIS, especially for the selection set corresponds with SelfTargetMSIS (see Section 5.1)? (3) Are the parameters of Dilithium still secure if we consider the hardness of sel-ISIS instead of SIS? We hope that these questions could be answered in the near future.

References

1. Alfonso Francisco De Abiega-L'Eglisse, Kevin Andrae Delgado-Vargas, Fernando Quetzalcoatl Valencia-Rodriguez, Víctor González Quiroga, Gina Gallegos-García, and Mariko Nakano-Miyatake. Performance of new hope and crystals-dilithium postquantum schemes in the transport layer security protocol. *IEEE Access*, 8:213968–213980, 2020.
2. Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 601–610. ACM, 2001.
3. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 717–746. Springer, 2019.
4. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016.
5. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 789–819. Springer, 2016.
6. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 10–24. SIAM, 2016.

7. Anja Becker, Nicolas Gama, and Antoine Joux. Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. *IACR Cryptol. ePrint Arch.*, page 522, 2015.
8. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.
9. Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate voronoi cells. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2019.
10. Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *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 I*, volume 10820 of *Lecture Notes in Computer Science*, pages 125–145. Springer, 2018.
11. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013.
12. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.
13. Léo Ducas, Thijs Laarhoven, and Wessel P. J. van Woerden. The randomized slicer for CVPP: sharper, faster, smaller, batchier. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2020.
14. Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within mordell's inequality. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 207–216. ACM, 2008.
15. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.
16. Denisa O. C. Greconici, Matthias J. Kannwischer, and Daan Sprenkels. Compact dilithium implementations on cortex-m3 and cortex-m4. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):1–24, 2021.
17. Louis Claude Guillou and Jean Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Proceedings on Advances in cryptology*, 1988.
18. Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. Speed-ups and time-memory trade-offs for tuple lattice sieving. In Michel Abdalla and Ricardo Dahab,

editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 407–436. Springer, 2018.

19. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *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*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.
20. Elena Kirshanova and Thijs Laarhoven. Lower bounds on lattice sieving and information set decoding. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 791–820. Springer, 2021.
21. Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2016.
22. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.
23. A.K. Lenstra, H.W. Lenstra, and L Lovsz. Factoring polynomials with rational coefficients.
24. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.
25. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.
26. Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1468–1480. SIAM, 2010.
27. Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 276–294. SIAM, 2015.
28. Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.*, 2(2):181–207, 2008.
29. C. Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, 1989.
30. Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In Lothar Budach, editor, *Fundamentals of Computation Theory, 8th International Symposium, FCT '91, Gosen,*

Germany, September 9-13, 1991, Proceedings, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991.
31. Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM J. Discret. Math.*, 23(2):715–731, 2009.

A The Comparison between solving $\mathbf{MSIS}_{m,k,\gamma}$ and $\mathbf{MSIS}_{m,k+1,\gamma}$

We compare the concrete hardness of $\mathrm{MSIS}_{m,k,\gamma}$ and $\mathrm{MSIS}_{m,k+1,\gamma}$. By the discussion in Section 3, the naive approach for solving $\mathrm{MSIS}_{m,k,\gamma}(\mathbf{A})$ is constructing a lattice $\mathcal{L} = \mathcal{L}(\begin{bmatrix} q\mathbf{I} & M(\mathbf{A}) \\ \mathbf{0} & -\mathbf{I} \end{bmatrix})$, where $M(\mathbf{A})$ is generated by expanding $\mathbf{A}$ into a $nm \times nk$ integer matrix, and solve (approximate)-SVP on this lattice. Similarly, solving $\mathrm{MSIS}_{m,k,\gamma}(\mathbf{A}|\mathbf{t})$ can be turned into solving (approximate)-SVP on a lattice $\mathcal{L}' = \mathcal{L}(\begin{bmatrix} q\mathbf{I} & M(\mathbf{A}|\mathbf{t}) \\ \mathbf{0} & -\mathbf{I} \end{bmatrix})$.

However, it is easy to see that a solution to $\mathrm{MSIS}_{m,k,\gamma}(\mathbf{A})$ is also a solution to $\mathrm{MSIS}_{m,k+1,\gamma}(\mathbf{A}|\mathbf{t})$ by appending 0, so if finding a short vector in $\mathcal{L}$ takes less time than finding a short vector in $\mathcal{L}'$, we can say that solving $\mathrm{MSIS}_{m,k+1,\gamma}$ is as hard as solving $\mathrm{MSIS}_{m,k,\gamma}$. Now we discuss the condition which the statement above holds. Since the fastest approximate SVP-solver is BKZ, we only need to show that finding a γ-length vector in $\mathcal{L}'$ needs a larger block size then finding a γ-length vector in $\mathcal{L}$.

In [8], the authors estimated the quality of a BKZ-reduced lattice basis, in the term of Hermite factor [14], which is defined as:

Definition A.1. *For a d-dimension lattice $\mathcal{L}$, let $\mathbf{B}$ be a basis of $\mathcal{L}$, and $\mathbf{b}_1$ be the shortest vector in $\mathbf{B}$. Then, the Hermite factor of $\mathbf{B}$ δ_0 is defined as:*

$$\delta_0(\mathbf{B})^d = \frac{\|\mathbf{b}_1\|}{\det(\mathcal{L})^{1/d}}.$$

The authors claimed that the Hermite factor of a β-BKZ-reduced lattice basis $\mathbf{B}$ can be estimated by:

$$\delta_0(\mathbf{B}) \approx (\frac{(\pi\beta)^{1/\beta}\beta}{2\pi e})^{\frac{1}{2(\beta-1)}}.$$

We suppose that the problem $\mathrm{MSIS}_{m,k,\gamma}$ can be solved by performing β-BKZ on $\mathcal{L}$, and the β-BKZ reduced basis is $\mathbf{B}$. We can see that the dimension of $\mathcal{L}$ is $n(m+k)$, and $\det(\mathcal{L}) = q^{nm}$, thus $gh(\mathcal{L}) = \sqrt{\frac{n(m+k)}{2\pi e}} q^{m/(m+k)}$. Let $\gamma = \tilde{\gamma} gh(\mathcal{L})$, so $\delta_0(\mathbf{B}) \approx (\tilde{\gamma}\sqrt{\frac{n(m+k)}{2\pi e}})^{\frac{1}{n(m+k)}}$.

Similarly, suppose that $\mathrm{MSIS}_{m,k+1,\gamma}$ can be solved by performing β'-BKZ on $\mathcal{L}'$ and the β'-BKZ reduced basis of $\mathcal{L}$ is $\mathbf{B}'$. The dimension of $\mathcal{L}'$ is $n(m+k+1)$, and $\det(\mathcal{L}) = q^{nm}$. So $\delta_0(\mathbf{B}') \approx (\tilde{\gamma}\sqrt{\frac{n(m+k)\cdot q^{\frac{m}{(m+k)(m+k+1)}}}{2\pi e}})^{\frac{1}{n(m+k+1)}}$.

We see that δ_0 is only related to the block size, and since $\beta \leq \beta'$, we have that $\delta_0(\mathbf{B}) \geq \delta_0(\mathbf{B}')$, which means that:

$$(\tilde{\gamma}\sqrt{\frac{n(m+k)}{2\pi e}})^{\frac{1}{n(m+k)}} \gtrapprox (\tilde{\gamma} \cdot q^{\frac{m}{(m+k)(m+k+1)}}\sqrt{\frac{n(m+k)}{2\pi e}})^{\frac{1}{n(m+k+1)}}.$$

We take logarithm on both side, after simplification, we get:

$$(m+k+1)(\log\tilde{\gamma} + \log\sqrt{\frac{n(m+k)}{2\pi e}}) \gtrapprox m\log q.$$

Now, we see that if $k > \frac{m\log q}{\log\tilde{\gamma}+\log\sqrt{\frac{n(m+k)}{2\pi e}}} - m - 1$, then the above condition holds, and solving $\text{MSIS}_{m,k+1,\gamma}$ is as hard as solving $\text{MSIS}_{m,k,\gamma}$. We can further check that the parameter in Section 4.2 satisfies the inequation, we omit the details here.