Efficient Fixed-Point Implementation of Mitaka

Shiduo Zhang, Paul Kirchner, Xiuhan Lin and Yang Yu

Institute for Advanced Study, Tsinghua University, Beijing, China
zsd@mail.tsinghua.edu.cn, kirchner@mail.tsinghua.edu.cn
lin-xiuhan@mail.tsinghua.edu.cn, yu-yang@mail.tsinghua.edu.cn

Abstract. The Mitaka signature scheme is a variant of Falcon (FN-DSA). It has a considerably simpler signing procedure that can be parallelized while retaining Falcon's efficiency advantage. Mitaka offers two different signing approaches: a default one relying on floating-point arithmetic and an alternative, called Mitaka$_{\mathbb{Z}}$, avoiding floating-point arithmetic at the cost of more Gaussian samples. Despite its better portability, Mitaka$_{\mathbb{Z}}$ has not previously been implemented in practice.

This work presents an efficient fixed-point implementation of the Mitaka signing algorithm. We optimize the Mitaka$_{\mathbb{Z}}$ Gaussian sampler to make it suited to fixed-point arithmetic while achieving higher efficiency. Specifically, we replace the wide integral Gram root decomposition used in Mitaka$_{\mathbb{Z}}$ with a fixed-point square Cholesky decomposition, which substantially reduces the cost of integer Gaussian sampling. We prove bounds on intermediate values and representation errors without empirical assumptions on their magnitudes. The distributional analysis models each integer Gaussian call as exact at its represented inputs. We provide a fixed-point implementation of the new Mitaka signing algorithm in C. On an Intel Core i7-1360P, our native AVX2 implementation generates signatures in an average of 361 kcycles for Mitaka-512 and 707 kcycles for Mitaka-1024, measured with Linux hardware performance events. On an ARM Cortex-M4F STM32F411, the corresponding measured costs are 4.0 and 7.8 Mcycles.

Keywords: Mitaka· Hash-and-sign lattice-based signatures · Gaussian sampling · fixed-point arithmetic

1 Introduction

In response to the threat posed by future quantum computers, public-key cryptosystems are migrating to post-quantum cryptography (PQC). To date, the U.S. NIST has selected five PQC algorithms for standardization. Among them, Falcon is a signature scheme featuring exceptionally small signatures and fast verification, making it suitable for applications where signature size and verification cost are priorities. Nevertheless, Falcon suffers from a rather complex signing algorithm that is difficult to implement correctly and heavily relies on floating-point arithmetic, which hinders its deployment.

To mitigate Falcon's implementation complications, Espitau *et al.* [EFG+22] proposed Mitaka, an implementation-friendly variant of Falcon. Mitaka uses the hybrid sampler [Pre15] to replace the fast Fourier sampler [DP16] used by Falcon, making signature generation considerably simpler and easier to parallelize. The hybrid sampler incurs some security loss compared with the fast Fourier sampler, yet this issue has been satisfactorily solved by well-designed trapdoor generation techniques [ENS+23]. Consequently, Mitaka now offers a nice trade-off among simplicity, security and efficiency. Its designs were later adopted by SOLMAE [KTW+24], which is now a Korean TTA standard and has been approved to proceed as a KS standard.

With the post-quantum transition underway, portability is emerging as an increasingly important criterion for post-quantum algorithms. Falcon's reliance on floating-point arithmetic is regarded as a potential pitfall, and some security risks [LTYZ25] have been identified in deterministic variants of Falcon. Substantial efforts have been made to improve Falcon in this regard. Pornin [Por19] gave an implementation of Falcon using integer-based emulated floating-point arithmetic. It is portable to a wide range of platforms and has been adopted as the reference implementation. Halmans *et al.* [HvVS+26] presented an implementation of Falcon, called TWFalcon, using triple-word arithmetic, which is compatible with embedded devices equipped only with a single-precision FPU. However, these implementations come with a substantial slowdown compared to the native floating-point version. Very recently, Braga *et al.* [DFLP26] provided a fixed-point implementation of Falcon along with careful boundedness and precision analysis and moderate algorithmic tweaks. The fixed-point implementation is approximately a factor of two slower than the original floating-point Falcon implementation, thereby offering a desirable trade-off between portability and efficiency. In response, the NIST FIPS 206 team has proposed specifying fixed-point arithmetic for FN-DSA [Per26].

Mitaka also takes portability into account. It provides an alternate signing approach, known as Mitaka$_{\mathbb{Z}}$, for platforms with poor support for floating-point numbers. The Mitaka$_{\mathbb{Z}}$ sampler follows Peikert's convolution sampling approach [Pei10] and employs an approximate Gram-Schmidt basis as the linear transformation, thereby retaining the same signature size and security as Mitaka. To remove the floating-point arithmetic in the signing procedure, Mitaka$_{\mathbb{Z}}$ uses the integral perturbation sampling technique [DGPY20] at the cost of many more integer Gaussian samples. While Mitaka$_{\mathbb{Z}}$ seems to offer better portability, to the best of our knowledge, no practical implementation has been published.

Contributions. In this work, we close the gap in the efficient implementation of Mitaka$_{\mathbb{Z}}$ by presenting a fixed-point implementation along with rigorous analysis.

We optimize the perturbation sampling of the Mitaka$_{\mathbb{Z}}$ sampler. Our main observation is that the perturbation vector can be replaced by the product of an approximate square Cholesky root and some zero-centered discrete Gaussian vector. This allows a perturbation vector to be generated with only $2n$ centered integer Gaussian samples, where n is the degree of the NTRU polynomial ring. In comparison, the original Mitaka$_{\mathbb{Z}}$ sampler requires $O(n \log n)$ integer Gaussian samples while the Mitaka sampler requires $2n$ continuous Gaussian samples, whose generation is costly and complex in fixed-point arithmetic.

We give a rigorous analysis of the boundedness and precision of intermediate variables in the optimized Mitaka$_{\mathbb{Z}}$ sampler. Thanks to the simpler sampler of Mitaka, our analysis does not rely on any empirical assumptions on the error bounds of intermediate values, which differs from the fixed-point implementation of Falcon [DFLP26]. Based on this, we compute the required precision to guarantee at most 1.5 bits of security loss for concrete Mitaka parameters accounting for the state-of-the-art trapdoor generation technique [ENS+23].

We offer a fixed-point implementation in C of signature generation with our new sampler for Mitaka-512 and Mitaka-1024. On an Intel Core i7-1360P, the native AVX2 implementation requires an average of 360 960 and 706 857 processor cycles, respectively, measured with Linux hardware performance events. On an ARM Cortex-M4F STM32F411, signature generation requires 3 980 099 cycles for Mitaka-512 and 7 818 552 cycles for Mitaka-1024. Section 5 describes the measurement conditions and reports memory requirements and the comparison with Falcon measurements on the same Cortex-M4 board.

2 Preliminaries

We use bold lowercase (resp., uppercase) letters for vectors (resp., matrices). All vectors are column vectors, and all logarithms are natural. The transpose of a matrix $\mathbf{H}$ is denoted

$\mathbf{H}^t$. For symmetric matrices, $\mathbf{\Sigma} \succeq 0$ denotes positive semidefiniteness, and $\mathbf{\Sigma}_1 \succeq \mathbf{\Sigma}_2$ means $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 \succeq 0$.

2.1 Lattices and Gaussians

For a full-rank matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$, the lattice spanned by $\mathbf{B}$ is $\mathcal{L}(\mathbf{B}) = \mathbf{B}\mathbb{Z}^n$. The dual of a lattice $\mathcal{L}$ is $\mathcal{L}^* = \{\mathbf{y} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{Z} \text{ for all } \mathbf{x} \in \mathcal{L}\}$.

Let $\rho_{\mathbf{M},\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{M}^{-1}(\mathbf{x}-\mathbf{c})\|_2^2)$ be the Gaussian function with invertible factor $\mathbf{M} \in \mathbb{R}^{n\times n}$ and center $\mathbf{c} \in \mathbb{R}^n$. Its scaled covariance is $\mathbf{\Sigma} = \mathbf{M}\mathbf{M}^t \succ 0$[1], and $\rho_{\mathbf{M},\mathbf{c}} = \rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}$. When $\mathbf{M} = s\mathbf{I}$ for $s > 0$, we write the subscript $\mathbf{M}$ as s, and call s the Gaussian width. We omit the center subscript when $\mathbf{c} = \mathbf{0}$. Let $\rho_{\mathbf{M},\mathbf{c}}(S) = \sum_{\mathbf{x}\in S} \rho_{\mathbf{M},\mathbf{c}}(\mathbf{x})$ for any countable set S. The discrete Gaussian over a lattice $\mathcal{L}$ is $D_{\mathcal{L},\mathbf{M},\mathbf{c}}(\mathbf{x}) = \rho_{\mathbf{M},\mathbf{c}}(\mathbf{x})/\rho_{\mathbf{M},\mathbf{c}}(\mathcal{L})$ for $\mathbf{x} \in \mathcal{L}$.

Let $\eta_\epsilon(\mathcal{L}) = \min\{s > 0 : \rho_{1/s}(\mathcal{L}^*) \le 1 + \epsilon\}$ be the smoothing parameter of a lattice $\mathcal{L}$ for $\epsilon > 0$ [MR04]. For an invertible $\mathbf{M}$, we write $\mathbf{M} \ge \eta_\epsilon(\mathcal{L})$ if $\eta_\epsilon(\mathbf{M}^{-1}\mathcal{L}) \le 1$. In particular, $\mathbf{M}\mathbf{M}^t \succeq s^2\mathbf{I}$ and $s \ge \eta_\epsilon(\mathcal{L})$ imply $\mathbf{M} \ge \eta_\epsilon(\mathcal{L})$.

Lemma 1 ([Pei10, Lemma 2.4])**.** *Let $0 < \epsilon < 1$ and $\mathbf{M} \ge \eta_\epsilon(\mathcal{L})$. Then, for every $\mathbf{c} \in \mathrm{span}(\mathcal{L})$, $\rho_{\mathbf{M},\mathbf{c}}(\mathcal{L}) \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \rho_{\mathbf{M}}(\mathcal{L})$.*

2.2 Rényi divergence

For distributions $\mathcal{P}, \mathcal{Q}$ with $\mathsf{Supp}(\mathcal{P}) \subseteq \mathsf{Supp}(\mathcal{Q})$ and $a > 1$, the Rényi divergence of order a is

$$R_a(\mathcal{P}\|\mathcal{Q}) = \left(\sum_x \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}}\right)^{1/(a-1)}.$$

The Rényi divergence satisfies the following two properties:

- $R_a(f(\mathcal{P})\|f(\mathcal{Q})) \le R_a(\mathcal{P}\|\mathcal{Q})$ for any function f;
- $\mathcal{P}(E) \le (\mathcal{Q}(E)R_a(\mathcal{P}\|\mathcal{Q}))^{(a-1)/a}$ for any event $E \subseteq \mathsf{Supp}(\mathcal{Q})$ and $a > 1$.

2.3 Power-of-Two Cyclotomics and NTRU

For $n = 2^\ell$ with $\ell \ge 1$, let $\mathcal{K} = \mathbb{Q}(\zeta_n) \simeq \mathbb{Q}[X]/(X^n+1)$, where ζ_n is a primitive $2n$-th root of unity, and let $\mathcal{R} = \mathbb{Z}[\zeta_n] \simeq \mathbb{Z}[X]/(X^n+1)$ be its ring of integers. For $f = \sum_{i=0}^{n-1} f_i X^i \in \mathcal{K}$, we identify f with its coefficient vector $(f_0, \ldots, f_{n-1})$. For a polynomial $f = \sum_{i=0}^{n-1} f_i X^i$, we write $\|f\|_2 = (\sum_{i=0}^{n-1} |f_i|^2)^{1/2}$.

The field $\mathcal{K}$ has n complex embeddings $\varphi_k : \mathcal{K} \to \mathbb{C}$, defined by $\varphi_k(f) = f(\zeta_n^{2k+1})$ for $0 \le k < n$. The canonical embedding is $\varphi(f) = (\varphi_k(f))_{k=0}^{n-1}$. It is natural to extend each φ_k entrywise to vectors and matrices over $\mathcal{K}$. In particular, for $\mathbf{H} = (H_{ij}) \in \mathcal{K}^{2\times 2}$,

$$\varphi_k(\mathbf{H}) = \begin{pmatrix} \varphi_k(H_{11}) & \varphi_k(H_{12}) \\ \varphi_k(H_{21}) & \varphi_k(H_{22}) \end{pmatrix} \in \mathbb{C}^{2\times 2}.$$

Then $\varphi_k(\mathbf{H}\mathbf{G}) = \varphi_k(\mathbf{H})\varphi_k(\mathbf{G})$.

For $f \in \mathcal{K}$, its matrix of multiplication in the basis $1, \zeta_n^1, \cdots, \zeta_n^{n-1}$ is denoted by $\mathcal{A}(f)$. Let $f^* = f(X^{-1})$. Then $\varphi_k(f^*) = \overline{\varphi_k(f)}$ and $\mathcal{A}(f^*) = \mathcal{A}(f)^t$. For $\mathbf{x}, \mathbf{y} \in \mathcal{K}^m$, let $\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{K}} = \sum_i x_i^* y_i$. For a matrix $\mathbf{H}$ over $\mathcal{K}$, we use $\mathbf{H}^*$ for its conjugate transpose, i.e., $(\mathbf{H}^*)_{ij} = H_{ji}^*$. Rounding $\lfloor\cdot\rceil$ is applied coefficientwise to polynomials and polynomial matrices. For $\mathbf{H} \in \mathcal{K}^{2\times 2}$, define $\|\mathbf{H}\|_2 = \max_{0\le k<n} \sigma_{\max}(\varphi_k(\mathbf{H}))$, $\sigma_{\min}(\mathbf{H}) = \min_{0\le k<n} \sigma_{\min}(\varphi_k(\mathbf{H}))$. If $\mathbf{H}$ is invertible, then $\|\mathbf{H}^{-1}\|_2 = 1/\sigma_{\min}(\mathbf{H})$, and $\|\mathbf{H}\|_F^2 = \sum_{k=0}^{n-1} \|\varphi_k(\mathbf{H})\|_F^2 = n\sum_{i,j} \|H_{ij}\|_{\mathrm{coef},2}^2$,

[1] The corresponding continuous Gaussian has covariance $\mathbf{\Sigma}/(2\pi)$.

where $\|\cdot\|_{\mathrm{coef},2}$ denotes the Euclidean norm of the coefficient vector. For $\mathbf{\Sigma} \in \mathcal{K}^{2\times 2}$, we write $\mathbf{\Sigma} \succ 0$ when $\mathbf{\Sigma} = \mathbf{\Sigma}^*$ and $\varphi_k(\mathbf{\Sigma})$ is positive definite for all k.

For an integer $q \geq 2$, an NTRU secret key consists of two short polynomials $f, g \in \mathcal{R}$, and the public key is $h = f^{-1}g \bmod q$. The corresponding NTRU module is $\Lambda_{h,q} = \{(u, v) \in \mathcal{R}^2 : u + vh = 0 \bmod q\mathcal{R}\}$, viewed as a $2n$-dimensional lattice under the coefficient embedding. Solving the NTRU equation $fG - gF = q$ yields the trapdoor basis

$$\mathbf{B} = \begin{pmatrix} g & G \\ -f & -F \end{pmatrix}, \qquad \mathbf{B}^{-1} = \frac{1}{q}\begin{pmatrix} -F & -G \\ f & g \end{pmatrix}.$$

Let $\mathbf{U}_a = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \in \mathcal{K}^{2\times 2}$. Then $\mathbf{U}_a^{-1} = \mathbf{U}_{-a}$. The Gram–Schmidt orthogonalization of the NTRU trapdoor basis $\mathbf{B}$ over $\mathcal{K}$ gives $\mathbf{B} = \widetilde{\mathbf{B}}\mathbf{U}_u$ such that

$$\widetilde{\mathbf{B}} = \begin{pmatrix} g & qf^*/a \\ -f & qg^*/a \end{pmatrix} \quad \text{with} \quad a = f^*f + g^*g$$

and the associated Gram-Schmidt coefficient $u = (g^*G + f^*F)/a$.

2.4 GPV Hash-and-Sign Signatures

Mitaka is a practical NTRU-based instantiation of the GPV hash-and-sign framework [GPV08], in which signing boils down to lattice Gaussian sampling with a short trapdoor basis. Algorithm 1 presents a brief description of the GPV signing procedure. The vector (s_1, s_2) is a centered Gaussian on $(t, 0)^t + \Lambda_{h,q}$; hence, it is short and satisfies $s_1 + s_2 h = H(\mathsf{salt} \,\|\, m) \bmod q$ for the public key h. In NTRU-based GPV signatures, the signature transcript consists of s_2 only, as s_1 can be recovered from s_2 and h.

Algorithm 1: NTRU hash-and-sign

Input: A message m, an expanded secret key sk, and a bound B_{sig}.
Output: A signature sig of m.

repeat
  $\mathsf{salt} \leftarrow \mathsf{FreshSalt}()$
  $t \leftarrow H(\mathsf{salt} \,\|\, m)$
  $\mathbf{z} \leftarrow \mathsf{SampleLattice}(\mathsf{sk}, (t, 0)^t)$
  $\mathbf{s} = (s_1, s_2)^t \leftarrow (t, 0)^t - \mathbf{z}$
until $\|\mathbf{s}\|_2^2 \leq B_{\mathrm{sig}}^2$
return $\mathsf{sig} = (\mathsf{salt}, s_2)$

2.5 Fixed-Point and Exact Integer Arithmetic

A signed w-bit integer ranges over $[-2^{w-1}, 2^{w-1} - 1]$, whereas an unsigned integer ranges over $[0, 2^w - 1]$. Exact integer arithmetic requires intermediate values as well as final results to fit in their respective representations. Arithmetic modulo an integer is exact in the corresponding residue ring. For pairwise coprime moduli with product P, the Chinese Remainder Theorem (CRT) uniquely determines an integer modulo P; its centered representative recovers the original signed integer whenever its absolute value is less than $P/2$.

A fixed-point number with f fractional bits represents $x = X2^{-f}$ for an integer X. A signed w-bit fixed-point format therefore has range $[-2^{w-f-1}, 2^{w-f-1} - 2^{-f}]$ and spacing 2^{-f}. Addition and subtraction of values with the same scale operate on their integer representations. Multiplication produces a result with scale 2^{-2f}; restoring scale 2^{-f} requires rescaling and, if necessary, rounding. The intermediate product must also fit in its chosen integer representation.

3 Reducing the Sampling Cost of Mitaka$_{\mathbb{Z}}$

3.1 The Mitaka$_{\mathbb{Z}}$ Sampler

Let us first recall the Mitaka$_{\mathbb{Z}}$ sampler. At a high level, it follows Peikert's convolution sampling paradigm [Pei10] while using the NTRU Gram-Schmidt basis, $\widetilde{\mathbf{B}} = \mathbf{B} \cdot \mathbf{U}_u^{-1}$, as the linear transformation to reduce the Gaussian width. To avoid high-precision arithmetic, the Mitaka$_{\mathbb{Z}}$ sampler first employs rational approximations of $\widetilde{\mathbf{B}}$ and $\mathbf{U}_u$. Specifically, it replaces $u \in \mathcal{K}$ by $\widehat{u} = \lfloor p_u u \rceil / p_u \in \frac{1}{p_u}\mathcal{R}$ for some integer p_u and sets $\widehat{\mathbf{B}} = \mathbf{B}\mathbf{U}_{-\widehat{u}} \in \frac{1}{p_u}\mathcal{R}^{2\times 2}$, which satisfies $\widehat{\mathbf{B}}\mathbf{U}_{\widehat{u}} = \mathbf{B}$. This allows online sampling to be performed entirely with integer arithmetic. To eliminate floating-point arithmetic from offline sampling, the Mitaka$_{\mathbb{Z}}$ sampler uses the integral perturbation sampling technique [DGPY20]. At the core of this approach is the integral Gram root decomposition of the perturbation covariance $\mathbf{\Sigma}_p = s^2\mathbf{I}_{2n} - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^*$, which outputs an integral but wide matrix $\mathbf{A}_Z$ such that $\mathbf{A}_Z\mathbf{A}_Z^* = p_u^2(\mathbf{\Sigma}_p - \mathbf{I}_{2n})$. The perturbation is computed by first multiplying $\mathbf{A}_Z$ by a Gaussian vector and then rounding the scaled product to an integer vector via Gaussian sampling. The latter step, termed Gaussian rounding, is used to ensure the integrality of the output perturbation. The detailed algorithmic description is given in Algorithm 2.

The Gaussian width achieved by the Mitaka$_{\mathbb{Z}}$ sampler is rs where $r \geq \eta_\epsilon(\mathbb{Z}^{2n})$ represents the width of integer Gaussians in the online sampling and $s > \|\widetilde{\mathbf{B}}\|_2 \cdot \left(1 + \sqrt{2n/p_u}\right) + 1$. For sufficiently large p_u, the minimum Gaussian width for Mitaka$_{\mathbb{Z}}$ approaches $\eta_\epsilon(\mathbb{Z}^{2n}) \cdot \|\widetilde{\mathbf{B}}\|_2$, matching the width of the Mitaka sampler. While achieving the same Gaussian width and better portability, the Mitaka$_{\mathbb{Z}}$ sampler requires $O(n \log n)$ Gaussian samples for perturbation sampling, which makes Mitaka$_{\mathbb{Z}}$ far less efficient.

3.2 An Improved Mitaka$_{\mathbb{Z}}$ Sampler

To overcome the efficiency bottleneck of Mitaka$_{\mathbb{Z}}$, we propose an efficient variant of the Mitaka$_{\mathbb{Z}}$ sampler, substantially reducing the sampling cost while preserving simplicity and portability. Fixed-point arithmetic allows us to replace the wide exact Gram root with an approximate square Cholesky root. We compute the structured Cholesky root $\mathbf{A}$ of the perturbation covariance $\mathbf{\Sigma}_p$ and represent it by $\widehat{\mathbf{A}} = \lfloor p_A \mathbf{A} \rceil / p_A$ at a sufficient precision p_A. This reduces the number of required integer Gaussian samples for perturbation generation by a factor of $O(\log n)$. The resulting discrepancy can be measured by bounding $\mathbf{\Sigma}_e = \mathbf{\Sigma}_p - \widehat{\mathbf{A}}\widehat{\mathbf{A}}^*$.

During offline key expansion, we evaluate $\mathbf{\Sigma}_p = s^2\mathbf{I} - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^*$ at each root of $X^n + 1$. For each resulting Hermitian matrix $C = (c_{ij})$, we compute its lower-triangular Cholesky factor L by

$$L_{00} = \sqrt{c_{00}}, \qquad L_{10} = \frac{c_{10}}{L_{00}}, \qquad L_{11} = \sqrt{c_{11} - |L_{10}|^2}, \qquad L_{01} = 0.$$

An inverse Fourier transform recovers the polynomial coefficients of $\mathbf{A}$, which are scaled by p_A and rounded to integers for storage. The supplementary C reference expansion uses `long double` arithmetic for these offline computations; signing uses the stored integer coefficients.

We further observe that the Gaussian rounding step can be removed with a small increase in the cost of online sampling. The centers of online integer Gaussian samples can be computed with exact integer arithmetic when the perturbation has a fixed denominator. We therefore omit the Gaussian rounding and directly use the scaled product of the approximate Cholesky root $\widehat{\mathbf{A}}$ and an integer vector sampled from a discrete Gaussian distribution. This further reduces the number of required integer samples for perturbation generation from $4n$ to $2n$.

Algorithm 2: Mitaka$_{\mathbb{Z}}$ Sampler

Input: The signing key $\widehat{\mathbf{B}}, \widehat{u}, \mathbf{A}_Z$; a center $\mathbf{c} \in \mathcal{R}^2$; parameters r, m, L_0, p_u.
Output: A lattice point $\mathbf{z} \in \mathcal{L}(\mathbf{B})$.
Perturbation sampling:
$\mathbf{x} \leftarrow D_{\mathbb{Z}^{mn}, L_0 r}$
$\mathbf{p}' \leftarrow \frac{1}{L_0 p_u} \mathbf{A}_Z \mathbf{x}$
$\mathbf{p} \leftarrow D_{\mathbb{Z}^{2n}, r, \mathbf{p}'}$
Online sampling:
$\widehat{\mathbf{c}} \leftarrow \widehat{\mathbf{B}}^{-1}(\mathbf{c} - \mathbf{p})$ // $\widehat{\mathbf{c}} \in \frac{1}{L_0 p_u q}\mathcal{R}^2$
$z_2 \leftarrow D_{\mathbb{Z}^n, r, \widehat{c}_2}$
$c_1' \leftarrow \widehat{c}_1 - \widehat{u} z_2$
$z_1 \leftarrow D_{\mathbb{Z}^n, r, c_1'}$
return $\mathbf{z} \leftarrow \mathbf{B}(z_1, z_2)^t$

Algorithm 3: New Mitaka$_{\mathbb{Z}}$ Sampler

Input: The signing key $\widehat{\mathbf{B}}, \widehat{u}, \widehat{\mathbf{A}}$; a center $\mathbf{c} \in \mathcal{R}^2$; parameters r, L, p_u, p_A.
Output: A lattice point $\mathbf{z} \in \mathcal{L}(\mathbf{B})$.
Perturbation sampling:
$\mathbf{x} \leftarrow D_{\mathbb{Z}^{2n}, Lr}$
$\mathbf{p} \leftarrow \frac{1}{L} \widehat{\mathbf{A}} \mathbf{x}$
Online sampling:
$\widehat{\mathbf{c}} \leftarrow \widehat{\mathbf{B}}^{-1}(\mathbf{c} - \mathbf{p})$ // $\widehat{\mathbf{c}} \in \frac{1}{L p_A p_u q}\mathcal{R}^2$
$z_2 \leftarrow D_{\mathbb{Z}^n, r, \widehat{c}_2}$
$c_1' \leftarrow \widehat{c}_1 - \widehat{u} z_2$
$z_1 \leftarrow D_{\mathbb{Z}^n, r, c_1'}$
return $\mathbf{z} \leftarrow \mathbf{B}(z_1, z_2)^t$

Our new Mitaka$_{\mathbb{Z}}$ sampler is described in Algorithm 3. As we shall show later, our new Mitaka$_{\mathbb{Z}}$ sampler does not incur any overhead on the Gaussian width. Table 1 compares the sampling costs of the Mitaka$_{\mathbb{Z}}$ sampler, the Mitaka sampler, and our new sampler.

Table 1: Comparison of the sampling cost for perturbation generation.

	Continuous Gaussians	Integer Gaussians	
		zero-centered	arbitrarily centered
Mitaka sampler	$2n$	0	0
Mitaka$_{\mathbb{Z}}$ sampler	0	$O(n \log n)$	$2n$
Our new sampler	0	$2n$	0

3.3 Correctness Analysis

This section shows the correctness of our new Mitaka$_{\mathbb{Z}}$ sampler (Algorithm 3) under certain parameter conditions. We focus on the annular NTRU trapdoor [ENS+23] such that for every primitive $2n$-th root of unity ζ,

$$\frac{q}{\alpha^2} \le |f(\zeta)|^2 + |g(\zeta)|^2 \le \alpha^2 q$$

for some $\alpha \ge 1$. Our analysis assumes exact integer arithmetic and perfect integer Gaussian samplers as in [DFLP26].

For the precision parameters p_u and p_A, we define

$$\beta(p_u) = \frac{\sqrt{q}}{2}\left(\sqrt{(\alpha + \alpha^{-1})^2 + \frac{\alpha^2 \csc^2(\pi/(2n))}{4p_u^2}} + \sqrt{(\alpha - \alpha^{-1})^2 + \frac{\alpha^2 \csc^2(\pi/(2n))}{4p_u^2}}\right)$$

and

$$\bar{\tau}(p_u, p_A) := \frac{(1+\sqrt{5})\csc(\pi/(2n))}{2sp_A}\sqrt{1 - \frac{q^2}{s^2\beta(p_u)^2}} + \frac{(3+\sqrt{5})\csc^2(\pi/(2n))}{8s^2p_A^2}.$$

We write $\bar{\tau} = \bar{\tau}(p_u, p_A)$ for short. Let $\mathcal{P}$ denote the output distribution of Algorithm 3 and $\mathcal{Q}_0 = D_{\mathcal{L}(\mathbf{B}), rs, \mathbf{c}}$. For $a > 1$ and $\kappa = (1+\epsilon)/(1-\epsilon)$, we further define

$$\mathcal{E}_a := \frac{9a}{4}(\log \kappa)^2 + \frac{3an^2 \sin^2(\pi/(2n))\,\bar{\tau}^2}{(3+\sqrt{5})(1+\bar{\tau})(1-(2a-1)\bar{\tau})} + \frac{5a\bar{\tau}^2}{(1-(2a-1)\bar{\tau})^2}.$$

Theorem 1 (Correctness of Algorithm 3)**.** *Let $0 < \epsilon \le 1/2$ and $a > 1$. Suppose that the following conditions hold:*

1. *$q/\alpha^2 \le |f(\zeta)|^2 + |g(\zeta)|^2 \le \alpha^2 q$ for every primitive $2n$-th root of unity ζ, with $\alpha \ge 1$;*
2. *$s > \beta(p_u)$;*
3. *$r \ge \eta_\epsilon(\mathbb{Z}^n)$ and $L \ge \frac{\beta(p_u)s\sqrt{1+\bar{\tau}}}{qr}\eta_\epsilon(\mathbb{Z}^{2n})$;*
4. *$(2a-1)\bar{\tau} < 1$ and $rs\sqrt{\frac{(1-\sqrt{\epsilon/2})(1-\bar{\tau})}{1+(2a-1)\bar{\tau}}} \ge \eta_\epsilon(\mathcal{L}(\mathbf{B}))$.*

Then $\log R_a(\mathcal{P}\|\mathcal{Q}_0) \le \mathcal{E}_a$.

The proof of Theorem 1 needs the following three lemmas. We defer their proofs to Appendix A.

Lemma 2. *Let $\mathbf{B}$ be the NTRU trapdoor basis satisfying the condition in Theorem 1 and u be its Gram-Schmidt coefficient. Then $\|\widehat{\mathbf{B}}\|_2 \le \beta(p_u)$ and $\sigma_{\min}(\widehat{\mathbf{B}}) \ge q/\beta(p_u)$. In particular, $s^2\mathbf{I}_{2n} - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^* \succ 0$ when $s > \beta(p_u)$.*

Lemma 3. *Using the notation of Algorithm 3, let $\mathbf{\Sigma}_e = \mathbf{\Sigma}_p - \widehat{\mathbf{A}}\widehat{\mathbf{A}}^*$ and $s > \beta(p_u)$. Then $\|\mathbf{\Sigma}_e\|_2 \le \bar{\tau}s^2$ and $\|\mathbf{\Sigma}_e\|_F \le \frac{2\sqrt{3}\,n\sin(\pi/(2n))}{1+\sqrt{5}}\,\bar{\tau}s^2$.*

Lemma 4. *Under the conditions in Theorem 1, the output distribution $\mathcal{P}$ has support $\mathcal{L}(\mathbf{B})$ and can be written as $\mathcal{P}(\mathbf{z}) = \mathcal{Q}_e(\mathbf{z})\frac{a_\mathbf{z}}{\mathbb{E}_{\mathcal{Q}_e}[a_\mathbf{z}]}$ with*

$$\mathcal{Q}_e = D_{\mathcal{L}(\mathbf{B}),r\sqrt{s^2\mathbf{I}_{2n}-\mathbf{\Sigma}_e},\mathbf{c}}$$

and $a_\mathbf{z} \in [\kappa^{-1}, \kappa^2]$. Consequently, for every $v > 1$, $\log R_v(\mathcal{P}\|\mathcal{Q}_e) \le \frac{9v}{8}(\log\kappa)^2$.

Proof of Theorem 1. By Hölder's inequality and Lemma 4, we have

$$\begin{aligned}\log R_a(\mathcal{P}\|\mathcal{Q}_0) &\le \frac{2a}{2a-1}\log R_{2a-1}(\mathcal{P}\|\mathcal{Q}_e) + \log R_{2a}(\mathcal{Q}_e\|\mathcal{Q}_0)\\ &\le \frac{9a}{4}(\log\kappa)^2 + \log R_{2a}(\mathcal{Q}_e\|\mathcal{Q}_0).\end{aligned}$$

Let $\theta_1, \ldots, \theta_{2n}$ be the eigenvalues of $s^{-2}\mathbf{\Sigma}_e$. Then $|\theta_i| \le \bar{\tau}$ by Lemma 3. For $t \in [0, 2a]$, define

$$\mathbf{V}_t = r^2(s^2\mathbf{I}_{2n} - \mathbf{\Sigma}_e)(\mathbf{I}_{2n} + (t-1)s^{-2}\mathbf{\Sigma}_e)^{-1}.$$

Its eigenvalues are $r^2s^2(1-\theta_i)/(1+(t-1)\theta_i)$, and hence $(2a-1)\bar{\tau} < 1$ ensures that $\mathbf{V}_t$ is positive definite. Set $\phi(t) = \log\big(\det(\mathcal{L}(\mathbf{B}))\det(\mathbf{V}_t)^{-1/2}\rho_{\sqrt{\mathbf{V}_t},\mathbf{c}}(\mathcal{L}(\mathbf{B}))\big)$. Then

$$\log R_{2a}(\mathcal{Q}_e\|\mathcal{Q}_0) = \frac{1}{2(2a-1)}\log\frac{\det(\mathbf{V}_{2a})\det(\mathbf{V}_0)^{2a-1}}{\det(\mathbf{V}_1)^{2a}} + \frac{\phi(2a) - 2a\phi(1) + (2a-1)\phi(0)}{2a-1}.$$

For the determinant term, diagonalizing $\mathbf{\Sigma}_e$ yields

$$\begin{aligned}\frac{1}{2(2a-1)}\log\frac{\det(\mathbf{V}_{2a})\det(\mathbf{V}_0)^{2a-1}}{\det(\mathbf{V}_1)^{2a}} &= a\sum_{i=1}^{2n}\theta_i^2\int_0^1\frac{x\,dx}{(1-x\theta_i)(1+(2a-1)x\theta_i)}\\ &\le \frac{a\|\mathbf{\Sigma}_e\|_F^2}{2s^4(1+\bar{\tau})(1-(2a-1)\bar{\tau})}.\end{aligned}$$

For the lattice-mass term, the eigenvalue bound gives

$$\mathbf{V}_t \succeq r^2s^2(1-\bar{\tau})(1+(2a-1)\bar{\tau})^{-1}\mathbf{I}_{2n}.$$

Hence the target-width condition ensures that $\sqrt{(1-\sqrt{\epsilon/2})\mathbf{V}_t}$ remains above $\eta_\epsilon(\mathcal{L}(\mathbf{B}))$ for all $t \in [0, 2a]$.

To bound the lattice-mass term, we apply the Poisson summation formula to the normalized Gaussian mass defining $\phi(t)$. Writing $\Lambda = \mathcal{L}(\mathbf{B})$ and $q_t(\mathbf{v}) = \pi \mathbf{v}^t \mathbf{V}_t \mathbf{v}$, we obtain $e^{\phi(t)} = \sum_{\mathbf{v}\in\Lambda^*} e^{-q_t(\mathbf{v})} e^{-2\pi\imath\langle \mathbf{v},\mathbf{c}\rangle}$. The center $\mathbf{c}$ appears in the phase factor, whose absolute value is one. Termwise differentiation gives

$$
\begin{aligned}
(e^{\phi(t)})' &= -\sum_{\mathbf{v}\in\Lambda^*} q_t'(\mathbf{v}) e^{-q_t(\mathbf{v})} e^{-2\pi\imath\langle \mathbf{v},\mathbf{c}\rangle}, \\
(e^{\phi(t)})'' &= \sum_{\mathbf{v}\in\Lambda^*} \big(q_t'(\mathbf{v})^2 - q_t''(\mathbf{v})\big) e^{-q_t(\mathbf{v})} e^{-2\pi\imath\langle \mathbf{v},\mathbf{c}\rangle}.
\end{aligned}
$$

Let $\delta = \bar{\tau}/(1-(2a-1)\bar{\tau})$ and $\xi = \sqrt{\epsilon/2}$. The eigenvalue bounds on $\mathbf{V}_t$ imply $|q_t'(\mathbf{v})| \le \delta q_t(\mathbf{v})$ and $|q_t''(\mathbf{v})| \le 2\delta^2 q_t(\mathbf{v})$. By the target-width condition, $\sum_{\mathbf{v}\neq 0} e^{-(1-\xi)q_t(\mathbf{v})} \le \epsilon$, which also implies $e^{\phi(t)} \ge 1-\epsilon$. Using $xe^{-\xi x} \le 1/(e\xi)$ and $x^2 e^{-\xi x} \le 4/(e^2\xi^2)$, we obtain $|(e^{\phi(t)})'| \le \delta\epsilon/(e\xi)$ and $|(e^{\phi(t)})''| \le \delta^2\epsilon(4/(e^2\xi^2) + 2/(e\xi))$. Consequently, since $\epsilon \le 1/2$,

$$
|\phi''(t)| \le \frac{|(e^{\phi(t)})''|}{e^{\phi(t)}} + \frac{|(e^{\phi(t)})'|^2}{e^{2\phi(t)}} \le \frac{5\bar{\tau}^2}{(1-(2a-1)\bar{\tau})^2}.
$$

Combining the determinant and lattice-mass terms gives

$$
\begin{aligned}
\log R_{2a}(\mathcal{Q}_e \| \mathcal{Q}_0) &\le \frac{a\|\mathbf{\Sigma}_e\|_F^2}{2s^4(1+\bar{\tau})(1-(2a-1)\bar{\tau})} + \frac{5a\bar{\tau}^2}{(1-(2a-1)\bar{\tau})^2} \\
&\le \frac{3an^2\sin^2(\pi/(2n))\,\bar{\tau}^2}{(3+\sqrt{5})(1+\bar{\tau})(1-(2a-1)\bar{\tau})} + \frac{5a\bar{\tau}^2}{(1-(2a-1)\bar{\tau})^2}.
\end{aligned}
$$

Combining this bound with the Hölder bound above completes the proof of Theorem 1. □

3.4 Fixed-Point Integer Gaussian Sampling

Algorithm 3 is implemented with exact integer arithmetic except for the integer Gaussian sampler. For these samples, we reuse the fixed-point SamplerZ implementation of Falcon [DFLP26]. The perturbation samples are centered at zero, while the two online stages sample around the rational centers $\widehat{c}_2$ and c_1'. All polynomial operations and center computations remain exact until these centers are passed to SamplerZ; only their fractional parts are converted to fixed point. We are therefore able to provably bound the errors on the Gaussian centers, unlike in the fixed-point Falcon case [DFLP26].

As in the fixed-point Falcon analysis, we model SamplerZ as a perfect Gaussian sampler for its represented inputs. Thus a call with represented center c^{fp} and width r is viewed as an exact sample from $D_{\mathbb{Z},r,c^{\mathrm{fp}}}$. The Gaussian width r is fixed and represented exactly in our implementation. Hence center quantization is the only input error considered in this step. The resulting distributional loss follows directly from the center-perturbation bound of Braga *et al.* [DFLP26, Lemma 4].

Lemma 5 ([DFLP26, Lemma 4])**.** *Let $r \ge \eta_\epsilon(\mathbb{Z})$, with $0 < \epsilon \le 1/4$, and let $a > 1$. If $|c^{\mathrm{fp}} - c| \le \Delta_c$, then*

$$
\log R_a\big(D_{\mathbb{Z},r,c^{\mathrm{fp}}} \,\big\|\, D_{\mathbb{Z},r,c}\big) \le \frac{\pi a(a+1)}{a-1}\frac{\Delta_c^2}{r^2}\big(1+O(\epsilon)\big).
$$

Only the $2n$ online integer samples incur this center-representation error; the perturbation samples have exact center zero. Since Lemma 5 holds uniformly in the center, it also applies to the history-dependent centers arising in the sequential online sampler. Applying

the Rényi composition bound to the $2n$ sequential calls to the integer Gaussian sampler gives

$$\log R_a(\mathcal{P}_{\mathrm{fp}} \| \mathcal{P}) \leq \frac{2\pi n a(a+1)}{a-1} \frac{\Delta_c^2}{r^2} \big(1 + O(\epsilon)\big).$$

4 Implementation Details

4.1 Concrete Parameters

We implement the Mitaka signature scheme with our new sampler (Algorithm 3). Our implementation uses the annular trapdoor generation technique in [ENS+23] and focuses on the corresponding parameter sets with $n \in \{512, 1024\}$ and $q = 12289$ for the NIST-I and NIST-V security levels, respectively. Both parameter sets satisfy the correctness and precision requirements presented in Section 3: the Rényi loss incurred by using the new sampler is at most 1.5 bits for up to $Q_s = 2^{64}$ signatures. Detailed parameters are summarized in Table 2.

Table 2: Concrete parameters in the fixed-point implementation of Mitaka$_{\mathbb{Z}}$.

	Mitaka-512	Mitaka-1024
Security parameter λ	128	256
Ring degree n	512	1024
NTRU modulus q	12289	12289
Trapdoor quality parameter α	1.15	1.23
Smoothing error ϵ	2^{-38}	$2^{-39.5}$
Denominator of $\widehat{u}$: p_u	2^{16}	2^{16}
Denominator of $\widehat{\mathbf{A}}$: p_A	2^{42}	2^{43}
Perturbation scaling factor L	2	2
Scaled Gaussian width s	127.49	138
Online integer Gaussian width r	3.305	3.34
CDT maximum for online integer Gaussian	14	14
CDT maximum for offline integer Gaussian	28	29
Perturbation denominator $d_p = Lp_A$	2^{43}	2^{44}
Second-center denominator $d_2 = qd_p$	$q \cdot 2^{43}$	$q \cdot 2^{44}$
First-center denominator $d_1 = p_u d_2$	$q \cdot 2^{59}$	$q \cdot 2^{60}$
Squared-norm bound for signatures	31894693	76344857
Measured failure probability per attempt	1.196%	0.0722%
Key recovery security (C/Q)	127/113	264/234
Forgery security (C/Q)	121/107	274/243

Key expansion is performed offline and stores the integer representations and transformed key-dependent quantities required in the online phase. The expanded key stores $\widehat{u}$ and the lower-triangular $\widehat{\mathbf{A}}$ as $u_{\mathrm{int}} = p_u \widehat{u} = \lfloor p_u u \rceil$ and $\mathbf{A}_{\mathrm{int}} = p_A \widehat{\mathbf{A}} = \lfloor p_A \mathbf{A} \rceil$ respectively. The coefficient bound $\|f\|_\infty, \|g\|_\infty, \|F\|_\infty, \|G\|_\infty \leq 127$ is inherited from the Falcon key representation [PFH+20] and retained by the implementation of [ENS+23]. For the Gram-Schmidt coefficient u, it holds that $|u_{\mathrm{int},i}| \leq p_u/2$ under the size-reduced condition. By

Lemma 2, we have

$$|(A_{\text{int}})_i| \le p_A\sqrt{s^2 - \frac{q^2}{\beta(p_u)^2}} + \frac{1}{2} < \begin{cases} 2^{49}, & n = 512, \\ 2^{50}, & n = 1024. \end{cases}$$

Thus these numbers fit in signed 64-bit words.

4.2 Integer Gaussian Samplers

Perturbation Sampling. The integer Gaussian samples for perturbation generation have center zero and width Lr. They are sampled via the CDT approach with a random sign flip. The tables contain $T_{\text{offline}} = 28$ thresholds for $n = 512$ and $T_{\text{offline}} = 29$ thresholds for $n = 1024$, giving magnitudes in $\{0, \ldots, 28\}$ and $\{0, \ldots, 29\}$, respectively. Each threshold has 88 bits.

Online Sampling. The online integer Gaussian sampler uses a fixed-point implementation adapted from the Falcon Gaussian sampler [DFLP26]. It draws a magnitude $z_0 \in \{0, \ldots, 14\}$ using $T_{\text{online}} = 14$ CDT thresholds of 88 bits each, then a uniform bit b gives the bimodal proposal $z = b + (2b - 1)z_0$. Rejection sampling produces the requested distribution. Fixed-point center computation is described in Section 4.4.

The acceptance probability is $\exp(-X)$, where the nonnegative exponent X uses a pair of unsigned 64-bit words. Native 128-bit arithmetic is needed on the CPU only for full products whose upper halves are used; the MCU computes those products from 32-bit limbs. The exponential polynomial has unsigned 64-bit coefficients.

The integer and fixed-point representations used in integer Gaussian sampling are summarized in Table 3.

Table 3: The data types of intermediate values used in integer Gaussian samplers.

Quantity	Representation
CDT entries	$3 \times$ `uint32_t`
Q64 center fraction	`uint64_t`
Exponent X	$2 \times$ `uint64_t`
Coefficients of the ApproxExp polynomial	`uint64_t`

Vectorization. For offline perturbation sampling, we adapt the vectorized CDT layout of VecFalcon [ZZ25], scanning our 88-bit thresholds over eight independent inputs in parallel using AVX2. For online sampling, each uniform sample consists of two 32-bit words and one 24-bit word. We broadcast one sample and compare its CDT thresholds in groups of eight AVX2 lanes. CDT lookups implement lexicographic comparison.

4.3 Exact Center Computations

The online sampling computes the two coordinates of $\widehat{\mathbf{B}}^{-1}(\mathbf{c} - \mathbf{p})$ exactly, where $\mathbf{c} = (t, 0)^t$ and $\mathbf{p} = \mathbf{A}_{\text{int}}\mathbf{x}/d_p$ with $d_p = Lp_A$. We move all key-dependent polynomial products to key expansion and retain exact integer numerator–denominator representations online.

The centers of the online integer Gaussian samplers are retained as exact rational numbers until they are passed to SamplerZ. Then the center $\widehat{c}_2$ has denominator $d_2 = qd_p$ and the center $c_1' = \widehat{c}_1 - u_{\text{int}}z_2/p_u$ has denominator $d_1 = p_u d_2$. Neither approximate division nor fixed-point conversion is introduced in the center computations.

Expanding $\widehat{\mathbf{B}}^{-1}$ gives $m_{11} = u_{\text{int}} f - p_u F$ and $m_{12} = u_{\text{int}} g - p_u G$. Since $\mathbf{A}_{\text{int}}$ is lower triangular, key expansion precomputes

$$\begin{aligned} C_{1,0} &= m_{11}(A_{\text{int}})_{00} + m_{12}(A_{\text{int}})_{10}, \quad & C_{1,1} &= m_{12}(A_{\text{int}})_{11}, \\ C_{2,0} &= f(A_{\text{int}})_{00} + g(A_{\text{int}})_{10}, & C_{2,1} &= g(A_{\text{int}})_{11}. \end{aligned}$$

Each coefficient is then split exactly as $C_{j,k} = d_p C^{\text{hi}}_{j,k} + C^{\text{lo}}_{j,k}$ with $-d_p/2 \le C^{\text{lo}}_{j,k} < d_p/2$. For the fresh perturbation vector $\mathbf{x}$, define $L_j = -C^{\text{lo}}_{j,0} x_0 - C^{\text{lo}}_{j,1} x_1$ and $H_2 = ft - C^{\text{hi}}_{2,0} x_0 - C^{\text{hi}}_{2,1} x_1$, $H'_1 = m_{11} t - C^{\text{hi}}_{1,0} x_0 - C^{\text{hi}}_{1,1} x_1 - q u_{\text{int}} z_2$. The exact center numerators are therefore $N_2 = d_p H_2 + L_2$ and $M_1 = d_p H'_1 + L_1$, so that $\widehat{c}_2 = N_2/d_2$ and $c'_1 = M_1/d_1$. The high bits/low bits split is only an exact change of integer representation: it introduces no approximation, but reduces the coefficient ranges that must be reconstructed by CRT. We present bounds on these intermediate values below.

Bound on L_j. Each coefficient of $C^{\text{lo}}_{j,k}$ has absolute value at most $d_p/2$, and that of $\mathbf{x}$, a Gaussian sample of $D_{\mathbb{Z},Lr}$, is bounded by T_{offline}. Therefore,

$$\|L_j\|_\infty \le T_{\text{offline}}(\|C^{\text{lo}}_{j,0}\|_1 + \|C^{\text{lo}}_{j,1}\|_1) \le n d_p T_{\text{offline}}.$$

Bound on $C^{\text{hi}}_{j,k}$. The decomposition of $C_{j,k}$ gives $\|C^{\text{hi}}_{j,k} - C_{j,k}/d_p\|_1 \le n/2$. By the condition (1) of Theorem 1, we have $\|(f,g)\|_2 \le \alpha\sqrt{q}$. Furthermore, the bounds on $\sigma_{\min}(\widehat{\mathbf{B}})$ and $\|\mathbf{\Sigma}_e\|_2$ in Lemma 2 and Lemma 3 imply

$$\|\widehat{\mathbf{A}}\|_2^2 \le s^2 - \sigma_{\min}(\widehat{\mathbf{B}})^2 + \|\mathbf{\Sigma}_e\|_2 \le s^2 - \frac{q^2}{\beta(p_u)^2} + s^2\bar{\tau} := \gamma_A^2.$$

Consequently,

$$\|C^{\text{hi}}_{2,k}\|_1 \le \frac{\|C_{2,k}\|_1}{d_p} + \frac{n}{2} \le \frac{\sqrt{n}\cdot\|C_{2,k}\|_2}{d_p} + \frac{n}{2} \le \frac{\sqrt{n}\cdot\|(f,g)\|_2\|\widehat{\mathbf{A}}\|_2}{L} + \frac{n}{2} \le \frac{\alpha\gamma_A\sqrt{qn}}{L} + \frac{n}{2}.$$

By Lemma 2, we have $\|(m_{11}, m_{12})\|_2 \le p_u\|\widehat{\mathbf{B}}\|_2 \le p_u\beta(p_u)$. The same argument yields

$$\|C^{\text{hi}}_{1,k}\|_1 \le \frac{\sqrt{n}\cdot\|C_{1,k}\|_2}{d_p} + \frac{n}{2} \le \frac{\sqrt{n}}{L}\|(m_{11}, m_{12})\|_2\|\widehat{\mathbf{A}}\|_2 + \frac{n}{2} \le \frac{p_u\beta(p_u)\gamma_A\sqrt{n}}{L} + \frac{n}{2}.$$

Bound on H_2. From $\|t\|_2 < q\sqrt{n}$ and the bound of $C^{\text{hi}}_{2,k}$, it follows that

$$\|H_2\|_\infty \le \|f\|_2\|t\|_2 + T_{\text{offline}}\sum_{k=0}^{1}\|C^{\text{hi}}_{2,k}\|_1 \le \alpha\sqrt{nq}\left(q + \frac{2\gamma_A T_{\text{offline}}}{L}\right) + nT_{\text{offline}}.$$

Bound on H'_1. Note that $\widehat{c}_2 = \frac{1}{q}\langle(f,g), \mathbf{c} - \mathbf{p}\rangle = \frac{1}{q}\langle(f,g), \mathbf{c} - \frac{1}{L}\widehat{\mathbf{A}}\mathbf{x}\rangle$ with $\|\mathbf{c}\|_2 < q\sqrt{n}$ and $\|\mathbf{x}\|_2 \le T_{\text{offline}}\sqrt{2n}$. Then

$$\|\widehat{c}_2\|_\infty \le \frac{\|(f,g)\|_2}{q}\left(\|\mathbf{c}\|_2 + \frac{\|\widehat{\mathbf{A}}\|_2\|\mathbf{x}\|_2}{L}\right) \le \frac{\alpha}{\sqrt{q}}\left(q\sqrt{n} + \frac{\gamma_A T_{\text{offline}}\sqrt{2n}}{L}\right).$$

Since z_2 is sampled from $D_{\mathcal{R},r,\widehat{c}_2}$, it follows that

$$\|z_2\|_\infty \le \|\widehat{c}_2\|_\infty + T_{\text{online}} + 1 \le \frac{\alpha}{\sqrt{q}}\left(q\sqrt{n} + \frac{\gamma_A T_{\text{offline}}\sqrt{2n}}{L}\right) + T_{\text{online}} + 1 := Z_2.$$

Combining the bounds on $C^{\mathrm{hi}}_{1,k}$ and z_2 yields

$$\begin{aligned}\|H'_1\|_\infty &\le \|m_{11}\|_2\|t\|_2 + T_{\mathrm{offline}} \sum_{k=0}^{1} \|C^{\mathrm{hi}}_{1,k}\|_1 + q\|u_{\mathrm{int}}\|_1\|z_2\|_\infty \\ &\le \sqrt{n}\, p_u \beta(p_u) \left(q + \frac{2\gamma_A T_{\mathrm{offline}}}{L} \right) + nT_{\mathrm{offline}} + \frac{qnp_u}{2} Z_2.\end{aligned}$$

Table 4 presents the concrete upper bounds for aforementioned values involved in the exact center computations.

Table 4: Upper bounds for intermediate values in the exact center computations.

	$n = 512$	$n = 1024$
$\|L_1\|_\infty, \|L_2\|_\infty$	$< 2^{57}$	$< 2^{59}$
$\|H_2\|_\infty$	$< 2^{26}$	$< 2^{27}$
$\|H'_1\|_\infty$	$< 2^{50}$	$< 2^{52}$

4.4 Fixed-Point Conversion of Centers

After exact reconstruction, each integer Gaussian center c is specified by two signed integers, a low part ℓ and a high part h:

$$c = \frac{2^b h + \ell}{q\, 2^{b+t}}, \qquad b = \log_2 d_p \in \{43, 44\}, \quad t \in \{0, 16\}.$$

For a coefficient of $\widehat{c}_2$, use $(\ell, h, t) = (L_{2,i}, H_{2,i}, 0)$; for a coefficient of c'_1, use $(\ell, h, t) = (L_{1,i}, H'_{1,i}, 16)$. Algorithm 4 extracts the integer part $k = \lfloor c \rfloor$ and the Q64 fractional code R without forming the full numerator $2^b h + \ell$. The same procedure covers both parameter sets and operates on one scalar pair at a time. For speed, the desktop implementation prepares three consecutive fixed-point centers together before sampling the corresponding coefficients in order. Batching center preparation preserves coefficient order, random-byte consumption, and the resulting signatures for the same conversion rule.

Algorithm 4: Fixed point center computation from low and high parts

Parameters : $q = 12289$, $b \in \{43, 44\}$, and $t \in \{0, 16\}$.
Input: Signed 64-bit integers (ℓ, h) satisfying the bounds of Section 4.3.
Output: $k = \lfloor c \rfloor$ and $R = \max\{1, \lfloor 2^{64}(c - k) \rfloor\}$, representing $c^{\mathrm{fp}} = k + R2^{-64}$.

$a \leftarrow \lfloor \ell / 2^b \rfloor$; $\ell_0 \leftarrow \ell - 2^b a$
$H \leftarrow h + a$
$v \leftarrow \lfloor H / 2^t \rfloor$; $h_0 \leftarrow H - 2^t v$
$k \leftarrow \mathrm{trunc}(v/q)$; $r \leftarrow v - qk$
$\delta \leftarrow [r < 0]$; $k \leftarrow k - \delta$; $r \leftarrow r + q\delta$
$w \leftarrow (2^b h_0 + \ell_0)\, 2^{64-b-t}$
$u \leftarrow r\, 2^{32} + \lfloor w / 2^{32} \rfloor$
$R_{\mathrm{hi}} \leftarrow \lfloor u/q \rfloor$; $r \leftarrow u - qR_{\mathrm{hi}}$
$u \leftarrow r\, 2^{32} + (w \bmod 2^{32})$
$R_{\mathrm{lo}} \leftarrow \lfloor u/q \rfloor$
$R \leftarrow R_{\mathrm{hi}}\, 2^{32} + R_{\mathrm{lo}}$
$R \leftarrow R \,\mathrm{OR}\, (R == 0)$ `// bitwise OR`
return (k, R)

Here $[P] \in \{0, 1\}$ is the indicator of P, and trunc rounds toward zero, as in signed C division. The correction by δ converts this quotient and remainder into Euclidean division,

including for negative inputs. The divisions by powers of two are floor splits, implemented by shifts and masks.

Immediately before the two 32-bit quotient digits are extracted, $0 \le r < q$, $0 \le w < 2^{64}$, and

$$c = k + \frac{r\,2^{64} + w}{q\,2^{64}}.$$

The two successive divisions by q therefore compute $\lfloor (r\,2^{64} + w)/q \rfloor$ exactly, without storing the wide dividend. Each temporary dividend u is smaller than $q\,2^{32} < 2^{46}$, and each quotient digit fits in 32 bits. All center-preparation intermediates fit in signed or unsigned 64-bit words under the preceding bounds. On the desktop, division by the fixed modulus $q = 12289$ compiles to reciprocal multiply-and-shift sequences. On Cortex-M4, the divisions by the fixed constant q use reciprocal multiplication and fixed corrections; they produce the same integer quotients and remainders. The final bitwise operation maps a zero fractional code to one without a conditional branch, leaving the integer part k unchanged. The acceptance arithmetic uses both R and its complement $2^{64} - R$; if $R = 0$, the complement needs 65 bits, and unsigned 64-bit negation would instead produce zero, giving an incorrect acceptance exponent. Restricting $1 \le R < 2^{64}$ keeps both values representable in one word and avoids a separate endpoint case.

This modification preserves the 2^{-64} error bound: writing $\theta = c - k \in [0, 1)$ and $R_0 = \lfloor 2^{64}\theta \rfloor$, we have

$$\begin{cases} 0 < c^{\mathrm{fp}} - c = 2^{-64} - \theta \le 2^{-64}, & R_0 = 0 \quad (0 \le \theta < 2^{-64}), \\ 0 \le c - c^{\mathrm{fp}} = \theta - R_0\,2^{-64} < 2^{-64}, & R_0 > 0. \end{cases}$$

Thus $|c^{\mathrm{fp}} - c| \le 2^{-64}$ even when zero is replaced by one, with equality exactly at integer centers. This meets the precision requirement of Lemma 5, which requires error smaller than 2^{-44}.

4.5 NTT

Our fixed-point implementation uses NTT instead of FFT for algebraic computations over $\mathcal{R}$. All polynomial products in the center computation are computed modulo two 31-bit auxiliary primes $\ell_0 = 2147473409$ and $\ell_1 = 2147389441$, for which residues fit in 32-bit words and individual residue products in 64-bit words, as $P = \ell_0\ell_1 < 2^{62}$. By CRT, reconstruction is exact for coefficients of magnitude below $\ell_0/2$ with one modulus, or below $P/2$ with two moduli. Since $\ell_i \equiv 1 \bmod 2048$, they support a complete NTT over $\mathcal{R}$.

As shown in Table 4, $\|H_2\|_\infty < \ell_0/2$, $\|L_j\|_\infty < P/2$ and $\|H_1'\|_\infty < P/2$, hence H_2 can be recovered exactly using one modulus, while L_j and H_1' can be exactly computed using two moduli. The center numerators N_2 and M_1 are then computed exactly from the recovered polynomials, without additional CRT reconstruction.

The NTT representations of key-dependent polynomials are precomputed during the key expansion stage. Table 5 reports the NTTs performed during signing, counting those over different auxiliary primes ℓ_0, ℓ_1 separately.

Norm checking in the NTT domain. For the final multiplication by $\mathbf{B}$, the NTT of z_2 is reused, so only z_1 requires an additional forward NTT. Let $(s_1, s_2)^t = \mathbf{c} - \mathbf{B}(z_1, z_2)^t$. Since only s_2 is included in the signature, its coefficients are recovered by one inverse NTT, while s_1 is used only for the norm check. The squared norm of s_1 is also computed from its NTT representation modulo ℓ_0, hence no inverse NTT is needed for this. In fact, as shown in Appendix B,

$$\|s_1\|_2^2 \equiv n^{-1} \sum_{j=0}^{n-1} \widehat{s}_{1,j}\widehat{s}_{1,n-1-j} \bmod \ell_0$$

Table 5: Number of NTTs and inverse NTTs per signing attempt.

	NTT	iNTT
NTT of t, x_0, x_1	3×2	0
Coefficient reconstruction of L_2	0	2
Coefficient reconstruction of H_2	0	1
NTT of z_2	2	0
Coefficient reconstruction of L_1	0	2
Coefficient reconstruction of H'_1	0	2
Multiplication by $\mathbf{B}$	1	1
Total	9	8

for $\widehat{s}_{1,j} = s_1(\psi^{2j+1})$ where ψ is a primitive $2n$-th root of unity modulo ℓ_0 and the probability of $\|s_1\|_2^2 > \ell_0$ is negligible.

AVX2 kernels. Both parameter sets use radix-2 AVX2 NTT kernels over two auxiliary primes. Twiddle factors are exact modular roots of unity. Shoup multiplication uses precomputed factors $\lfloor w2^{32}/p \rfloor$ for twiddle w and auxiliary prime p. Forward and inverse transforms use Shoup reduction. Products with precomputed NTT keys continue to use Montgomery reduction.

We pack eight canonical 31-bit residues into the 32-bit lanes of each 256-bit register. The Shoup quotient estimate uses the high words of even- and odd-lane 64-bit products; packed low-word products reconstruct the remainder, followed by one conditional subtraction of the prime. At both security levels, the inverse transform uses unscaled butterflies and divides by n modulo the auxiliary prime in the final layer. For butterfly strides of at least eight coefficients, we load contiguous vectors of left and right operands. At smaller strides, register permutations and blends collect the operands and restore their coefficient order.

Paired forward transforms share twiddle loads between two polynomials during perturbation precomputation; inverse transforms are called individually. Fused pointwise operations accumulate products before storing final sums. The modular norm computation vectorizes products of paired transform coefficients and accumulates their contributions before final reduction.

Cortex-M4 kernels. NTT butterflies use Montgomery multiplication with radix 2^{32} and twiddle factors stored in Montgomery form, implemented with the Thumb-2 `UMULL` and `UMLAL` instructions. Both security levels share the same NTT assembly. The $n = 512$ kernels use one radix-2 stage followed by four radix-4 stages, each fusing two radix-2 stages; the $n = 1024$ kernels use five radix-4 stages. Fusing stages keeps intermediate values in registers and reduces loads and stores. Inverse butterflies are unscaled, with one final multiplication by n^{-1} per output. They reuse each forward twiddle stage in reverse order and reverse the subtraction to absorb the sign, so no backward twiddle tables are stored. The two forward tables occupy 4 096 bytes at $n = 512$ and 8 192 bytes at $n = 1024$, included in the expanded key.

5 Performance Evaluation

FXP-Mitaka denotes our fixed-point implementation of Mitaka$_{\mathbb{Z}}$ signature generation using Algorithm 3. We evaluate both security levels ($n = 512$ and $n = 1024$) on two platforms: a

native AVX2 implementation on an Intel Core i7-1360P and a Cortex-M4F implementation on an STM32F411 microcontroller. Complete-signing costs include rejected candidates and retries. One kcycle denotes 1 000 processor cycles.

5.1 Desktop Implementation

Experimental setup. The desktop computer has an Intel Core i7-1360P with four performance cores, eight efficiency cores, sixteen logical CPUs, and an 18 MiB shared L3 cache. The FXP-Mitaka measurements use Ubuntu 26.04.1 LTS with Linux 7.0.0-38-generic. We compile both parameter sets with GCC 15.2.0 using `-O3`, `-mavx2`, `-mno-fma`, `-fno-lto`, and generic x86-64 tuning. Both parameter sets use the same signing source. Each benchmark is single-threaded and pinned to performance-core logical CPU 6. CPU frequency is not fixed; the other hardware thread on that core and background system activity are not controlled.

We measure user-mode processor core cycles with the Linux `cpu_core/cycles/u` hardware event, configured through `perf_event_open`. Serialized `RDPMC` reads bracket blocks of 512 complete signing calls. The counter runs without multiplexing. Process CPU time is measured separately with `CLOCK_PROCESS_CPUTIME_ID`; cycles are not inferred from time or a nominal clock frequency.

For each parameter set, we reuse one fixed public test key for 32 batches of 512 signing calls, giving 16 384 timed signatures. Key setup and workspace preparation are performed once, outside the timed region. Each build first makes 16 untimed warm-up calls. Messages contain 127 bytes. The uninstrumented and component-instrumented builds alternate execution order between batches and produce identical signatures; all outputs, including warm-up, pass public verification outside the timed region. The timed calls require 16 590 attempts at $n = 512$ and 16 401 at $n = 1024$: an attempt constructs a candidate signature, and norm rejection triggers another attempt within the same signing call. All such retries are included in the timings. No observations are discarded.

Signature generation. FXP-Mitaka uses a two-stage sampler. The offline phase generates a fresh perturbation and performs the number-theoretic transform (NTT) precomputation that depends on it. The online phase performs the message-dependent center computations and sampling, followed by signature construction, norm checking, and encoding. Table 6 reports direct measurements of the combined signing call; its totals are not sums of separately instrumented phase measurements. Mean signature generation costs are about 361 kcycles at $n = 512$ and 707 kcycles at $n = 1024$. Mean process CPU times are 74.281 and 150.281 microseconds per signature, respectively.

Table 6: Mean hardware cycles per signature for FXP-Mitaka AVX2 on the Intel Core i7-1360P.

	$n = 512$	$n = 1024$
Complete signature generation	360 960	706 857
Forward NTTs (9 per attempt)	37 235	79 479
Inverse NTTs (8 per attempt)	29 533	62 674
Offline perturbation sampler	42 711	88 212
Online sampler with center computations	150 763	298 912

Component cycle measurements. The component rows report cycles per completed signature, including work from rejected attempts. Each sampler generates $2n$ coordinates

per attempt; per-coordinate and per-transform costs below divide by the corresponding number of executed operations. The NTT measurements include inverse scaling and exclude pointwise products, residue conversion, and CRT reconstruction. The mean costs per transform are 4 086 and 8 822 cycles for forward NTTs, and 3 646 and 7 826 cycles for inverse NTTs, at $n = 512$ and $n = 1024$, respectively. Offline sampling includes random-byte reads and refills, cumulative distribution table (CDT) lookups, and sign sampling, but excludes stream derivation and perturbation-dependent NTTs. Online sampling includes fixed-point conversion of centers, CDT lookups, sign sampling, rejection sampling, random-byte reads and refills, and immediate coordinate translation; it excludes initial stream derivation and other polynomial arithmetic. The corresponding mean costs per coordinate are 41.19 and 43.03 cycles offline, and 145.40 and 145.80 cycles online. Other signing work includes hashing, pointwise arithmetic, CRT reconstruction, norm checks, encoding, and cleanup. Each component interval is corrected by subtracting the minimum of 10 000 empty counter-read intervals. Component instrumentation can change code layout and cache behavior, so the complete-signing costs are measured in a separate build without component instrumentation.

Comparison with fixed-point Falcon. We benchmark the optimized fixed-point Falcon implementation of [DFLP26] at commit `ad1e5bf`[2], with `FALCON_FPR_MODE=2`. Signing is measured after key expansion. It uses Clang 19.1.7 and the optimization and target options listed above.

Table 7 compares these measurements with the complete FXP-Mitaka signing costs above. The cycle-count ratios are 2.56× for $n = 512$ and 2.73× for $n = 1024$. Both experiments use the same Intel Core i7-1360P computer, P-core logical CPU 6, and Linux 7.0.0-38-generic, with one test key per parameter set. Falcon uses Clang 19.1.7, whereas FXP-Mitaka uses GCC 15.2.0.

Table 7: Comparison of fixed-point Falcon and FXP-Mitaka on an Intel Core i7-1360P.

	Cycles	CPU time (μs)	Cycle speedup
Fixed-point Falcon-512	923 812	187.078	1.00×
FXP-Mitaka-512	360 960	74.281	2.56×
Fixed-point Falcon-1024	1 929 756	400.893	1.00×
FXP-Mitaka-1024	706 857	150.281	2.73×

5.2 ARM Cortex-M4F implementation

Experimental setup. The implementation runs on a WeAct Black Pill board with an STM32F411CEU6 operating at 84 MHz. We compile with GCC 14.2.1 for ARM using `-O3` and link-time optimization. The Data Watchpoint and Trace (DWT) cycle counter measures 100 complete signing calls for each implementation and parameter set, using one fixed key. Both implementations and both parameter sets use the same board, clock configuration, compiler flags, and flash execution. Each call signs the four-byte message `data` with a distinct request identifier for FXP-Mitaka or a distinct seed for Falcon. FXP-Mitaka uses an expanded private key in flash; Pornin's RAM-efficient Falcon implementation uses a compact private key in flash. Key generation, expansion, workspace preparation, input preparation, and output checks are outside the timed intervals. Counter-read overhead is subtracted from each interval. Rejections and retries within signing are

[2] `fixed-point-fndsa/fixed-point-c`, commit `ad1e5bf`.

included: FXP-Mitaka makes 102 attempts for 100 signatures at $n = 512$ and 100 attempts at $n = 1024$.

FXP-Mitaka uses the Cortex-M4 NTT kernels of Section 4.5. It stores expanded private keys and NTT twiddle factors in flash, and reuses transform buffers for hashing and random number generation to limit writable RAM. The remaining arithmetic uses 32-bit and 64-bit integer operations.

Memory scheduling. The center computations use the high/low split of Section 4.3 and NTTs modulo the auxiliary primes ℓ_0 and ℓ_1. Intermediate arrays share storage when their lifetimes do not overlap. Three n-word transform slots and one accumulator modulo ℓ_0 occupy $16n$ bytes in total. Two signed 64-bit coefficient arrays occupy another $16n$ bytes: one holds the low part of the split computation, and the other holds L_2, the low-part contribution to the second center, before being reused for the updated high part. Contexts, the message seed, salt, and alignment bring the complete temporary buffer to $32n + 112$ bytes: 16 496 bytes at $n = 512$ and 32 880 bytes at $n = 1024$. The two primes are processed sequentially. During perturbation generation, a retained residue modulo ℓ_0 is packed into the otherwise unused upper word of an L_2 entry; it moves back to a transform slot when CRT reconstruction replaces the packed L_2 residues. SHAKE state and permutation scratch occupy transform storage only while that storage is free; the offline ChaCha state similarly occupies the low array before its coefficients become live. The signature output buffer first holds perturbation coefficients, then the raw challenge and online sampler state, and finally the encoded signature. These overlays are used only after the preceding values have been consumed. Keeping the persistent seed and salt in the workspace also avoids retaining them in the call stack.

Two additional transforms avoid retaining intermediate arrays. First, the forward NTT of the challenge t modulo ℓ_0 is recomputed before the final basis multiplication and norm check. Its first transform is consumed in forming the high-part accumulators, after which the same slot is reused for the transform modulo ℓ_1; only the compact 16-bit coefficients of t are retained. Second, after using the NTT of the sampled polynomial z_2 modulo ℓ_1 to update the first center's high-part accumulator H_1', an extra inverse NTT recovers z_2 in coefficient form. The coefficients are centered, reduced modulo ℓ_0, and transformed in place, avoiding a separate retained coefficient array. This ℓ_0 transform is then reused for both the remaining H_1' update and the final basis multiplication. Thus the extra transforms are one forward NTT of t modulo ℓ_0 and one inverse NTT of z_2 modulo ℓ_1, giving 10 forward and 9 inverse NTTs per candidate on Cortex-M4, compared with 9 forward and 8 inverse NTTs in the desktop implementation.

Mean signature generation costs are 3 980 099 cycles (47.382 ms) at $n = 512$ and 7 818 552 cycles (93.078 ms) at $n = 1024$. For the two security levels, the signing workspaces occupy 16 496 and 32 880 bytes, respectively. The maximum observed stack usage over 100 signatures is 456 and 464 bytes, giving workspace-plus-stack SRAM totals of 16 952 and 33 344 bytes. These totals exclude signature output, caller-owned inputs and results, and the benchmark driver and USB helper. Flash sizes in Table 8 are taken from the timing firmware and include its code, tables, private key, and benchmark driver. The code was designed for speed; flash storage could be reduced by adding NTTs.

Comparison with Falcon on Cortex-M4. We benchmark Pornin's RAM-efficient Falcon implementation [Por26] at commit `1fcdede`[3], with its Cortex-M4 assembly routines on the same STM32F411 at 84 MHz. Its mean costs are 13 521 413 cycles (160.969 ms) at $n = 512$ and 28 661 235 cycles (341.205 ms) at $n = 1024$. FXP-Mitaka uses 70.6% and 72.7% fewer cycles, respectively. We use the same memory convention for both implementations:

[3] `pornin/c-fn-dsa-alt`, commit `1fcdede`.

temporary buffer plus stack. Pornin reports 992 bytes of stack; its temporary-buffer sizes are 10 271 and 20 511 bytes.

Table 8: Mean complete-signing costs over 100 signatures per row on the STM32F411 Cortex-M4F at 84 MHz. Memory requirements are in bytes; SRAM is buffer plus stack, and flash covers the benchmark image.

	SRAM	Flash	kcycles	Speedup
FXP-Mitaka-512	16 952	68 672	3 980	3.40×
Falcon-512 [Por26]	11 263	57 292	13 521	1.00×
FXP-Mitaka-1024	33 344	117 696	7 819	3.67×
Falcon-1024 [Por26]	21 503	58 508	28 661	1.00×

A Proofs for the Sampler Analysis

This appendix gives the proofs of the three auxiliary lemmas used in Section 3.3. We retain the notation of that section throughout. The proof of the main theorem is given there.

A.1 Proof of Lemma 2

Let $\delta_u = u - \widehat{u}$. Coefficientwise rounding gives $\|\delta_u\|_\infty \le 1/(2p_u)$. For every canonical embedding $\zeta = e^{\imath\theta}$,

$$\begin{aligned} |\delta_u(\zeta)| &= \Re\Big(e^{-\imath \arg \delta_u(\zeta)}\delta_u(\zeta)\Big) \\ &\le \|\delta_u\|_\infty \sum_{j=0}^{n-1} |\cos(j\theta - \arg \delta_u(\zeta))| \le \frac{\csc(\pi/(2n))}{2p_u}. \end{aligned}$$

Write $\widetilde{\mathbf{B}} = \mathbf{B}\mathbf{U}_u^{-1}$ and $\gamma_\zeta = |f(\zeta)|^2 + |g(\zeta)|^2$. The two columns of $\widetilde{\mathbf{B}}(\zeta)$ are orthogonal with squared norms γ_ζ and q^2/γ_ζ. Since $\widehat{\mathbf{B}} = \widetilde{\mathbf{B}}\mathbf{U}_{\delta_u}$, the Gram matrix of $\widehat{\mathbf{B}}(\zeta)$ has determinant q^2 and trace

$$\gamma_\zeta\big(1 + |\delta_u(\zeta)|^2\big) + \frac{q^2}{\gamma_\zeta} \le q\left[\alpha^2\left(1 + \frac{\csc^2(\pi/(2n))}{4p_u^2}\right) + \alpha^{-2}\right].$$

Applying the quadratic formula to the two eigenvalues gives $\sigma_{\max}(\widehat{\mathbf{B}}(\zeta)) \le \beta(p_u)$. The canonical embedding then yields $\|\widehat{\mathbf{B}}\|_2 \le \beta(p_u)$. Since the product of the two singular values at every embedding equals q, $\sigma_{\min}(\widehat{\mathbf{B}}) \ge q/\beta(p_u)$.

A.2 Proof of Lemma 3

Let $\mathbf{E} = \mathbf{A} - \widehat{\mathbf{A}}$. Each coefficient of $\mathbf{E}$ has absolute value at most $1/(2p_A)$. Since $\mathbf{A}$ is lower triangular, the same is true of $\mathbf{E}$, and the coefficient-to-evaluation bound used above gives

$$\|\mathbf{E}\|_2 \le \frac{(1+\sqrt{5})\csc(\pi/(2n))}{4p_A}, \qquad \|\mathbf{E}\|_F \le \frac{\sqrt{3}\,n}{2p_A}.$$

The spectral constant is $\|\left(\begin{smallmatrix}1 & 0\\ 1 & 1\end{smallmatrix}\right)\|_2 = (1+\sqrt{5})/2$, while the Frobenius estimate follows from Parseval over the three nonzero polynomial entries.

Lemma 2 gives $\|\mathbf{A}\|_2 \le \sqrt{s^2 - q^2/\beta(p_u)^2}$. Together with the identity $\mathbf{\Sigma}_e = \mathbf{A}\mathbf{E}^* + \mathbf{E}\mathbf{A}^* - \mathbf{E}\mathbf{E}^*$, this yields

$$\|\mathbf{\Sigma}_e\|_2 \le (2\|\mathbf{A}\|_2 + \|\mathbf{E}\|_2)\|\mathbf{E}\|_2,$$
$$\|\mathbf{\Sigma}_e\|_F \le (2\|\mathbf{A}\|_2 + \|\mathbf{E}\|_2)\|\mathbf{E}\|_F.$$

Substitution of the preceding bounds gives exactly the two estimates stated in Lemma 3.

The same identities also imply the sampling-scale estimate used in the main text. Indeed,

$$\begin{aligned} 1 + \|\widehat{\mathbf{B}}^{-1}\widehat{\mathbf{A}}\|_2^2 &= \lambda_{\max}\Big(\widehat{\mathbf{B}}^{-1}(s^2\mathbf{I}_{2n} - \mathbf{\Sigma}_e)(\widehat{\mathbf{B}}^{-1})^*\Big) \\ &\le \frac{s^2 + \|\mathbf{\Sigma}_e\|_2}{\sigma_{\min}(\widehat{\mathbf{B}})^2} \le \frac{\beta(p_u)^2 s^2(1+\bar{\tau})}{q^2}. \end{aligned}$$

A.3 Proof of Lemma 4

Fix $\mathbf{z} \in \mathcal{L}(\mathbf{B})$ and write $(z_1, z_2)^t = \mathbf{B}^{-1}\mathbf{z}$. For every perturbation input $\mathbf{x} \in \mathbb{Z}^{2n}$, the centers in Algorithm 3 satisfy

$$(z_1 - c_1', z_2 - \widehat{c}_2)^t = \widehat{\mathbf{B}}^{-1}\big(\mathbf{z} - \mathbf{c} + \widehat{\mathbf{A}}\mathbf{x}/L\big).$$

Hence

$$\begin{aligned} \mathcal{P}(\mathbf{z}) &= \sum_{\mathbf{x}\in\mathbb{Z}^{2n}} \frac{\rho_{Lr}(\mathbf{x})}{\rho_{Lr}(\mathbb{Z}^{2n})} \frac{\rho_{r,c_1'}(z_1)\rho_{r,\widehat{c}_2}(z_2)}{\rho_{r,c_1'}(\mathbb{Z}^n)\rho_{r,\widehat{c}_2}(\mathbb{Z}^n)} \\ &= \frac{1}{\rho_{Lr}(\mathbb{Z}^{2n})} \sum_{\mathbf{x}\in\mathbb{Z}^{2n}} \frac{\rho_{Lr}(\mathbf{x})\rho_{r\widehat{\mathbf{B}}}(\mathbf{z} - \mathbf{c} + \widehat{\mathbf{A}}\mathbf{x}/L)}{\rho_{r,c_1'}(\mathbb{Z}^n)\rho_{r,\widehat{c}_2}(\mathbb{Z}^n)} \\ &\in [1, \kappa^2] \frac{\sum_{\mathbf{x}\in\mathbb{Z}^{2n}} \rho_{Lr}(\mathbf{x})\rho_{r\widehat{\mathbf{B}}}(\mathbf{z} - \mathbf{c} + \widehat{\mathbf{A}}\mathbf{x}/L)}{\rho_{Lr}(\mathbb{Z}^{2n})\rho_r(\mathbb{Z}^n)^2}, \end{aligned}$$

where the last line uses $r \ge \eta_\epsilon(\mathbb{Z}^n)$ for the two online normalizers.

Define

$$\mathbf{C} = \Big(\mathbf{I}_{2n} + \widehat{\mathbf{A}}^*(\widehat{\mathbf{B}}\widehat{\mathbf{B}}^*)^{-1}\widehat{\mathbf{A}}\Big)^{-1}, \qquad \mathbf{m}_\mathbf{z} = -L\mathbf{C}\widehat{\mathbf{A}}^*(\widehat{\mathbf{B}}\widehat{\mathbf{B}}^*)^{-1}(\mathbf{z} - \mathbf{c}).$$

Completing the square in $\mathbf{x}$ gives

$$\begin{aligned} &\sum_{\mathbf{x}\in\mathbb{Z}^{2n}} \rho_{Lr}(\mathbf{x})\rho_{r\widehat{\mathbf{B}}}(\mathbf{z} - \mathbf{c} + \widehat{\mathbf{A}}\mathbf{x}/L) \\ &\qquad = \rho_{r\sqrt{s^2\mathbf{I}_{2n} - \mathbf{\Sigma}_e}}(\mathbf{z} - \mathbf{c})\, \rho_{Lr\sqrt{\mathbf{C}},\mathbf{m}_\mathbf{z}}(\mathbb{Z}^{2n}). \end{aligned}$$

The smallest eigenvalue of $\mathbf{C}$ equals $(1 + \|\widehat{\mathbf{B}}^{-1}\widehat{\mathbf{A}}\|_2^2)^{-1}$. The choice of L in the main text therefore implies $Lr\sqrt{\mathbf{C}} \ge \eta_\epsilon(\mathbb{Z}^{2n})$. Applying smoothing once more,

$$\rho_{Lr\sqrt{\mathbf{C}},\mathbf{m}_\mathbf{z}}(\mathbb{Z}^{2n}) \in [\kappa^{-1}, 1]\, \rho_{Lr\sqrt{\mathbf{C}}}(\mathbb{Z}^{2n}).$$

Combining the preceding relations yields

$$\mathcal{P}(\mathbf{z}) = C_0\, a_\mathbf{z}\, \rho_{r\sqrt{s^2\mathbf{I}_{2n} - \mathbf{\Sigma}_e},\mathbf{c}}(\mathbf{z}), \qquad a_\mathbf{z} \in [\kappa^{-1}, \kappa^2],$$

for a constant $C_0 > 0$ independent of $\mathbf{z}$. Normalizing over $\mathcal{L}(\mathbf{B})$ gives

$$\mathcal{P}(\mathbf{z}) = \mathcal{Q}_e(\mathbf{z}) \frac{a_\mathbf{z}}{\mathbb{E}_{\mathcal{Q}_e}[a_\mathbf{z}]}.$$

All Gaussian masses above are positive, so $\mathsf{Supp}(\mathcal{P}) = \mathcal{L}(\mathbf{B})$.

For the Rényi bound, let $F(t) = \log \mathbb{E}_{\mathcal{Q}_e}[a_\mathbf{z}^t]$. Under the tilted distribution proportional to $\mathcal{Q}_e(\mathbf{z})a_\mathbf{z}^t$, $F''(t) = \mathrm{Var}(\log a_\mathbf{z})$. Since $\log a_\mathbf{z}$ lies in an interval of length $3\log\kappa$, Popoviciu's inequality gives $F''(t) \le 9(\log\kappa)^2/4$. Integrating twice yields $\log R_v(\mathcal{P}\|\mathcal{Q}_e) \le 9v(\log\kappa)^2/8$.

B Removing the Final Inverse NTT

For the modular norm check of Section 4.5, write the candidate before norm rejection as $\mathbf{s} = \mathbf{c} - \mathbf{z} = (s_1, s_2)^t$, and put $N_i = \|s_i\|^2_{\mathrm{coef},2}$. The exact check accepts when $N_1 + N_2 \le T$, with $T < \ell_0 = 2147473409$ for both parameter sets in Table 2. The optimized check replaces N_1 by its least nonnegative residue modulo ℓ_0 and recovers s_2 with one inverse NTT.

Norm computation in the transform domain. Let $\psi \in \mathbb{Z}_{\ell_0}$ have order $2n$, put $\xi_k = \psi^{2k+1}$, and write $\widehat{s}_{1,k} = s_1(\xi_k)$. Since $\xi_{n-1-k} = \xi_k^{-1}$, orthogonality of these roots gives

$$\begin{aligned} n^{-1}\sum_{k=0}^{n-1} \widehat{s}_{1,k}\widehat{s}_{1,n-1-k} &= \sum_{i,j=0}^{n-1} s_{1,i}s_{1,j}\left(n^{-1}\sum_{k=0}^{n-1}\xi_k^{i-j}\right) \\ &= \sum_{i=0}^{n-1} s_{1,i}^2 = N_1 \pmod{\ell_0}. \end{aligned} \tag{1}$$

Here n^{-1} denotes inversion modulo ℓ_0, and the spectra are indexed by their evaluation roots; an implementation may permute them. Thus $N_1 \bmod \ell_0$ requires only pointwise products and a sum. Whenever $\|\mathbf{s}\|_2^2 < \ell_0$, this residue equals N_1 as an integer. Every coefficient of s_2 then has magnitude below $\sqrt{\ell_0} < \ell_0/2$, so its centered reconstruction is also exact. Consequently, the two norm checks agree outside the event $E = \{\|\mathbf{s}\|_2^2 \ge \ell_0\}$.

Lemma 6 (Modular norm check)**.** *Under the conditions of Theorem 1, define* $\sigma_+^2 = r^2 s^2(1+\bar{\tau})$ *and* $\kappa = (1+\epsilon)/(1-\epsilon)$*. For a candidate produced by Algorithm 3 with exact scalar Gaussian sampling, the probability that the modular and exact norm checks differ is at most*

$$p_{\mathrm{NTT}} := \kappa^4 2^n \exp\left(-\frac{\pi\ell_0}{2\sigma_+^2}\right). \tag{2}$$

For both parameter sets in Table 2, $p_{\mathrm{NTT}} < 2^{-20000}$*.*

Proof. We use the Gaussian mass comparison underlying Banaszczyk's tail bound [Ban93]. Let $\Lambda = \mathcal{L}(\mathbf{B})$, $\mathbf{W} = r\sqrt{s^2\mathbf{I}_{2n} - \mathbf{\Sigma}_e}$, $\Lambda' = \mathbf{W}^{-1}\Lambda$, and $\mathbf{v} = \mathbf{W}^{-1}\mathbf{c}$. For $\mathbf{z} \sim \mathcal{Q}_e$, the vector $\mathbf{y} = \mathbf{W}^{-1}(\mathbf{c} - \mathbf{z})$ follows the width-one discrete Gaussian on $\Lambda' + \mathbf{v}$. The covariance bound gives $\|\mathbf{W}\|_2^2 \le \sigma_+^2$, so E implies $\|\mathbf{y}\|_2^2 \ge \ell_0/\sigma_+^2$. The theorem's smoothing condition also implies $\eta_\epsilon(\Lambda') \le 1$.

For any $R > 0$, splitting the Gaussian exponent in half gives

$$\sum_{\substack{\mathbf{y}\in\Lambda'+\mathbf{v}\\ \|\mathbf{y}\|_2 \ge R}} e^{-\pi\|\mathbf{y}\|_2^2} \le e^{-\pi R^2/2}\rho_{\sqrt{2}}(\Lambda' + \mathbf{v}).$$

Poisson summation and Gaussian mass scaling give $\rho_{\sqrt{2}}(\Lambda' + \mathbf{v}) \le \rho_{\sqrt{2}}(\Lambda') \le 2^n\rho_1(\Lambda')$. Lemma 1 gives $\rho_1(\Lambda'+\mathbf{v}) \ge \kappa^{-1}\rho_1(\Lambda')$. Dividing the preceding tail mass by this normalizer, with $R^2 = \ell_0/\sigma_+^2$, yields

$$\Pr_{\mathcal{Q}_e}[E] \le \kappa 2^n \exp\left(-\frac{\pi\ell_0}{2\sigma_+^2}\right).$$

By Lemma 4, $\mathcal{P}(\mathbf{z})/\mathcal{Q}_e(\mathbf{z}) = a_{\mathbf{z}}/\mathbb{E}_{\mathcal{Q}_e}[a_{\mathbf{z}}] \le \kappa^3$. This proves (2). For both concrete parameter sets, $\sigma_+^2 < 213000$. Using this bound, $n \le 1024$, and $\epsilon \le 2^{-38}$ in (2) gives $\log_2 p_{\mathrm{NTT}} < -21823$, hence the stated bound. □

References

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296:625–635, 1993.

[DFLP26] Daniel De Almeida Braga, Pierre-Alain Fouque, Bachir Lachguel, and Thomas Prest. Toward a secure fixed-point implementation of the Falcon signature scheme. Cryptology ePrint Archive, Paper 2026/1531, 2026.

[DGPY20] Léo Ducas, Steven Galbraith, Thomas Prest, and Yang Yu. Integral matrix gram root and lattice gaussian sampling without floats. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 608–637. Springer, Cham, May 2020.

[DP16] Léo Ducas and Thomas Prest. Fast Fourier Orthogonalization. pages 191–198, New York, NY, USA, July 19-22 2016. ACM.

[EFG+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Cham, May / June 2022.

[ENS+23] Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation - making mitaka as secure as falcon. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 3–36. Springer, Singapore, December 2023.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[HvVS+26] Stef Halmans, Christine van Vredendaal, Tobias Schneider, Frank Custers, and Tim Güneysu. Twfalcon: Triple-word arithmetic for falcon: Giving falcon the precision to fly securely. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(2):347–371, 2026.

[KTW+24] Kwangjo Kim, Mehdi Tibouchi, Alexandre Wallet, Thomas Espitau, Yang Yu, and Y Kim. Solmae: Faster and simpler quantum-safe signature based on ntru-lattices. Technical report, Ver. 2.0, 2024. Available at: https://kpqc.or.kr/images/pdf/SOLMAE.pdf.

[LTYZ25] Xiuhan Lin, Mehdi Tibouchi, Yang Yu, and Shiduo Zhang. Do not disturb a sleeping falcon - floating-point error sensitivity of the falcon sampler and its consequences. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 213–244. Springer, Cham, May 2025.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Berlin, Heidelberg, August 2010.

[Per26] Ray Perlner. New plan for FN-DSA. NIST PQC Forum, on behalf of the FIPS 206 team. Available at https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/7z-y5qp-W2Y, 2026.

[PFH+20] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Por19] Thomas Pornin. New efficient, constant-time implementations of falcon. Cryptology ePrint Archive, Paper 2019/893, 2019.

[Por26] Thomas Pornin. A RAM-efficient implementation of Falcon. Cryptology ePrint Archive, Paper 2026/1915, 2026.

[Pre15] Thomas Prest. *Gaussian Sampling in Lattice-Based Cryptography*. PhD thesis, École Normale Supérieure, Paris, France, 2015.

[ZZ25] Jipeng Zhang and Jiaheng Zhang. Vectorized falcon-sign implementations using SSE2, AVX2, AVX-512F, NEON, and RVV. Cryptology ePrint Archive, Report 2025/1867, 2025.