

BiT

Bimodal Triangular distribution based lattice signatures

Principal Submitter: Xianhui Lu^{1,2}, +86-15010131069, luxianhui@iie.ac.cn

Contributors:

- **Design:** Yongjian Yin^{1,2}, yinyongjian@iie.ac.cn

Yijian Liu^{1,2}, liuyijian@iie.ac.cn

Dongshu Cai^{1,2}, caidongshu@iie.ac.cn

- **Analysis:** Dingding Jia^{1,2}, jiadignding@iie.ac.cn

Lei Bi^{1,2}, bilei@iie.ac.cn

Yamin Liu³, liuyamin3@huawei.com

Bing Shi^{1,2}, shibing@iie.ac.cn

Jingnan He^{1,2}, hejingnan@iie.ac.cn

- **Implementation:** Ya Shi^{1,2}, shiya@iie.ac.cn

Hang Zhang^{1,2}, zhanghang2024@iie.ac.cn

Ying Liu^{1,2}, liuying@iie.ac.cn

Yu Zhang^{1,2}, zhangyu@iie.ac.cn

Organizations: 1. State Key Laboratory of Cyberspace Security Defense,

Institute of Information Engineering, CAS

2. University of Chinese Academy of Sciences

3. Huawei Technologies

Owner: Xianhui Lu

Postal Address: No. 19, Shucun Road, Beijing 100093, China

Contact: liuying@iie.ac.cn, +86-18633352026

Contents

1	Introduction	4
1.1	Innovation	4
1.2	Design Overview	5
1.3	Security and Results	8
2	Preliminaries	9
2.1	Notations	9
2.2	Digital Signature Scheme	10
2.3	Lattice-Based Hard Problems	10
2.4	Triangular Distribution and Rejection Sampling	11
2.5	Challenge Sampling	12
2.6	High/Low Bits Extraction	13
3	Algorithm Description	13
3.1	Compression Technique	13
3.1.1	Signature Compression by hint	13
3.1.2	Partitioned Rejection Sampling	13
3.1.3	Public Key Compression	14
3.2	Scheme Description	14
3.2.1	Rejection Conditions	15
3.3	Correctness	15
3.4	Rejection Counts	16
3.5	Pseudocode Description	17
3.6	NTRU Version	17
4	Design Rationale	17
4.1	Design Principles and Hard Problems	17
4.1.1	Design Principles	17
4.1.2	Mathematical Hard Problems	20
4.2	Criterion for Parameter Selection	20
4.3	The post-quantum security for the entropy of the challenge	21
4.3.1	Grover Search Algorithm	22
4.3.2	Quantum Circuit Depth	22
4.3.3	Quantum Security Corresponding to Different Challenge Entropies	22
5	Security Statements and Analysis	23
5.1	Theoretical Security Analysis	23
5.2	Practical Security	27
5.3	Security against physical attacks	29

6	Performance Evaluation	30
6.1	Performance Analysis	30
6.2	Performance Test	31
6.2.1	x86 Platforms	31
6.2.2	Performance Test Results	32
6.3	Parameter, Security, and Size Summary	32
7	Feature Statements	34

List of Figures

1	The Graph of Function f, F, G	6
2	BiT (framework)	8
3	RejectSample	12
4	Compressed BiT	15
5	Pseudocode of BiT	18
6	BiT (NTRU Version)	19
7	Fiat-Shamir with Abort	19
8	Rejection Sampling in BiT	19
9	The Security Games	24

List of Tables

1	Efficiency, Security and Size	9
2	Function Declaration	17
3	Parameter Settings	21
4	MAXDEPTH	22
5	Quantum Security for Different Challenge Entropies	23
6	Practical Security	28
7	Efficiency and Size of SM3-Based Implementation	32
8	Efficiency of FIPS202-Based Implementation	33
9	Parameter, Security, Size (128,256,512-bit security level)	33
10	Parameter, Security, Size (80,192,384-bit security level)	34

1 Introduction

We propose a lattice-based digital signature scheme, BiT. The scheme is based on the Fiat-Shamir with Abort paradigm, and its security relies on the Module Short Integer Solution (MSIS) and Module Learning With Errors (MLWE) assumptions. We provide six parameter sets for different security levels. The scheme supports 128-bit, 256-bit, and 512-bit security, and also includes 80-bit, 192-bit, and 384-bit security levels for other application scenarios. We also provide a description of an NTRU cryptosystem-based variant.¹

1.1 Innovation

This work improves rejection sampling for Fiat-Shamir lattice signatures by providing better masking while keeping the implementation simple and efficient. Our rejection sampling procedure uses a bimodal structure of the form $\mathbf{y} + (-1)^b \mathbf{v}$, where the masking vector $\mathbf{y}$ is sampled from a triangular distribution.² This design retains the low rejection rate of bimodal Gaussian techniques, but avoids floating-point arithmetic and Gaussian sampling. As a result, it admits a simple constant-time implementation and provides stronger protection against side-information leakage.

Overall, the signature algorithm uses a masking distribution to hide the secret terms through a masking method, producing the source distribution. Rejection sampling then converts the source distribution into a target distribution that depends only on public information. Improvements to the signature algorithm can be summarized in the following aspects:

- **Simple and efficient masking distribution.** To avoid floating-point operations, we use a triangular masking distribution. Compared with discrete Gaussian sampling, triangular sampling is simpler and easier to implement in constant time. We further combine it with a bimodal structure to reduce the rejection rate.
- **Optimized target distribution design.** Previous rejection-sampling methods usually use a target distribution with the same shape as the masking distribution. In contrast, we study the target-distribution selection problem and optimize it for masking effectiveness. We prove that, for triangular masking, a trapezoidal target distribution maximizes the masking effectiveness. We further show that, within the bimodal framework, this triangular-to-trapezoidal construction provides stronger masking than the standard Gaussian-to-Gaussian approach.
- **Efficient rejection sampling procedure.** Finally, we provide a simple and efficient algorithm for performing rejection sampling from the triangular source distribution to the trapezoidal target distribution.

¹Compared with the MLWE-based BiT scheme, it offers no advantage in size or performance. Therefore, only the scheme description is provided, without corresponding parameter sets.

²The triangular distribution is obtained as the difference of two uniformly sampled equal-length integers. It can therefore be implemented using only uniform sampling, retaining the simplicity and side-channel robustness of uniform samplers while providing a smoother masking distribution.

1.2 Design Overview

Fiat-Shamir with Abort: The Fiat-Shamir with Abort paradigm converts a three-pass identification protocol into a signature scheme using the Fiat-Shamir transformation. It was first applied to lattice cryptography in [21, 22]. To prevent the signature from leaking information about the secret key, an abort mechanism is used to shape the signature distribution.

Taking SIS-based schemes as an example, key generation follows the construction of SIS instances. The signing key is a small-norm matrix $\mathbf{S}$, and the verification key is $(\mathbf{A}, \mathbf{T} = \mathbf{AS} \bmod q)$, where $\mathbf{A}$ is a uniformly random matrix modulo q .

To sign a message μ , the signer computes a signature $(\mathbf{z} = \mathbf{y} + \mathbf{Sc}, \mathbf{c})$. Here, $\mathbf{y}$ is a masking vector that hides the secret term $\mathbf{Sc}$, and the challenge $\mathbf{c}$ is obtained by applying the random oracle RO to $\mathbf{Ay} \bmod q \parallel \mu$. Before the signature is output, rejection sampling is used to make the signature distribution independent of the secret key.

To verify the signature, the verifier first checks that $\mathbf{z}$ is a short vector. It then verifies that $\mathbf{c} = \mathbf{H}(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$, where $\mathbf{H}$ denotes the random oracle.

Bimodal Technique In rejection sampling for $\mathbf{z} := \mathbf{y} + \mathbf{Sc}$, the term $\mathbf{v} := \mathbf{Sc}$ can be viewed as an offset applied to $\mathbf{y}$. Rejection sampling essentially removes this offset. Smaller offsets lead to simpler sampling and lower rejection rates.

The bimodal technique generates $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{Sc}$ with $b \leftarrow \{0, 1\}$. The two symmetric offsets $\pm \mathbf{Sc}$ partially cancel each other, effectively reducing the overall offset.

However, the additional randomness introduced by b breaks the standard verification equation. To complete the verification, it needs to modify the key generation such that $\mathbf{AS} = \hat{q}\mathbf{I}$, where $\hat{q} = 2^{-1} \bmod q$, and modifying the verification equation to $\mathbf{c} = \mathbf{H}(\text{High}(\mathbf{Az} + \hat{q}\mathbf{c}), \mu)$, where High extracts the high-order bits of the input³. Under this setting, we obtain

$$\begin{aligned} \mathbf{Az} + \hat{q}\mathbf{c} &= \mathbf{A}[\mathbf{y} + (-1)^b \mathbf{Sc}] + \hat{q}\mathbf{c} \\ &= \mathbf{Ay} + [(-1)^b + 1]\hat{q}\mathbf{c} = \mathbf{Ay} + (1 - b)\mathbf{c} \end{aligned}$$

The second equation follows from $\mathbf{AS} = \hat{q}\mathbf{I}_n$. Considering that when $b = 0$, $((-1)^b + 1)\hat{q} = 2\hat{q} = 1$, and when $b = 1$, $((-1)^b + 1)\hat{q} = 0$, we obtain the third equation. Since $\mathbf{c}$ is sufficiently small, $\text{High}(\mathbf{Ay} + (1 - b)\mathbf{c})$ is nearly identical to $\text{High}(\mathbf{Ay})$. Accordingly, the signing algorithm queries RO on $\text{High}(\mathbf{Ay}) \parallel \mu$. Finally, the signing procedure adds the condition $\text{High}(\mathbf{Ay} + (1 - b)\mathbf{c}) = \text{High}(\mathbf{Ay})$ to ensure correct verification.

Rejection Sampling Optimization: We optimize the masking distribution, the target distribution, and the execution process.

- (1) **Masking Distribution-Triangular Distribution.** BiT adopts a Gaussian-like yet easily constructed distribution, aiming to retain the advantages of Gaussian sampling while avoiding floating-point operations. Specifically, our scheme uses the triangular distribution to sample the masking term. The sampler uniformly selects two integers of fixed length

³This technique originates from the NTRU+Sign signature scheme [29]. In the original bimodal scheme BLISS [8], the key relation is $\mathbf{AS} = q\mathbf{I}_n \bmod 2q$. We wish to retain $\bmod q$ instead of $\bmod 2q$.

and outputs their difference. Thus, it has implementation complexity and side-channel resistance comparable to those of uniform sampling.

(2) Target Distribution-Trapezoidal Distribution close to Source Distribution.

In essence, rejection sampling eliminates the discrepancy between the source and target distributions. Therefore, choosing a target distribution that better matches the source distribution can reduce the rejection rate. BiT adopts a trapezoidal distribution that more closely matches the source distribution.

In BiT, the source distribution is $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{v}$, where $\mathbf{y}$ is the masking distribution follows a triangular distribution, $\mathbf{v}$ is secret term and $b \leftarrow \{0, 1\}$ is introduced by the bimodal technique. Let $\|\mathbf{v}\|_\infty = \beta_1$. For the one-dimensional case with $0 < \beta_1 < \gamma$, define the triangular distribution is $f(x, \gamma)$,

$$f(x, \gamma) = \begin{cases} \frac{\gamma - |x|}{\gamma^2}, & x \in [-\gamma, \gamma] \\ 0, & \text{else} \end{cases}$$

Then, the source distribution is $F(x, \gamma, \beta_1) = f(x - \beta_1, \gamma)/2 + f(x + \beta_1, \gamma)/2$. Next, we construct the target distribution using public information. Let $\max_v \|\mathbf{v}\|_\infty = \beta_2$, which is public. For $0 < \beta_1 < \beta_2 < \gamma$, define the target distribution is,

$$G(x, \gamma, \beta_2) = \begin{cases} f(x - \beta_2, \gamma)/2 + f(x + \beta_2, \gamma)/2, & x \in [-\beta_2, \beta_2] \\ f(x, \gamma), & \text{else} \end{cases}$$

The masking distribution $f(x, \gamma)$, the source distribution $F(x, \gamma, \beta_1)$, and the target distribution $G(x, \gamma, \beta_2)$ are illustrated in Figure 1, corresponding to the red, green, and blue curves, respectively. The green dashed curves represent $f(x - \beta_1, \gamma)$ and $f(x + \beta_1, \gamma)$, and their superposition forms the distribution F .

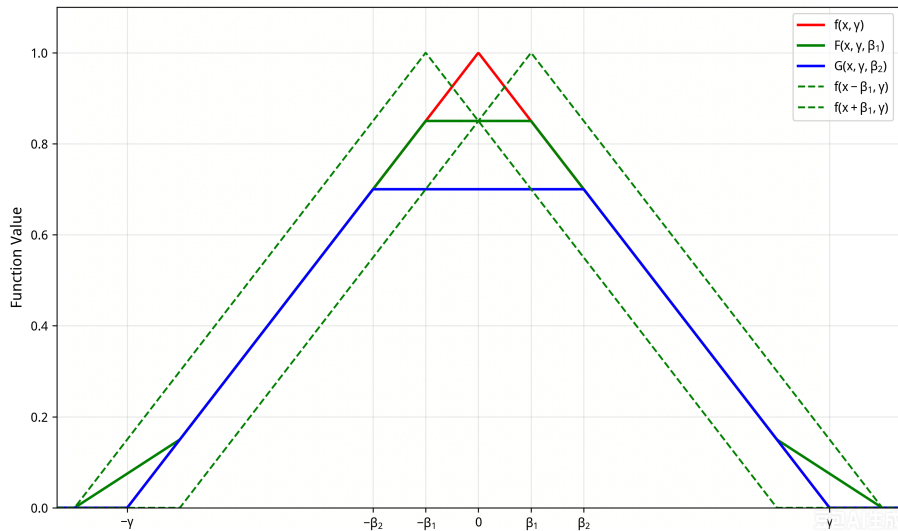


Figure 1: The Graph of Function f, F, G

The rejection sampling algorithm samples from F and outputs samples following G (which is not normalized here). The rejection rate equals the uncovered area between the two distributions and is β_2^2/γ^2 .

(3) Execution Process.

The proposed rejection sampling transforms the distribution corresponding to F into that of G . In practice, the algorithm first checks the norm of the input term. If it lies in the interval $[-\gamma + \beta_1, -\beta_2] \cup [\beta_2, \gamma - \beta_1]$ (i.e.: the overlapping part of F and G in Figure 1), the sample is accepted directly without rejection. Since $\gamma \gg \beta_i$ for $i \in \{1, 2\}$, this interval contains most samples, and only a small fraction require additional processing, which significantly reduces computational overhead.

For the remaining samples, the algorithm computes the rejection probability and performs acceptance using random coin flipping. Our rejection probability computation avoids floating-point number. Moreover, as discussed above, all secret terms with norm in $[0, \beta_2]$ have the same rejection rate β_2^2/γ^2 . In contrast, when Gaussian masking distributions are used, the rejection rate depends on the size of the secret term. Thus, our rejection sampling provides stronger side-channel resistance.

Polynomial Ring: To further reduce the size of public keys and signatures, the proposed scheme is constructed over polynomial rings. The signing key $\mathbf{S}$ and verification key $\mathbf{T}$, which are matrices in the original paradigm, are replaced by polynomial vectors.

Within the bimodal framework, the signing key is defined as a short-norm polynomial vector $\mathbf{s} = (1, \mathbf{s}_0, \mathbf{e})$. The verification key is defined as a polynomial matrix $\mathbf{A} = [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0, \mathbf{I}]$, where $\mathbf{j}_n$ is a vector whose first entry is 1 and all other entries are 0, $\mathbf{A}_0$ is a uniformly random matrix, and $\mathbf{b} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}$.

The public key does not need to store the entire matrix $\mathbf{A}$. Instead, it only stores the seed of $\mathbf{A}_0$ and the vector $\mathbf{b}$.

Modulus selection for arithmetic efficiency: We use an NTT-friendly prime q to enable efficient ring operations. The scheme performs all ring arithmetic using the Number Theoretic Transform (NTT) and uses a fully split polynomial ring to improve efficiency. For example, if the multiplicative group contains an element of order 256, a suitable prime such as $q = 36353$ allows the polynomial ring to split completely.

Overview of the Scheme: Based on the above design principles, the overall framework of the proposed scheme over $\mathcal{R}_q$ is shown in Figure 2.

- **KGen:** The key generation algorithm randomly selects two short-norm secret vectors $\mathbf{s}_0$ and $\mathbf{e}$, together with a public matrix $\mathbf{A}_0$. The signing key is $\mathbf{s} = [1, \mathbf{s}_0^T, \mathbf{e}^T]^T$, and the verification key is $\mathbf{A} = [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0, \mathbf{I}_n]$, which satisfies $\mathbf{A}_0\mathbf{S} = \hat{q}\mathbf{I}_n$. Here, $\hat{q} = 2^{-1} \bmod q$, $\mathbf{j}_n = (1, 0, \dots, 0)^T \in \mathcal{R}_q^n$, $\mathbf{b} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}$, and $\mathbf{I}_n$ is the $n \times n$ identity matrix.

KGen (1^κ)	Sign (pk, sk, μ)
1 : $\mathbf{s}_0, \mathbf{e} \xleftarrow{\$} S_\eta^m \times S_\eta^n \subset \mathcal{R}_q^m \times \mathcal{R}_q^n$	1 : $\mathbf{y} \xleftarrow{\$} T_{\gamma_1}^{m+n+1}$
2 : $\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n \times m}$	2 : $\mathbf{w} := \mathbf{A}\mathbf{y}$
3 : $\mathbf{b} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}$	3 : $c \in \mathcal{R}_q := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$
4 : $\mathbf{A} = [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0, \mathbf{I}_n]$	4 : $b \xleftarrow{\$} \{0, 1\}$
5 : $\mathbf{s} := (1, \mathbf{s}_0^T, \mathbf{e}^T)^T$	5 : $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$
6 : $\text{pk} := \mathbf{A}; \text{sk} := \mathbf{s}$	6 : $\mathbf{z} := \text{RejectSample}(\mathbf{z}, \gamma_1, \mathbf{s}c, \beta)$
	7 : if $\mathbf{z} = \perp$, go into step 1
Verify (pk, σ , μ)	8 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1
1 : if $\ \mathbf{z}\ _\infty > B_\infty$, then reject	9 : if $\ \mathbf{z}\ _\infty > B_\infty$, then restart
2 : $\hat{\mathbf{w}}' = \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c$	10 : return $\sigma := (\mathbf{z}, c)$
3 : $\mathbf{w}' = \text{HighBits}(\hat{\mathbf{w}}', \gamma_2)$	
4 : Accept if $c = \text{H}(\mathbf{w}', \mu)$	

Figure 2: BiT (framework)

- **Sign** : To sign a message μ , the signing algorithm samples a masking vector $\mathbf{y}$ from the distribution T_γ^{m+n+1} , where T_γ denotes the triangular distribution over $(-\gamma, \gamma)$.⁴ It then computes $\mathbf{A}\mathbf{y}$ and queries RO on $\text{High}(\mathbf{A}\mathbf{y}) \parallel \mu$ to obtain the challenge c . Here, c is an n -dimensional polynomial with exactly τ nonzero coefficients in $\{\pm 1\}$, and all remaining coefficients are 0. The function **High** extracts the high-order coefficients of $\mathbf{w}$. Next, the algorithm samples $b \leftarrow \{0, 1\}$ and computes $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}c$.

To ensure correctness and security, several rejection checks are applied. Rejection sampling **RejectSampling** is used to truncate the signature distribution so that the output is independent of secret information. The condition $\text{High}(\mathbf{w}) = \text{High}(\mathbf{w} + (-1)^b \mathbf{j}_n c)$ ensures correct verification while hiding the value of b , which encodes the offset direction of the secret term and must also be kept hidden from the adversary. The condition $\|\mathbf{z}\|_\infty \leq B_\infty$ ensures a bounded signature norm, since otherwise an adversary can construct large signatures that still pass the remaining checks. Finally, the algorithm outputs the signature $(\mathbf{z}, c)$.

- **Verify** : To verify a signature, the verification algorithm first checks whether $\|\mathbf{z}\|_\infty \leq B_\infty$. It then computes $\hat{\mathbf{w}}' = \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c$ and $\mathbf{w}' = \text{High}(\hat{\mathbf{w}}')$. Finally, the algorithm accepts the signature if $c = \text{H}(\mathbf{w}', \mu)$.

1.3 Security and Results

This section summarizes the efficiency, security strength and size of the scheme. Figure 1⁵ provides a concise overview of the results. The security of the scheme is measured in terms of **core-SVP**. The values in parentheses correspond to the security levels for strong unforgeability of the randomized signature scheme. The complete parameters and test results are provided in

⁴The triangular distribution is generated by uniformly sampling two fixed-BiT-length integers from $[0, \gamma)$ and outputting their difference.

⁵Performance results are based on the SM3-based implementation. Results for the FIPS 202-based implementation are presented in Section 6.2.2.

Table 1: Efficiency, Security and Size

Security Level	80	128	192	256	384	512
Efficiency AVX (clcyes)						
KGen	\	491240	\	1344984	\	4135509
Sign	\	1955243	\	2696404	\	12178299
Verify	\	436272	\	1369603	\	3996773
MLWE Hardness (Core-SVP)						
BKZ block-size b	348	443	659	877	1342	1752
Classical	101	129	192	256	392	512
Quantum	93	118	176	234	359	465
MSIS Hardness (Core-SVP)						
BKZ block-size (SUF)	346(277)	531(439)	816(658)	1031(877)	1523(1316)	2045(1754)
Classical (SUF)	101 (80)	155(128)	238(192)	301(256)	444(384)	597(512)
Quantum (SUF)	91 (73)	140(116)	216(174)	273(232)	403(348)	541(464)
Size (bytes)						
pk	505	1048	1293	2144	2435	5056
sig	1057	1504	2242	3456	5251	6695
pk+sig	1563	2552	3535	5600	7686	11751

2 Preliminaries

2.1 Notations

The operations mod and mod^+ are defined as follows: for $\forall a, q \in \mathbb{Z}$, it holds that $a \bmod q \in [-\frac{q}{2}, \frac{q}{2})$ and $a \bmod^+ l \in [0, l)$. For $n \in \mathbb{Z}$, denote $[n] = \{0, 1, \dots, n-1\}$. We use $[a, b]_{\mathbb{Z}}$ and $(a, b)_{\mathbb{Z}}$ to denote the discrete interval $[a, b] \cap \mathbb{Z}$ and $(a, b) \cap \mathbb{Z}$, respectively.

Vectors are represented by bold lowercase letters, e.g. $\mathbf{v}$ and matrices by bold uppercase letters, e.g. $\mathbf{A}$. All vectors are assumed to be column vectors. The symbols $\mathbf{v}^T$ and $\mathbf{A}^T$ denote the transpose of the vector $\mathbf{v}$ and the matrix $\mathbf{A}$, respectively. Additionally, $\mathbf{v}_i$ indicates the i -th component of $\mathbf{v}$.

$\mathbb{R}_{>0}$ denotes $(0, +\infty) \cap \mathbb{R}$. $\mathbb{Z}_q$ and $\mathcal{R}_q$ denote $[-\frac{q}{2}, \frac{q}{2}) \cap \mathbb{Z}$ and $\mathbb{Z}_q[x]/\langle x^d + 1 \rangle$, respectively. Let S_k be the set of polynomials in $\mathcal{R}_q$ whose coefficients lie in $[-k, k]$. Let $\mathbf{j}_n = (1, 0, \dots, 0)^T \in \mathcal{R}_q^n$. The coefficient vector of a polynomial $f(x) = a_0 + a_1x + \dots + a_{d-1}x^{d-1} \in \mathcal{R}_q$ is written as $(a_0, a_1, \dots, a_{d-1})^T \in \mathbb{Z}_q^d$.

The ℓ_p -norm of a vector $\mathbf{v}$ is represented as $\|\mathbf{v}\|_p$. For $\mathbf{v} = (\mathbf{v}_0, \dots, \mathbf{v}_{n-1})^T \in \mathbb{Z}_q^n$, the ℓ_p -norm is defined as $\|\mathbf{v}\|_p = (|\mathbf{v}_0|^p + \dots + |\mathbf{v}_{n-1}|^p)^{1/p}$. For $\mathbf{w} = (\mathbf{w}_0, \dots, \mathbf{w}_{n-1})^T \in \mathcal{R}_q^n$, $\|\mathbf{w}\|_p$ is defined as $[(\|\mathbf{w}_0\|_p)^p + \dots + (\|\mathbf{w}_{n-1}\|_p)^p]^{1/p}$. Unless otherwise stated, all operations in $\mathbb{Z}_q$ and $\mathcal{R}_q$ are performed modulo q .

For a distribution D , $x \xleftarrow{\$} D$ denotes sampling x from D . For a set S , $x \xleftarrow{\$} S$ denotes uniform sampling from S . All logarithms are base 2.

2.2 Digital Signature Scheme

Definition 1 (Digital Signature Scheme). *A digital signature scheme consists of three probabilistic polynomial-time (PPT) algorithms: (KGen, Sign, Verify).*

KGen: *Given 1^κ as input, generate the signing key sk and the verification key pk , where κ is the security parameter.*

Sign: *Given the signing key sk and a message μ as inputs, generate a signature σ for μ as output.*

Verify: *Given the verification key pk , a message μ , and a signature σ as inputs, output 0 to indicate rejection or 1 to indicate acceptance.*

Correctness. For all pairs $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(1^\kappa)$ and all messages μ , it always holds that $\text{Verify}(\text{pk}, \mu, \sigma) = 1$, where $\sigma \leftarrow \text{Sign}(\text{sk}, \mu)$.

Definition 2 (Unforgeability under Chosen Message Attack). *A digital signature scheme is called (t, ϵ, q_S, q_H) -UF-CMA secure (unforgeability under chosen message attack) if, for any PPT adversary $\mathcal{A}$ with running time t , q_S signature queries and q_H hash queries, there is:*

$$\Pr \left[\text{Verify}(\text{pk}, \mu, \sigma) = 1 \wedge \mu \notin M_{\text{Sign}} : (\mu, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_H}(\text{pk}) \right] \leq \epsilon,$$

where $\mathcal{O}_S$ and $\mathcal{O}_H$ denote the signing and hash oracles, which respond with signatures and hash values, respectively. M_{Sign} is the set of messages that have been queried to $\mathcal{O}_S$.

Based on the definition of UF-CMA security, other security notions can be classified as follows: If $q_S = 0$, it is referred to as (t, ϵ, q_H) -UF-NMA-secure (unforgeability under no message attack); If the adversary is allowed to forge a new signature for messages that have already been queried, the security notion is referred to as (t, ϵ, q_S, q_H) -SUF-CMA-secure (strong unforgeability under chosen message attack).

2.3 Lattice-Based Hard Problems

Definition 3 ($\text{MSIS}_{q,n,m,\beta}$ Problem). *The $\text{MSIS}_{q,n,m,\beta}$ problem is called (t, ϵ) -hard if, for any PPT adversary $\mathcal{A}$ with running time t ,*

$$\text{Adv}_{\mathcal{A}}^{\text{MSIS}} = \Pr[[\mathbf{I}|\mathbf{A}] \cdot \mathbf{s} = \mathbf{0} \wedge \|\mathbf{s}\|_\infty \leq \beta | \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m}; \mathbf{s} \in \mathcal{R}_q^{n+m} / \{\mathbf{0}\} \leftarrow \mathcal{A}(\mathbf{A})] \leq \epsilon$$

Definition 4 ($\text{MLWE}_{q,n,m,\phi,\varphi}$ instance). *For parameters q, n, m , two probability distributions ϕ and φ on $\mathcal{R}_q$, a $\text{MLWE}_{q,n,m,\phi,\varphi}$ instance is that, randomly selecting a matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m}$, a pair $(\mathbf{s}, \mathbf{e}) \in \phi^m \times \varphi^n$, and outputting $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$.*

Definition 5 (Decision $\text{MLWE}_{q,n,m,\phi,\varphi}$ Problem). *The decision $\text{MLWE}_{q,n,m,\phi,\varphi}$ problem is called (t, ϵ) -hard if, for any PPT adversary $\mathcal{A}$ with running time t ,*

$$\text{Adv}_{\mathcal{A}}^{\text{MLWE}} = |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\text{MLWE}}}(1^\kappa)] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\text{uniform}}}(1^\kappa)]| \leq \epsilon$$

where $\mathcal{O}_{\text{MLWE}}$ and $\mathcal{O}_{\text{uniform}}$ denote the MLWE and uniform oracles, respectively. The oracle $\mathcal{O}_{\text{MLWE}}$ outputs MLWE instances, while $\mathcal{O}_{\text{uniform}}$ returns a uniformly random pair $(\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{R}_q^{n \times m} \times \mathcal{R}_q^n$.

Definition 6 (The BimodalSelfTargetMSIS $_{q,n,m,\gamma}$ problem). *For parameters $q, n, m, \gamma \in \mathbb{N}$, the ring $\mathcal{R}_q$ and the hash function $H : \{0, 1\}^* \rightarrow \mathcal{C} \subset \mathcal{R}_q$, the BimodalSelfTargetMSIS $_{q,n,m,\gamma}$ problem is called (t, ϵ) -hard, if for any PPT adversary $\mathcal{A}$ with running time t ,*

$$\Pr \left[\begin{array}{l} 0 \leq \|\mathbf{y}\|_\infty \leq \gamma \\ H(\mathbf{A}\mathbf{y} + \hat{q}\mathbf{j}_n c, \mu) = c \end{array} \middle| \begin{array}{l} (\mathbf{A}_0, \mathbf{b}) \xleftarrow{\$} \mathcal{R}_q^{n \times m} \times \mathcal{R}_q^n; (\mathbf{y}, \mu, c) \leftarrow \mathcal{A}^{\mathcal{O}_H}(\mathbf{A}) \\ \mathbf{A} := [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0, \mathbf{I}_n] \end{array} \right] \leq \epsilon,$$

where $\mathbf{y} \in \mathcal{R}_q^{n+m+1}$, μ is a string and $c \in \mathcal{R}_q$, and $\mathcal{O}_H$ denotes the hash oracle. The oracle $\mathcal{O}_H$ outputs hash values for queries.

Definition 7 (NTRU $_{q,\phi}$ instance). *For parameters q , $\hat{q} = 2^{-1} \bmod q$, a probability distribution ϕ on $\mathcal{R}_q$, an NTRU $_{q,\phi}$ instance is that, randomly selecting two polynomial $\mathbf{f}, \mathbf{g} \xleftarrow{\$} \phi^2$, where $\mathbf{f}$ is invertible, and outputting $(\mathbf{g} + \hat{q})/\mathbf{f}$.*

Definition 8 (Decisional NTRU $_{q,\phi}$ problem). *The decisional NTRU $_{q,\phi}$ problem is (t, ϵ) -hard if, for any PPT adversary $\mathcal{A}$ with running time t ,*

$$\text{Adv}_{\mathcal{A}}^{\text{NTRU}} = |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\text{NTRU}}}(1^\kappa)] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\text{uniform}}}(1^\kappa)]| \leq \epsilon,$$

where the oracle $\mathcal{O}_{\text{NTRU}}$ replies with NTRU instances, and the oracle $\mathcal{O}_{\text{uniform}}$ replies with the uniform samples from $\mathcal{R}_q$. κ is the security parameter.

2.4 Triangular Distribution and Rejection Sampling

We adopt triangular distribution

$$f(x, \gamma) = \begin{cases} \frac{\gamma - |z|}{\gamma^2}, & z \in (-\gamma, \gamma)\mathbb{Z} \\ 0, & \text{else} \end{cases}$$

as the distribution of the masking term $\mathbf{y}$.

For the signature component $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{v}$, where $b \leftarrow \{0, 1\}$ and $\mathbf{v}$ is the secret term. We first define rejection sampling in one dimension, which simplifies the scenario to $\mathbf{z}_i := \mathbf{y}_i + (-1)^b \mathbf{v}_i$. Let $|\mathbf{v}_i| = \beta_1$ and $\max_v \|\mathbf{v}\|_\infty = \beta_2$. For $0 < \beta_1 < \beta_2 < \gamma$, define the source distribution

$F(x, \gamma, \beta_1)$ and the target distribution $G(x, \gamma, \beta_2)$ as:

$$F(z, \gamma, \beta_1) = f(z - \beta_1, \gamma)/2 + f(z + \beta_1, \gamma)/2$$

$$G(z, \gamma, \beta_2) = \begin{cases} f(z - \beta_2, \gamma)/2 + f(z + \beta_2, \gamma)/2, & z \in (-\beta_2, \beta_2)\mathbb{Z} \\ f(z, \gamma), & \text{else} \end{cases}$$

For $\mathbf{z}_i \in (-\gamma, \gamma)\mathbb{Z}$ and $0 < \beta_1 < \beta_2 < \gamma$, the rejection sampling algorithm **RejectSample** accepts $\mathbf{z}_i$ with probability $\frac{G(z, \gamma, \beta_2)}{F(z, \gamma, \beta_1)}$. And the heuristic rejection rate of single-coefficient rejection sampling equals β_2^2/γ^2 . Let $\hat{G}$ denote the normalized form of G , and let $T_{\hat{G}_{\gamma, \beta_2}}$ be the corresponding distribution induced by $\hat{G}$. Then the output of **RejectSample** follows the distribution $T_{\hat{G}_{\gamma, \beta_2}}$.

The algorithm naturally extends to polynomial vectors by applying rejection sampling coefficient-wise. The rejection sampling algorithm **RejectSample** is described as Figure 3.

RejectSample ($\mathbf{z}, \gamma, \mathbf{v}, \beta_2$)	
1 :	for i -th component $(\mathbf{z}_i, \mathbf{v}_i)$ of $(\mathbf{z}, \mathbf{v})$
2 :	$\beta_{1,i} := \mathbf{v}_i $
3 :	$z_i := \mathbf{z}_i$
4 :	return $\mathbf{z}_i$ with probability $G(z_i, \gamma, \beta_2)/F(z_i, \gamma, \beta_{1,i})$
5 :	else return $\perp$

Figure 3: **RejectSample**

2.5 Challenge Sampling

The adapted hash function maps the input to the ball B_τ . Here, B_τ is defined as a set of polynomials over $\mathcal{R}_q$, where each polynomial contains exactly τ coefficients of 1 and zeros for the remaining coefficients, with its size given by $|B_\tau| = \binom{d}{\tau}$.

The hash implementation in the proposed signature scheme consists of two steps. First, a hash function maps $\{0, 1\}^*$ to $\{0, 1\}^d$. This step enables a concise and compact representation of elements in B_τ , and its output is much shorter than that of the second step. Incorporating the hash value generated in the first step into the signature effectively reduces the overall signature size. In the second step, an XOF is utilized to map the output of the first stage to an element in B_τ for subsequent computation. The detailed description of the second-step function is presented as follows:

SampleInBall (ρ)	
01	initialize $\mathbf{c} = c_0 c_1 \dots c_{d-1} = 00 \dots 0$
02	for $i := d - \tau$ to $d - 1$
03	$j \leftarrow \{0, 1, \dots, i\}$
04	$c_i := c_j$
05	$c_j := 1$
06	return c

2.6 High/Low Bits Extraction

The function **HighBits** and **LowBits** extract the high and low bits of the input through modulo operation.

LowBits ($\mathbf{w}, \gamma$)	HighBits ($\mathbf{w}, \gamma$)
1 : return $\mathbf{w} \bmod \gamma$	1 : return $(\mathbf{w} - \text{LowBits}(\mathbf{w}))/\gamma$

3 Algorithm Description

3.1 Compression Technique

A description of the simplified version **BiT** has been provided in 2. Combined with the special structure of the signing and public keys in the **MLWE** setting, both the signature size and the public key size can be further reduced.

3.1.1 Signature Compression by hint

In the **BiT** signature scheme, $\mathbf{A} = [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0, \mathbf{I}_n]$ can be rewritten as $[\mathbf{A}_1, \mathbf{I}_n]$, where $\mathbf{A}_1 = [\hat{q}\mathbf{j}_n - \mathbf{b}, \mathbf{A}_0]$. Similarly, the signing key $\mathbf{s} := (1, \mathbf{s}_0^T, \mathbf{e}^T)^T$ can be decomposed as $(\mathbf{s}_1^T, \mathbf{s}_2^T)^T$, where $\mathbf{s}_1 = (1, \mathbf{s}_0^T)$ and $\mathbf{s}_2 = \mathbf{e}$.

Accordingly, the masking vector is split into $(\mathbf{y}_1, \mathbf{y}_2)^6$ to mask $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively. As a result, the signature format changes from $(\mathbf{z}, c)$ to $(\mathbf{z}_1, \mathbf{z}_2, c)$.

During verification, the core condition $c = \text{H}(\text{HighBits}(\mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_nc), \mu)$ is replaced with $c = \text{H}(\text{HighBits}(\mathbf{A}_1\mathbf{z}_1 + \mathbf{z}_2 + \hat{q}\mathbf{j}_nc), \mu)$. Since $\mathbf{z}_2$ is short, $\text{HighBits}(\mathbf{A}_1\mathbf{z}_1 + \mathbf{z}_2 + \hat{q}\mathbf{j}_nc)$ is almost identical to $\text{HighBits}(\mathbf{A}_1\mathbf{z}_1 + \hat{q}\mathbf{j}_nc)$ with differences arising only from carry and borrow errors introduced by $\mathbf{z}_2$. Therefore, we replace the full vector $\mathbf{z}_2$ with a compact hint $\mathbf{h}$, reducing the signature size by nearly half and outputting only $(\mathbf{z}_1, \mathbf{h}, c)$.

3.1.2 Partitioned Rejection Sampling

In the **MLWE** variant, the signing key has the form $(1, \mathbf{s}_0, \mathbf{e})$. The scheme uses $\mathbf{y}$ to mask the secret term. Since $\mathbf{s}_0$ and $\mathbf{e}$ follow the same distribution, they should be masked using random vectors of the same scale. In contrast, the constant term 1 has much smaller magnitude, allowing the corresponding masking component to be significantly shorter.

Exploiting this structure, we sample $\mathbf{y} = (\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2)$ to mask $(-1)^b \mathbf{s}c = ((-1)^b c, (-1)^b \mathbf{s}_0 c, (-1)^b \mathbf{e}c)$, where $\mathbf{y}_{1,0}$ is sampled from a smaller distribution.

An additional advantage is that most coefficients of $(-1)^b c$ are zero. Therefore, rejection sampling only needs to process the nonzero coefficients instead of all coefficients.

⁶The input to RO is changed to $\text{HighBits}(\mathbf{A}_1\mathbf{y}_1 + \mathbf{y}_2, \gamma_2) \parallel \mu$.

3.1.3 Public Key Compression

In practical deployment, only the seed ρ of $\mathbf{A}_0$ and the polynomial vector $\mathbf{b}$ are transmitted as the public key. To further reduce the public key size, we compress $\mathbf{b}$ by its high-order part. Specifically, the verification key is reconstructed as $[\hat{q}\mathbf{j}_n - \mathbf{b}_1 \| \mathbf{0}, \mathbf{A}_0, \mathbf{I}_n]$, where $\mathbf{b} = \mathbf{b}_1 \| \mathbf{b}_0$ and $\mathbf{b}_1$ denotes the high-order component of vector $\mathbf{b}$.

This modification affects the verification procedure. The computed value $\hat{\mathbf{w}}' = \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c$ changes from $\mathbf{w} + (1 - b)\mathbf{j}_n c$ to $\mathbf{w} + (1 - b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$, and verifier can not get the information about $(-1)^b \mathbf{b}_0 c$. In the following, we reformulate the signature procedure to complete verification, while keeping the output size minimal.

- **Directly include $\mathbf{b}_0$ in the signature:** Under this setting, the signature is $(\mathbf{z}, c, \mathbf{b}_0)$. Since $\mathbf{b}_0$ is public information, this does not introduce security risks. However, providing only $\mathbf{b}_0$ is insufficient for verification, as the verifier cannot recover the information about $(-1)^b$.
- **Use $(-1)^b \mathbf{b}_0$ as the signature component:** Under this setting, the signature is $(\mathbf{z}, c, (-1)^b \mathbf{b}_0)$. This enables full verification while preserving security. First, $(-1)^b \mathbf{b}_0$ follows the same distribution as $\mathbf{b}_0$. Moreover, in the joint distribution $(\mathbf{z}, c, (-1)^b \mathbf{b}_0)$, rejection sampling ensures that $\mathbf{z}$ is independent of b , and the random oracle ensures that c is independent. Hence, $(\mathbf{z}, c)$ is independent of $(-1)^b \mathbf{b}_0$, and the modified signature is distributionally equivalent to $(\mathbf{z}, c, \mathbf{b}_0)$, providing no additional advantage to the adversary.
- **Compress $(-1)^b \mathbf{b}_0$ using a hint:** Following the compression idea for $\mathbf{z}_2$, where a short hint $\mathbf{h}$ replaces the full vector $\mathbf{z}_2$, we similarly integrate $(-1)^b \mathbf{b}_0$ into the hint generation process to further reduce the signature size.

3.2 Scheme Description

The description of the Compressed BiT signature scheme is shown in Figure 4.

First, in **KGen**, the public key $\mathbf{b}$ is split due to the public key compression in Section 3.1.3 (step 4). Specifically, the algorithm decomposes $\mathbf{b}$ into a high-order part $\mathbf{b}_1$ and a low-order part $\mathbf{b}_0$, which are incorporated into the verification key and signing key, respectively. The term $\gamma_b \mathbf{b}_1 = \mathbf{b} - \mathbf{b}_0$ in the matrix $\mathbf{A}$ is essentially equivalent to zero-padding the low-order bits of $\mathbf{b}_1$.

In **Sign**, following the partitioned rejection sampling mechanism in Section 3.1.2, the algorithm independently samples the corresponding masking components $(\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2)$ for $(1, \mathbf{s}_0, \mathbf{e})$ (Step 1), and performs rejection sampling on the corresponding signature components $(\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2)$ (Steps 6–9), respectively. Subsequently, **Sign** generates the hint $\mathbf{h}$ for $\mathbf{z}_2$ and $(-1)^b \mathbf{b}_0$ to compress the signature (Steps 11–12). In addition, the norm verification of $\mathbf{z} := (\mathbf{z}_1, \mathbf{z}_2)$ is replaced with that of $(\mathbf{z}_1, \gamma_2 \mathbf{h})$ (Step 13). Considering that $\mathbf{h}$ can be regarded as $\mathbf{z}_2 \bmod \gamma_2$, this adjustment is effectively equivalent to the original norm check on $\mathbf{z}$.

In **Verify**, the algorithm uses the hint $\mathbf{h}$ to complete the signature verification (Step 3).

KGen (1^κ)	Sign ($\text{pk}, \text{sk}, \mu$)
1: $\mathbf{s}_0, \mathbf{e} \xleftarrow{\$} S_\eta^m \times S_\eta^n \subset \mathcal{R}_q^m \times \mathcal{R}_q^n$	1: $\mathbf{y} := (\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2) \xleftarrow{\$} T_{\gamma_{1,0}} \times T_{\gamma_{1,1}}^m \times T_{\gamma_{1,2}}^n$
2: $\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n \times m}$	2: $\mathbf{w} := \mathbf{A}\mathbf{y}$
3: $\mathbf{b} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}$	3: $c \in \mathcal{R}_q := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$
4: $(\mathbf{b}_1, \mathbf{b}_0) := (\text{HighBits}(\mathbf{b}, \gamma_b), \text{LowBits}(\mathbf{b}, \gamma_b))$	4: $b \xleftarrow{\$} \{0, 1\}$
5: $\mathbf{A} = [\hat{q}\mathbf{j}_n - \gamma_b\mathbf{b}_1, \mathbf{A}_0, \mathbf{I}_n] = [\mathbf{A}_1, \mathbf{I}_n]$	5: $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$
6: $\mathbf{s} := (1, \mathbf{s}_0, \mathbf{e})$	6: $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) \in \mathcal{R}_q \times \mathcal{R}_q^m \times \mathcal{R}_q^n := \text{Split}(\mathbf{z})$
7: $\text{pk} := \mathbf{A}; \text{sk} := (\mathbf{s}, \mathbf{b}_0)$	7: $\mathbf{z}_{1,0} := \text{RejectSample}(\mathbf{z}_{1,0}, \gamma_{1,0}, c, 1)$
Verify (pk, σ, μ)	8: $\mathbf{z}_{1,1} := \text{RejectSample}(\mathbf{z}_{1,1}, \gamma_{1,1}, \mathbf{s}_0c, \beta)$
1: if $\ (\mathbf{z}_1, \gamma_2\mathbf{h})\ _\infty > B_\infty$, then reject	9: $\mathbf{z}_2 := \text{RejectSample}(\mathbf{z}_2, \gamma_{1,2}, \mathbf{e}c, \beta)$
2: $\hat{\mathbf{w}}' = \mathbf{A}_1\mathbf{z}_1 + \hat{q}\mathbf{j}_nc$	10: if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_nc, \gamma_2)$, go into step 1
3: $\mathbf{w}' = \text{HighBits}(\hat{\mathbf{w}}', \gamma_2) + \mathbf{h}$	11: $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_nc + (-1)^b \mathbf{b}_0c$
4: Accept if $c = \text{H}(\mathbf{w}', \mu)$	12: $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$
	13: if $\ (\mathbf{z}_1, \gamma_2\mathbf{h})\ _\infty > B_\infty$, then restart
	14: return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$

Figure 4: Compressed BiT

3.2.1 Rejection Conditions

There are three rejection conditions in **Sign**: (1) $\mathbf{z} := \text{RejectSample}(\mathbf{z})$ (Steps 7–9); (2) $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_nc, \gamma_2)$ (Step 10); (3) $\|(\mathbf{z}_1, \gamma_2\mathbf{h})\|_\infty > B_\infty$ (Step 13). Their functions are explained here.

- (1) $\mathbf{z} := \text{RejectSample}(\mathbf{z})$: This condition ensures that $\mathbf{z}_1$ and $\mathbf{h}$ in the signature do not reveal information about the key $\mathbf{s}$, where $\mathbf{h}$ can be seen as a compression of $\mathbf{z}_2$.
- (2) $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_nc, \gamma_2)$: Without the signature compression techniques of Section 3.1, this check ensures normal signature verification, as verifiers cannot directly access b . In the modified scheme, where b is embedded in the hint $\mathbf{h}$, the check prevents adversaries from gaining information about b .

Specifically, b can take two values, 0 or 1, which lead to different distributions of $\mathbf{h}$. When $b = 0$, $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\mathbf{w} - \mathbf{z}_2 + \mathbf{j}_nc, \gamma_2)$, and when $b = 1$, $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\mathbf{w} - \mathbf{z}_2, \gamma_2)$. This check ensures that $\text{HighBits}(\mathbf{w}, \gamma_2) = \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_nc, \gamma_2)$, so that for $b = 0$, $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\mathbf{w} - \mathbf{z}_2 + \mathbf{j}_nc, \gamma_2) = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\mathbf{w} - \mathbf{z}_2, \gamma_2)$, matching the case when $b = 1$. This ensures consistency in $\mathbf{h}$ and prevents leakage of b .

- (3) $\|(\mathbf{z}_1, \gamma_2\mathbf{h})\|_\infty > B_\infty$: This step performs a norm check on $(\mathbf{z}_1, \gamma_2\mathbf{h})$, which can be viewed as a norm check of $\mathbf{z} := (\mathbf{z}_1, \mathbf{z}_2)$. We describe the criteria for setting B_∞ here. Since $\mathbf{z}$ and $\mathbf{y}$ have the same length, $\|\mathbf{z}_1\|_\infty$ is bounded by $\gamma_{1,1} - 1$. For $\gamma_2\mathbf{h}$, its infinite norm is $\gamma_2\|\mathbf{h}\|_\infty$. As a result, B_∞ is set as $\max\{\gamma_{1,1} - 1, \gamma_2\|\mathbf{h}\|_\infty\}$.

3.3 Correctness

Theorem 1. *The signature scheme described in Figure 4 is perfectly correct.*

Proof. According to the text $\|(\mathbf{z}_1, \gamma_2 \mathbf{h})\|_\infty > B_\infty$ in the step 13 of **Sign**, for the valid signature, $\|(\mathbf{z}_1, \gamma_2 \mathbf{h})\|_\infty \leq B_\infty$ always hold true. Thus, we only focus on the condition $\mathbf{c} = \mathbf{H}(\mathbf{w}', \mu)$ in **Verify**.

$$\begin{aligned}
\hat{\mathbf{w}}' &= \mathbf{A}_1 \mathbf{z}_1 + \hat{q} \mathbf{j}_n c \\
&= \mathbf{A}_1 [\mathbf{y}_1 + (-1)^b \mathbf{s}_1 c] + \hat{q} \mathbf{j}_n c + [\mathbf{y}_2 + (-1)^b \mathbf{s}_2 c] - \mathbf{z}_2 \\
&= (\mathbf{A}_1 \mathbf{y}_1 + \mathbf{y}_2) + (-1)^b (\mathbf{A}_1 \mathbf{s}_1 + \mathbf{s}_2) c + \hat{q} \mathbf{j}_n c - \mathbf{z}_2 \\
&= \mathbf{A} \mathbf{y} + [(-1)^b + 1] \hat{q} \mathbf{j}_n c + (-1)^b \mathbf{b}_0 c - \mathbf{z}_2 \\
&= \mathbf{A} \mathbf{y} - \mathbf{z}_2 + (1 - b) \mathbf{j}_n c + (-1)^b \mathbf{b}_0 c = \hat{\mathbf{w}}
\end{aligned}$$

The fourth equation comes from $\mathbf{A} \mathbf{s} = \hat{q} \mathbf{j}_n + \mathbf{b}_0$. Considering that when $b = 0$, $((-1)^b + 1) \hat{q} = 2 \hat{q} = 1$, and when $b = 1$, $((-1)^b + 1) \hat{q} = 0$, we obtain the fifth equation. Thus, $\mathbf{w}' = \text{HighBits}(\mathbf{w}, \gamma_2)$. This means that $\mathbf{c} = \mathbf{H}(\mathbf{w}', \mu)$. $\square$

3.4 Rejection Counts

Lemma 1. *In the scheme in Figure 4, the heuristic expected repetition time is less than $M = 1/\rho$, where $\rho \approx (1 - 1/\gamma_{1,0}^2)^\tau \cdot (1 - \beta^2/\gamma_{1,1}^2)^{dm} \cdot (1 - \beta^2/\gamma_{1,2}^2)^{dn} \cdot (1 - 1/\gamma_2)^{n\tau} > 0$ represents the probability of generating a valid signature within a single rewind, where d is the degree of the polynomial.*

Proof. According to the analysis in Section 3.2, rejections in **Sign** arise from two sources: (1) $\mathbf{z} := \text{RejectSample}(\mathbf{z})$ and (2) $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, with the first being the primary cause. Its rejection rate corresponds to the portion of F not covered by G . Thus, its acceptance rate is

$$P_1 = (1 - 1/\gamma_{1,0}^2)^\tau \cdot (1 - \beta^2/\gamma_{1,1}^2)^{dm} \cdot (1 - \beta^2/\gamma_{1,2}^2)^{dn}$$

Here, $(1 - 1/\gamma_{1,0}^2)^\tau$ comes from the rejection sampling for $\mathbf{z}_{1,0}$, where τ reflects that only τ non-zero terms in $(-1)^b c$ need masking; The term $(1 - \beta^2/\gamma_{1,1}^2)^{dm}$ and $(1 - \beta^2/\gamma_{1,2}^2)^{dn}$ corresponds to the rejection sampling for $(\mathbf{z}_{1,1}, \mathbf{z}_2)$, where m and n are the dimensionality of $(\mathbf{z}_{1,1}, \mathbf{z}_2)$ and d is the polynomial degree.

For the condition $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, the term $(-1)^b \mathbf{j}_n c$ affects only one bit, so the rejection rate for a single coefficient is $1/\gamma_2$. Since c has τ non-zero terms, we have

$$P_2 = (1 - 1/\gamma_2)^\tau$$

The acceptance rate of the signature algorithm is

$$P_1 \cdot P_2 = (1 - 1/\gamma_{1,0}^2)^\tau \cdot (1 - \beta^2/\gamma_{1,1}^2)^{dm} \cdot (1 - \beta^2/\gamma_{1,2}^2)^{dn} \cdot (1 - 1/\gamma_2)^\tau$$

$\square$

3.5 Pseudocode Description

The pseudocode of the BiT signature scheme is shown in Figure 5, with the adopted functions summarized in Table 2. We add a norm check on $\mathbf{h}$ at Step 22 of the signing procedure to reject a small fraction of oversized hints for storage space. The chosen bound B_h ensures that the rejection rate in this step is less than 7%.

H	Instantiated as <i>SM3</i>
SampleInBall	Maps a seed $\tilde{c}$ to a polynomial $c \in \mathcal{R}_q$
ExpandA	Maps a seed ρ to a matrix $\mathbf{A} \in \mathcal{R}_q^{m \times n}$ in NTT domain representation
ExpandS	Maps a seed $\rho' \in \{0, 1\}^\kappa$ to a pair $(\mathbf{s}_0, \mathbf{e}) \in S_\eta^m \times S_\eta^n$
ExpandMask	Maps a seed $seed_y$ and a counter value counter to $\mathbf{y} \in S_{\gamma_1}^m$

Table 2: Function Declaration

3.6 NTRU Version

The description of the BiT signature scheme (NTRU Version) is shown in Figure 6.

4 Design Rationale

This section first introduces the design principles of BiT and the underlying hard mathematical problems. It then describes the parameter design criteria that balance security and efficiency, and finally presents the rules for selecting the challenge entropy.

4.1 Design Principles and Hard Problems

4.1.1 Design Principles

This scheme follows the Fiat-Shamir with Abort paradigm. Under the random oracle (RO) model, it transforms an identification (ID) protocol into a signature scheme. A standard ID protocol consists of three rounds of interaction between the prover and the verifier. First, the prover sends a commitment w to the verifier. The verifier then returns a random challenge c , and the prover responds with z , forming the transcript (w, c, z) . The verifier checks the validity of the transcript and outputs 1 if it is valid, and 0 otherwise.

This transformation is naturally suited to number-theoretic signature schemes. In these schemes, the signer can completely hide the secret term using a masking vector, so the signature distribution does not leak information about the secret key. In contrast, the security of lattice-based schemes relies on the hardness of finding short lattice vectors. As a result, the masking vector must also be short, making it impossible to completely hide the secret key.

To prevent statistical attacks on the signature distribution, a rejection sampling mechanism is introduced to ensure that the final signature distribution is independent of the secret key. Compared with the standard ID protocol, the transformed scheme adds a response algorithm, $\text{Rep}_{\text{Abort}}$, in the third step. This algorithm selectively outputs the response z or aborts, thereby removing the dependence of the signature distribution on the secret key. Here, $\perp$ denotes an

KGen(1^κ)

```

1 :  $\rho, \rho' \leftarrow \{0, 1\}^\kappa$ 
2 :  $\mathbf{A}_0 \in \mathcal{R}_q^{n \times m} := \text{ExpandA}(\rho)$ 
3 :  $(\mathbf{s}_0, \mathbf{e}) \in S_\eta^m \times S_\eta^n := \text{ExpandS}(\rho')$ 
4 :  $\mathbf{b} := \mathbf{A}_0 \mathbf{s}_0 + \mathbf{e}$ 
5 :  $(\mathbf{b}_1, \mathbf{b}_0) := (\text{HighBits}(\mathbf{b}, \gamma_b), \text{LowBits}(\mathbf{b}, \gamma_b))$ 
6 :  $tr := H(\text{pk})$ 
7 : return  $(\text{pk}, \text{sk}) := ((\rho, \mathbf{b}_1), (\rho, \mathbf{b}_1, \rho', tr, \mathbf{s}_0, \mathbf{e}, \mathbf{b}_0))$ 

```

Sign(sk, M)

```

1 :  $\mathbf{A}_0 \in \mathcal{R}_q^{n \times m} := \text{ExpandA}(\rho)$ 
2 :  $\mathbf{A} := (\hat{q}\mathbf{j}_n - \gamma_b \mathbf{b}_1, \mathbf{A}_0, \mathbf{I}_n)$   $\hat{q} = 2^{-1} \bmod q$ 
3 :  $\mathbf{s} := (1, \mathbf{s}_0, \mathbf{e})$ 
4 :  $\mu \in \{0, 1\}^\kappa := H(tr || M)$ 
5 : counter := 0
6 :  $rnd \leftarrow \{0, 1\}^\kappa$ 
7 :  $seed_y \in \{0, 1\}^\kappa := \text{XOF}(\rho' || rnd || \mu)$ 
8 :  $\mathbf{y} := (\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2) \in T_{\gamma_{1,0}}^m \times T_{\gamma_{1,1}}^m \times T_{\gamma_{1,2}}^m := \text{ExpandMask}(seed_y, \text{counter})$ 
9 :  $\mathbf{w} := \mathbf{A}\mathbf{y}$ ; counter := counter +  $m$ 
10 :  $\mathbf{w}_1 := \text{HighBits}(\mathbf{w}, \gamma_2)$ 
11 :  $\tilde{c} \in \{0, 1\}^\kappa := H(\mu || \mathbf{w}_1)$ 
12 :  $c \in B_\tau := \text{SampleInBall}(\tilde{c})$ 
13 :  $b \leftarrow \{0, 1\}$ 
14 :  $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$ 
15 :  $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) := \text{Split}(\mathbf{z}) \in \mathcal{R}_q \times \mathcal{R}_q^m \times \mathcal{R}_q^n$ 
16 :  $\mathbf{z}_{1,0} := \text{RejectSample}(\mathbf{z}_{1,0}, \gamma_{1,0}, c, 1)$ 
17 :  $\mathbf{z}_{1,1} := \text{RejectSample}(\mathbf{z}_{1,1}, \gamma_{1,1}, \mathbf{s}_0 c, \beta)$ 
18 :  $\mathbf{z}_2 := \text{RejectSample}(\mathbf{z}_2, \gamma_{1,2}, \mathbf{e}c, \beta)$ 
19 : if  $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$ , go into step 8
20 :  $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1 - b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 
21 :  $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 
22 : if  $\|\mathbf{h}\|_\infty \geq B_h$ , go into step 8
23 : if  $\|(\mathbf{z}_1, \gamma_2 \mathbf{h})\|_\infty > B_\infty$ , go into step 8
24 : return  $\sigma := (\mathbf{z}_1, \mathbf{h}, \tilde{c})$ 

```

Verify($\text{pk}, M, \sigma = (\mathbf{z}_1, \mathbf{h}, \tilde{c})$)

```

1 :  $\mathbf{A}_0 \in \mathcal{R}_q^{n \times m} := \text{ExpandA}(\rho)$ 
2 :  $\mathbf{A}_1 := (\hat{q}\mathbf{j}_n - \gamma_b \mathbf{b}_1, \mathbf{A}_0)$ 
3 :  $tr := H(\text{pk})$ 
4 :  $\mu \in \{0, 1\}^\kappa := H(tr || M)$ 
5 :  $c := \text{SampleInBall}(\tilde{c})$ 
6 :  $\hat{\mathbf{w}}' = \mathbf{A}_1 \mathbf{z}_1 + \hat{q}\mathbf{j}_n c$ 
7 :  $\mathbf{w}' = \text{HighBits}(\hat{\mathbf{w}}', \gamma_2) + \mathbf{h}$ 
8 : return  $\llbracket \|(\mathbf{z}_1, \gamma_2 \mathbf{h})\|_\infty \leq B_\infty \rrbracket$  and  $\llbracket \tilde{c} = H(\mu || \mathbf{w}') \rrbracket$ 

```

Figure 5: Pseudocode of BiT

KGen (1^κ)	Sign (pk, sk, μ)
1: $f, g \xleftarrow{\$} S_\eta \times S_\eta \subset \mathcal{R}_q \times \mathcal{R}_q$ 2: if f is not invertible in $\mathcal{R}_q$, go into step 1 3: $s = (f, -g)$ 4: $\hat{a} = (g + \hat{q})/f \in \mathcal{R}_q$ 5: $a := (\hat{a}, 1) \in \mathcal{R}_q^2$ 6: $(\text{pk}, \text{sk}) := (a, s)$	1: $y = (y_1, y_2) \xleftarrow{\$} T_{\gamma_1} \times T_{\gamma_1}$ 2: $w := ay \pmod q$ 3: $c = \text{H}(\text{HighBits}(w, \gamma_2), \mu)$ 4: $b \xleftarrow{\$} \{0, 1\}$ 5: $z = (z_1, z_2) := y + (-1)^b sc$ 6: $z := \text{RejectSample}(z, \gamma_1, sc, \beta)$ 7: if $z = \perp$, go into step 1 8: if $\text{HighBits}(w, \gamma_2) \neq \text{HighBits}(w + (-1)^b c, \gamma_2)$, go into step 1 9: $\hat{w} = w - z_2 + (1 - b)c$ 10: $h = \text{HighBits}(w, \gamma_2) - \text{HighBits}(\hat{w}, \gamma_2)$ 11: if $\ (z_1, \gamma_2 h)\ _\infty > B_\infty$, then restart 12: return $\sigma := (z_1, h, c)$
Verify (pk, σ , μ)	
1: if $\ (z_1, \gamma_2 h)\ _\infty > B_\infty$, return reject 2: $\hat{w}' = \hat{a}z_1 + \hat{q}c \pmod q$ 3: $w' = \text{HighBits}(\hat{w}', \gamma_2) + h$ 4: Accept if $c = \text{H}(w', \mu)$	

Figure 6: BiT (NTRU Version)

ID Scheme	Signature Scheme	Signature with Abort
Interactive 1: Prover $\xrightarrow{(w, \text{st})}$ Verifier 2: Prover $\xleftarrow{c}$ Verifier 3: Prover $\xrightarrow{z}$ Verifier 4: Output (w, c, z)	Signing 1: $w \leftarrow \text{Com}(\text{sk})$ 2: $c \leftarrow \text{RO}(w, \mu)$ 3: $z \leftarrow \text{Rep}(w, c)$ 4: Output (w, c, z)	Signing 1: $w \leftarrow \text{Com}(\text{sk})$ 2: $c \leftarrow \text{RO}(w, \mu)$ 3: $z \leftarrow \text{Rep}_{\text{Abort}}(w, c)$ 4: If $z = \perp$, restart; otherwise, output (w, c, z)
Verification 1: Accept if $V(w, c, z) = 1$	Verification 1: Accept if $V(w, c, z) = 1$	Verification 1: Accept if $V(w, c, z) = 1$

Figure 7: Fiat-Shamir with Abort

abort. The components that differ between schemes are highlighted in the boxes. Figure 7 illustrates the complete Fiat-Shamir with Aborts transformation process.

Rejection Sampling: We have introduced rejection sampling and its basic principle in the Introduction. Here, we further explain its advantages.

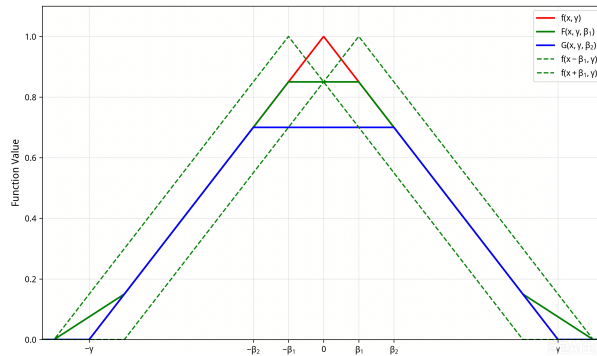


Figure 8: Rejection Sampling in BiT

As discussed in the Introduction, the rejection rate equals the area between the functions F and G , which can be easily computed as β^2/γ_1^2 . In contrast, the rejection sampling used in the NIST-standardized ML-DSA scheme is based on the sampled vector $\mathbf{z} := \mathbf{y} + \mathbf{s}c$ and rejects whenever $\|\mathbf{z}\|_\infty > \gamma_1 - \beta$. In the one-dimensional case, its rejection rate is β/γ_1 , which is significantly higher than that of our scheme. The gap becomes even larger in higher dimensions.

Signature Compression: The signing key can be decomposed into $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$, and the signature is similarly decomposed into $(\mathbf{z}_1, \mathbf{z}_2)$. During verification, $\mathbf{z}_2$ appears only as an additive term and is never involved in multiplication. Moreover, $\mathbf{z}_2$ has a small norm and therefore has only a limited effect on the verification result. This property allows its influence to be further reduced by truncating the high-order bits of the computation results.

To eliminate the residual influence caused by high-order deviations, two mainstream approaches are commonly used. The first introduces an additional check during signing: if the high-order bits computed by the signer do not match those computed by the verifier, the current signature is rejected, and $\mathbf{z}_2$ is not output. However, this approach significantly increases the rejection rate, reduces computational efficiency, and limits parameter selection and signature size. The second approach computes the difference between the high-order bits obtained by the signer and the verifier, and encodes it as auxiliary hint information $\mathbf{h}$. In essence, $\mathbf{h}$ compresses the original component $\mathbf{z}_2$, thereby significantly reducing the signature size. Since $\mathbf{h}$ is included in the signature, rejection sampling on $\mathbf{z}_2$ is still required to ensure security.

Since the optimized rejection sampling mechanism proposed in this work, the overhead introduced by the first approach is much higher than the cost of rejection sampling on $\mathbf{z}_2$. Therefore, this scheme adopts the second, hint-based compression approach to reduce the signature size efficiently.

4.1.2 Mathematical Hard Problems

The existential unforgeability of lattice-based signature schemes is typically reduced to the hardness of standard lattice problems, including the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). In general, the security of a signature scheme relies on two components:

1. **Key generation security.** The public key must be computationally indistinguishable from a uniformly random element over the corresponding space. For MLWE-based schemes, this property relies on the hardness of the MLWE problem.
2. **Signature generation security.** The scheme must prevent an adversary from forging valid short signatures. This property is ultimately reduced to the hardness of the Module Short Integer Solution (MSIS) problem.

The standardized definitions of all lattice hard problems involved in this paper are provided in Section 2.3.

4.2 Criterion for Parameter Selection

Parameter selection is guided by two criteria: security and computational efficiency.

Specifically, the target security level determines the maximum entropy of the challenge polynomial $\mathbf{c}$, thereby limiting the value of parameter τ . The security of key generation and the unforgeability of signatures are respectively reduced to the hardness of the MLWE and MSIS problems. Core parameters, including the public key dimension and polynomial degree, are then chosen according to the complexity of these underlying problems, ensuring that all hardness assumptions meet or exceed the target security level. The detailed security reductions are presented in Section 5.

In addition, the overhead of rejection sampling is taken into account during parameter selection. In this work, the expected number of signing attempts required to generate a valid signature is approximately 5–6, providing a good balance between computational efficiency and security.

The parameters are respectively provided in Tables 3.

Table 3: Parameter Settings

Security Level	80	128	192	256	384	512
η [size of key]	1	1	1	1	1	1
q [modulus]	3329	26881	15361	119297	40961	520193
d [order of the polynomial]	512	256	1024	512	2048	1024
(n, m) [dimension of pk]	(1, 1)	(3, 3)	(1,1)	(3, 3)	(1, 1)	(3,3)
τ [hamming weight of $\mathbf{c}$]	17	30	31	58	60	115
β [∞ -norm of $\mathbf{sc}$ ($\eta\tau$)]	17	30	31	58	60	115
$(\gamma_{1,0}, \gamma_{1,1}, \gamma_{1,2})$ [size of $\mathbf{y}$]	$(2^3, 2^9, 2^9)$	$(2^3, 2^{10}, 2^{11})$	$(2^3, 2^{10}, 2^{11})$	$(2^3, 2^{12}, 2^{13})$	$(2^3, 2^{13}, 2^{13})$	$(2^4, 2^{13}, 2^{15})$
γ_2 [parameters for HighBits]	363	682	1065	5517	4288	69481
γ_b [Parameters for compressing pk]	2^4	2^4	2^4	2^6	2^6	2^6
B_∞	2^9	2^{11}	2^{11}	2^{13}	2^{13}	2^{17}
$\ \mathbf{h}\ _\infty$	2	3	2	2	2	1

4.3 The post-quantum security for the entropy of the challenge

In this section, we present the criteria for setting the challenge entropy (i.e., the Hamming weight τ of $\mathbf{c}$) to ensure quantum security of the hash function under Grover's search with bounded circuit depth.

4.3.1 Grover Search Algorithm

The Grover search algorithm (GSA) is a quantum algorithm for unstructured search. Given a search space of size $N = 2^n$, it finds a marked element in $O(\sqrt{N})$ iterations, providing a quadratic speedup over classical brute-force search.

Each Grover iteration consists of an oracle that marks the target element and a diffusion operator. Starting from a uniform superposition over the search space, if there are M marked elements, the probability of measuring a marked element is maximized after $\left\lfloor \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rfloor$ iterations. When $M \ll N$, this can be approximated by $\left\lfloor \frac{\pi}{4} \sqrt{N} \right\rfloor$.

4.3.2 Quantum Circuit Depth

The depth of a quantum circuit is the minimum number of sequential layers obtained by grouping all quantum gates that can be executed in parallel into the same layer. It determines the execution time of the circuit and is ultimately limited by the coherence time of the qubits.

To evaluate post-quantum security under realistic quantum computing assumptions, NIST defines three maximum quantum circuit depth thresholds, denoted by MAXDEPTH. These thresholds are listed in Table 4.

Table 4: MAXDEPTH

MAXDEPTH	Description
2^{40}	The approximate number of gates that presently envisioned quantum computing architectures are expected to serially perform in a year
2^{64}	The approximate number of gates that current classical computing architectures can perform serially in a decade
2^{96}	The approximate number of gates that atomic scale qubits with speed of light propagation times could perform in a millennium

4.3.3 Quantum Security Corresponding to Different Challenge Entropies

Let g and d denote the total number of quantum gates and the circuit depth of a single Grover iteration for a Grover search attack on SHA-3, respectively. Under the maximum quantum circuit depth constraint MAXDEPTH, the number of Grover iterations is bounded by $\left\lfloor \frac{\text{MAXDEPTH}}{d} \right\rfloor$. Accordingly, the searchable space satisfies $N \leq \left(\frac{4}{\pi} \frac{\text{MAXDEPTH}}{d} \right)^2$. When the challenge entropy satisfies $2^k > \left(\frac{4}{\pi} \frac{\text{MAXDEPTH}}{d} \right)^2$, the total number of quantum gates required for a Grover attack is

$$G = \frac{2^k}{N} \times g \frac{\pi}{4} \sqrt{N} > 2^k \left(\frac{\pi}{4} \right)^2 \frac{gd}{\text{MAXDEPTH}}.$$

In each Grover iteration, the diffusion operator requires far fewer quantum gates and much smaller circuit depth than the SHA-3 oracle, so its cost can be neglected. According to the latest quantum resource estimation for SHA-3, $g = 2605074$ and $d = 1156$. Taking $\text{MAXDEPTH} = 2^{64}$ as an example, the quantum security for different challenge entropies is shown in Table 5.

Table 5: Quantum Security for Different Challenge Entropies

Security Level	80	128	192	256	384	512
d [order of the polynomial]	512	256	1024	512	2048	1024
τ [hamming weight of c]	17	30	31	58	60	115
entropy of the challenge	104.28	129.74	196.68	256.81	386.61	514.33
quantum security	$2^{71.065}$	$2^{96.53}$	$2^{163.46}$	$2^{223.60}$	$2^{353.40}$	$2^{481.12}$

5 Security Statements and Analysis

5.1 Theoretical Security Analysis

The security analysis follows the standard Fiat–Shamir proof framework. We first establish standard UF-CMA security in two steps. In the first step, we reduce UF-CMA security to UF-NMA security, where the adversary is not allowed to make signing queries. In the second step, we reduce UF-NMA security to the hardness of the underlying lattice problems. We then address the additional case arising in the proof of SUF-CMA security, namely, when the adversary produces a valid forgery on a message that has already been queried to the signing oracle.

Theorem 2. *Let the $\text{MLWE}_{q,n,m,S_\eta,S_\eta}$ problem be $(t_{\text{MLWE}}, \epsilon_{\text{MLWE}})$ -hard, the $\text{MSIS}_{q,n,(m+1),\gamma_{\text{MSIS}}}$ problem be $(t_{\text{MSIS}}, \epsilon_{\text{MSIS}})$ -hard for $\gamma_{\text{MSIS}} = 2B_\infty + \gamma_2 + 1$, the $\text{BiomodalSelfTargetMSIS}_{q,n,m,\gamma_{\text{Self}}}$ problem be $(t_{\text{Self}}, \epsilon_{\text{Self}})$ -hard for $\gamma_{\text{Self}} = B_\infty + \gamma_2/2 + 1$. Then BiT signature scheme in Figure 4 is (t, ϵ, q_S, q_H) -SUF-CMA secure in programmable random oracle model, where $t \approx t_{\text{MLWE}} + t_{\text{MSIS}} + t_{\text{Self}}$ and*

$$\epsilon \leq \epsilon_{\text{MLWE}} + \epsilon_{\text{Self}} + \epsilon_{\text{MSIS}} + \text{negl}$$

$$\text{where } \text{negl} = [\kappa\rho^{-1}q_S(\kappa\rho^{-1}q_S + q_H) + q_S(q_S + q_H)] \left(\frac{2\gamma_2+1}{q}\right)^n + q_S 2^{-\kappa}.$$

Proof. The proof consists of three parts. First, in Lemma 2, the schemes CMA security is reduced to its NMA security. Next, in Lemma 3, we reduce the NMA security to the BiomodalSelfTargetMSIS problem. Finally, we establish the reduction from SUF-CMA to UF-CMA security under the MSIS assumption in Lemma 4. $\square$

Lemma 2. *Let $\mathcal{A}$ be an algorithm that (t, ϵ, q_S, q_H) -breaks UF-CMA security of the BiT signature scheme in Figure 4, and the decision $\text{MLWE}_{q,n,m,S_\eta,S_\eta}$ problem be $(t_{\text{MLWE}}, \epsilon_{\text{MLWE}})$ -hard. Then, there exists an algorithm that $(t_{\text{Game}}, \epsilon_{\text{Game}}, q_S, q_H)$ -forges a signature for Game in Figure 9, where $t_{\text{Game}} \approx t + t_{\text{MLWE}}$ and*

$$\epsilon_{\text{Game}} \geq \epsilon - \text{negl} - \epsilon_{\text{MLWE}}$$

$$\text{where } \text{negl} = [\kappa\rho^{-1}q_S(\kappa\rho^{-1}q_S + q_H) + q_S(q_S + q_H)] \left(\frac{2\gamma_2+1}{q}\right)^n + q_S 2^{-\kappa}.$$

KGen : (1^κ) Real; Game 1; Game 2; Game 3	Sign (sk, pk, μ) Game 2
1 : $\mathbf{s}_0, \mathbf{e} \xleftarrow{\$} S_\eta^m \times S_\eta^n \subset \mathcal{R}_q^m \times \mathcal{R}_q^n$ 2 : $\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n \times m}$ 3 : $\mathbf{b} = \mathbf{A}_0 \mathbf{s}_0 + \mathbf{e}$ 4 : $(\mathbf{b}_1, \mathbf{b}_0) := (\text{HighBits}(\mathbf{b}, \gamma_b), \text{LowBits}(\mathbf{b}, \gamma_b))$ 5 : $\mathbf{A} = [\hat{q}\mathbf{j}_n - \gamma_b \mathbf{b}_1, \mathbf{A}_0, \mathbf{I}_n] = [\mathbf{A}_1, \mathbf{I}_n]$ 6 : $\mathbf{s} := (1, \mathbf{s}_0, \mathbf{e})$ 7 : $\text{pk} := \mathbf{A}; \text{sk} := (\mathbf{s}, \mathbf{b}_0)$	1 : $\mathbf{y} := (\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2) \xleftarrow{\$} T_{\gamma_{1,0}}^m \times T_{\gamma_{1,1}}^m \times T_{\gamma_{1,2}}^n$ 2 : $c \xleftarrow{\$} \mathcal{C}$ 3 : $b \xleftarrow{\$} \{0, 1\}$ 4 : $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$ 5 : $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) := \text{Split}(\mathbf{z}) \in \mathcal{R}_q \times \mathcal{R}_q^m \times \mathcal{R}_q^n$ 6 : $\mathbf{z} := \text{RejectSample}(\mathbf{z})$ 7 : $\mathbf{w} := \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c - [(1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c]$ 8 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1 9 : $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 10 : $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 11 : if $\ (\mathbf{z}_1, \gamma_2 \mathbf{h})\ _\infty > B_\infty$, then restart 12 : return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$ 13 : $\text{program } c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$
KGen : (1^κ) Game 4; Game	
1 : $\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n \times m}$ 2 : $\mathbf{b} \xleftarrow{\$} \mathcal{R}_q^n$ 3 : $(\mathbf{b}_1, \mathbf{b}_0) := (\text{HighBits}(\mathbf{b}, \gamma_b), \text{LowBits}(\mathbf{b}, \gamma_b))$ 4 : $\mathbf{A} = [\hat{q}\mathbf{j}_n - \gamma_b \mathbf{b}_1, \mathbf{A}_0, \mathbf{I}_n] = [\mathbf{A}_1, \mathbf{I}_n]$ 5 : $\text{pk} := \mathbf{A}$	
Sign (sk, pk, μ) Real	Sign (sk, pk, μ) Game 3; Game 4
1 : $\mathbf{y} := (\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2) \xleftarrow{\$} T_{\gamma_{1,0}}^m \times T_{\gamma_{1,1}}^m \times T_{\gamma_{1,2}}^n$ 2 : $\mathbf{w} := \mathbf{A}\mathbf{y}$ 3 : $c \in \mathcal{R}_q := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$ 4 : $b \xleftarrow{\$} \{0, 1\}$ 5 : $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$ 6 : $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) := \text{Split}(\mathbf{z}) \in \mathcal{R}_q \times \mathcal{R}_q^m \times \mathcal{R}_q^n$ 7 : $\mathbf{z} := \text{RejectSample}(\mathbf{z})$ 8 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1 9 : $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 10 : $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 11 : if $\ (\mathbf{z}_1, \gamma_2 \mathbf{h})\ _\infty > B_\infty$, then restart 12 : return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$	1 : $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) \xleftarrow{\$} T_{\hat{G}_{\gamma_{1,0},1}}^m \times T_{\hat{G}_{\gamma_{1,1},\beta}}^m \times T_{\hat{G}_{\gamma_{1,2},\beta}}^n$ 2 : $c \xleftarrow{\$} \mathcal{C}$ 3 : $b \xleftarrow{\$} \{0, 1\}$ 4 : $\mathbf{w} := \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c - [(1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c]$ 5 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1 6 : $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 7 : $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 8 : if $\ (\mathbf{z}_1, \gamma_2 \mathbf{h})\ _\infty > B_\infty$, then restart 9 : return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$ 10 : $\text{program } c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$
Sign (sk, pk, μ) Game 1	Sign (pk, μ) Game
1 : $\mathbf{y} := (\mathbf{y}_{1,0}, \mathbf{y}_{1,1}, \mathbf{y}_2) \xleftarrow{\$} T_{\gamma_{1,0}}^m \times T_{\gamma_{1,1}}^m \times T_{\gamma_{1,2}}^n$ 2 : $c \xleftarrow{\$} \mathcal{C}$ 3 : $b \xleftarrow{\$} \{0, 1\}$ 4 : $\mathbf{z} := \mathbf{y} + (-1)^b \mathbf{s}c$ 5 : $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) := \text{Split}(\mathbf{z}) \in \mathcal{R}_q \times \mathcal{R}_q^m \times \mathcal{R}_q^n$ 6 : $\mathbf{w} := \mathbf{A}\mathbf{y}$ 7 : $\text{program } c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$ 8 : $\mathbf{z} := \text{RejectSample}(\mathbf{z})$ 9 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1 10 : $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 11 : $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 12 : if $\ (\mathbf{z}_1, \gamma_2 \mathbf{h})\ _\infty > B_\infty$, then restart 13 : return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$	1 : $(\mathbf{z}_1, \mathbf{z}_2) := (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}, \mathbf{z}_2) \xleftarrow{\$} T_{\hat{G}_{\gamma_{1,0},1}}^m \times T_{\hat{G}_{\gamma_{1,1},\beta}}^m \times T_{\hat{G}_{\gamma_{1,2},\beta}}^n$ 2 : $c \xleftarrow{\$} \mathcal{C}$ 3 : $b \xleftarrow{\$} \{0, 1\}$ 4 : $\mathbf{b}_0 \xleftarrow{\$} \mathcal{R}_{\gamma_b}^n$ 5 : $\mathbf{w} := \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c - [(1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c]$ 6 : if $\text{HighBits}(\mathbf{w}, \gamma_2) \neq \text{HighBits}(\mathbf{w} + (-1)^b \mathbf{j}_n c, \gamma_2)$, go into step 1 7 : $\hat{\mathbf{w}} = \mathbf{w} - \mathbf{z}_2 + (1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c$ 8 : $\mathbf{h} = \text{HighBits}(\mathbf{w}, \gamma_2) - \text{HighBits}(\hat{\mathbf{w}}, \gamma_2)$ 9 : if $\ (\mathbf{z}_1, \gamma_2 \mathbf{h})\ _\infty > B_\infty$, then restart 10 : return $\sigma := (\mathbf{z}_1, \mathbf{h}, c)$ 11 : $\text{program } c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$

Figure 9: The Security Games

Proof. To prove the lemma, we introduce intermediate games **Game 1**, **Game 2**, **Game 3**, **Game 4** and **Game** as shown in Figure 9. Let ϵ_1 ($\epsilon_2, \epsilon_3, \epsilon_4, \epsilon_f$, respectively) denote the distinguishing advantage between **Real** and **Game 1** (**Game 1** and **Game 2**, **Game 2** and **Game 3**, **Game 3** and **Game 4**, **Game 4** and **Game**, respectively). The boxed parts highlight the modifications from the previous game.

Real denotes the real signing algorithm. For simplicity, the rejection sampling procedures for $\mathbf{z}_{1,0}$, $\mathbf{z}_{1,1}$, and $\mathbf{z}_2$ are uniformly referred to as the rejection sampling for $\mathbf{z}$.

Real $\rightsquigarrow$ **Game 1**. In **Game 1**, when answering signing queries, the challenger no longer computes $c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$. Instead, it samples $c \xleftarrow{\$} \mathcal{C}$ uniformly at random and later programs $\text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu) = c$ for each iteration.

Game 1 simulates **RO** in the real algorithm **Real**. The only difference is that, in **Game 1**, outputs are directly assigned to random oracle inputs if the **RO** query is invoked by the signing query, whereas in **Real**, the random oracle first checks whether the input has appeared before and returns the previously recorded output if so. For direct **RO** queries, it still checks whether the input has appeared before and returns the previously recorded output if so.

Clearly, **Game 1** perfectly simulates **Real** as long as all random oracle inputs are distinct. Therefore, the distinguishing advantage introduced by this modification is bounded by the probability of random oracle input collisions. Since the adversary may query the same message μ multiple times, we focus on the probability of $\text{HighBits}(\mathbf{w}, \gamma_2) = \text{HighBits}(\mathbf{w}', \gamma_2)$.

$$\begin{aligned} & \Pr[\text{HighBits}(\mathbf{w}, \gamma_2) = \text{HighBits}(\mathbf{w}', \gamma_2)] \\ &= \Pr[\text{HighBits}(\mathbf{A}\mathbf{y}, \gamma_2) = \text{HighBits}(\mathbf{A}\mathbf{y}', \gamma_2)] \\ &\leq \Pr[\|\mathbf{A}\mathbf{y} - \mathbf{A}\mathbf{y}'\|_\infty \leq \gamma_2] \\ &= \Pr[\|\mathbf{u}\|_\infty \leq \gamma_2 | \mathbf{u} \xleftarrow{\$} \mathcal{R}_q^n] \\ &= \left(\frac{2\gamma_2 + 1}{q}\right)^n \end{aligned}$$

The inequality follows from the definition of **HighBits**. Since **HighBits** extracts the high-order bits by performing modulo γ_2 , two inputs whose difference exceeds γ_2 must produce different outputs under **HighBits**. The second equation follows by heuristically modeling $\mathbf{A}(\mathbf{y} - \mathbf{y}')$ as a uniformly random element $\mathbf{u} \in \mathcal{R}_q^n$.

The probability of a single rewind in the scheme generating a valid signature is ρ . Excluding the probability of $q_S 2^{-\kappa}$, the maximum number of rejections in **Game 1** is κ/ρ . The number of defined points of the **RO** within **Game 1** is upper bounded by $\kappa\rho^{-1}q_S + q_H$. Additionally, there are at most $\kappa\rho^{-1}q_S$ reprogramming steps. Therefore, we have

$$\epsilon_1 \leq \kappa\rho^{-1}q_S(\kappa\rho^{-1}q_S + q_H) \left(\frac{2\gamma_2 + 1}{q}\right)^n + q_S 2^{-\kappa} \quad (1)$$

Game 1 $\rightsquigarrow$ **Game 2**. In **Game 2**, the generation of $\mathbf{w}$ is changed from $\mathbf{w} := \mathbf{A}\mathbf{y}$ to $\mathbf{w} := \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c - [(1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c]$. It is straightforward to verify that $\mathbf{A}\mathbf{y} = \mathbf{A}\mathbf{z} + \hat{q}\mathbf{j}_n c - [(1-b)\mathbf{j}_n c + (-1)^b \mathbf{b}_0 c]$, thus this modification introduces no distinguishing advantage.

Additionally, the programming position is modified. In **Game 2**, $c := \text{H}(\text{HighBits}(\mathbf{w}, \gamma_2), \mu)$ is executed after the signature is output. Similar to the analysis in **Real** $\rightsquigarrow$ **Game 1**, **Game 1** and

Game 2 are identical as long as all random oracle inputs are distinct. Hence, the distinguishing advantage is bounded by the probability of random oracle input collisions, which is $\left(\frac{2\gamma_2+1}{q}\right)^n$.

Note that in Game 2, only output signatures are programmed into RO. Therefore, it suffices to consider collisions involving output signatures only. The adversary can issue at most q_S signing queries and q_H RO queries. Hence, the number of defined RO points in Game 2 is bounded by $q_H + q_S$, and there are at most q_S output signatures. Therefore, we have $\epsilon_2 \leq q_S(q_S + q_H) \left(\frac{2\gamma_2+1}{q}\right)^n$.

Game 2 $\rightsquigarrow$ Game 3. In Game 3, the generation of the signature component $\mathbf{z}$ is modified. Instead of computing $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}c$, the challenger directly samples $\mathbf{z} \leftarrow T_{\hat{G}_{\gamma_1,0,1}} \times T_{\hat{G}_{\gamma_1,1,\beta}}^m \times T_{\hat{G}_{\gamma_1,2,\beta}}^n$. The correctness of this replacement follows from the rejection sampling algorithm `RejectSample`. Since the distribution of $\mathbf{z}$ is identical in Game 2 and Game 3, and $\mathbf{z}$ is independent of all other information in Game 3, the output signature distribution remains unchanged. We have $\epsilon_3 = 0$.

Game 3 $\rightsquigarrow$ Game 4. Game 4 modifies the generation of the verification key. In Game 3, $(\mathbf{A}_0, \mathbf{b})$ is sampled as an MLWE instance, and the verification key $\mathbf{A}$ is derived from it. In contrast, in Game 4, $(\mathbf{A}_0, \mathbf{b})$ is sampled uniformly from $\mathcal{R}_q^{n \times m} \times \mathcal{R}_q^n$. The difference between these two games is bounded by the MLWE assumption. Therefore, $\epsilon_4 \leq \epsilon_{\text{MLWE}}$.

Game 4 $\rightsquigarrow$ Game. Finally, Game completes the last replacement and can simulate the output of Real without the secret key. In Game, $\mathbf{b}_0$ is sampled uniformly from $\mathcal{R}_{\gamma_b}^n$. In Game 4, $\mathbf{b}$ is sampled uniformly from $\mathcal{R}_q^n$, and $\mathbf{b}_0$ is defined as its low-order bits. Since the coefficients of $\mathbf{b}$ are mutually independent and uniformly distributed, this change does not alter the distribution of $\mathbf{b}_0$. Therefore, $\epsilon_f = 0$.

Thus, we have $\epsilon_{\text{Game}} \geq \epsilon - \epsilon_{\text{MLWE}} - [\kappa\rho^{-1}q_S(\kappa\rho^{-1}q_S + q_H) + q_S(q_S + q_H)] \left(\frac{2\gamma_2+1}{q}\right)^n + q_S 2^{-\kappa}$. $\square$

Lemma 3. Let $\mathcal{F}$ be an adversary that $(t_{\text{Game}}, \epsilon_{\text{Game}}, q_S, q_H)$ -forges a signature in Game in Figure 9, then there is also an adversary $\mathcal{F}'$ with running time $t_{\text{Self}} \approx t_{\text{Game}}$ that can solve the $\text{BimodalSelfTargetMSIS}_{q,n,m,\gamma_{\text{Self}}}$ problem for $\gamma_{\text{Self}} = B_\infty + \gamma_2/2 + 1$. The probability ϵ_{Self} is lower bounded by: $\epsilon_{\text{Self}} \geq \epsilon_{\text{Game}}$.

Proof. If $\mathcal{F}$ successfully forges the signature $(\mathbf{z}_1, \mathbf{h}, c)$ on message μ , then

$$\text{H}(\text{HighBits}(\mathbf{A}_1 \mathbf{z}_1 + \hat{q} \mathbf{j}_n c, \gamma_2) + \mathbf{h}, \mu) = c$$

Then, there is

$$\gamma_2 \cdot [\text{HighBits}(\mathbf{A}_1 \mathbf{z}_1 + \hat{q} \mathbf{j}_n c, \gamma_2) + \mathbf{h}] = \mathbf{A}_1 \mathbf{z}_1 + \hat{q} \mathbf{j}_n c + \gamma_2 \mathbf{h} + \mathbf{u}$$

where $\|\mathbf{u}\|_\infty \leq \gamma_2/2 + 1$ comes from removal operation for `HighBits`. Then

$$\text{H}\left(\frac{1}{\gamma_2}(\mathbf{A}_1 \mathbf{z}_1 + \hat{q} \mathbf{j}_n c + \gamma_2 \mathbf{h} + \mathbf{u}), \mu\right) = c$$

Express the equation as a multiplication involving matrices and vectors:

$$H\left(\frac{1}{\gamma_2}([\mathbf{A}_1|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z}_1 \\ \gamma_2 \mathbf{h} + \mathbf{u} \end{bmatrix} + \hat{q}\mathbf{j}_n c), \mu\right) = c$$

Define $H'(\gamma_2 \mathbf{x}, \mu) = H(\mathbf{x}, \mu)$, then

$$H'([\mathbf{A}_1|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z}_1 \\ \gamma_2 \mathbf{h} + \mathbf{u} \end{bmatrix} + \hat{q}\mathbf{j}_n c, \mu) = c$$

Let $\mathbf{y} = \begin{bmatrix} \mathbf{z}_1 \\ \gamma_2 \mathbf{h} + \mathbf{u} \end{bmatrix}$, then $\|\mathbf{y}\|_\infty \leq \max(\|\gamma_2 \mathbf{h} + \mathbf{u}\|_\infty, \|\mathbf{z}_1\|_\infty) \leq B_\infty + \gamma_2/2 + 1$. This is exactly the solution of the `BimodalSelfTargetMSIS` problem. $\square$

Lemma 4. *If there exists an adversary $\mathcal{F}$ that $(t_{\text{SUF}}, \epsilon_{\text{SUF}}, q_S, q_H)$ -forges another signature for the message that already has an existing signature in the scheme in 4, then there also exists an adversary $\mathcal{F}'$ with running time $t_{\text{MSIS}} \approx t_{\text{SUF}}$ that can solve the $\text{MSIS}_{q,n,(m+1),\gamma_{\text{MSIS}}}$ problem for parameters $n, m, q \in \mathbb{N}$, $\gamma_{\text{MSIS}} = 2B_\infty + \gamma_2 + 1$ and the ring $\mathcal{R}_q$. The probability ϵ_{MSIS} is lower bounded by: $\epsilon_{\text{MSIS}} \geq \epsilon_{\text{SUF}}$.*

Proof. The `SUF-CMA` security is guaranteed by the hardness of the `MSIS` problem. To break the `SUF-CMA` security of an `UF-CMA` secure signature scheme, the adversary $\mathcal{F}$ is allowed to make signing queries and must output a new signature $(\mathbf{z}'_1, \mathbf{h}', c)$ for a message μ with an existing signature $(\mathbf{z}_1, \mathbf{h}, c)$, where the challenge c remains unchanged by the forking lemma.

$$H(\text{HighBits}(\mathbf{A}_1 \mathbf{z}'_1 + \hat{q}\mathbf{j}_n c, \gamma_2) + \mathbf{h}', \mu) = c$$

$$H(\text{HighBits}(\mathbf{A}_1 \mathbf{z}_1 + \hat{q}\mathbf{j}_n c, \gamma_2) + \mathbf{h}, \mu) = c$$

According to the collision resistance of H , there is

$$\text{HighBits}(\mathbf{A}_1 \mathbf{z}'_1 + \hat{q}\mathbf{j}_n c, \gamma_2) + \mathbf{h}' = \text{HighBits}(\mathbf{A}_1 \mathbf{z}_1 + \hat{q}\mathbf{j}_n c, \gamma_2) + \mathbf{h}$$

$$\mathbf{A}_1 \mathbf{z}'_1 + \hat{q}\mathbf{j}_n c + \gamma_2 \mathbf{h}' = \mathbf{A}_1 \mathbf{z}_1 + \hat{q}\mathbf{j}_n c + \gamma_2 \mathbf{h} + \mathbf{u}$$

$$[\mathbf{A}_1|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z}_1 - \mathbf{z}'_1 \\ \gamma_2(\mathbf{h} - \mathbf{h}') + \mathbf{u} \end{bmatrix} = \mathbf{0}$$

$$[\mathbf{A}_1|\mathbf{I}] \cdot \mathbf{y} = \mathbf{0}$$

where $\mathbf{u}$ comes from the removal of `HighBits` and $\|\mathbf{u}\|_\infty \leq \gamma_2 + 1$, then $\|\mathbf{y}\|_\infty \leq \|\gamma_2(\mathbf{h} - \mathbf{h}')\|_\infty + \|\mathbf{u}\|_\infty \leq 2B_\infty + \gamma_2 + 1$. Thus we obtain a solution of $\text{MSIS}_{q,n,m,\gamma_{\text{MSIS}}}$ problem. $\square$

5.2 Practical Security

Following the methodology of the NIST PQC standardization documents for lattice-based signatures, our concrete security evaluation considers both key-recovery and signature-forgery attacks. Key recovery corresponds to the decisional `MLWE` problem induced by the public key, while forgery security is based on `SIS`-type problems appearing in the reductions of Section 5.1.

Specifically, UF-CMA security reduces to the hardness of MLWE and BimodalSelfTargetMSIS, whereas SUF-CMA security additionally relies on the hardness of the corresponding MSIS problem.

We estimate the hardness of structured module/ring problems by treating them as the corresponding unstructured lattice problems over $\mathbb{Z}_q$. This provides a common and conservative basis for comparison with other lattice-based submissions, avoiding reliance on unproven additional hardness from the module or ring structure. In particular, the MLWE and MSIS instances are evaluated as ordinary LWE and SIS instances of the corresponding dimensions. The parameters are listed in Table 3.

We used the lattice-estimator [1, 2] (commit `3e48ef4`) to estimate all proposed parameter sets. The estimator implements the standard attack families currently used for LWE and SIS, including primal lattice attacks, dual lattice attacks, hybrid variants combining lattice reduction with guessing, and the non-lattice attacks enabled in the estimator, such as BKW-type attacks and Arora–Ge algebraic attacks when their preconditions are relevant. In our estimation, all of these attacks were enabled and compared under the same parameter description of the instances. BKZ costs were estimated using the core-SVP model (ADPS16 in the estimator), which measures the cost of BKZ reduction by the cost of solving exact SVP subproblems in the BKZ blocks [3, 4]. Specifically, once the estimator determines the BKZ block size β for a successful attack, classical and quantum costs are derived from the best-known asymptotic costs of lattice sieving in dimension β . This simple model enables direct comparison with earlier estimates based on BKZ plus classical or quantum sieving.

Table 6: Practical Security

Security Level	80	128	192	256	384	512
MLWE Hardness (Core-SVP)						
BKZ block-size b	348	443	659	877	1342	1752
Classical	101	129	192	256	392	512
Quantum	93	118	176	234	359	465
MSIS Hardness (Core-SVP)						
BKZ block-size (SUF)	346(277)	531(439)	816(658)	1031(877)	1523(1316)	2045(1754)
Classical (SUF)	101 (80)	155(128)	238(192)	301(256)	444(384)	597(512)
Quantum (SUF)	91 (73)	140(116)	216(174)	273(232)	403(348)	541(464)

The resulting concrete forgery hardness is reported in Table 6. MLWE Hardness denotes the difficulty of the MLWE problem, while MSIS Hardness indicates the difficulty of the MSIS problem. Security levels outside parentheses correspond to unforgeability (UF-CMA), which is based on the BimodalSelfTargetMSIS problem and ultimately reduced to MSIS, whereas values in parentheses correspond to strong unforgeability (SUF-CMA). Under SUF-CMA, the reduction involves a larger MSIS-type short-solution bound, yielding lower concrete security estimates.

For the concrete assessment of each hard problem, following the methodology of ML-DSA [23] and HAETAE [7], the BimodalSelfTargetMSIS _{$q,n,m,\gamma_{\text{Self}}$} problem is evaluated using the corre-

sponding $\text{MSIS}_{q,n,m,\gamma_{\text{Self}}}$ problem with the same parameters.

Moreover, the MSIS instance in the scheme differs from the standard MSIS , as the short solution vector has unbalanced amplitudes. Specifically, the solution takes the form

$$\begin{pmatrix} \mathbf{z}_1 - \mathbf{z}'_1 \\ \gamma_2(\mathbf{h} - \mathbf{h}') + \mathbf{u} \end{pmatrix}, \quad \mathbf{z}_1 = (\mathbf{z}_{1,0}, \mathbf{z}_{1,1}),$$

with $\|\mathbf{z}_{1,0} - \mathbf{z}'_{1,0}\|_\infty \leq 2(\gamma_{1,0} - 1)$, $\|\mathbf{z}_{1,1} - \mathbf{z}'_{1,1}\|_\infty \leq 2(\gamma_{1,1} - 1)$ and $\|\gamma_2(\mathbf{h} - \mathbf{h}') + \mathbf{u}\|_\infty \leq 2B_\infty + \gamma_2 + 1$. This imbalance makes it harder for an adversary to find a solution. Following the standard LWE attack methodology, which balances the variance of the secret and the error via a scale factor, we scale the components to align their amplitudes. Concretely, we have

$$q' := \left(\frac{2B_\infty + \gamma_2 + 1}{2(\gamma_{1,0} - 1)} \right)^{\frac{m}{m+n+1}} \cdot \left(\frac{2B_\infty + \gamma_2 + 1}{2(\gamma_{1,1} - 1)} \right)^{\frac{1}{m+n+1}} \cdot q,$$

and we estimate the concrete security based on the instance $\text{MSIS}_{q',n,m,2B_\infty+\gamma_2+1}$, thus yielding an accurate concrete security estimate.

5.3 Security against physical attacks

The resistance to side-channel attacks is an important metric for evaluating the practical security of a cryptographic implementation. In practice, a common source of side-channel vulnerabilities in compact lattice-based signature schemes, such as BLISS [9], Falcon [14], and HAWK [12], is the use of discrete Gaussian sampling. This requires sampling from a complex non-uniform distribution and is difficult to implement securely [15, 13, 25, 16, 17].

Our scheme avoids Gaussian sampling. Instead, it uses uniform sampling and triangular sampling derived from uniform sampling. This simplifies the sampling procedure and makes secure implementation easier. In addition, the main operations, such as polynomial multiplication, modular reduction, and norm checking, can all be implemented in a regular constant-time manner.

Compared with ML-DSA [23], BiT reduces the risk of several power-based side-channel attacks through its design.

- **NTT-domain secret multiplication.** BiT does not compute secret-challenge products such as $\mathbf{sc}$ and $\mathbf{ec}$ by point-wise multiplication in the NTT domain. This follows from the parameter selection and the structure of the challenge polynomial c , where sparse multiplication is more efficient than NTT-based multiplication. As a result, attacks targeting secret-dependent NTT operations in ML-DSA, such as $\mathbf{cs}_1$ or $\mathbf{cs}_2$, cannot be directly applied to BiT [6, 28, 19, 26].
- **Reduced leakage from intermediate values.** BiT mainly uses the high-order bits $\mathbf{w}_1 := \text{HighBits}(\mathbf{w}, \gamma_2)$ during signing and reduces the use of intermediate values such as the low-order bits $\mathbf{w}_0$. Therefore, attacks that rely on leakage from these intermediate values are not directly applicable to BiT [5].

- **Additional protection from the bimodal structure.** In ML-DSA, side-channel attacks often exploit the relation

$$\mathbf{z} := \mathbf{y} + c\mathbf{s}_1,$$

where leakage from $\mathbf{y}$ (such as zero coefficients or partial coefficient bits) or from rejected signatures $(\mathbf{z}, c)$ may reveal information about $c\mathbf{s}_1$ [20, 27, 30].

In BiT, this computation is replaced by

$$\mathbf{z}_{1,0} := \mathbf{y}_{1,0} + (-1)^b c, \quad \mathbf{z}_{1,1} := \mathbf{y}_{1,1} + (-1)^b s_0 c, \quad \mathbf{z}_2 := \mathbf{y}_2 + (-1)^b e c.$$

Here, $\mathbf{z}_{1,0}$ is independent of the secret vectors $\mathbf{s}_0$ and $\mathbf{e}$, and $\mathbf{z}_2$ is not included in the final signature. The remaining component,

$$\mathbf{z}_{1,1} := \mathbf{y}_{1,1} + (-1)^b s_0 c,$$

also depends on the secret bit b . Therefore, even if similar side-channel leakage is available, the key-recovery techniques developed for ML-DSA cannot be directly applied to recover $s_0 c$ without also recovering b .

Nevertheless, in scenarios requiring resistance to power-based side-channel attacks, additional masking and related countermeasures are still needed to protect sensitive variables. For example, consider $\mathbf{z}_{1,0} := \mathbf{y}_{1,0} + (-1)^b c$. Following the approach in [27], if leakage reveals that a coefficient of $\mathbf{y}_{1,0}$ is zero, an adversary may recover the value of b . This information can then be combined with further leakage indicating that a coefficient of $\mathbf{y}_{1,1}$ is zero to derive equations or constraints on $s_0 c$ from $\mathbf{z}_{1,1} := \mathbf{y}_{1,1} + (-1)^b s_0 c$. Compared with uniform sampling, such leakage is more likely under triangular sampling because its probability mass function has a higher value at 0. Moreover, each coefficient in triangular sampling is generated as the difference of two fixed-width uniform samples. Therefore, implementations should treat these intermediate values as sensitive to prevent leakage about the corresponding coefficient of $\mathbf{y}$. In addition, for $\mathbf{z}_{1,1} := \mathbf{y}_{1,1} + (-1)^b s_0 c$, rejection sampling determines coefficient-wise acceptance according to the magnitudes of the corresponding coefficients of $\mathbf{z}_{1,1}$ and $s_0 c$. Consequently, leakage from rejected signatures $(\mathbf{z}_{1,1}, c)$ or from the probability mass function of the source distribution may reveal constraints on the secret term $s_0 c$. Therefore, implementations targeting resistance to power-based side-channel attacks should protect the triangular mask $\mathbf{y}$, the secret bit b , and the rejection-sampling procedure using appropriate masking, hiding, or shuffling techniques.

6 Performance Evaluation

6.1 Performance Analysis

In BiT, the signing process consists of the following main steps: sampling the masking vector $\mathbf{y}$, computing $\mathbf{A}\mathbf{y}$, generating the challenge $\mathbf{c}$, performing rejection sampling on $\mathbf{z}$, and producing the hint $\mathbf{h}$.

- **Sampling $\mathbf{y}$:** BiT samples $\mathbf{y}$ from a triangular distribution, which is the difference of two

uniformly sampled variables. This is more efficient than Gaussian sampling.

- **Computing $\mathbf{Ay}$:** All computations are performed in the NTT domain, and fully split polynomial rings are used. The efficiency is comparable to mainstream schemes.
- **Generating the challenge $\mathbf{c}$:** The challenge is generated using two hash functions, following the same structure as mainstream lattice-based signature schemes.
- **Computing $\mathbf{cs}$:** The challenge c is sparse, with Hamming weight τ and coefficients in $\{0, 1\}$. The products $\mathbf{cs}$, $\mathbf{ce}$, and $\mathbf{cb}_0$ are therefore computed by sparse multiplication instead of the NTT, which is fast because τ is small and the nonzero entries are all 1, so only additions are required.
- **Rejection sampling on $\mathbf{z}$:** We use a simple rejection sampling method. Most candidate signatures $\mathbf{z}$ can be verified by checking their norms only, while a small fraction require an additional coin-flipping step to decide acceptance.
- **Producing the hint $\mathbf{h}$:** The generation of $\mathbf{h}$ has similar cost to mainstream schemes.
- **Optimization notes:** The improved rejection sampling allows smaller parameters. For example, BiT uses modulus 26881, while ML-DSA uses 8380417 at the 128-bit security level. As a result, polynomial matrix-vector multiplication in BiT is significantly faster, making the scheme more efficient than ML-DSA.

In addition, during rejection sampling of $\mathbf{z}$, if the i -th coefficient is rejected, the algorithm updates the first i coefficients and reuses the remaining ones. This reduces sampling overhead. This is secure because the unrejected coefficients have not been processed by rejection sampling, remain fully hidden, and are independent of whether earlier coefficients are rejected.

6.2 Performance Test

6.2.1 x86 Platforms

Reference Implementation: The performance of the reference implementation was evaluated using the C implementation of BiT. To assess the impact of different symmetric components on the overall running time, we report two sets of timing results: one instantiated with SM3 and the other instantiated with FIPS202. All benchmarks were executed on an Intel Core i7-8750H processor running at 2.2 GHz, with Turbo Boost and hyper-threading disabled to reduce measurement variance. The test platform was equipped with 16 GiB of memory and ran Ubuntu 24.04.2 LTS. The implementation was built using GNU Make 4.3 and compiled with GNU GCC 13.3.0 using the flags `-std=c99`, `-Wpedantic`, `-Wall`, `-Wextra`, and `-O2`. Each reported value represents the average number of CPU cycles over 100000 independent runs.

Optimized Implementation with AVX2: The performance of the optimized implementation was evaluated using the x86 AVX2 implementation of BiT. As before, we report two sets of timing results to assess the impact of the symmetric component. The first set corresponds to

the SM3-based instantiation, where AVX2 optimizations are applied to all operations except the symmetric component, while SM3 itself remains implemented in standard C; since we do not provide a dedicated AVX2 implementation of SM3, these results include the cost of the scalar SM3 component and are therefore less competitive than a fully vectorized symmetric instantiation. The second set corresponds to the FIPS202-based instantiation, for which a four-way parallel AVX2 implementation of the symmetric component is provided.

6.2.2 Performance Test Results

Table 7 presents the efficiency and size results for the SM3-based implementation. Table 8 supplements these results with efficiency results for the FIPS 202-based implementation. The results are reported for the 128-bit, 256-bit, and 512-bit security levels, including both AVX and REF implementations. The results for other security levels are given in Section 6.3.

Table 7: Efficiency and Size of SM3-Based Implementation

Security Level	128	256	512
Efficiency AVX (clcyes)			
KGen	491240	1344984	4135509
Sign	1955243	2696404	12178299
Verify	436272	1369603	3996773
Efficiency REF (clcyes)			
KGen	579986	1600281	4653932
Sign	2903445	3899580	16052002
Verify	586399	1696159	4630660
Size (bytes)			
pk	1048	2144	5056
sk	1864	4160	9024
sig	1504	3456	6695
pk+sig	2552	5600	11751

- **Transmission and Storage Cost:** The transmission and storage cost is measured by the sizes of the public key, secret key, and signature. Since the public key and signature are usually transmitted together, the combined size `pk + sig` is also reported.

6.3 Parameter, Security, and Size Summary

Table 9 and table 10 summarize the parameter settings, security levels, and size metrics of the BiT signature schemes.

Table 8: Efficiency of FIPS202-Based Implementation

Security Level	128	256	512
Efficiency AVX (clcyes)			
KGen	74515	195360	504948
Sign	333420	498220	1541878
Verify	76089	226266	543833
Efficiency REF (clcyes)			
KGen	269659	657525	1243540
Sign	1862472	2302217	7932956
Verify	291908	751294	1452871

Table 9: Parameter, Security, Size (128,256,512-bit security level)

Security Level	128	256	512
η [size of key]	1	1	1
q [modulus]	26881	119297	520193
d [order of the polynomial]	256	512	1024
(n, m) [dimension of $\mathbf{pk}$]	(3, 3)	(3, 3)	(3,3)
τ [hamming weight of c]	30	58	115
β [∞ -norm of $\mathbf{sc}(\eta\tau)$]	30	58	115
$(\gamma_{1,0}, \gamma_{1,1}, \gamma_{1,2})$ [size of $\mathbf{y}$]	$(2^3, 2^{10}, 2^{11})$	$(2^4, 2^{12}, 2^{13})$	$(2^4, 2^{13}, 2^{15})$
γ_2 [parameters for HighBits]	682	5517	69481
γ_b [Parameters for compressing $\mathbf{pk}$]	2^4	2^6	2^6
B_∞	2^{11}	2^{13}	2^{17}
$\ \mathbf{h}\ _\infty$	3	3	1
acceptance rate	18%	53%	33%
MLWE Hardness (Core-SVP)			
BKZ block-size b	443	877	1751
Classical	129	256	512
Quantum	118	234	465
MSIS Hardness (Core-SVP)			
BKZ block-size (SUF)	531(439)	1031(877)	2045(1754)
Classical (SUF)	155(128)	301(256)	597(512)
Quantum (SUF)	140(116)	273(232)	541(464)
pk (bytes)	1048	2144	5056
sig (bytes)	1504	3456	6695
pk+sig (bytes)	33552	5600	11751

Table 10: Parameter, Security, Size (80,192,384-bit security level)

Security Level	80	192	384
η [size of key]	1	1	1
q [modulus]	3329	15361	40961
d [order of the polynomial]	512	1024	2048
(n, m) [dimension of $\mathbf{pk}$]	(1, 1)	(1,1)	(1, 1)
τ [hamming weight of c]	17	31	60
β [∞ -norm of $\mathbf{sc}(\eta\tau)$]	17	31	60
$(\gamma_{1,0}, \gamma_{1,1}, \gamma_{1,2})$ [size of $\mathbf{y}$]	$(2^3, 2^9, 2^9)$	$(2^3, 2^{10}, 2^{11})$	$(2^3, 2^{13}, 2^{13})$
γ_2 [parameters for HighBits]	363	1065	4288
γ_b [Parameters for compressing $\mathbf{pk}$]	2^4	2^4	2^6
B_∞	2^9	2^{11}	2^{13}
$\ \mathbf{h}\ _\infty$	2	2	2
acceptance rate	24%	18%	31%
MLWE Hardness (Core-SVP)			
BKZ block-size b	348	659	1342
Classical	101	192	392
Quantum	93	176	359
MSIS Hardness (Core-SVP)			
BKZ block-size (SUF)	346(277)	816(658)	1523(1316)
Classical (SUF)	101 (80)	238(192)	444(384)
Quantum (SUF)	91 (73)	216(174)	403(348)
$\mathbf{pk}$ (bytes)	505	1293	2435
sig (bytes)	1057	2242	5251
$\mathbf{pk} + \mathbf{sig}$ (bytes)	1562	3535	7686

7 Feature Statements

The proposed scheme is designed according to the following principles.

Hardness of underlying assumptions: The scheme is based on MLWE problem and MSIS problem. The hardness of these problems has been extensively studied and is widely accepted, providing a solid security foundation for the proposed scheme.

Compact size and efficient implementation: Most existing compact Fiat–Shamir signature schemes employ Gaussian sampling [21, 22, 10]. However, Gaussian sampling requires floating-point arithmetic, making high-precision sampling difficult to implement efficiently. In addition, the corresponding rejection sampling also incurs a high computational cost. By contrast, schemes based on uniform sampling [11, 24] use a simpler rejection sampling procedure but produce longer

signatures. Our scheme adopts the triangular distribution [18], combining the implementation efficiency of uniform sampling with the compact signatures of Gaussian sampling. As a result, it achieves both short signatures and low computational overhead.

Conservative parameter selection: To ensure long-term security, the scheme adopts a conservative parameter selection strategy. The parameters are chosen so that the corresponding **core-SVP** problem remains sufficiently hard, providing resilience against potential future attacks.

Minimizing the combined size of the public key and signature: In many applications, such as certificate chains, the public key and signature are transmitted together. Therefore, the combined size of the two is a primary design objective. Our scheme focuses on minimizing combined size of the public key and signature.

Modular design: The scheme is constructed over module lattices using the polynomial ring $\mathbb{Z}_q[x]/\langle x^d + 1 \rangle$ at all security levels. The main operations in signing and verification consist of XOF expansion, SM3 hashing, and polynomial multiplication over $\mathbb{Z}_q[x]/\langle x^d + 1 \rangle$. Supporting different security levels only requires adjusting the corresponding ring parameters and XOF settings. Consequently, optimized implementations can be easily adapted across different security levels.

References

- [1] Martin R. Albrecht. Lattice Estimator. <https://github.com/malb/lattice-estimator>.
- [2] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, pages 351–367, 2018.
- [3] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Newhope without reconciliation. *IACR Cryptology ePrint Archive*, 2016:1157, 2016.
- [4] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24, 2016.
- [5] Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in dilithium: a template-based approach. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(4):188–210, 2023.
- [6] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu Jing. An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In *2021*

- IEEE 39th International Conference on Computer Design (ICCD)*, pages 583–590. IEEE, 2021.
- [7] Jung Hee Cheon, Hyeonmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. HAETAE: Shorter lattice-based fiat-shamir signatures. *Cryptology ePrint Archive*, Paper 2023/624, 2023.
 - [8] Léo Ducas, Alain Durmus, Tancrede Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. pages 40–56, 2013.
 - [9] Léo Ducas, Alain Durmus, Tancrede Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Annual Cryptology Conference*, pages 40–56. Springer, 2013.
 - [10] Léo Ducas, Alain Durmus, Tancrede Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, Lecture Notes in Computer Science, pages 40–56. Springer, 2013.
 - [11] Léo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - dilithium: Digital signatures from module lattices. *IACR Cryptol. ePrint Arch.*, 2017:633, 2017.
 - [12] Léo Ducas, Eamonn W Postlethwaite, Ludo N Pulles, and Wessel van Woerden. Hawk: Module lip makes lattice signatures fast, compact and simple. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 65–94. Springer, 2022.
 - [13] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, pages 1857–1874, New York, NY, USA, 2017. Association for Computing Machinery.
 - [14] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NISTs post-quantum cryptography standardization process*, 36(5):1–75, 2018.
 - [15] 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 *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 323–345, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
 - [16] Morgane Guereau, Ange Martinelli, Thomas Ricosset, and Mélissa Rossi. The hidden parallelepiped is back again: Power analysis attacks on falcon. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 141–164, 2022.
 - [17] Morgane Guereau and Mélissa Rossi. A not so discrete sampler: power analysis attacks on hawk signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(4):156–178, 2024.

- [18] Seungwoo Lee, Joo Woo, Jonghyun Kim, and Jong Hwan Park. Generalized centered binomial distribution for bimodal lattice signatures. *IEEE Access*, 12:138641–138651, 2024.
- [19] Yong Liu, Yuejun Liu, Yongbin Zhou, Yiwen Gao, Zehua Qiao, and Huaxin Wang. A novel power analysis attack against crystals-dilithium implementation. In *2024 IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2024.
- [20] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Transactions on Information Forensics and Security*, 16:1868–1879, 2020.
- [21] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, Lecture Notes in Computer Science, pages 598–616. Springer, 2009.
- [22] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, Lecture Notes in Computer Science, pages 738–755. Springer, 2012.
- [23] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>.
- [24] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical Report FIPS 204, U.S. Department of Commerce, Washington, D.C., August 2024.
- [25] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongswan’s implementation of post-quantum signatures. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS ’17*, pages 1843–1855, New York, NY, USA, 2017. Association for Computing Machinery.
- [26] Tolun Tosun and Erkey Savas. Zero-value filtering for accelerating non-profiled side-channel attack on incomplete ntt-based implementations of lattice-based cryptography. *IEEE Transactions on Information Forensics and Security*, 19:3353–3365, 2024.
- [27] Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all. In *International Conference on Selected Areas in Cryptography*, pages 3–32. Springer, 2022.
- [28] Huaxin Wang, Yiwen Gao, Yuejun Liu, Qian Zhang, and Yongbin Zhou. In-depth correlation power analysis attacks on a hardware implementation of crystals-dilithium. *Cybersecurity*, 7(1):21, 2024.
- [29] J. Woo, J. Kim, and Ga Hee Hong. Ntru+sign: compact ntru-based signatures using bimodal distributions. *Designs, Codes and Cryptography*, 94(2), 2026.

- [30] Yuanyuan Zhou, Weijia Wang, Yiteng Sun, and Yu Yu. Rejected signatures challenges pose new challenges: Key recovery of crystals-dilithium via side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(4):817–847, 2025.