Mitaka: A Simpler, Parallelizable, Maskable Variant of Falcon

Thomas Espitau[1][0000−0002−7655−9594], Pierre-Alain Fouque[2][0000−0003−4997−2276], François Gérard[3], Mélissa Rossi[4][0000−0002−9268−3034], Akira Takahashi[5][0000−0001−8556−3053], Mehdi Tibouchi[1][0000−0002−2736−2963], Alexandre Wallet[2][0000−0003−0492−2435], and Yang Yu[6][0000−0003−0120−0648]

[1] NTT Corporation, Tokyo, Japan
{thomas.espitau.ax,mehdi.tibouchi.br}@hco.ntt.co.jp
[2] IRISA, Univ Rennes 1, Inria, Rennes Bretagne-Atlantique Center, Rennes, France
pa.fouque@gmail.com, alexandre.wallet@inria.fr
[3] University of Luxembourg, Esch-sur-Alzette, Luxembourg
francois.gerard@uni.lu
[4] ANSSI, Paris, France
melissa.rossi@ssi.gouv.fr
[5] Aarhus University, Aarhus, Denmark
takahashi@cs.au.dk
[6] BNRist, Tsinghua University, Beijing, China
yu-yang@mail.tsinghua.edu.cn

Abstract. This work describes the Mitaka signature scheme: a new hash-and-sign signature scheme over NTRU lattices which can be seen as a variant of NIST finalist Falcon. It achieves comparable efficiency but is considerably simpler, online/offline, and easier to parallelize and protect against side-channels, thus offering significant advantages from an implementation standpoint. It is also much more versatile in terms of parameter selection.

We obtain this signature scheme by replacing the FFO lattice Gaussian sampler in Falcon by the "hybrid" sampler of Ducas and Prest, for which we carry out a detailed and corrected security analysis. In principle, such a change can result in a substantial security loss, but we show that this loss can be largely mitigated using new techniques in key generation that allow us to construct much higher quality lattice trapdoors for the hybrid sampler relatively cheaply. This new approach can also be instantiated on a wide variety of base fields, in contrast with Falcon's restriction to power-of-two cyclotomics.

We also introduce a new lattice Gaussian sampler with the same quality and efficiency, but which is moreover compatible with the integral matrix Gram root technique of Ducas et al., allowing us to avoid floating point arithmetic. This makes it possible to realize the *same* signature scheme as Mitaka efficiently on platforms with poor support for floating point numbers.

Finally, we describe a provably secure masking of Mitaka. More precisely, we introduce novel gadgets that allow provable masking at any

order at much lower cost than previous masking techniques for Gaussian sampling-based signature schemes, for cheap and dependable side-channel protection.

1 Introduction

The third round finalists for signatures in the NIST postquantum standardization process consist of just three candidates: Rainbow [9], a multivariate scheme, Dilithium [12,29], a lattice-based scheme in the Fiat–Shamir with aborts framework, and Falcon [38], a hash-and-sign signature over NTRU lattices. They occupy fairly different positions within the design space of post-quantum signature schemes, and it is therefore important to understand, for each of them, to what extent they could possibly be improved by exploring similar designs that overcome some of their limitations. This paper aims at doing so for the Falcon signature scheme.

Hash-and-sign lattice-based signatures. Falcon fits within the long and hectic history of hash-and-sign signatures based on lattices. In those schemes, the signing key is a "good" representation of a lattice, the *trapdoor*, which makes it possible, given an arbitrary point in the ambient space, to find lattice points that are relatively close to it (i.e. solve the *approximate closest vector* problem, ApproxCVP[7]); the verification key, on the other hand, is a "bad" representation: it allows anyone to check if a point is in the lattice, but not to solve ApproxCVP. In order to sign a message, it is then hashed to a random point in the ambient space, and the signature is a lattice point close to it, obtained using the trapdoor. To verify, one checks that the signature is in the lattice and sufficiently close to the hash digest.

Early constructions along those lines, such as the GGH signature scheme [20] and multiple iterations of NTRUSign [22,21], were later shown to be insecure due to a common critical vulnerability: the lattice points obtained as signatures would leak information about the trapdoor used to compute them, which could then be recovered using more or less advanced statistical techniques [33,14]. One of the first round NIST candidates was in fact broken using the same idea [40].

It is thus crucial for security to prove that signatures are sampled according to a distribution that is *statistically independent* of the trapdoor. The first approach to do so, which remains the state of the art,[8] is due to Gentry, Peikert and Vaikuntanathan (GPV) [18]: sample the ApproxCVP solution according to a discrete Gaussian distribution centered at the target point and supported over the lattice, with covariance independent from the trapdoor (usually spherical).

[7] Sometimes, this is also seen as a *bounded distance decoding* problem, BDD, but with large enough decoding bound that there are exponentially many solutions, instead of a unique one as is typically the case in the traditional formulation of BDD.

[8] Other techniques have been proposed that avoid Gaussian distributions, as in [30], but they tend not to be competitive.

This type of lattice discrete Gaussian sampling can be carried out by randomizing known deterministic algorithms for ApproxCVP, like Babai rounding and Babai's nearest plane algorithm.

Within the overall GPV framework, specific signature schemes vary according to the lattices over which they are instantiated, the construction of the corresponding trapdoors, and the lattice Gaussian sampling algorithms they rely on based on those trapdoors. The security level achieved by such a scheme is then essentially determined by the *quality* of the trapdoor and of the Gaussian sampling algorithm, defined as the minimal standard deviation achievable in Gaussian sampling, while still preserving the statistical independence of the output.

A complete overview of existing proposals for each of those elements is beyond the scope of the current work. We focus instead on the particular case of NTRU lattices with the usual NTRU trapdoors first considered in NTRUSign, as those lattices appear to offer the most efficient implementations by a significant margin, thanks to their compact trapdoors.

Hash-and-sign signatures over NTRU lattices. NTRU lattices are, in essence, free rank 2 module lattices over cyclotomic rings, and the NTRU designers showed how to construct good trapdoors for them, even though the original signature schemes based on them proved insecure.

They were brought within the GPV framework (and thus gained provable security) thanks to the work of Ducas, Lyubashevsky and Prest (DLP) [13], who combined them with the Gaussian sampling algorithm obtained by randomizing Babai's nearest plane algorithm (this randomization is sometimes called the *Klein sampler* for lattice Gaussians). They analyzed the security of the construction and provided what became the first reasonably efficient implementation of a signature scheme in the GPV framework.

This DLP signature scheme offers relatively compact keys and signatures, but suffers from a relatively long signing time, quadratic in the $\mathbb{Z}$-rank of the underlying lattice. This is because the nearest plane computation is carried out after descending to $\mathbb{Z}$, essentially ignoring the module structure of the lattice.

FALCON is a direct improvement of this scheme, obtained by replacing this quadratic Gaussian sampling by a quasilinear one, derived from the quasilinear nearest plane algorithm described in the Fast Fourier Orthogonalization paper of Ducas and Prest [15] (and refining the parameter selection using a tighter statistical analysis based on the Rényi divergence). The computation still ultimately descends to $\mathbb{Z}$, but takes advantage of the tower field structure of the underlying number field (assumed to be a power-of-two cyclotomic) to achieve a better complexity.

These two approaches are equivalent in terms of the quality of the resulting Gaussian sampler, which is essentially the best possible for the given NTRU lattice. However, DLP does so at the cost of a slow signing algorithm, whereas FALCON, while fast, suffers from a very complex signing algorithm that is hard to implement, poorly suited for parallelization and difficult to protect against side-channel attacks. On the last point, both schemes have been shown to suffer

from potential vulnerabilities with respect to side-channel leakage [17,25], and even though the most recent implementation of FALCON appears to be protected against timing attacks [35,23], countermeasures against stronger side-channel attacks like DPA seem difficult to achieve. FALCON is also limited to NTRU lattices over power-of-two cyclotomic fields, which limits its flexibility in terms of parameter selection. That latter limitation can be overcome to some extent by extending the construction to higher rank modules, as done in MODFALCON [7], but the other drawbacks remain.

Another possibility is to instantiate the randomized ApproxCVP algorithm directly over the underlying ring, instead of doing so over $\mathbb{Z}$. For the randomized version of Babai rounding, this gives rise to (the ring version of) Peikert's sampler, as introduced in [34]. This can also be done for Babai's nearest plane algorithm, leading to what Ducas and Prest call the *hybrid sampler*. The resulting algorithms consist of a constant number of ring multiplications, so that quasilinear complexity is obtained "for free" as long as the underlying ring has a fast multiplication algorithm (as certainly holds for arbitrary cyclotomics). This makes them highly versatile in terms of parameter selection. They are also much simpler than FALCON, easy to parallelize, and support fairly inexpensive masking for side-channel protection.

Their downside, however, is the lower quality of the corresponding samplers compared to FALCON and DLP. Indeed, by not descending to $\mathbb{Z}$ but only to the ring itself, the ApproxCVP algorithm achieves less tight of a bound compared to the Klein sampler, and hence the Gaussian sampling has a larger standard deviation. This is analyzed in details in Prest's Ph.D. thesis [37] (although certain heuristic assumptions are incorrect), and results in a substantially lower security level than FALCON and DLP.

Our contributions: the MITAKA *signature scheme.* In this work, we revisit in particular the hybrid sampler mentioned above, and show that the security loss compared to FALCON can be largely mitigated using a novel technique to generate higher quality trapdoors. The resulting scheme, MITAKA,[9] offers an attractive alternative to FALCON in many settings since:

- it is considerably simpler from an algorithmic and an implementation standpoint, while keeping the same complexity (in fact, it is likely faster at the same dimension due to better cache locality);
- signature generation is parallel(izable);
- like Peikert's sampler, it has an online/offline structure, with the online part requiring only one-dimensional discrete Gaussian sampling with very small, constant standard deviation and simple linear operations;

[9] Trivia: Mitaka is a neighborhood in Tokyo, Japan whose name means "the three falcons". It sounded fitting considering the maskable, parallelizable nature of our scheme and its strong points compared to FALCON.

– it can be instantiated over arbitrary cyclotomic fields[10], which makes it quite versatile in terms of parameter selection;
– it is easier to protect against side-channels and can be cheaply masked even at high order.

The main idea that allows us to achieve higher security than previously expected is as follows. It is well-known that, given NTRU generators (f, g), it is easy to compute the quality of the corresponding NTRU trapdoor for the hybrid sampler (in particular, it can be done without computing the whole trapdoor). It is thus very easy to check whether a given (f, g) reaches a certain threshold in terms of bit security, and as a result, the costly part of key generation is the sampling of the random ring elements f and g themselves (with discrete Gaussian coefficients). One can therefore achieve a greatly improved security level at the same cost in terms of randomness and not much more computation time if one can "recycle" already sampled ring elements f and g.

We propose several ways of doing so. The simplest one is to generate lists $\{f_i\}$, $\{g_j\}$ of candidate elements for f and g, and test the pairs (f_i, g_j): this increases the space of candidates quadratically, instead of linearly, in the amount of generated randomness. One can also generate the f_i's, g_j's themselves as sums of Gaussians with smaller standard deviation (as long as it remains above the smoothing parameters), and consider the Galois conjugates of a given g_j. By combining these techniques appropriately, we achieve a substantial security increase, of around 15 bits for typical parameter sizes. Concretely, we achieve the same security level as Dilithium–II [12] (which was argued to reach NIST Level-I) in dimension $d = 512$, and attain roughly NIST Level–V in dimension $d = 1024$, with intermediate parameter settings possible.

We also provide a detailed security analysis of our construction, and while most of the presentation focuses on power-of-two cyclotomics for simplicity's sake and easier comparison with previous work, we also show that intermediate NIST security levels can be conveniently achieved using other base fields, e.g. of dimension 648 (same security as FALCON–512), 768 (NIST Level–II), 864 (NIST Level–III) and 972 (NIST Level–IV).

As an additional contribution, we also introduce a novel, alternate lattice Gaussian sampler for MITAKA that achieves the same complexity and the same quality as the hybrid sampler, but has a different structure, closer to Peikert's sampler. The advantage of that alternate sampler is that it is compatible with the integral lattice Gram root technique of Ducas et al. [11], making it possible to instantiate it *without floating point arithmetic*. We call the resulting construction MITAKA$_{\mathbb{Z}}$. We stress that MITAKA and MITAKA$_{\mathbb{Z}}$ are two different approaches to implement the *same* signature scheme (in the sense that the generated signatures have statistically close distributions), and one can choose one or the other as preferred depending on whether the target platform has fast floating point arithmetic or not.

[10] In principle, even more general number fields are possible as well, provided a good basis is known for their canonical embedding. The corresponding security analysis is cumbersome, however.

Finally, we introduce a new, concrete approach to mask those signature generation algorithms efficiently. In previous work, efficiently masking signature schemes using Gaussian sampling has proved quite challenging: even for the case of 1-dimensional *centered* discrete Gaussians, as in the BLISS signature scheme [10], this is far from straightforward [4]. Since MITAKA and MITAKA$_{\mathbb{Z}}$, like FALCON and DLP, require discrete Gaussian sampling with *variable* centers, a naive approach to masking is unlikely to yield fast results. Instead, we introduce and prove a novel gadget for sampling Gaussian distribution with an arithmetically masked center and a fixed standard deviation. This allows us to completely avoid masking Gaussian sampling operations in the online phase:[11] this works for a masked center, because picking a uniform center in $[0, M)$ with fixed denominator and sampling a discrete Gaussian around that center results in a close to uniform distribution modulo M. Carrying out this share-by-share sampling directly causes a slight decrease in the quality of the resulting sampler (depending on the number of shares), but this can be overcome completely with careful use of rejection sampling. Combining these statistical techniques with usual provable masking methodology, we achieve very efficient side-channel protection for both MITAKA and MITAKA$_{\mathbb{Z}}$.

Organization of the paper. We start in Section 2 with preliminary material. In Section 3, we show some Gaussian samplers over modules that are the building blocks of MITAKA. Section 4 introduces the techniques to improve the quality of NTRU trapdoors, which elevates the security level achievable using MITAKA's samplers. The security analysis and preliminary implementation results of MITAKA are respectively provided in Sections 5 and 6. In Section 7, we describe a provably secure masking of MITAKA. Finally, Section 8 provides some concluding remarks.

2 Preliminaries

For any $a \in \mathbb{R}$ and $q > 0$, let $[a]_q = \lfloor aq \rceil / q \in (1/q)\mathbb{Z}$.

2.1 Linear algebra and lattices

Write $\mathbf{A}^t$ for the transpose of any matrix $\mathbf{A}$. Let $s_1(\mathbf{A}) = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$ the largest singular value of $\mathbf{A}$. Let $\Sigma \in \mathbb{R}^{n\times n}$ be a symmetric matrix. We write $\Sigma \succ 0$ when Σ is *positive definite*, i.e. $\mathbf{x}^t\Sigma\mathbf{x} > 0$ for all non-zero $\mathbf{x} \in \mathbb{R}^n$. We also write $\Sigma_1 \succ \Sigma_2$ when $\Sigma_1 - \Sigma_2 \succ 0$. It holds that $\Sigma \succ 0$ if and only if $\Sigma^{-1} \succ 0$ and that $\Sigma_1 \succ \Sigma_2 \succ 0$ if and only if $\Sigma_2^{-1} \succ \Sigma_1^{-1} \succ 0$. A lattice $\mathscr{L}$ is a discrete additive subgroup of a Euclidean space. When the space is $\mathbb{R}^m$, and if it is generated by (the columns of) $\mathbf{B} \in \mathbb{R}^{m\times d}$, we also write $\mathscr{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^d\}$. If $\mathbf{B}$ has full column rank, then we call $\mathbf{B}$ a basis and d the rank of $\mathscr{L}$. The volume of $\mathscr{L}$ is $\mathrm{Vol}(\mathscr{L}) = \det(\mathbf{B}^t\mathbf{B})^{\frac{1}{2}}$ for any basis $\mathbf{B}$.

[11] The same idea can be adapted to the offline phase by masking the zero center. This is a bit less compelling, however, as it requires more shares, and replaces centered Gaussian sampling by variable center sampling.

2.2 Power-of-two cyclotomic fields

For the sake of simplicity and readability, we focus in the rest of this article on the case where the number field is a cyclotomic field of conductor a power of 2. In any case, the content of this section generalizes straightforwardly to other cyclotomic number fields, as well as most of our results. Besides, the use of cyclotomic fields is nowadays pervasive in lattice-based cryptography. In this section we therefore keep only the minimum amount of notation and definitions to follow the article. More details can be found in the full version [16].

Let $d = 2^\ell$ for some integer $\ell \geqslant 1$ and ζ_d to be a $2d$-th primitive root of 1. Then for a fixed d, $\mathscr{K} := \mathbb{Q}(\zeta_d)$ is the d-th power-of-two cyclotomic field, and its ring of algebraic integers is $\mathscr{R} := \mathbb{Z}[\zeta_d]$. The field automorphism $\zeta_d \mapsto \zeta_d^{-1} = \overline{\zeta_d}$ corresponds to the complex conjugation, and we write the image f^* of f under this automorphism. We have $\mathscr{K} \simeq \mathbb{Q}[x]/(x^d+1)$ and $\mathscr{R} \simeq \mathbb{Z}[x]/(x^d+1)$, and both are contained in $\mathscr{K}_\mathbb{R} := \mathscr{K} \otimes \mathbb{R} \simeq \mathbb{R}[x]/(x^d+1)$. Each $f = \sum_{i=0}^{d-1} f_i \zeta_d^i \in \mathscr{K}_\mathbb{R}$ can be identified[12] with its coefficient vector $(f_0, \ldots, f_{d-1}) \in \mathbb{R}^d$. The adjoint operation extends naturally to $\mathscr{K}_\mathbb{R}$, and $\mathscr{K}_\mathbb{R}^+$ is the subspace of elements satisfying $f^* = f$.

The cyclotomic field $\mathscr{K}$ comes with d complex field embeddings $\varphi_i : \mathscr{K} \to \mathbb{C}$ which map f seen as a polynomial to its evaluations at the odd powers of ζ_d. This defines the so-called *canonical embedding* $\varphi(f) := (\varphi_1(f), \ldots, \varphi_d(f))$. It extends straightforwardly to $\mathscr{K}_\mathbb{R}$ and identifies it to the space $\mathcal{H} = \{\mathbf{v} \in \mathbb{C}^d : v_i = \overline{v_{d/2+i}}, 1 \leqslant i \leqslant d/2\}$. Note that $\varphi(fg) = (\varphi_i(f)\varphi_i(g))_{i \leqslant d}$. When needed, this embedding extends entry-wise to vectors or matrices over $\mathscr{K}_\mathbb{R}$. We let $\mathscr{K}_\mathbb{R}^{++}$ be the subset of $\mathscr{K}_\mathbb{R}^+$ which have all positive coordinates in the canonical embedding.

2.3 Matrices of algebraic numbers and NTRU modules

2.3.1 2×2 $\mathscr{K}$-valued matrices. This work deals with free $\mathscr{R}$-modules of rank 2 in $\mathscr{K}^2$, or in other words, groups of the form $\mathscr{R}\mathbf{x} + \mathscr{R}\mathbf{y}$ where $\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)$ span $\mathscr{K}^2$. There is a natural $\mathscr{K}$-bilinear form over $\mathscr{K}^2$ defined by $\langle \mathbf{x}, \mathbf{y} \rangle_\mathscr{K} := x_1^* y_1 + x_2^* y_2 \in \mathscr{K}$. It can be checked that for all $\mathbf{x} \in \mathscr{K}^2$, $\langle \mathbf{x}, \mathbf{x} \rangle_\mathscr{K} \in \mathscr{K}_\mathbb{R}^{++}$. This form comes with a corresponding notion of orthogonality. In particular, the well-known Gram-Schmidt orthogonalization procedure for a pair of linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathscr{K}^2$ is defined as

$$\widetilde{\mathbf{b}}_1 := \mathbf{b}_1, \ \widetilde{\mathbf{b}}_2 := \mathbf{b}_2 - \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle_\mathscr{K}}{\langle \mathbf{b}_1, \mathbf{b}_1 \rangle_\mathscr{K}} \cdot \widetilde{\mathbf{b}}_1.$$

One readily checks that $\langle \widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2 \rangle_\mathscr{K} = 0$. The Gram-Schmidt matrix with columns $\widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2$ is denoted by $\widetilde{\mathbf{B}}$ and we have $\det \widetilde{\mathbf{B}} = \det \mathbf{B}$.

For $\Sigma \in \mathscr{K}_\mathbb{R}^{2\times 2}$, we write Σ^* its conjugate-transpose, where $*$ is the conjugation in $\mathscr{K}_\mathbb{R}$. We extend the notion of positive definiteness for matrices with entries in $\mathscr{K}_\mathbb{R}$: $\Sigma \in \mathscr{K}_\mathbb{R}^{2\times 2}$ is positive definite when $\Sigma = \Sigma^*$ and all the d matrices induced by the field embeddings are positive definite. We then write $\Sigma \succ 0$. For

[12] This is the so-called coefficient embedding.

example, $\mathbf{B}^*\mathbf{B}$ is a positive definite matrix for all $\mathbf{B} \in \mathscr{K}_{\mathbb{R}}^{2\times 2}$. Positive definite matrices admit "square roots", that is, matrices $\sqrt{\Sigma}$ such that $\sqrt{\Sigma}\sqrt{\Sigma}^* = \Sigma$.

This work uses fundamental quantities for matrices over $\mathscr{K}$. The first is defined as $|\mathbf{B}|_{\mathscr{K}} := \max_{1\leqslant i\leqslant 2} \|\varphi(\langle \widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_i\rangle_{\mathscr{K}})\|_\infty^{1/2}$. Since the eigenvalues λ_1, λ_2 of $\mathbf{B}^*\mathbf{B}$ are all in $\mathscr{K}^{++}$, coordinate-wise square roots are well-defined. The largest singular value of (the embeddings of) $\mathbf{B}$ is recovered as $s_1(\mathbf{B}) := \max_{1\leqslant i\leqslant 2} \|\varphi(\sqrt{\lambda_i})\|_\infty$.

NTRU Modules. Given $f, g \in \mathscr{R}$ such that f is invertible modulo some prime $q \in \mathbb{Z}$, we let $h = f^{-1}g \bmod q$. The NTRU module determined by h is $\mathscr{L}_{\mathrm{NTRU}} = \{(u, v) \in \mathscr{R}^2 \,:\, uh - v = 0 \bmod q\}$. Two bases of this free module are of particular interest for cryptography:

$$\mathbf{B}_h = \begin{bmatrix} 1 & 0 \\ h & q \end{bmatrix} \text{ and } \mathbf{B}_{f,g} = \begin{bmatrix} f & F \\ g & G \end{bmatrix},$$

where $F, G \in \mathscr{R}$ are such that $fG - gF = q$. This module is usually seen as a lattice of volume q^d in $\mathbb{R}^{2d}$ thanks to the coefficient embedding. Lemma 1 shows the formulas for the associated quality parameters of $\mathbf{B}_{f,g}$.

Lemma 1 ([37], adapted). *Let $\mathbf{B}_{f,g}$ be a basis of an NTRU module. We have $\sqrt{q} \leqslant |\mathbf{B}_{f,g}|_{\mathscr{K}} \leqslant s_1(\mathbf{B}_{f,g})$ and :*

$$|\mathbf{B}_{f,g}|_{\mathscr{K}}^2 = \max\left(\|\varphi(ff^* + gg^*)\|_\infty, \left\|\frac{q^2}{\varphi(ff^* + gg^*)}\right\|_\infty\right),$$
$$s_1(\mathbf{B}_{f,g})^2 = \frac{1}{2}\|\varphi\left(T + \sqrt{T^2 - 4q^2}\right)\|_\infty,$$

where $T := ff^* + gg^* + FF^* + GG^*$. We have $|\mathbf{B}_{f,g}|_{\mathscr{K}} = s_1(\widetilde{\mathbf{B}})$, where $\widetilde{\mathbf{B}}$ is the Gram-Schmidt orthogonalization (over $\mathscr{K}$) of $\mathbf{B}_{f,g}$.

2.4 Gaussians over rings

The Gaussian function on $\mathbb{R}^d$ centered at $\mathbf{c}$ and with covariance matrix $\Sigma \succ 0$ is defined as $\rho_{\mathbf{c},\Sigma}(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^t\Sigma^{-1}(\mathbf{x} - \mathbf{c}))$. If $\Sigma = s^2\mathbf{I}_d$, we write also $\rho_{\mathbf{c},s} = \exp(-\|\mathbf{x} - \mathbf{c}\|^2/(2s^2))$ and call the associated Gaussian *spherical*. We omit $\mathbf{c}$ if it is $\mathbf{0}$. The normal distribution $\mathcal{N}_\Sigma$ of covariance Σ then has density probability function $((2\pi)^d \cdot \det\Sigma)^{-1/2}\rho_\Sigma$. When we write $\mathcal{N}_{\mathscr{K}_{\mathbb{R}},s}$, we mean that $(z_1, \ldots, z_d) \leftarrow (\mathcal{N}_{s/\sqrt{d}})^d$ is sampled and $(z_1 + iz_2, \ldots, z_{d-1} + iz_d)$ is outputted.

The discrete Gaussian distribution over a full rank lattice $\mathscr{L}$, centered at $\mathbf{c}$ and with covariance matrix $\Sigma \succ 0$ has density function given by

$$\forall \mathbf{x} \in \mathscr{L}, D_{\mathscr{L},\mathbf{c},\Sigma}(\mathbf{x}) = \frac{\rho_{\mathbf{c},\Sigma}(\mathbf{x})}{\rho_{\mathbf{c},\Sigma}(\mathscr{L})}.$$

For $c \in \mathscr{K}_{\mathbb{R}}$ and $s > 0$, we also use the notation $\lfloor c \rceil_s$ to denote the distribution $D_{\varphi(\mathscr{R}),\varphi(c),s}$. It extends coordinate-wise to vectors in $\mathscr{K}_{\mathbb{R}}^2$. For $\varepsilon > 0$, the smoothing parameter of a lattice $\mathscr{L}$ is $\eta_\varepsilon(\mathscr{L}) = \min\{s > 0 \,:\, \rho_{1/s}(\mathscr{L}^\vee) \leqslant 1 + \varepsilon\}$, where

Algorithm 1: RingPeikert sampler

Input: A matrix $\mathbf{B} \in \mathcal{K}^{2\times 2}$ such that $\mathcal{L} = \varphi(\mathbf{B}\mathcal{R}^2)$ and a target center $\mathbf{c} \in \mathcal{K}_{\mathbb{R}}^2$.
Result: $\mathbf{z} \in \mathcal{L}$ with distribution negligibly far from $D_{\mathcal{L},\mathbf{c},\Sigma}$.

1 *Precomputed: a parameter* $r \geqslant \eta_\varepsilon(\mathcal{R}^2)$*, and* $\Sigma_0 \in \mathcal{K}_{\mathbb{R}}^{2\times 2}$ *such that* $\Sigma_0\Sigma_0^* = \Sigma - r^2\mathbf{B}\mathbf{B}^*$
2 $\mathbf{x} \leftarrow \Sigma_0 \cdot (\mathcal{N}_{\mathcal{K}_{\mathbb{R}},1})^2$
3 $\mathbf{z} \leftarrow \lceil \mathbf{B}^{-1}(\mathbf{c} - \mathbf{x}) \rfloor_r$
4 **return** $\mathbf{Bz}$

$\mathcal{L}^\vee$ is the dual lattice. The exact definition of the lattice dual is not needed in this work, and when $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^d$, it is enough to know the matrix $\mathbf{B}^{-t}$ encodes it. We say that $\sqrt{\Sigma} \geqslant \eta_\varepsilon(\mathcal{L})$ when $\rho_1(\sqrt{\Sigma}^*\mathcal{L}^\vee) = \rho_{\Sigma^{-1}}(\mathcal{L}^\vee) \leqslant 1 + \varepsilon$. In particular, one checks that $r\mathbf{B} \succ \eta_\varepsilon(\varphi(\mathbf{B}\mathcal{R}^2))$ when $r \geqslant \eta_\varepsilon(\mathcal{R}^2)$. We use the following bound.

Lemma 2 (Adapted from [18]). *Let $\mathbf{B}\mathcal{R}^2$ be free $\mathcal{R}$-module, and let $\mathcal{L} = \mathrm{M}(\mathbf{B})\mathbb{Z}^{2d}$ be the associated rank d lattice in $\mathbb{R}^{2d}$. For all $\varepsilon > 0$,*

$$\eta_\varepsilon(\mathcal{L}) \leqslant |\mathbf{B}|_{\mathcal{K}} \cdot \frac{1}{\pi}\sqrt{\frac{\log(2d(1+1/\varepsilon))}{2}}.$$

3 Sampling discrete Gaussians in $\mathcal{R}$-modules

We present three approaches to sample discrete Gaussian over rings. The first two are respectively Peikert's perturbative approach adapted from [34], and the hybrid sampler of Ducas and Prest [37], which is core to MITAKA and uses the first as a subroutine. Then we describe a new sampler based on [11] which can involve integer arithmetic only and combines the ideas of the other two others.

3.1 Peikert's sampler

In [34], Peikert presented an efficient algorithm to sample discrete Gaussians in a target lattice, using small continuous Gaussian perturbation. On a high level, it can be thought of as a randomized version of Babai's round-off algorithm, using random (normal) perturbations to hide the lattice structure, and can be formulated directly over the algebra $\mathcal{K}_{\mathbb{R}}$. The pseudo-code in Algorithm 1 outputs discrete Gaussians in a free rank 2 $\mathcal{R}$-module $\mathcal{L}$ described by a basis $\mathbf{B} \in \mathcal{K}^{2\times 2}$, with an arbitrary center in $\mathcal{K}_{\mathbb{R}}^2$. When $\Sigma \succ r^2\mathbf{B}\mathbf{B}^*$, the existence of Σ_0 below is guaranteed.

Theorem 1 ([34], adapted). *Let $\mathcal{D}$ be the output distribution of Algorithm 1. If $\varepsilon \leqslant 1/2$ and $\sqrt{\Sigma} \geqslant s_1(\mathbf{B}) \cdot \eta_\varepsilon(\mathcal{R}^2)$, then the statistical distance between $\mathcal{D}$ and*

Algorithm 2: RingPeikert, one-dimensional version

Input: A target center $c \in \mathscr{K}_{\mathbb{R}}$.
Result: $z \in \mathscr{R}$ with distribution negligibly far from $D_{\mathscr{R},c,\Sigma}$.

1 *Precomputed: a parameter* $r \geqslant \eta_\varepsilon(\mathscr{R})$, *and* $\sigma_0 \in \mathscr{K}_{\mathbb{R}}$ *such that* $\sigma_0^*\sigma_0 = \Sigma - r^2$
2 $x \leftarrow \sigma_0 \cdot \mathcal{N}_{\mathscr{K}_{\mathbb{R}},1}$
3 **return** $\lceil c - x \rfloor_r$

$D_{\mathscr{L},\mathbf{c},\Sigma}$ is bounded by 2ε. Moreover, we have

$$\sup_{\mathbf{x}\in\mathbf{B}\mathscr{R}^2} \left| \frac{\mathscr{D}(\mathbf{x})}{D_{\mathscr{L}(\mathbf{B}),\mathbf{c},\Sigma}(\mathbf{x})} - 1 \right| \leqslant 4\varepsilon.$$

From Lemma 1 and Lemma 2, note that the condition in the statement ensures that we are above the smoothing parameter of the target lattice. In practice, the covariance parameter is a scalar multiple of the identity matrix, or a positive real "constant" if seen in $\mathscr{K}_{\mathbb{R}}^{++}$. We highlight in Algorithm 2 the one-dimensional version of Peikert's sampler, that is, outputting discrete Gaussians in $\mathscr{R}$, because it appears as a subroutine of the hybrid sampler in the next section.

3.2 Ducas & Prest's hybrid sampler

In [37], a so-called hybrid sampler is presented that outputs discrete Gaussians in free $\mathscr{R}$ modules of finite rank. On a high level, this hybrid sampler follows Klein's approach, which is a randomized version of the Nearest Plane algorithm. In the ring context, the randomization subroutine happens "at the ring level" thanks to a ring Gaussian sampler, instead of "at the integer level". To again hide the lattice structure, perturbations are also involved but their distribution now depends on the target center. The hybrid sampler is described in Algorithm 3, which makes use of floating-point arithmetic, and is core to the MITAKA scheme.

It relies on a ring sampler which can be instantiated by Algorithm 2. For the sake of clarity, these "Peikert sampling" steps are made explicit in lines 4–6 and 9–11. We restrict to "totally spherical" standard deviation parameters (that is, scalar matrices) as they are the main use-case of this work.

Theorem 2 ([37], Theorem 5.10, adapted). *Let $\mathscr{D}$ be the output distribution of Algorithm 3. If $\varepsilon \leqslant 2^{-5}$ and $\sqrt{\Sigma} \geqslant |\mathbf{B}|_{\mathscr{K}} \cdot \eta_\varepsilon(\mathscr{R}^2)$, then the statistical distance between $\mathscr{D}$ and $D_{\mathscr{L},\mathbf{c},\Sigma}$ is bounded by 7ε. Moreover, we have*

$$\sup_{\mathbf{x}\in\mathbf{B}\mathscr{R}^2} \left| \frac{\mathscr{D}(\mathbf{x})}{D_{\mathscr{L},\mathbf{c},\Sigma}(\mathbf{x})} - 1 \right| \leqslant 14\varepsilon.$$

Algorithm 3: Hybrid Gaussian sampler

Input: A target center $\mathbf{c} \in \mathscr{K}_{\mathbb{R}}^2$, a matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ such that $\mathscr{L} = \varphi(\mathbf{B}\mathscr{R}^2)$ and its GSO $[\widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2]$ over $\mathscr{K}$, a parameter $\sigma > 0$ (corresponding to $(\sigma, \ldots, \sigma) \in \mathscr{K}_{\mathbb{R}}$).

Result: $\mathbf{z}$ with distribution negligibly far from $D_{\mathscr{L},\mathbf{c},\sigma^2\mathbf{I}_{2d}}$.

Precomputed: $\sigma_i := \sqrt{\frac{\sigma^2}{\langle \widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_i \rangle} - r^2} \in \mathscr{K}_{\mathbb{R}}^{++}$.

$\mathbf{c}_2 \leftarrow \mathbf{c}, \mathbf{v}_2 \leftarrow 0$

$d_2 \leftarrow \frac{\langle \widetilde{\mathbf{b}}_2, \mathbf{c}_2 \rangle_{\mathscr{K}}}{\langle \widetilde{\mathbf{b}}_2, \widetilde{\mathbf{b}}_2 \rangle_{\mathscr{K}}}$

$u_2 \leftarrow \mathcal{N}_{\mathscr{K}_{\mathbb{R}},1}$

$y_2 \leftarrow \sigma_2 \cdot u_2$

$x_2 \leftarrow \lfloor d_2 - y_2 \rceil_r$

$\mathbf{c}_1 \leftarrow \mathbf{c}_2 - x_2\mathbf{b}_2, \mathbf{v}_1 \leftarrow x_2\mathbf{b}_2$

$d_1 \leftarrow \frac{\langle \widetilde{\mathbf{b}}_1, \mathbf{c}_1 \rangle_{\mathscr{K}}}{\langle \widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_1 \rangle_{\mathscr{K}}}$

$u_1 \leftarrow \mathcal{N}_{\mathscr{K}_{\mathbb{R}},1}$

$y_1 \leftarrow \sigma_1 \cdot u_1$

$x_1 \leftarrow \lfloor d_1 - y_1 \rceil_r$

$\mathbf{v}_0 \leftarrow \mathbf{v}_1 + x_1\mathbf{b}_1$

return $\mathbf{v}_0$

In our integer arithmetic friendly sampler presented in the next section, we rely on a specific variant where the target lattice is described by an upper triangular matrix $\mathbf{U}$, or equivalently, when the Gram-Schmidt orthogonalization is the identity matrix. It is presented in Algorithm 4, and is core to MITAKA$_{\mathbb{Z}}$. In particular, in MITAKA$_{\mathbb{Z}}$, the ring sampler becomes a *discrete* Gaussian sampler that can be emulated in integer arithmetic. This is emphasized below by RingSampler$_{\mathbb{Z}}$.

3.3 An integer arithmetic friendly sampler

To clarify the presentation, in this section we identify matrices over $\mathscr{K}$ to their structured version over $\mathbb{Q}$. Fundamentally, our new sampler for $D_{\mathscr{L}(\mathbf{B}),\mathbf{c},s}$ combines Peikert's approach of Algorithm 1 and hybrid sampling in the case where the Gram-Schmidt is the identity. What allows us to restrict to integer arithmetic is to rely on the work of [11]. There, the authors showed how to generate small *integral* perturbation vectors, relying on a generalization of the Cholesky decomposition that can be also computed purely in integer arithmetic.

On the "hybrid side", as observed in the previous section, it is enough for us to have access to a discrete Gaussian sampler in integer arithmetic. Multiplying the output of Algorithm 4 by the Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}} = \mathbf{B}\mathbf{U}^{-1}$ of the target lattice basis, one would obtain vector with the correct support. The Gram-Schmidt basis may however contain entries in $\mathscr{K}$ that may have very large denominators. We avoid this thanks to an approximation $\widehat{\mathbf{B}} \in (1/(pq))\mathscr{R}^{2\times 2}$ of

Algorithm 4: Hybrid Gaussian sampler, $\mathbf{U}$ version

Input: A target center $\mathbf{c} = (c_1, c_2) \in \mathscr{K}^2$, an upper triangular matrix $\mathbf{U} = [(1, 0), (u, 1)]$ with $u \in \mathscr{K}$, a parameter $r > 0$ (corresponding to $(r, \ldots, r) \in \mathscr{K}_{\mathbb{R}}$).
Result: $\mathbf{z}$ with distribution negligibly far from $D_{\mathscr{L}(\mathbf{U}),\mathbf{c},r}$.

$z_2 \leftarrow \mathsf{RingSampler}_{\mathbb{Z}}(c_2, r)$
$c_1' \leftarrow c_1 - z_2 u$
$z_1 \leftarrow \mathsf{RingSampler}_{\mathbb{Z}}(c_1', r)$
return $\mathbf{z} = \mathbf{U}(z_1, z_2)$.

Algorithm 5: Integer arithmetic ring Gaussian sampler

Input: a matrix $\widehat{\mathbf{B}} \in \mathscr{R}^{2\times 2}$ such that $\widehat{\mathbf{B}}\mathbf{U}_{\hat{u}} = \mathbf{B} = \widetilde{\mathbf{B}}\mathbf{U}_u$, where $\hat{u} = [u]_p \in \frac{1}{p}\mathscr{R}$, a center $\mathbf{c} \in \mathscr{R}^2$, and parameters $r, s > 0$.
Result: $\mathbf{z}$ with distribution negligibly far from $D_{\mathscr{L}(\mathbf{B}),\mathbf{c},rs}$.

Precomputed: $\Sigma_p = s^2\mathbf{I} - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^t$ and $\mathbf{A} \leftarrow \mathsf{IntGram}(p^2(\Sigma_p - \mathbf{I}))$
/* $\mathbf{A}\mathbf{A}^t = p^2(\Sigma_p - \mathbf{I})$ */
$\mathbf{p} \leftarrow$ Algorithm 6$(p, \mathbf{A})$ /* $\mathbf{p} \sim D_{\mathscr{R}^2, r^2\Sigma_p}$ */
$\hat{\mathbf{c}} \leftarrow \widehat{\mathbf{B}}^{-1}(\mathbf{c} - \mathbf{p})$
$\mathbf{z}' \leftarrow$ Algorithm 4$(\hat{u}, \hat{\mathbf{c}}, s)$ /* $\mathbf{z}' \sim D_{\mathscr{L}(\mathbf{U}_{\hat{u}}),\hat{\mathbf{c}},s}$ */
return $\mathbf{z} = \widehat{\mathbf{B}}\mathbf{z}'$

$\widetilde{\mathbf{B}}$, obtained by p-rounding of the upper right coefficient of $\mathbf{U}$. The quality of this approach is essentially driven by $|\mathbf{B}_{f,g}|_{\mathscr{K}} = s_1(\widetilde{\mathbf{B}})$.

Algorithm 5 describes this approach. The notation $\mathbf{U}_{\hat{u}}$ denotes that the upper-right coefficient of the matrix is $\hat{u}$. The procedure IntGram is fully described in [11], and impacts the choice of parameters for the algorithm to be actually correct. On a high level, given in input a positive definite matrix $\Sigma \in \mathscr{R}^{2\times 2}$, it outputs a matrix $\mathbf{A} \in \mathscr{R}^{2\times m}$ such that $\mathbf{A}\mathbf{A}^t = \Sigma$, and where $m \geqslant 2$. In our context, the input is a small perturbation covariance matrix $\Sigma_p = s^2\mathbf{I} - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^t$, where s is a large enough integer.

The offline sampler in Algorithm 6 is adapted from [11] and outputs from the expected distribution as long as $\mathbf{A}$ has been suitably computed. In terms of notation, recall that $\Lambda(\mathbf{A})^\perp \subset \mathscr{R}^m$ is the lattice of integer solutions of $\mathbf{A}\mathbf{x} = \mathbf{0}$.

We now state the correctness of Algorithm 5, stressing that the statement is correct as long as the integral root decomposition could be carried out and $p \geqslant d$.

Theorem 3. *Keep the notation of Algorithm 5, assuming also that* IntGram *correctly computes* $\mathbf{A}$. *For* $\varepsilon \in (0, 1)$, *let* $s > |\mathbf{B}_{f,g}|_{\mathscr{K}}(1 + \sqrt{2d/p}) + 1$ *be an integer and* $r \geqslant \eta_\varepsilon(\mathbb{Z}^{2d})$. *Then the distribution* $\mathscr{D}$ *of the output of Algorithm 5*

Algorithm 6: Offline sampler

Input: An integer $p > 0$, a matrix $\mathbf{A} \in \mathscr{R}^{2\times m}$.
Result: $\mathbf{p} \in \mathscr{R}^2$ with distribution negligibly far from $D_{\mathscr{R}^2, r^2\Sigma}$, where $\Sigma = \frac{1}{p^2}\mathbf{A}\mathbf{A}^t + \mathbf{I}$.

1 *Precomputed:* integers $r > \eta_\varepsilon(\mathscr{R}^2)$ and L such that $Lr \geqslant \eta_\varepsilon(\Lambda(A)^\perp)$.
2 $\mathbf{x} \leftarrow (\lfloor 0 \rceil_{Lr})^m$
3 $\mathbf{p}' \leftarrow \frac{1}{pL}\mathbf{A}\mathbf{x}$
4 $\mathbf{p} \leftarrow \lfloor \mathbf{p}' \rceil_r$
5 **return** $\mathbf{p}$.

is at statistical distance at most 15ε *from* $D_{\mathscr{L}(\mathbf{B}),\mathbf{c},sr}$. *Moreover, we have*

$$\sup_{\mathbf{z}\in\mathscr{L}(\mathbf{B})} \left| \frac{\mathscr{D}(\mathbf{z})}{D_{\mathscr{L}(\mathbf{B}),\mathbf{c},sr}(\mathbf{z})} - 1 \right| \leqslant 30\varepsilon.$$

3.4 Asymptotic security of the samplers

Hash-and-sign signatures over lattices are constructed, following the GPV framework [18], by hashing a message to the ambient space of the lattice, and returning as a signature a lattice point close to that hash digest. This is done using a "good" representation of the lattice, called the *trapdoor*, that enables the signer to solve the ApproxCVP problem with a relatively small approximation factor. Moreover, to prevent signatures from leaking information about the secret trapdoor, the close lattice points need to be sampled according to a distribution that is statistically independent of the trapdoor: usually a spherical discrete Gaussian distribution supported over the lattice and centered at the hash digest. This is where the algorithms from the previous sections come into play.

The security of the resulting signature scheme depends on the standard deviation of the discrete Gaussian distribution output by the sampler: the smaller the standard deviation, the closer the distance to the hash digest, the harder the corresponding ApproxCVP problem, and hence the higher the security level. As we have seen, however, there is a lower bound (depending on the trapdoor) to how small of a standard deviation the sampler can achieve while remaining statistically close to the desired spherical Gaussian: lower than that, and the distribution may start to deviate from that Gaussians in ways that could expose information about the secret trapdoor, and thus compromise the security of the signing key.

In the case of NTRU lattices, the trapdoor is the secret basis: $\mathbf{B}_{f,g} = \begin{bmatrix} f & F \\ g & G \end{bmatrix}$, and the standard deviation of the discrete Gaussian obtained from this trapdoor varies depending on the sampling algorithm, as discussed in particular in [37, §6]. It can be written as:

$$\sigma = \alpha \cdot \eta_\varepsilon(\mathscr{R}^2) \cdot \sqrt{q} \tag{1}$$

Table 1. Comparison of the best achievable trapdoor quality α for the various Gaussian samplers over NTRU lattices.

Sampler	$\alpha\sqrt{q}$	Best achievable α
Peikert	$s_1(\mathbf{B}_{f,g})$	$O(d^{1/4}\sqrt{\log d})$ [37, §6.5.2]
Hybrid (Mitaka)	$\lvert\mathbf{B}_{f,g}\rvert_{\mathscr{K}}$	$O(d^{1/8}\log^{1/4} d)$ [full version of this paper [16]]
Klein (Falcon)	$\lVert\mathbf{B}_{f,g}\rVert_{\mathrm{GS}}$	$O(1)$ [37, §6.5.1]

where the factor $\alpha \geqslant 1$, which we call the *quality*, depends on the sampler for a given trapdoor.

For the so-called Klein sampler used in DLP and Falcon, $\alpha\sqrt{q}$ is the Gram–Schmidt norm $\lVert\mathbf{B}_{f,g}\rVert_{\mathrm{GS}} := \max_{1\leqslant i\leqslant 2d}\lVert\widetilde{\mathbf{b}}_i^{\mathbb{Z}}\rVert_2$ of $\mathbf{B}_{f,g}$ *over the integers*. For the Peikert sampler over $\mathscr{K}$, Theorem 1 shows that $\alpha\sqrt{q} = s_1(\mathbf{B}_{f,g})$. Finally, for the hybrid sampler, Theorem 2 shows that $\alpha\sqrt{q} = \lvert\mathbf{B}_{f,g}\rvert_{\mathscr{K}}$.

For a given sampler, the generators f, g should be sampled appropriately to minimize the corresponding α. In his thesis [37], Prest analyzed the optimal choices both theoretically (under suitable heuristics) and experimentally. The resulting optimal choices for α are as follows (after correcting the flawed heuristic analysis of Prest in the case of the hybrid sampler; detailed discussion is in the full version [16]):

- heuristically, the quality of the Peikert sampler satisfies $\alpha = O(d^{1/4}\sqrt{\log d})$ [37, §6.5.2];
- for the hybrid sampler, we show $\alpha = O(d^{1/8}\log^{1/4} d)$ (and not $O(\sqrt{\log d})$ contrary to what was claimed in [37, §6.5.2] based on flawed heuristics);
- for the Klein sampler (used in DLP, and in modified form, Falcon), the heuristic analysis in [37, §6.5.1] show that it can be taken as low as $\sqrt{e/2} \approx 1.17$ independently of the dimension, and in particular $\alpha = O(1)$.

These properties are summarized in Table 1.

3.5 The Mitaka signature scheme

The previous samplers can be plugged directly into the GPV framework [18] to construct secure hash-and-sign signature schemes in the random oracle model. The idea is to sign a message by first hashing it as a point in the ambient space of the lattice, and then using the sampler to construct a lattice point close to that target. The signature is then the difference between the target and the lattice point (a small vector in the lattice coset defined by the target). The signing procedure is described more precisely in Algorithm 8. Both Mitaka and Mitaka$_{\mathbb{Z}}$ are specific instantiations of this paradigm, using the samplers of Algorithms 3 and 5 respectively. Mitaka generates the key pair in a similar manner to Falcon, but some techniques are introduced to improve the quality of the trapdoor (see Section 4 for details). In particular, as done in Falcon, it is also possible to trade secret-key size for efficiency by giving only 3 of the

Algorithm 7: Key generation

Input: Global parameter $\Psi = (\mathscr{R}, q, \sigma, r)$, quality parameter α.
Result: A key pair $(\mathsf{pk}, \mathsf{sk})$.

$(f, g) \leftarrow \mathsf{FirstVec}(\Psi, \alpha)$ satisfying $|\mathbf{B}_{f,g}|^2_{\mathscr{K}} \leqslant \alpha^2 q$ `/* Algorithm 10 */`
if f *is not invertible* mod q **then** restart
$(F, G) \leftarrow \mathsf{NTRUSolve}(\Psi, (f, g))$ satisfying $fG - gF = q$
$[\mathbf{b}_1, \mathbf{b}_2] \leftarrow [(f, g), (F, G)]$
$\widetilde{\mathbf{b}}_2 \leftarrow \mathbf{b}_2 - \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle_{\mathscr{K}}}{\langle \mathbf{b}_1, \mathbf{b}_1 \rangle_{\mathscr{K}}} \mathbf{b}_1 \in \mathscr{K}^2$
$n_i \leftarrow \langle \widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_i \rangle_{\mathscr{K}} \in \mathscr{K}^{++}_{\mathbb{R}}$ for $i = 1, 2$
$\sigma_i \leftarrow \sqrt{\frac{\sigma^2}{n_i} - r^2} \in \mathscr{K}^{++}_{\mathbb{R}}$ for $i = 1, 2$
$\mathsf{pk} \leftarrow f^{-1} g \bmod q$, $\mathsf{sk} = (\mathbf{b}_1, \mathbf{b}_2, n_1^{-1}\mathbf{b}_1, n_2^{-1}\widetilde{\mathbf{b}}_2, \sigma_1, \sigma_2)$
return $(\mathsf{pk}, \mathsf{sk})$

4 polynomials in the basis and enough information to recover Gram-Schmidt data by on-the-fly computations, all in Fast Fourier format. The verification algorithms of MITAKA and FALCON are exactly the same. For completeness, we describe the key generation and verification in Algorithms 7 and 9.

In Algorithm 8, the acceptance bound γ for signatures is chosen slightly larger than $\sigma\sqrt{d}$, for σ the standard deviation of the sampler given by Eq. (1) above, in order to ensure a low repetition rate for signature generation. (In the concrete security evaluation of Section 5, γ is selected so as to ensure $< 10\%$ rejection; this gives e.g. $\gamma = 1.042\sigma\sqrt{2d}$ for $d = 512$). Signature verification simply recovers the second component $s_2 = s_1 \cdot h + c \bmod q$ and checks that the vector $\mathbf{s} = (s_1, s_2)$ is of length at most γ.

The security argument of Gentry, Peikert, and Vaikuntanathan reduces the security of the signature scheme to the hardness of SIS in the underlying lattice up to bound 2γ. It is therefore invalidated if an attacker can obtain two distinct outputs of the sampler with the same center (since their difference would be a solution to this SIS problem) [18, Section 6.1]. This is avoided in the signature scheme by randomizing the hash value associated with the message using a sufficiently long random salt $\mathsf{r} \in \{0, 1\}^k$. To avoid collisions, it suffices to pick $k \geqslant \lambda + \log_2 q_s$ for λ bits of security and q_s signature queries. The choice of $k = 320$ as in [38,7] suffices for up to 256 bits of security.

4 Improved Trapdoor Generation

The Peikert, hybrid and FALCON samplers for an NTRU basis $\mathbf{B}_{f,g}$ all have essentially the same complexity, and the first two are significantly simpler, easier to implement, slightly faster in the same dimension, and offer better avenues for parallelization and side-channel resistance (see Section 7). It would therefore be desirable to adopt one of the first two for practical implementations.

Algorithm 8: Signing

Input: A message m, a secret key sk, a bound γ.
Result: A signature sig of m.

do
  $\mathsf{r} \xleftarrow{\$} \{0,1\}^k$, $c \leftarrow H(\mathsf{r} \| \mathsf{m})$
  $\mathbf{z} \leftarrow \mathsf{Sampler}(\mathsf{sk}, (0, c))$ /* Algorithm 3 or 5 */
  $\mathbf{s} \leftarrow (s_1, s_2) = (0, c) - \mathbf{z}$ /* $s_1 \cdot h - s_2 \equiv -c \bmod q$ */
while $\|\mathbf{s}\|^2 > \gamma^2$
return $\mathsf{sig} = (\mathsf{r}, s_1)$.

Algorithm 9: Verification

Input: A signature $\mathsf{sig} = (\mathsf{r}, s_1)$ of m, a public key $\mathsf{pk} = h$, a bound γ.
Result: Accept or reject.

$c \leftarrow H(\mathsf{r} \| \mathsf{m})$
$s_2 \leftarrow c + s_1 \cdot h \bmod q$
if $\|(s_1, s_2)\|^2 > \gamma^2$ **then return** Reject.
return Accept.

However, as seen in Section 3.4, the Falcon sampler has a substantial advantage in terms of security, since its Gaussian standard deviation is proportional to $\|\mathbf{B}_{f,g}\|_{\mathrm{GS}}$, whereas the Peikert and hybrid samplers are proportional to $s_1(\mathbf{B}_{f,g})$ and $|\mathbf{B}_{f,g}|_{\mathscr{K}}$ respectively, which are both larger. This results in a significant difference in asymptotic terms, as shown in Table 1, and also in bit security terms as will become apparent in the next section.

To increase the security level achievable using the first two samplers, and in particular the hybrid sampler, we propose a new technique to significantly improve the quality of NTRU trapdoors. We note in passing that it also applies to Falcon: while it cannot yield significant improvements in terms of security, since the standard deviation it achieves is already a very small factor away from the theoretical optimum, it can be used to speed up key generation substantially. The idea is as follows.

Recall that NTRU trapdoor generation for Falcon, say, works by sampling f, g with discrete Gaussian coefficient, computing the $\|\mathbf{B}_{f,g}\|_{\mathrm{GS}}$ of the resulting NTRU basis, and checking if this value is below the desired quality threshold. If not, we try again, and if so, the NTRU basis is completed and kept as the secret key. Trapdoor sampling for the hybrid sampler is similar. (On the other hand, for Peikert, completion has to be recomputed at each step to evaluate $s_1(\mathbf{B}_{f,g})$).

In this process, the costly operations are, on the one hand, the generation of the discrete Gaussian randomness, which has to be repeated several dozen times over in order to reach the desired threshold (this is not explicitly quantified by the authors of Falcon, but experiments suggest that, in many instances,

upwards of 50 iterations are necessary), and, on the other hand, the completion of the basis (still costly despite recent optimizations [36]), which is only carried out once at the end and not for each iteration[13].

To optimize the process, our idea is to amortize the cost of discrete Gaussian sampling, by constructing several candidate trapdoors from the same randomness. We propose three main ideas to do so.

Lists of candidates for f and g. The usual key generation algorithm for FALCON, as already mentioned, normally ends up generating many pairs (f_i, g_i), and tests each of them as a candidate first vector for the NTRU lattice.

Since we are generating Gaussian vectors f_i and g_i anyway, we can easily recycle this generated randomness by testing all the mixed pairs (f_i, g_j) instead: this results in a set of possible candidates which increases quadratically with the number of random vectors we generate, instead of just linearly.

Generating the Gaussian vectors as linear combinations. Independently, one can generate each candidate vector f as a linear combination $\sum_{k=1}^{\ell} f^{(k)}$ where each $f^{(k)}$ is sampled from a discrete Gaussian of standard deviation $\sigma_0/\sqrt{\ell}$, for σ_0 the desired standard deviation of f. It is well-known that this results in the correct distribution provided that $\sigma_0/\sqrt{\ell}$ remains above the smoothing parameter of $\mathbb{Z}$ [32]. In fact, the FALCON implementation already does so for $d = 512$, where the candidate vectors are sums of two Gaussians vectors of standard deviation $\sqrt{2}$ times lower.

Now, when generating several f_i's, one obtains ℓ lists $L_k = \{f_i^{(k)}\}_i$ of Gaussian vectors. It is again possible to recycle this generated randomness by mixing and matching among those lists, and constructing candidates f of the form $\sum f_{i_k}^{(k)}$ for varying indices i_k, so that the total set of candidates is in bijection with $\prod_k L_k$. Its size increases like the ℓ-th power of the size of the lists.

Using the Galois action. Finally, one can expand the set of candidates for g, say, by applying the action of the Galois group. In principle, other unitary transformations of g, even less structured ones like randomly permuting the coefficients in the power basis, could also be considered, but the Galois action in particular is convenient as it is expressed as a circular permutation on the embeddings $\varphi_i(g)$ of g (i.e., the Fourier coefficients), and for the hybrid sampler, the computation of the quality is entirely carried out in the Fourier domain.

Concretely, recall from Lemma 1 that the quality parameter α of the hybrid sampler associated with $\mathbf{B}_{f,g}$ satisfies:

$$\alpha^2 = \frac{|\mathbf{B}_{f,g}|_{\mathscr{K}}^2}{q} = \max\left(\frac{\max_i z_i}{q}, \frac{q}{\min_i z_i}\right)$$
$$\text{where } z_i = \varphi_i(ff^* + gg^*) = |\varphi_i(f)|^2 + |\varphi_i(g)|^2 \in \mathbb{R}^+.$$

[13] This is the case at least for FALCON and for the hybrid sampler, as for both of them, one can compute the quality of the trapdoor given only (f, g). This is especially fast for the hybrid sampler. For the Peikert sampler, however, doing so without also obtaining (F, G) seems difficult, and is left as an open problem.

It is easy to compute the embeddings z_i^τ associated to $\mathbf{B}_{f,\tau(g)}$ for some Galois automorphism τ of $\mathscr{K}$ simply by applying the corresponding permutation on the components of $\varphi(g)$. Moreover, we see from this representation that the conjugation $\tau_*\colon g \mapsto g^*$ leaves this quality invariant, so the relevant Galois elements to consider are a set of representatives of $\mathrm{Gal}(\mathscr{K}/\mathbb{Q})/\langle\tau_*\rangle$. For power-of-two cyclotomics, one can for example use τ_5^k for $k = 0, \ldots, d/2-1$, where $\tau_5(\zeta_d) = \zeta_d^5$.

Security considerations. The techniques above can potentially skew the distribution of f and g somewhat compared to the case when each tested (f, g) that fails to pass the security threshold is thrown away. However, this is not really cause for concern: the precise distribution of f and g is not known to affect the security of the signature scheme other than in two ways:

- the extent to which it affects the geometry of the trapdoor, as encoded in the quality parameter α already; and
- the length of (f, g) itself as it affects key recovery security, but this length is always at least as large in our setting as in Falcon.

This indicates that our optimized secret keys do not weaken the scheme.

Concrete example. In Algorithm 10, we describe an example key generation procedure that combines all three techniques presented above: we construct lists of candidates for f and g and test all possible pairs. Moreover, each f and g itself is sampled as a sum of $\ell = 2$ narrower Gaussians, and the list of g's is expanded using the Galois action. Of course, different combinations of the techniques are also possible, but this particular one offers a good balance between efficiency and achievable security.

Using this approach, as shown in Fig. 1, we are able to efficiently generate trapdoors with $\alpha \leqslant 2.04$ for $d = 512$, and $\alpha \leqslant 2.33$ for $d = 1024$ by $m \approx 16$ (corresponding to generating 64 narrow Gaussian vectors to select one candidate (f, g), largely in line with Falcon).

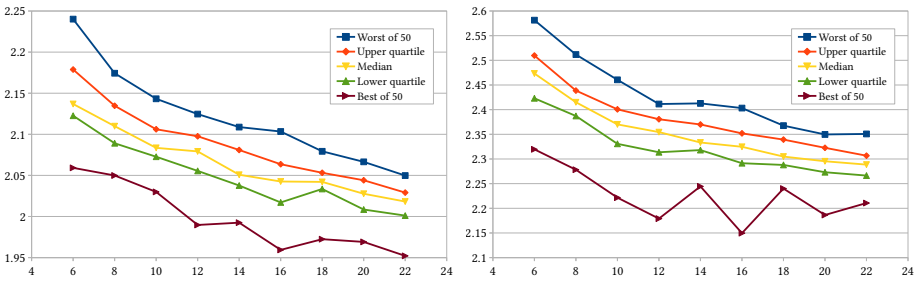

Fig. 1. Quality α reached by the optimized sampler of Algorithm 10 for various choices of m (50 trials each, $\sigma_0 = 1.17\sqrt{q/2d}$, G coset representatives of $\mathrm{Gal}(\mathscr{K}/\mathbb{Q})/\langle\tau_*\rangle$). Reachable α in dimension 512 (left) and 1024 (right).

Algorithm 10: MITAKA optimized key generation

Input: Desired standard deviation σ_0 of f and g, target quality α of the Gaussian, number of samples m to generate, set G of Galois automorphisms to apply. The total search space is of size $\#G \cdot m^4$ for $4m$ generated discrete Gaussian vectors.

Result: NTRU first trapdoor vector (f, g) with quality better than α.

1 **for** $i \in [1, m]$ **do**
2 $\quad f'_i \leftarrow D_{\mathscr{R},\sigma_0/\sqrt{2}}$, $f''_i \leftarrow D_{\mathscr{R},\sigma_0/\sqrt{2}}$
3 $\quad g'_i \leftarrow D_{\mathscr{R},\sigma_0/\sqrt{2}}$, $g''_i \leftarrow D_{\mathscr{R},\sigma_0/\sqrt{2}}$
4 **end for**
5 $L_f \leftarrow \{f'_i + f''_j \mid i, j \in [1, m]\}$
6 $L_g \leftarrow \{\tau(g'_k + g''_\ell) \mid k, \ell \in [1, m], \tau \in G\}$
7 $L_u \leftarrow \{(f, \varphi(ff^*)) \mid f \in L_f\}$
8 $L_v \leftarrow \{(g, \varphi(gg^*)) \mid g \in L_g\}$
9 **for** $(f, u) \in L_u, (g, v) \in L_v$ **do**
10 $\quad z \leftarrow u + v$
11 $\quad$ **if** $q/\alpha^2 \leqslant z_i \leqslant \alpha^2 q$ *for all* i **then return** (f, g)
12 **end for**
13 **restart**

Improved search via early aborts and filtering. Key generation using the technique above involves an exhaustive search in a relatively large set of candidates $L_u \times L_v$, and testing each candidate involves $O(d)$ comparisons:

$$q/\alpha^2 \leqslant u_i + v_i \leqslant \alpha^2 q \quad \text{for } 1 \leqslant i \leqslant d/2,$$

as done in Step 11 of Algorithm 10. One can of course reject a candidate immediately as soon as one of the comparison fails, but this can happen arbitrarily late in the loop through the indices.

However, the lower bound condition is much more likely to fail than the upper bound for a given candidate (see the full version [16] for detailed analysis). Moreover, if we fix u, then it is more likely to fail on any given v for the indices i such that u_i is small. One can therefore improve the algorithm by a wide margin by carrying out a simple precomputation on u: extract the list of indices $S_u(w)$ of the w smallest elements of u for some $w \ll d/2$ (this can be done without sorting, in time $O(d)$). Then, for each corresponding candidate v, first check in time $O(w)$ whether the lower bound condition holds on the indices in $S_u(w)$: if so, the comparison is carried out normally, and otherwise v is rejected early.

Picking for example $w = 25$, we find that around 99.8% of candidates are rejected early in that way for our parameters, greatly reducing the number of full-fledged comparisons. All in all, this lets us achieve a speed-up of more than 5- to 10-fold as d ranges from 512 to 1024.

An additional, very simple optimization is to filter out values u, v such that $\|u\|_\infty > \alpha^2 q$ (and similarly for v) from the lists L_u and L_v, since such candidates clearly cannot satisfy the comparison.

5 Security analysis of Mitaka

Concrete security. In order to assess the concrete security of our signature scheme, we proceed using the usual cryptanalytic methodology of estimating the complexity of the best attacks against *key recovery attacks* on the one hand, and *signature forgery* on the other. For the parameter choices relevant to our scheme (in which the vectors of the trapdoor basis are not unusually small), key recovery is always harder than signature forgery, and therefore the cost of signature forgery is what determines the security level. The security of the forgery is a function of the standard deviation of the lattice Gaussian sampler used in the signature function, which itself depends on the quality α of the trapdoor, as discussed in Section 3.4. This analysis translates into concrete bit-security estimates following the methodology of NewHope [1], sometimes called "core-SVP methodology". In this model [5,26], the bit complexity of lattice sieving (which is asymptotically the best SVP oracle) is taken as $\lfloor 0.292\beta \rfloor$ in the classical setting and $\lfloor 0.265\beta \rfloor$ in the quantum setting in dimension β. The resulting security in terms of α is given in Fig. 2 in dimensions 512 and 1024. This allows us to compare Mitaka with Falcon as well as with a "naive" version of the hybrid sampler that would omit the optimizations of Section 4; the results are presented in Table 2. Detailed and comprehensive analysis is given in the full version [16].

In addition, as mentioned earlier, our construction can be instantiated over more general base fields than power-of-two cyclotomics, which enables us to choose security level in a much more flexible way than Falcon. Example security levels which can be reached in this way are presented in Table 4. For such fields, we can choose the modulus q to be the first prime which is congurant to 1 modulo the conductor. Again, detailed analysis is given in the full version [16].

Asymptotic security. As for all signature schemes in the GPV framework, the EUF–CMA security of our scheme in an asymptotic sense reduces, both in the classical [18] and quantum random oracle models [6], to the SIS problem in the underlying lattice (in this case, an instance of Module–SIS [27]). However, as is the case for Falcon (and as holds, similarly, for Dilithium), the SIS bound

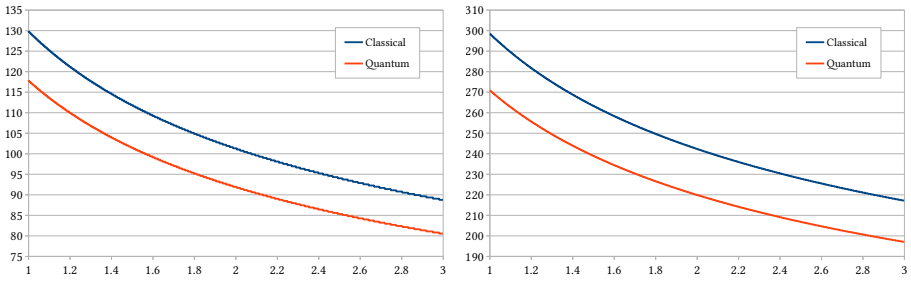

Fig. 2. Security (classical and quantum) against forgery as a function of the quality $1 \leqslant \alpha \leqslant 3$ of the lattice sampler (left: dimension 512 and right: dimension 1024).

Table 2. Concrete values for sampler quality and associated bit security level.

	$d = 512$				$d = 1024$			
	Quality α	Classical	Quantum	NIST Level	Quality α	Classical	Quantum	NIST Level
FALCON	1.17	124	112	I	1.17	285	258	V
Naive Hybrid[a]	3.03	90	82	below I	3.58	207	188	IV
MITAKA	2.04	**102**	**92**	I[b]	2.33	**233**	**211**	V

[a] Key generation with the same median amount of randomness as MITAKA Algorithm 10 with $m = 16$, but without the optimizations of Section 4.
[b] Taking into account the heavy memory cost of sieving. This is the same level as Dilithium–II; see [28, §5.3].

Table 3. Intermediate parameters and security levels for MITAKA.

	$d = 512$	$d = 648$	$d = 768$	$d = 864$	$d = 972$	$d = 1024$
Conductor	2^{10}	$2^3 \cdot 3^5$	$2^8 \cdot 3^2$	$2^5 \cdot 3^4$	$2^2 \cdot 3^6$	2^{11}
Security (C/Q)	102/92	136/123	167/151	192/174	220/199	233/211
NIST level	$\mathrm{I}^{-(a)}$	$\mathrm{I}^{(b)}$	II	III	IV	V
Modulus q	12289	3889	18433	10369	17497	12289
Quality α	2.04	2.13	2.20	2.25	2.30	2.33
Sig. size (bytes)	713	827	1080	1176	1359	1405

[a] Above round 2 Dilithium–II. [b] Above FALCON–512; arguably NIST level II.

in Euclidean norm for the standard parameter choice ($q = 12289$) makes the underlying assumption vacuous. This is not known to lead to any attack, and can be addressed by increasing q if so desired, or reducing to the SIS problem in infinity norm instead.

6 Implementation Results

In order to assess the practicality of MITAKA, we carried out a preliminary, pure C implementation of the scheme (using the sampler described in Algorithm 3). For easier and fairer comparison, we reused the polynomial arithmetic and FFT of the reference implementation of FALCON, as well as its pseudorandom generator (an implementation of ChaCha20). Our implementation is available at https://github.com/espitau/Mitaka-EC22.

An important caveat is that the current version of our code includes direct calls to floating point transcendental functions, and therefore cannot be guaranteed to run in constant time as is. It is well-known that this can be addressed using the polynomial approximation techniques used e.g. in [41,4,23], but full precision estimates for the required functions are left as future work.

The result of those tests run on a single core of an Intel Core i7–1065G7 @ 1.30GHz laptop can be found in Table 4. We can see that MITAKA is approximately twice as fast as the other lattice-based candidates. However, it should be noted that all those tests were performed using reference implementations. While the comparison with FALCON can be seen as fair due to the shared code, the comparison with a non-optimized implementation of Dilithium is somewhat

Table 4. Performance comparisons of Mitaka with Falcon and Dilithium at the lowest and highest security levels. The private key size corresponds to the expanded key for all schemes (including the Falcon tree for Falcon, the expanded public matrix for Dilithium, and the precomputed sampling data for Mitaka).

	Lowest security level			Highest security level		
	Falcon–512	Dilithium–2	Mitaka–512	Falcon–1024	Dilithium–5	Mitaka–1024
Security (C/Q)	124/112	121/110	102/92	285/258	230/210	233/209
Claimed NIST level	I	II	I^-	V	V	V
Sig. size (bytes)	666	2420	713	1280	4595	1405
Pub. key size (bytes)	896	1312	896	1792	2592	1792
Priv. key size (kB)	56	18	16	120	82	32
Sig. time (kcycles)	502	466	248	1191	931	514

less relevant. Still, we believe that these preliminary results are quite promising for Mitaka.

Furthermore, performance is mainly driven by the cost of the continuous and discrete one-dimensional Gaussian samplers. Since signature generation can be split in an offline part and an online part, Mitaka can offer even better speed results if some computations can be performed between signatures. While these results are favorable to Mitaka, optimized implementations on specific architectures would be needed for a definitive comparison to Falcon and Dilithium.

7 Side-channel Countermeasure

First, our signature scheme can be easily made isochronous. According to [23], isochrony ensures independence between the running time of the algorithm and the secret values. For our signature, the absence of conditional branches implies that one can implement our signature isochronously using standard techniques.

In a second step, we turn our signature scheme into an equivalent one which is protected against more powerful side-channel attacks that exploit the leakage of several executions. More precisely, following the seminal work due to Ishai, Sahai, and Wagner [24], we aim to protect our samplers for Mitaka and Mitaka$_{\mathbb{Z}}$ alternative described in Section 3 from the so-called *t-probing side-channel adversary*, who is able to peek at up to t intermediate variables per each invocation of an algorithm. The *masking* countermeasure is a technique to mitigate such attacks, by additively *secret-sharing* every sensitive variables into $t+1$ values in $\mathcal{R}$. The integer t is often referred to as *masking order*. Essentially, we will provide two functionally equivalent alternative algorithms for Mitaka and Mitaka$_{\mathbb{Z}}$ where any set of at most t intermediate variables is independent from the secret. In this paper, we consider the masking order as a—potentially large—arbitrary variable t. Clearly, high masking order allows a side-channel protected implementation to tolerate stronger probing attacks with larger number of probes.

For a ring element $a \in \mathcal{R}$, we say that a vector $(a_i)_{0 \leqslant i \leqslant t} \in \mathcal{R}^{t+1}$ is an arithmetic masking of a if $a = \sum_{i \in [0,t]} a_i$. For readability, we often write $[\![a]\!] := (a_i)_{0 \leqslant i \leqslant t}$.

The masking of our signature presents three unprecedented difficulties in masked lattice-based schemes:

1. Compared to Fiat-Shamir with aborts, masking the Gaussian sampling is unavoidable. We here present a novel technique to efficiently mask Gaussian sampling in Section 7.2.1. Notably, our approach only requires arithmetic masking, allowing us to avoid any conversion between arithmetic and Boolean shares during the online phase.
2. The computations are performed in $\mathbb{Z}$ instead of a modular ring. This feature does not appear in any other lattice-based scheme. Thus, we need to fix a bound on the size of the masks and make sure that the computations will never pass this bound. Let Q^{mask} be the bound on the largest manipulated integer, the shares of $[\![a]\!]$ are implicitly reduced modulo Q^{mask}. Sometimes we refine the notation $[\![\cdot]\!]$ into $[\![\cdot]\!]_M$ to explicitly specify a modulus $M < \mathsf{Q}^{\text{mask}}$ for secret-sharing.
3. Some polynomial multiplications need both inputs to be masked. This unusual operation does not appear in LWE-based schemes where the multiplications are performed between a public matrix of polynomial and a masked vector. We handle this problem with a function in Section 7.2.2.

7.1 Preliminaries on masking countermeasure

The most basic security notion for a masking countermeasure is the *t-privacy* of a gadget G [24]. This notion guarantees that any set of at most t intermediate variables observed during the computation is independent of the secret input. While the idea behind the notion is relatively simple, t-private gadgets are unfortunately not composable, meaning that a gadget consisting of multiple t-private sub-gadgets may not be necessarily secure. Hence in this work we rely on the following more handy security notions introduced by Barthe et al. [2].

Definition 1 (t-NI, t-SNI). *Let G be a gadget with inputs $(x_i)_{0 \leqslant i \leqslant t} \in \mathcal{R}^{t+1}$ and outputs $(y_i)_{0 \leqslant i \leqslant t} \in \mathcal{R}^{t+1}$. Suppose that for any set of t_1 intermediate variables and any subset of $O \subseteq [1,t]$ of output indices with $t_1 + |O| \leqslant t$, there exists a subset of indices $I \subseteq [1,t]$ such that the output distribution of the t_1 intermediate variables and the output variables $(y_i)_{i \in O}$ can be simulated from $(x_i)_{i \in I}$. Then*

- *if $|I| \leqslant t_1 + |O|$ we say G is* t-non-interfering (t-NI)*; and*
- *if $|I| \leqslant t_1$ we say G is* t-strong-non-interfering (t-SNI).

It is easy to check that t-SNI implies t-NI which implies t-probing security. The above notion can be naturally extended for a gadget with multiple input and output sharings. Note that *linear* operations performed share-wise (such as addition of two sharings, or multiplication by a constant) are trivially t-NI, as each computation on share i can be simulated from the input share x_i. Building blocks satisfying either NI or SNI can be easily composed with each other,

Table 5. Masking properties of known and new gadgets

Gadget name	Security property	Reference
SecMult	t-SNI	[24,39,2]
Refresh	t-SNI	[8,2]
Unmask	t-NIo	[3]
MaskedCDT	t-NI	[4,19]
SecNTTMult	t-SNI	This work, Lemma 4
GaussShareByShare	t-NIo	This work, Lemma 3

by inserting the Refresh gadgets at suitable locations to re-randomize shares [2, Proposition 4]. It is also internally used in the Unmask gadget before taking the sum of shares, so that a probe on any partial sum doesn't leak more information than one input share [3].

Typically, the non-interference notions only deal with gadgets where all of the inputs and outputs are sensitive. To also handle public, non-sensitive values, a weaker notion called *NI with public output* (t-NIo) was proposed in [3]. As stated in [3, Lemma 1], if a gadget G is t-NI secure it is also t-NIo secure for any public outputs.

In the sequel, we use the SecMult gadget that computes the multiplication of two masked inputs. It is one of the key building blocks of masking theory and has been introduced in [24,39] and proved t-SNI in [2].

We also use the MaskedCDT gadget that generates a masked sample that follows a tabulated Gaussian distribution of a fixed center c and a fixed standard deviation r. The table values are not sensitive so they are the same as for the unmasked implementation. This masked CDT algorithm was introduced in [4,19] and proved t-NI.

7.2 Two new gadgets

In Table 5, we introduce the known and new gadgets necessary for our sampler along with their properties. These properties will be proved in the following subsections.

7.2.1 Share-by-share Gaussian sampling In this section, we present a novel technique for generating a masked Gaussian sampling with an arbitrary masked center $[\![c]\!]$ of $c \in 1/C \cdot \mathbb{Z}$ for some fixed integer C. Note that $1/C \cdot \mathbb{Z}$ is not a ring, and thus the multiplication is not well-defined for shares in $1/C \cdot \mathbb{Z}$. This is not an issue in our application, since we never carry out multiplication of two sharings in this form.

We aim at considering a share by share generation. A direct and fast approach is to generate $z_i \leftarrow D_{\mathbb{Z},c_i,r/\sqrt{t+1}}$ for each share of c and to output $(z_0, \cdots, z_t)$ as $[\![z]\!]$. To ensure $z \sim D_{\mathbb{Z},c,r}$, it requires $r \geq \sqrt{2(t+1)}\eta_\epsilon(\mathbb{Z})$ according to [31], which

Algorithm 11: GaussShareByShare

Input: An unmasked standard deviation r. An arithmetic masking $[\![c]\!]$ of the center $c \in 1/C \cdot \mathbb{Z}$. Let $B := \lceil \sqrt{2(t+1)} \rceil$.
Result: An arithmetic masking $[\![z]\!]$ with z's distribution negligibly far from $D_{\mathbb{Z},c,r}$.

1 **for** $i \in [0, t]$ **do**
2 $\quad z_i \leftarrow D_{1/B \cdot \mathbb{Z}, c_i, r/\sqrt{t+1}}$
3 **end for**
`/* Extracting the fractional part of` z `*/`
4 $[\![\bar{z}]\!]_1 \leftarrow (z_0 \bmod 1, \ldots, z_t \bmod 1)$ `/* secret-sharing in` $(\frac{1}{B} \cdot \mathbb{Z})/\mathbb{Z}$ `*/`
5 **if** Unmask$([\![\bar{z}]\!]_1) \neq 0$ **then**
6 $\quad$ **restart** to step 1
7 **end if**
8 **return** $(z_0, \ldots, z_t)$

yields a considerable security loss. To overcome this issue, we propose a different approach sampling shares over $1/B \cdot \mathbb{Z}$ with $B := \lceil \sqrt{2(t+1)} \rceil$ and utilizing rejection sampling to keep the masked output over $\mathbb{Z}$. Our masked Gaussian sampling algorithm is presented in Algorithm 11.

Correctness We now show that Algorithm 11 is correct for $r \geq \eta_\epsilon(\mathbb{Z})$. Since $r \geq \eta_\epsilon(\mathbb{Z}) \geq \frac{\sqrt{2(t+1)}}{B} \eta_\epsilon(\mathbb{Z})$, by [31, Theorem 3], in step 4, $z = \sum_{i=0}^{t} z_i$ follows $D_{1/B \cdot \mathbb{Z}, c, r}$. Thanks to the rejection sampling, the support of the final output z is $\mathbb{Z}$ and noticing that the probability of each output z is proportional to $\rho_{r,c}(z)$, it follows that the distribution of z is $D_{\mathbb{Z},c,r}$. The rejection rate is $\frac{\rho_{r,c}(\mathbb{Z})}{\rho_{r,c}(1/B \cdot \mathbb{Z})} \approx 1/B$ as $r \geq \eta_\epsilon(\mathbb{Z}) \geq \eta_\epsilon(1/B \cdot \mathbb{Z})$. All in all, we have shown that Algorithm 11 provides $[\![z]\!] \sim D_{\mathbb{Z},c,r}$ at the cost of about $\sqrt{2(t+1)}$ average rejections.

Masking security Let $\bar{z} = \sum_i \bar{z}_i \bmod 1$. As the Unmask gadget is only `NIo` secure with public output $\bar{z}$, we need to show that $\bar{z}$ does not leak sensitive information, i.e. the output z and the center c. Indeed, the output only occurs when $\bar{z} = 0$, hence $\bar{z}$ is independent of the output. The support of $\bar{z}$ is $\frac{1}{B}\{0, 1, \cdots, B-1\}$ and $\Pr[\bar{z} = \frac{i}{B}] \propto \rho_{c,r}(\mathbb{Z} + \frac{i}{B}) = \rho_{c-\frac{i}{B},r}(\mathbb{Z}) \in [\frac{1-\epsilon}{1+\epsilon}, 1]\rho_r(\mathbb{Z})$ due to the smoothness condition $r \geq \eta_\epsilon(\mathbb{Z})$. Therefore the distribution of $\bar{z}$ is negligibly close to uniform independent of c. Consequently, $\bar{z}$ can be securely unmasked. As all the operations are performed share by share and assuming uniformly distributed shares of the input center c, we can deduce the following lemma.

Lemma 3. *The gadget* GaussShareByShare *is t-`NIo` secure with public output $\bar{z}$.*

In the implementation, one needs to instantiate an unmasked Gaussian sampling with arbitrary center and fixed standard deviation (line 2 of Algorithm 11). We chose a table based approach and follow the technique of [32] to use a reduced number of tables.

7.2.2 Polynomial multiplication In some lattice-based schemes such as Kyber or Dilithium, polynomial multiplication is always performed between a sensitive and a public polynomial. This means that, using polynomials protected with arithmetic masking, one can multiply each share independently by the public unmasked polynomial and obtain an arithmetic sharing of the result of the multiplication. In this work, we have polynomials multiplications with both operand in arithmetic masked form. Given $[\![a]\!]$ and $[\![b]\!] \in \mathscr{R}^{t+1}$, we want to compute $[\![c]\!] \in \mathscr{R}^{t+1}$ such that $\sum_{i=0}^{t} c_i = \left(\sum_{i=0}^{t} a_i\right) \cdot \left(\sum_{i=0}^{t} b_i\right)$. To perform this masked polynomial multiplication, we propose to rely on an NTT-based multiplication. Using NTT, the product of two polynomials $a, b \in \mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}[x]/(x^d+1)$ is given by

$$\mathsf{NTT}^{-1}(\mathsf{NTT}(a) \circ \mathsf{NTT}(b))$$

with $\circ$ the *coefficient-wise* product between two vectors in $\mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}$. Since the NTT is linear, it can be applied on each share independently and we only have to mask the coefficient-wise multiplication between elements of $\mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}$ using the technique of [24]. While a naive multiplication algorithm would require d^2 ISW multiplications, we only need d of them. Since we want to multiply the polynomials in $\mathbb{Z}$ and not in $\mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}$, we need to work with a modulus large enough to avoid any reduction in the result. Recall that it is also possible to use several $\mathsf{Q}^{\mathrm{mask}}$ with CRT techniques to reduce the size.

Let us define $\mathsf{SecNTTMult}$, the masked product of two polynomials $[\![a]\!], [\![b]\!]$ arithmetically masked in $\mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}[x]/(x^d+1)$ by

$$\mathsf{NTT}^{-1}((\mathsf{SecMult}(\mathsf{NTT}([\![a]\!])_j, \mathsf{NTT}([\![b]\!])_j)_{0 \le j \le d-1}).$$

The algorithm for this product is detailed in the full version of our paper [16].

Lemma 4. $\mathsf{SecNTTMult}$ *is* t-$\mathtt{SNI}$ *secure.*

Note that here the shares are entire polynomials containing d coefficients. So, t probes actually provide $t \times d$ coefficients to the attacker.

Proof. Let $\delta \le t$ be the number of observations made by the attacker. Assume the following distribution of the attacker δ observations of intermediate shared polynomials: δ_1 observations on the first NTT computation on $\hat{a}$, δ_2 observations on the first NTT computation on $\hat{b}$, δ_3 observations on the $\mathsf{SecMult}$ part (which provides the knowledge of the $d \times \delta_3$ *coefficients* of the probed polynomials), δ_4 observations on the last NTT^{-1} computation, and δ_5 observations of the returned values. Finally, we have $\sum_{i=1}^{5} \delta_i \le \delta$. The algorithm NTT^{-1} is linear, thus it is t-$\mathtt{NI}$ and all the observations on steps 6 and 7 can be perfectly simulated with at most $\delta_4 + \delta_5$ shares of $\hat{c}$. The algorithm $\mathsf{SecMult}$ is applied coefficient-wise, thus each i-th execution has δ_3 observations of intermediate values (here coefficients) and $\delta_4 + \delta_5$ observations on the outputs (here coefficients too). By applying d times the t-$\mathtt{SNI}$ property for each $\mathsf{SecMult}$ operation, we can conclude that every observation from steps 3 to 7 can be perfectly simulated with at most δ_3 shared (polynomials) of $\hat{a}$ and $\hat{b}$. The linearity of the NTT with arithmetic masking allows to finish proving that every set of size at most t observations

Algorithm 12: Masked$\text{MITAKA}_{\mathbb{Z}}$Sampler

Input: A masked secret key in the following form: $([\![\widetilde{\mathbf{B}^*}]\!], [\![\widetilde{\mathbf{B}^*}^{-1}]\!], [\![\widetilde{v}]\!], [\![\mathbf{A}]\!])$ and a masked vector $[\![\mathbf{c}]\!]$ for a center $\mathbf{c} \in \mathscr{R}^2$, both arithmetically masked mod Q^{mask}.
Result: An unmasked sample $\mathbf{z} \sim D_{\mathscr{L}(\mathbf{B}),s,\mathbf{c}}$.

1 **Offline**
2 $[\![\mathbf{p}]\!] \leftarrow \mathsf{MaskedOfflineSampling}([\![\mathbf{A}]\!])$
3 **Online**
4 $[\![\mathbf{c}^{\text{pert}}]\!] \leftarrow [\![\mathbf{c}]\!] - [\![\mathbf{p}]\!]$
5 $[\![\mathbf{c}^{\text{pert}}]\!] \leftarrow \mathsf{SecNTTMult}([\![\widetilde{\mathbf{B}^*}^{-1}]\!], [\![\mathbf{c}^{\text{pert}}]\!])$
6 $[\![\mathbf{v}]\!] \leftarrow \mathsf{MaskedOnlineSampling}([\![\widetilde{v}]\!], [\![\mathbf{c}^{\text{pert}}]\!])$
7 $[\![\mathbf{z}]\!] \leftarrow \mathsf{SecNTTMult}([\![\widetilde{\mathbf{B}^*}]\!], [\![\mathbf{v}]\!])$
8 **return** $\sum_{i=0}^{t} \mathbf{z}_i \bmod \mathsf{Q}^{\text{mask}}$

containing $\sum_{i=1}^{4} \delta_i$ (resp. δ_5) intermediate (resp. returned) *polynomial shares* can be perfectly simulated with at most $\sum_{i=1}^{4} \delta_i$ polynomial shares of each input.

In the following, we extrapolate this polynomial multiplication technique to matrices of polynomials and keep the same notation SecNTTMult. We also remark that although SecNTTMult are sometimes called back-to-back in the masked samplers, this can be further optimized in practice: to minimize the number of $\mathsf{NTT}/\mathsf{NTT}^{-1}$ invocations in an implementation, one could keep the NTT representation as much as possible, and then bring it back to the coefficient domain whenever it encounters GaussShareByShare, as explicitly described in the full version of our paper [16].

7.3 Masking the $\text{MITAKA}_{\mathbb{Z}}$ sampler

The detailed overall structure of the sampler is presented in Algorithm 12; the algorithms for the online and offline samplings are detailed in the full version of this paper [16]. We remark that Algorithm 12 consists in a linear succession of gadgets with no dependency cycle, i.e. each line depends on freshly computed masked inputs. Thus, one can show that this algorithm is t-NI, as proved in Theorem 4 below. The proof is detailed in the full version of our paper [16].

Theorem 4. *The masked* $\text{MITAKA}_{\mathbb{Z}}$ *sampler (Alg. 12) is* t-NIo *with public output* $\mathbf{z}$.

7.4 Masking the MITAKA samplers

Our masked version of the RingPeikert sampler of Algorithm 1 and of the Hybrid sampler of Algorithm 3 are provided in the full version [16]. Although masked MITAKA is instantiated with the MaskedHybrid sampler, we also include MaskedRingPeikert

for completeness because the former can be essentially obtained by extending the basic masking paradigm outlined in the latter.

Contrary to MITAKA$_{\mathbb{Z}}$, one can remark that we here need to mask floating-point arithmetic. However, we can avoid it by representing each sensitive variable from $\mathscr{K}_{\mathbb{R}}$ as a fixed-point number. Concretely, an element $x \in \mathscr{K}_{\mathbb{R}}$ is approximated by $\tilde{x} \in \mathscr{K}_{\mathbb{R}}$ such that every coefficient of $q^k\tilde{x}$ is an integer, where k is a parameter determining the precision. Then we can secret-share $q^k\tilde{x}$ in $\mathbb{Z}^d_{\mathsf{Q}^{\mathrm{mask}}}$ for $\mathsf{Q}^{\mathrm{mask}} \gg q^k$. Since we do not perform many multiplication operations, an accumulated scaling factor does not break the correctness of sampler if we choose sufficiently large $\mathsf{Q}^{\mathrm{mask}}$.

We also remark that a secret-shared center in fixed-point representation must be divided by a scaling factor q^k for the following 1-dimensional discrete Gaussian sampling to work share-by-share . This division can be performed in floating-point arithmetic in practice. For the sum of shares to represent the correct center in the MaskedRingPeikert sampler, we further set $\mathsf{Q}^{\mathrm{mask}} = q^{k+\ell}$ for some $\ell > 0$. The resulting shares after division form a sharing of $\mathbf{v} = [(v_{1,j})_{j\in[0,d-1]}, (v_{2,j})_{j\in[0,d-1]}]$ over $(\mathbb{Q}/q^\ell\mathbb{Z})^{2d}$. Thanks to our GaussShareByShare introduced earlier, we are able to perform the discrete Gaussian sampling independently w.r.t each share of the center, while avoiding a factor of $\sqrt{t+1}$ overhead incurred in the standard deviation. As a result we obtain shares of discrete Gaussian samples $[\![z_{i,j}]\!]_{q^\ell}$ such that the distribution of $z_{i,j}$ is statistically close to $D_{\mathbb{Z},v_{i,j},r} \bmod q^\ell$ for every $i = 1, 2$ and $j \in [0, d-1]$. Since the output values of the signature are defined mod q we can further map the shares to $[\![\cdot]\!]_q$ and the remaining computations can be performed mod q.

Since we invoke the above routine twice in the MaskedHybrid sampler, the initial masking modulus needs to be increased so that no wrap-around occurs during the masked computation of the second nearest plane. Concretely, the first nearest plane operations are computed with modulus $\mathsf{Q}^{\mathrm{mask}} = q^{2k+\ell}$; the second nearest plane operations are performed on $[\![\cdot]\!]_{q^{k+\ell}}$, with corresponding arithmetic shares of sensitive inputs; the output values can be represented in $[\![\cdot]\!]_q$ as in the MaskedRingPeikert.

Although a naive implementation of the MITAKA sampler should rely on floating-point arithmetic and thus naturally carry out FFT-based polynomial multiplications, we instead make use of NTT in our masked algorithms. Notice that the masked instances only deal with a multiplication between polynomials mapped to $\mathbb{Z}_{\mathsf{Q}^{\mathrm{mask}}}[x]/(x^d+1)$ (or $\mathbb{Z}_{q^{\ell+k}}[x]/(x^d+1)$ during the second nearest plane of the Hybrid sampler) thanks to the fixed-point representation. This allows us to exploit SecNTTMult as in MaskedMITAKA$_{\mathbb{Z}}$Sampler. One caveat is that in the current setting $\mathsf{Q}^{\mathrm{mask}}$ is restricted to a *power* of q, but we are able to show such a choice is indeed NTT-friendly. Recall that the prime q is usually chosen such that $q = 1 \bmod 2d$, so that x^d+1 has exactly d roots $(\zeta, \zeta^3, \ldots, \zeta^{2d-1})$ over $\mathbb{Z}_q$. Now thanks to the Hensel lifting, one can construct another set of d roots $(\omega, \omega^3, \ldots, \omega^{2d-1})$ over $\mathbb{Z}_{q^2}$, such that $\omega = \zeta \bmod q$. By iterating this procedure until the roots for a sufficiently large modulus $\mathsf{Q}^{\mathrm{mask}}$ are obtained, we can indeed

utilize the NTT for evaluating $f(x) \in \mathbb{Z}_{\mathbf{Q}^{\text{mask}}}[x]/(x^d + 1)$ on the primitive $2d$-th roots of unity.

We are able to prove that both masked samplers meet the standard security notion (t-NIo) for masked signature schemes. The proof is detailed in the full version [16].

Theorem 5. *The masked* MITAKA *sampler is t-NIo secure with public output* $\mathbf{v}_0$.

8 Conclusion

The FALCON signature scheme, one of the NIST round 3 finalists, is a very attractive postquantum scheme for real-world deployment: it has fast signing and verification, the best bandwidth requirements (as measured in combined verification key and signature size) of all round 2 signatures, as well as a solid, well-understood security. However, it suffers from a number of short-comings: it has a complex structure that makes it hard to implement correctly; it is not flexible in terms of parameter selection (only supporting NIST security levels I and V, and no intermediate level); it is inefficient on architectures without fast native floating point arithmetic; and it is difficult to protect against side-channels.

In this paper, we introduced MITAKA, a simpler variant based on similar design principles, which manages to maintain the advantages of FALCON while largely mitigating those shortcomings: as we have seen, it has performance on par or superior to FALCON, its bandwidth requirements are similar (not quite as good as FALCON, but still far better than all other NIST candidates), and its security relies on the same assumptions. However, unlike FALCON, it is also relatively simple to implement; it supports a wide range of parameter settings covering all NIST security levels; it can be implemented using the MITAKA$_{\mathbb{Z}}$ sampler in a way that avoids floating-point arithmetic; and it can be efficiently protected against side-channel attacks, for example with our proposed masking countermeasure. It also has nice additional properties such as its online/offline structure.

Thus, we showed that NTRU-based hash-and-sign signatures have even more potential than previously expected to replace, e.g., elliptic curve-based schemes in a postquantum world. Some questions remain for future work, such as:

- to what extent can signature size be reduced? (while smaller than other lattice schemes, it is substantially larger than classical elliptic curve-based signatures, or postquantum candidates based on multivariate cryptography and isogenies);
- can security for the hybrid sampler be improved further with better trapdoor generation? (there is still a substantial gap between the hybrid sampler and Klein–GPV);
- how fast can MITAKA perform with a fully optimized, constant-time implementation on, e.g., Intel CPUs with AVX2? What about embedded microcontrollers with and without masking?

Acknowledgements. We would like to thank Léo Ducas, Thomas Prest and Damien Stehlé for valuable comments and discussions. The second and seventh authors were supported by the European Union H2020 Research and Innovation Program Grant 780701 (PROMETHEUS). The third author was supported by the ERC Advanced Grant No. 787390. The fifth author has been supported by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No. 803096 (SPEC). The eighth author has been supported by the National Natural Science Foundation of China (No. 62102216), the National Key Research and Development Program of China (Grant No. 2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008) and Major Scientific and Techological Innovation Project of Shandong Province, China (Grant No. 2019JZZY010133).

References

1. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 327–343. USENIX Association (Aug 2016)
2. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 116–129. ACM Press (Oct 2016).
3. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Grégoire, B., Rossi, M., Tibouchi, M.: Masking the GLP lattice-based signature scheme at any order. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 354–384. Springer, Heidelberg (Apr / May 2018).
4. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2147–2164. ACM Press (Nov 2019).
5. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Krauthgamer, R. (ed.) 27th SODA. pp. 10–24. ACM-SIAM (Jan 2016).
6. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (Dec 2011).
7. Chuengsatiansup, C., Prest, T., Stehlé, D., Wallet, A., Xagawa, K.: ModFalcon: Compact signatures based on module-NTRU lattices. In: Sun, H.M., Shieh, S.P., Gu, G., Ateniese, G. (eds.) ASIACCS 20. pp. 853–866. ACM Press (Oct 2020).
8. Coron, J.S.: Higher order masking of look-up tables. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 441–458. Springer, Heidelberg (May 2014).
9. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D., Yang, B.Y., Kannwischer, M., Patarin, J.: Rainbow. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

10. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (Aug 2013).
11. Ducas, L., Galbraith, S., Prest, T., Yu, Y.: Integral matrix gram root and lattice gaussian sampling without floats. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 608–637. Springer, Heidelberg (May 2020).
12. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). , https://tches.iacr.org/index.php/TCHES/article/view/839
13. Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 22–41. Springer, Heidelberg (Dec 2014).
14. Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 433–450. Springer, Heidelberg (Dec 2012).
15. Ducas, L., Prest, T.: Fast Fourier orthogonalization. Cryptology ePrint Archive, Report 2015/1014 (2015), https://eprint.iacr.org/2015/1014
16. Espitau, T., Fouque, P.A., Gérard, F., Rossi, M., Takahashi, A., Tibouchi, M., Wallet, A., Yu, Y.: Mitaka: a simpler, parallelizable, maskable variant of falcon. Cryptology ePrint Archive, Report 2021/1486 (2021), https://ia.cr/2021/1486
17. Fouque, P.A., Kirchner, P., Tibouchi, M., Wallet, A., Yu, Y.: Key recovery from Gram-Schmidt norm leakage in hash-and-sign signatures over NTRU lattices. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 34–63. Springer, Heidelberg (May 2020).
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008).
19. Gérard, F., Rossi, M.: An efficient and provable masked implementation of qtesla. In: Belaïd, S., Güneysu, T. (eds.) CARDIS 2019. Lecture Notes in Computer Science, vol. 11833, pp. 74–91. Springer (2019)
20. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (Aug 1997).
21. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: Performance improvements and a baseline parameter generation algorithm for NTRUSign. Cryptology ePrint Archive, Report 2005/274 (2005), https://eprint.iacr.org/2005/274
22. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital signatures using the NTRU lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (Apr 2003).
23. Howe, J., Prest, T., Ricosset, T., Rossi, M.: Isochronous gaussian sampling: From inception to implementation. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 53–71. Springer, Heidelberg (2020).
24. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003).
25. Karabulut, E., Aysu, A.: Falcon down: Breaking Falcon post-quantum signature scheme through side-channel attacks (2021)

26. Laarhoven, T.: Search problems in cryptography. Ph.D. thesis, Eindhoven University of Technology (2015)
27. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr. **75**(3), 565–599 (2015)
28. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2019), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
29. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
30. Lyubashevsky, V., Wichs, D.: Simple lattice trapdoor sampling from a broad class of distributions. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 716–730. Springer, Heidelberg (Mar / Apr 2015).
31. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (Aug 2013).
32. Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 455–485. Springer, Heidelberg (Aug 2017).
33. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. Journal of Cryptology **22**(2), 139–160 (Apr 2009).
34. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (Aug 2010).
35. Pornin, T.: New efficient, constant-time implementations of Falcon. Cryptology ePrint Archive, Report 2019/893 (2019), https://eprint.iacr.org/2019/893
36. Pornin, T., Prest, T.: More efficient algorithms for the NTRU key generation using the field norm. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 504–533. Springer, Heidelberg (Apr 2019).
37. Prest, T.: Gaussian Sampling in Lattice-Based Cryptography. Ph.D. thesis, École Normale Supérieure, Paris, France (2015)
38. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
39. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (Aug 2010).
40. Yu, Y., Ducas, L.: Learning strikes again: The case of the DRS signature scheme. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 525–543. Springer, Heidelberg (Dec 2018).
41. Zhao, R.K., Steinfeld, R., Sakzad, A.: FACCT: Fast, compact, and constant-time discrete gaussian sampler over integers. IEEE Transactions on Computers **69**(1), 126–137 (2020)