ℤalcon: an alternative FPA-free NTRU sampler for Falcon

Pierre-Alain Fouque, François Gérard, Mélissa Rossi and Yang Yu

Submission to 3rd NIST PQC workshop

Abstract. Falcon is a very efficient and compact lattice-based signature scheme following the hash-and-sign GPV paradigm. The scheme is in the third round of the NIST Post-Quantum competition. It relies on the fast FFO sampler proposed by Ducas and Prest for sampling a Gaussian distribution over a lattice, that require floating-point operations. Floating-point operations are complex to protect against side-channel attack. We propose to tweak Falcon into ℤalcon, an FPA-free alternative. We slightly modify the key generation and replace the FFO sampler with a new sampler based on Ducas *et al.* paper (Eurocrypt 2020). We specify the latter and show that it can be implemented without floating-point arithmetic operations. We additionally separate the sampling into an off-line phase that can be done in preprocessing and a fast and simple on-line sampling. This alternative is useful in constraint environments like smart cards where the on-line phase should be both fast and protected against side-channels. In this work-in-progress report, we also provide a provable masking and an implementation of the on-line sampler. We believe that it is possible to secure the off-line sampler as well.

1 Introduction

Falcon is one of the most efficient and compact lattice-based signature scheme proposed to the NIST Post-Quantum Competition. It follows the Hash-and-Sign paradigm proposed by Gentry, Peikert and Vaikuntanathan in [GPV08] with an efficient Gaussian sampler proposed by Ducas and Prest [DP16]. Its security is based on the NTRU assumption, which provides a short trapdoor. The main drawback of this scheme is that currently it relies on floating-point operations to compute the Gaussian sampler. This also makes it difficult to protect against side-channel attacks since it is an open-problem to define secure masking scheme for such representation. Consequently, designing a masked implementation of Falcon is an open-problem. Falcon is a very interesting scheme and many researches around it have been made [CPS+20, FKT+20, HPRR20]. To progress in this research vein, we propose to define a variant of Falcon, ℤalcon, using only integers in its sampling. Hence, we provide a portable alternative, easier to mask. The main idea is to modify the Gaussian sampler with a one that uses integer operations, that are more suitable to mask.

Gaussian Sampling over a lattice. To prevent statistical attack on NTRU-based signature à la Nguyen-Regev[NR06], Gentry, Peikert, and Vaikuntanathan [GPV08] propose to use a randomized closest vector approximation algorithm. Hashing the message gives a vector $\mathbf{v}$ in some space and instead of computing the closest lattice vector to this point as the signature $\mathbf{s}$, we sample a spherical discrete Gaussian distribution centered at $\mathbf{v}$ on the lattice points. Outputting the closest vector leaks information about the secret short basis since the distribution of $\mathbf{v} - \mathbf{s}$ is a publicly known Gaussian. Therefore the signature process consists in sampling a discrete Gaussian over a lattice for a variable center and where the widths of the Gaussians in each direction depend on the length of the secret basis.

Sampling over a lattice boils down to sampling over a discrete Gaussian over the integers, and using CVP algorithms. Klein and Peikert samplers are respectively a generalization of Babai's nearest plane and Babai's round-off algorithms that instead of outputting one lattice point, return a point according to a discrete Gaussian over a lattice. Klein sampler [GPV08] is inherently sequential and needs the Gram-Schmidt orthogonalization. Consequently, it requires to work with either integer operations for rationals with very large denominators, or floating-point approximations. In [Pei10], Peikert proposed a parallel alternative using Babai round-off algorithm where most of the expensive computational part, including floating-point arithmetic, can be done in an offline phase, independent of the message to sign at the cost of increasing the

width of the sampled Gaussian. The offline step samples a non-spherical Gaussian and requires a Cholesky decomposition of the covariance matrix that is of a high precision. The online sampling can be reduced to integer Gaussian sampling that is relatively easy and can be done with only integer operations. Recently, Ducas *et al.* [DGPY20] shows that by replacing the high-precision Cholesky decomposition with an integral Gram root, one can also remove the need of floating-point operation during the offline step.

Gaussian Sampling over NTRU lattice. Prest [Pre15] describes the hybrid sampler, which is an intermediate sampler between Klein and Peikert trying to reach the best of both samplers. Klein and nearest plane give short vectors but are sequential (with complexity $O(n^2)$), while Peikert and Babai round-off output larger vector but are parallel (complexity $O(n \log n)$ for NTRU). Moreover, Klein sampler is not adapted to structured lattice since the Gram-Schmidt orthogonalization process breaks the underlying algebraic structure. In a security point of view, short vectors are better, since this will make the attacker task harder as to forge a signature, he will have to find a closer vector to $\mathbf{v}$. Klein sampler outputs vectors of size proportional to the length of the Gram-Schmidt basis vector, while Peikert sampler are proportional to the first singular value of the basis. The latter value is usually larger than the former by a factor at least $\sqrt{\log n}$ [Pre15]. Prest sampler uses Klein sampler as the main algorithm and uses Peikert sampler as a subroutine. In the case of NTRU lattices, module of rank-2 over a polynomial ring of dimension n, it is particularly efficient since Klein/Hybrid sampler is reduced to 2 steps of Peikert sampler. The quality is better than Peikert sampler with a running time close to Peikert.

The Fast Fourier Orthogonalization sampler proposed in [DP16] fully exploit the algebraic structure and output very short vector as close to Klein sampler. However, it requires to compactly store the Cholesky decomposition, LDL, computed over the ring. FFO works over a tower of subrings in the case of cyclotomic rings and usually computations are performed with floating-point operations since when we descend in the tower of subfields, the denominators of intermediate values will grow rapidly. It is hard in this procedure to control the required precision.

Gaussian Sampling over $\mathbb{Z}$. There are several algorithms to sample from a discrete Gaussian over the integers. The algorithms can be split into several categories according to their ability to sample with a variable center and a variable and large width. For Falcon, we need to the last two properties. There are currently two samplers for these two tasks: MW was proposed by Micciancio and Walter [MW17] and the other one by Karney [Kar16]. Karney sampler can be implemented over the integers. For our range of parameters, we prefer to use MW sampler since Karney sampler relies on the rejection sampling method. Indeed, on the one hand, one needs to study whether this non-constant time rejection sampling does not leak information and on the other hand, the rejection sampling requires the evaluation of a polynomial on a large input with very high precision, even with Rényi divergence arguments. This leads to storing unrealistically large integers. MW can be implemented efficiently using integer operations.

Security of the Implementation of Falcon. Falcon is the successor of the DLP scheme designed by Ducas, Lyubashevsky, and Prest [DLP14a] using a NTRU trapdoor with a Klein sampler, where the sampler has been changed to a more efficient one given in [DP16] and called the FFO sampler. For DLP, an efficient attack on the non-secure implementation of DLP has been described in [FKT+20] exploiting some timing information, allowing to recover the lengths of the Gram-Schmidt vectors. In particular, this attack shows that for NTRU, it is possible to recover the secret basis given only the size of the Gram-Schmidt vectors. For Falcon, this attack is more costly since the FFO tree makes the information harder to exploit, however the recent isochronous implementation [HPRR20] avoid this leakage.

The standard implementation of Falcon to the NIST follows the constant-time paradigm, meaning that in the execution tree, there is no branching on secret values and all memory accesses are independent of the secret. Except, a fault attack described in [MHS+19] by McCarthy *et al.*, there is no attack or side-channel attacks reported. However, we aim to propose a side-channel implementation. We do not target cache effect since on Cortex M4, usually only one process is executed at one time.

Our Contributions. We propose a variant of the Falcon signature scheme where the Gaussian sampling is changed so that computations can be performed over the integers since our final goal is to propose a

portable and masked implementation for such scheme. Only the key generation part requires floating point arithmetic. It is possible to remove the FP in this part and we did it for small modulus. However, more study remains be done to avoid problems in large range of parameters. Since this operation is less sensitive and performed only once, we decided to not go into this step in this work.

Our scheme is based on Ducas *et al.*'s algorithm [DGPY20] which requires two base samplers: $D_{\mathbb{Z},Lr}$ and $D_{\mathbb{Z},r,\mathbf{c}}$ with $\mathbf{c} \in \frac{1}{L} \cdot \mathbb{Z}$, where L is any integer larger than some polynomial bound and can be chosen as a power of 2. The off-line and on-line steps can be implemented over the integers. To improve the quality of our sampler and achieve better security level, we propose to tweak the key generation to look for specific NTRU parameters with small first singular values without increasing the number of Gaussian sampling we need.

In our implementation, we rely on the MW base sampler using fixed-point operations. MW is a generic sampler that relies on smaller base samplers over the integers. We use CDT tables to implement the samplers with small widths and for the different values of small number of bits of the centers.

For the masking of the base samplers, we have to deal with the masking of the table look-up search. Moreover, contrary to other lattice-based schemes, we need to mask polynomial products over $\mathbb{Z}[X]$ with two secret polynomial. We use NTT for a large modulus and multiply only n values with ISW. Since the NTT and inverse NTT are linear operations, these steps can be easily masked. The modulus is sufficiently large so that no wrap around occurs.

Related Work. Masking lattice-based cryptography is an interesting problem addressed in many work. For signature schemes, Fiat-Shamir with Abord signature are easier to mask. For instance, the GLP signature scheme [GLP12] has been protected by Barthe *et al.* in [BBE+18] and a concrete implementation for Dilithium has been given in [MGTF19] by Migliore *et al.*. In the same line of research, Gérard and Rossi concretely implement a masked version of qTelsa Signature scheme [BAA+19] in [GR19]. For BLISS signature scheme [DDLL13], after many side-channel attacks [EFGT17, BDE+18], a formal masking scheme has finally been proposed in [BBE+19]. Recently, some attacks on the NIST KEM candidates have been presented [RRCB20, RBRC20b, RBRC20a] showing that these schemes are fragile against side-channel attacks.

Organization of the paper. In section 2, we recall basic results regarding lattices, gaussians, and NTRU properties. Then, we describe our Signature scheme called ℤalcon and analyze its security. Section 4 presents our integer Gaussian samplers with details, and finally we give some experiments and masking properties in sections 5 and 6.

2 Preliminaries

2.1 Notations

We denote vectors (resp. matrices) with bold lower case letters (resp. bold upper case letters). Vectors are in column form. Let $\|\mathbf{v}\|$ be the Euclidean norm of the vector $\mathbf{v}$. We use log to denote the logarithm of base 2. Let $\epsilon > 0$ be a small number; we write $\hat{\epsilon} = \epsilon + O(\epsilon^2)$. For $q > 0$, let $\lfloor a \rceil_q = \lfloor aq \rceil / q \in (1/q) \cdot \mathbb{Z}$ for $a \in \mathbb{R}$.

2.2 Linear algebra

For $\mathbf{A} \in \mathbb{R}^{n \times m}$, let $\mathbf{A}_{(i,j)}$ be the element in the i-th row and j-th column of $\mathbf{A}$, and $\mathbf{A}^t$ the transpose of $\mathbf{A}$. Let $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \sqrt{e_1(\mathbf{A}^t\mathbf{A})}$ and $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} \mathbf{A}^2_{(i,j)}}$. It is known that $\|\mathbf{A}^t\|_2 = \|\mathbf{A}\|_2$, $\|\mathbf{AB}\|_2 \leq \|\mathbf{A}\|_2 \|\mathbf{B}\|_2$ and $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$. Let $\|\mathbf{A}\|_{\max} = \max |\mathbf{A}_{(i,j)}|$.

Let $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{n-1}) \in \mathbb{F}^{m \times n}$ of rank n over some field $\mathbb{F}$. It can be decomposed as $\mathbf{B} = \mathbf{B}^*\mathbf{U}$ where $\mathbf{B}^* = (\mathbf{b}^*_0, \cdots, \mathbf{b}^*_{n-1})$ consists of pairwise orthogonal column vectors and $\mathbf{U}$ is an upper triangle matrix with 1 on the diagonal. We call $\mathbf{B}^*$ and $\mathbf{U}$ the *Gram-Schmidt orthogonal* and the *Gram-Schmidt coefficient matrix* of $\mathbf{B}$. Let $\|\mathbf{B}\|_{GS} = \max_i \{\|\mathbf{b}^*_i\|\}$.

Let $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ be a symmetric matrix. We write $\mathbf{\Sigma} \succ 0$ (resp. $\mathbf{\Sigma} \succeq 0$) when $\mathbf{\Sigma}$ is *positive definite* (*positive semi-definite*), i.e. $\mathbf{x}^t\mathbf{\Sigma}\mathbf{x} > 0$ ($\mathbf{x}^t\mathbf{\Sigma}\mathbf{x} \geq 0$) for all non-zero $\mathbf{x} \in \mathbb{R}^n$. We also write $\mathbf{\Sigma}_1 \succ \mathbf{\Sigma}_2$ (resp. $\mathbf{\Sigma}_1 \succeq \mathbf{\Sigma}_2$) when $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 \succ 0$ (resp. $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 \succeq 0$). It holds that $\mathbf{\Sigma} \succ 0$ if and only if $\mathbf{\Sigma}^{-1} \succ 0$ and that $\mathbf{\Sigma}_1 \succ \mathbf{\Sigma}_2 \succ 0$ if and only if $\mathbf{\Sigma}_2^{-1} \succ \mathbf{\Sigma}_1^{-1} \succ 0$. For a positive semi-definite $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$, let $e_1(\mathbf{\Sigma}) \geq e_2(\mathbf{\Sigma}) \geq \cdots \geq e_n(\mathbf{\Sigma}) \geq 0$ be its eigenvalues. If $\mathbf{\Sigma} \succ 0$, then $e_i(\mathbf{\Sigma}) \cdot e_{n-i}(\mathbf{\Sigma}^{-1}) = 1$.

2.3 Lattices

A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. If $\mathcal{L}$ is generated by $\mathbf{B} \in \mathbb{R}^{m\times n}$, we write $\mathcal{L} := \mathcal{L}(\mathbf{B}) = \{\mathbf{Bv} \mid \mathbf{v} \in \mathbb{Z}^n\}$. If $\mathbf{B}$ has a full column rank, we call $\mathbf{B}$ a basis, n the rank and m the dimension of $\mathcal{L}$. When the rank of $\mathcal{L}$ equals its dimension, $\mathcal{L}$ is full-rank.

Given a lattice $\mathcal{L}$, its dual lattice is $\widehat{\mathcal{L}} = \{\mathbf{u} \in \mathrm{span}(\mathcal{L}) \mid \forall \mathbf{v} \in \mathcal{L}, \langle \mathbf{u}, \mathbf{v}\rangle \in \mathbb{Z}\}$. Let n be the rank of $\mathcal{L}$. For $k \leq n$, the k-th minimum $\lambda_k(\mathcal{L})$ is the smallest value $r \in \mathbb{R}$ such that there are at least k linearly independent vectors in $\mathcal{L}$ with lengths $\leq r$.

Given $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with $m \geq n$, let $\Lambda^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0}\}$ be the *orthogonal lattice* defined by $\mathbf{A}$. When the rank of $\mathbf{A}$ is n, the rank of $\Lambda^\perp(\mathbf{A})$ is $(m-n)$.

2.4 Gaussians

Let $\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi(\mathbf{x}-\mathbf{c})^t\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c})\right)$ be the n-dimensional Gaussian weight with center $\mathbf{c} \in \mathbb{R}^n$ and (scaled)[1] covariance matrix $\mathbf{\Sigma}$. We also denote $r\sqrt{\mathbf{\Sigma}} = \sqrt{r^2 \cdot \mathbf{\Sigma}}$ for $r > 0$. When $\mathbf{c} = \mathbf{0}$, the Gaussian function is written as $\rho_{\sqrt{\mathbf{\Sigma}}}$ and is called *centered*. When $\mathbf{\Sigma} = s^2\mathbf{I}_n$, we write the subscript $\sqrt{\mathbf{\Sigma}}$ as s directly, and call the Gaussian *spherical* of *width* s.

The *discrete Gaussian distribution* over a lattice $\mathcal{L}$ with center $\mathbf{c}$ and covariance matrix $\mathbf{\Sigma}$ is defined by the probability function $D_{\mathcal{L},\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathcal{L})}$ for any $\mathbf{x} \in \mathcal{L}$. We recall some notions related to the *smoothing parameter*.

Definition 1 ([MR07], Definition 3.1). *Given a lattice $\mathcal{L}$ and $\epsilon > 0$, the ϵ-smoothing parameter of $\mathcal{L}$ is* $\eta_\epsilon(\mathcal{L}) = \min\left\{s \mid \rho_{1/s}\left(\widehat{\mathcal{L}}\right) \leq 1+\epsilon\right\}$.

Definition 2 ([Pei10], Definition 2.3). *Given a full-rank lattice $\mathcal{L}$, $\epsilon > 0$ and $\mathbf{\Sigma} \succ 0$, we write $\sqrt{\mathbf{\Sigma}} \geq \eta_\epsilon(\mathcal{L})$ if $\eta_\epsilon\left(\sqrt{\mathbf{\Sigma}}^{-1} \cdot \mathcal{L}\right) \leq 1$ i.e. $\rho_{\sqrt{\mathbf{\Sigma}^{-1}}}\left(\widehat{\mathcal{L}}\right) \leq 1+\epsilon$.*

We list two basic facts: for a full-rank lattice $\mathcal{L}$ and $\epsilon \in (0,1)$,

1. $\eta_\epsilon(r\mathcal{L}) = r \cdot \eta_\epsilon(\mathcal{L})$ for arbitrary $r > 0$;
2. if $\mathbf{\Sigma}_1 \succeq \mathbf{\Sigma}_2$ and $\sqrt{\mathbf{\Sigma}_2} \geq \eta_\epsilon(\mathcal{L})$, then $\sqrt{\mathbf{\Sigma}_1} \geq \eta_\epsilon(\mathcal{L})$.

We define $\overline{\eta_\epsilon}(\mathbb{Z}^n) = \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}}$.

Lemma 3 ([MR07], Lemma 3.3). *Let $\mathcal{L}$ be an n-dimensional lattice and $\epsilon \in (0,1)$, $\eta_\epsilon(\mathcal{L}) \leq \overline{\eta_\epsilon}(\mathbb{Z}^n) \cdot \lambda_n(\mathcal{L})$.*

We recall the convolution theorem for discrete Gaussians introduced in [Pei10].

Theorem 4 (Adapted from Theorem 3.1 [Pei10]). *Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 \in \mathbb{R}^{n\times n}$ be positive definite matrices. Let $\mathbf{\Sigma} = \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2$ and let $\mathbf{\Sigma}_3 \in \mathbb{R}^{n\times n}$ be such that $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$. Let $\mathcal{L}_1, \mathcal{L}_2$ be two full-rank lattices in $\mathbb{R}^n$ such that $\sqrt{\mathbf{\Sigma}_1} \geq \eta_\epsilon(\mathcal{L}_1)$ and $\sqrt{\mathbf{\Sigma}_3} \geq \eta_\epsilon(\mathcal{L}_2)$ for $\epsilon \in (0, 1/2)$. Let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$. Then the distribution of $\mathbf{x}_1 \hookleftarrow D_{\mathcal{L}_1,\sqrt{\mathbf{\Sigma}_1},\mathbf{x}_2-\mathbf{c}_2+\mathbf{c}_1}$ where $\mathbf{x}_2 \hookleftarrow D_{\mathcal{L}_2,\sqrt{\mathbf{\Sigma}_2},\mathbf{c}_2}$ is within max-log distance $4\hat{\epsilon}$ of $D_{\mathcal{L}_1,\sqrt{\mathbf{\Sigma}},\mathbf{c}_1}$.*

[1] The scaling factor is 2π and we omit it in this paper for convenience.

2.5 Power-of-2 Cyclotomics

Let $\mathcal{R}_n = \mathbb{Z}[x]/(x^n+1)$ with n a power-of-2. Let $\mathcal{K}_n = \mathbb{Q}[x]/(x^n+1)$. When the context is clear, we write $\mathcal{R} = \mathcal{R}_n$ and $\mathcal{K} = \mathcal{K}_n$ for omission. Let $\overline{f}$ denote the conjugate of $f \in \mathcal{K}$, i.e. $\overline{f} = f(x^{-1})$. Let $\|f\| = \sqrt{\sum_{i=0}^{n-1} f_i^2}$ and $\|f\|_\infty = \max_{i=0}^{n-1} |f_i|$ be the ℓ_2 and ℓ_∞ norm of $f \in \mathcal{K}_n$ respectively. For $\mathbf{a} = (a_0, \cdots, a_{m-1}), \mathbf{b} = (b_0, \cdots, b_{m-1}) \in \mathcal{K}^m$, let $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathcal{K}} = \sum_{i=0}^{m-1} a_i \overline{b_i} \in \mathcal{K}$ and $\|\mathbf{a}\|_{\mathcal{K}} = \langle \mathbf{a}, \mathbf{a} \rangle_{\mathcal{K}}$. We extend $\lfloor a \rceil_q$ coefficient-wise to $\mathcal{K}$. Let $\mathbf{U}_{a,n} = \begin{pmatrix} 1 & a \\ & 1 \end{pmatrix} \in \mathcal{K}_n^{2\times 2}$, then $\mathbf{U}_{a,n}^{-1} = \mathbf{U}_{-a,n}$.

Each $f \in \mathcal{K}_n$ can be represented by $f_e, f_o \in \mathcal{K}_{n/2}$ as $f(x) = f_e(x^2) + x f_o(x^2)$. The multiplication matrix of f can be represented as a 2×2 block matrix as follows:

$$\mathcal{F}_n(f) = \begin{pmatrix} \mathcal{F}_{n/2}(f_e) & \mathcal{F}_{n/2}(x f_o) \\ \mathcal{F}_{n/2}(f_o) & \mathcal{F}_{n/2}(f_e) \end{pmatrix}.$$

In this paper, we use $\mathcal{F}_n(f)$ as the matrix representation of f. If $\mathcal{F}_n(f) \succ 0$, we call f is positive definite and write $e_i(f) = e_i(\mathcal{F}_n(f))$ for short.

For $f \in \mathcal{R}_n$, let $\sigma_i(f) = f(x^{2i+1}) \in \mathcal{R}_n$ be a conjugate of f. Let ξ_{2n} be a $2n$-th primitive root of 1. We call $f(\xi_{2n}^{2i+1}) \in \mathbb{C}$ an embedding of f.

2.6 NTRU

Given $f, g \in \mathcal{R}$ such that f is invertible modulo some $q \in \mathbb{Z}$, we let $h = f^{-1} g \bmod q$. The NTRU lattice determined by h is

$$\mathcal{L}_{NTRU} = \{(u, v) \in \mathcal{R}^2 \mid u - vh = 0 \bmod q\}.$$

In NTRU cryptosystems, the secret key is commonly (f, g) where $f, g \in \mathcal{R}^2$ are short, while h is the public key. Some NTRU schemes use the trapdoor basis that is some $\mathbf{B} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ and $gF - fG = q$. There are many (F, G)'s resulting in a trapdoor basis for the same (f, g). Yet these bases have the same orthogonalization, i.e.

$$\mathbf{B}^* = \begin{pmatrix} g & G^* = G - vg \\ f & F^* = F - vf \end{pmatrix} = \begin{pmatrix} g & -\frac{q\overline{f}}{f\overline{f}+g\overline{g}} \\ f & \frac{q\overline{g}}{f\overline{f}+g\overline{g}} \end{pmatrix} \in \mathcal{K}^{2\times 2}$$

where $v = \frac{F\overline{f}+G\overline{g}}{f\overline{f}+g\overline{g}}$ and $\langle \mathbf{b}_0^* = (g, f), \mathbf{b}_1^* = (G^*, F^*) \rangle_{\mathcal{K}} = 0$. As shown in [DLP14b], $\|\mathbf{B}\|_{GS} = \max\{\|(f, g)\|, \|(G^*, F^*)\|\}$. This paper is interested in optimal NTRU bases, that is $\|(g, f)\| \approx \|(G^*, F^*)\|$.

3 ℤalcon Signature Scheme

In this section, we present a variant of Falcon signature, called ℤalcon. ℤalcon is also an instantiation of the GPV hash-and-sign scheme over NTRU lattices, but its Gaussian sampling procedure does not require floating-point arithmetic, which provides satisfactory determinism and feasibility of an masked implementation.

The lattice Gaussian sampler of ℤalcon uses the framework of Peikert's approach [Pei10], consisting of on-line and off-line phases. The original Peikert's sampler requires larger parameters compared with the GPV sampler [GPV08]. To mitigate the overhead, we propose to use $s^2\mathbf{I} - r^2\mathbf{B}^*\mathbf{B}^{*t}$ instead of $s^2\mathbf{I} - r^2\mathbf{B}\mathbf{B}^t$ as the perturbation covariance, where $\mathbf{B}^*$ is the Gram-Schmidt orthogonalization (with respect to $\mathcal{K}_n$) of $\mathbf{B}$. This modification effectively improves the parameter, as $\|\mathbf{B}^*\|_2$ is asymptotically smaller than $\|\mathbf{B}\|_2$. To avoid high-precision arithmetic, we further replace $\mathbf{B}^*$ with its approximation and combine the recent techniques in [DGPY20] for the perturbation sampling.

Prior to the formal algorithmic description, let us first summarize all system parameters in Table 1.

Table 1. Description and bounds of all the system parameters.

	Description	Requirement
n	degree of the ring	2^ℓ
$\mathcal{R}_n$	$\mathbb{Z}[x]/(x^n+1)$	
q	modulus	
ϵ	closeness parameter	very small
$\hat{\epsilon}$	closeness parameter	$\hat{\epsilon} = \epsilon + O(\epsilon^2)$
r	base Gaussian width	$r \geq \overline{\eta_\epsilon}(\mathbb{Z}^{2n})$
$A_{n,q}$	upper bound of $\|\mathbf{B}^*\|_2$	pre-computed experimentally
s'	scaled Gaussian width	$s' \in \mathbb{N}$, $s' \geq A_{n,q}+1$
s	Gaussian width	$s = s'r$
p	approximation precision	$p \in \mathbb{N}$
B	upper bound of $\left\|p\widetilde{\mathbf{B}^*}\right\|_2$	$B \in \mathbb{N}$, $B = p \cdot A_{n,q} = \omega(n^2)$
b	base for gadget decomposition	$b \in \mathbb{N}$, $b^3 \geq (n+1)\left(B + \frac{n}{8}\right) + 6nb^2$, $p^2 s'^2 - p^2 - B^2 - 1 \geq \frac{b^6-1}{b^2-1}\ell + b^3$
m	integers s.t. the Gram root $\mathbf{A} \in \mathcal{R}_n^{2\times m}$	$m = 6\ell + 9$
L	upper bound of $\lambda_{mn-n}(\Lambda^\perp(\mathbf{A}))$	$L \in N$, $L \geq 2b^2\sqrt{n}$

3.1 Key Generation

The key generation of ℤalcon is similar to that of Falcon, consisting of NTRU trapdoor generation and some pre-computation for signing. The secret key is an NTRU trapdoor basis $\mathbf{B} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ such that $\|\mathbf{B}^*\|_2$ is bounded by a pre-computed threshold $A_{n,q}$. This is different from that in Falcon case, mainly because we change the sampler for signing. The choice of $A_{n,q}$ is discussed in Section 3.2 according to experimental measures. In addition, ℤalcon and Falcon execute totally different pre-computation in their key generation. For Falcon, the pre-computation is constructing the so-called Falcon tree. The Falcon tree contains all intermediate values of Gram-Schmidt orthogonalization of $\mathbf{B}$, thus it is of high-precision and large size. In contrast, the pre-computation by ℤalcon is computing an integral Gram root of the perturbation covariance. The resulting integral Gram root is over integers and can be stored efficiently. The formal description of the key generation is provided in Algorithm 1.

In order to obtain better compactness and higher security, we search among all $(f, \sigma_k(g))$'s for the smallest $\max\left\{e_1\left(d_0^{(k)}\right), e_1\left(d_1^{(k)}\right)\right\}$ that determines the signature size. For any $d \in \mathcal{K}_n$, the set of all eigenvalues of $\mathcal{F}_n(d)$ is $\{d(\xi_{2n}^{2i+1}) \mid i \in \mathbb{Z}_n\}$. Immediately, it follows that

$$\max\left\{e_1\left(d_0^{(k)}\right), e_1\left(d_1^{(k)}\right)\right\} = \max\left\{\max_{i\in\mathbb{Z}_n}\left\{d_0^{(k)}(\xi_{2n}^{2i+1})\right\}, \frac{q^2}{\min_{i\in\mathbb{Z}_n}\left\{d_0^{(k)}(\xi_{2n}^{2i+1})\right\}}\right\}.$$

Since all $\sigma_k(g)$'s have the same set of n embeddings, we only need to compute the embeddings of f and g, and then can quickly get those of $d_0^{(k)}$. An experimental estimate of $\min_k\left\{\max\left\{e_1(d_0^{(k)}), e_1\left(d_1^{(k)}\right)\right\}\right\}$ is given in Section 3.2. We are not aware of the impact on security by this technique, and similar idea also applies to Falcon.

The algorithm $\mathsf{NTRUSolve}(f, g)$ is explicated in [PP19] and we omit its description. The implementation of $\mathsf{IntGram}(\mathbf{\Sigma})$ follows the ring-based approach in [DGPY20]: we first compute $\mathbf{C} = \left\lfloor\sqrt{d'\mathbf{I} - \mathbf{\Sigma}}\right\rfloor \in \mathcal{R}_n^{2\times 2}$ for some $d' \in \mathbb{N}$ and then decompose $(d-d')\mathbf{I} - (\mathbf{C}\mathbf{C}^t - d'\mathbf{I} + \mathbf{\Sigma})$ that is diagonally dominant. Algorithm 2 presents a sketch of $\mathsf{IntGram}(\mathbf{\Sigma})$. While $\mathbf{A}$ is much wider than the trapdoor itself, it can be stored efficiently as shown in [DGPY20].

Algorithm 1 Key Generation KeyGen

Input: system parameters (see Table 1)
Output: public key $pk = h \in \mathcal{R}_n$ and
secret key $sk = (\mathbf{B}, \widetilde{v}, \mathbf{A})$ where $\mathbf{B} \in \mathcal{R}_n^{2\times 2}$, $\widetilde{v} \in \frac{1}{p}\mathcal{R}_n$ and $\mathbf{A} \in \mathcal{R}_n^{2\times m}$.

if $n = 512$ **then**
  $r_{sk} \leftarrow 1.17\sqrt{2\pi}\sqrt{\frac{q}{2n}}$
end if
if $n = 1024$ **then**
  $r_{sk} \leftarrow 1.17\sqrt{2\pi}\sqrt{\frac{q}{2n}}$
end if
$f', g' \leftarrow D_{\mathcal{R}_n, r_{sk}}$
if f' or g' is not invertible over $\mathcal{R}_n/q\mathcal{R}_n$ **then**
  restart
end if
find $k \in \mathbb{Z}_n$ minimizing $\max\left\{e_1\left(d_0^{(k)}\right), e_1\left(d_1^{(k)}\right)\right\}$ where $d_0^{(k)} = f'\overline{f'} + \sigma_k(g')\overline{\sigma_k(g')}$, $d_1^{(k)} = q^2/d_0^{(k)}$
if $\max\left\{e_1\left(d_0^{(k)}\right), e_1\left(d_1^{(k)}\right)\right\} > A_{n,q}^2$ **then**
  restart
end if
$(f, g) \leftarrow (f', \sigma_k(g'))$
$(F, G) \leftarrow \mathsf{NTRUSolve}(f, g)$ $\quad \{F, G \in \mathcal{R}_n\colon gF - fG = q\}$
$\mathbf{B} \leftarrow \begin{pmatrix} g & G \\ f & F \end{pmatrix}$
$\widetilde{v} \leftarrow \lfloor \frac{F\overline{f} + G\overline{g}}{f\overline{f} + g\overline{g}} \rceil_p$
$\widetilde{\mathbf{U}} \leftarrow \mathbf{U}_{\widetilde{v},n}$
$\widetilde{\mathbf{B}^*} \leftarrow \mathbf{B}\mathbf{U}_{-\widetilde{v},n}$
$\mathbf{A} \leftarrow \mathsf{IntGram}\left(p^2\widetilde{\mathbf{B}^*}\widetilde{\mathbf{B}^*}^t\right)$ $\quad \{\mathbf{A} \in \mathcal{R}_n^{2\times m}\colon \mathbf{A}\mathbf{A}^t = p^2\left((s'^2 - 1)\mathbf{I} - \widetilde{\mathbf{B}^*}\widetilde{\mathbf{B}^*}^t\right)\}$
$h = f^{-1}g \bmod q$
return $sk = (\mathbf{B}, \widetilde{v}, \mathbf{A})$ and $pk = h$

3.2 Signing

Since ℤalcon follows the GPV hash-and-sign paradigm, the signing procedure is essentially to sample from some lattice Gaussian centered at the hashed message. We therefore directly exhibit the new Gaussian sampler.

To sample $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$, our sampler proceeds as follows. First it generates a perturbation vector $\mathbf{p}$ from $D_{\mathcal{R}^2,\sqrt{\mathbf{\Sigma}_p}}$ where $\mathbf{\Sigma}_p = s^2\mathbf{I} - r^2\widetilde{\mathbf{B}^*}\widetilde{\mathbf{B}^*}^t$. This can be done in the off-line phase. Then the remaining work is to sample from $D_{\mathcal{L}(\mathbf{B}),r\sqrt{\widetilde{\mathbf{B}^*}\widetilde{\mathbf{B}^*}^t},\mathbf{c}-\mathbf{p}}$ in the on-line phase. By a linear transformation, it suffices to sample a vector $\mathbf{v}'$ from $D_{\mathcal{L}(\widetilde{\mathbf{U}}),r,\widetilde{\mathbf{B}^*}^{-1}(\mathbf{c}-\mathbf{p})}$ and then to output $\mathbf{v} = \widetilde{\mathbf{B}^*}\mathbf{v}'$. The formal algorithmic description is given in Algorithm 3.

With the auxiliary matrix $\mathbf{A}$, we can sample a perturbation vector fully over integers. Precisely, the sampling procedures consist of $D_{\mathbb{Z},Lr}$ and $D_{\mathbb{Z},r,c}$ with $c \in \frac{1}{pL}\mathbb{Z}$. Algorithm 4 shows the details.

The on-line sampling follows the framework of the GPV sampler [GPV08] but does not need floating-point arithmetic neither. Indeed the center is of the form $\frac{1}{pq}\mathcal{R}_n^2$, because

$$\widetilde{\mathbf{B}^*}^{-1} = \mathbf{U}_{\widetilde{v},n}\mathbf{B}^{-1} = \frac{1}{q}\mathbf{U}_{\widetilde{v},n}\begin{pmatrix} F & -G \\ -f & g \end{pmatrix} \in \frac{1}{pq}\cdot\mathcal{R}_n^{2\times 2}.$$

Algorithm 2 Integral Gram root decomposition $\mathsf{IntGram}(\mathbf{\Sigma})$

Input: a positive definite matrix $\mathbf{\Sigma} \in \mathcal{R}_n^{2\times 2}$ such that $B^2 \geq \|\mathbf{\Sigma}\|_2$
Output: $\mathbf{A} \in \mathcal{R}_n^{2\times m}$ such that $\mathbf{A}\mathbf{A}^t = (p^2 s'^2 - p^2)\mathbf{I} - \mathbf{\Sigma}$ and $\mathbf{A} \cdot \mathcal{R}_n^m = \mathcal{R}_n^2$.
1: $\mathbf{C} \leftarrow \lfloor \sqrt{B^2\mathbf{I} - \mathbf{\Sigma}} \rceil \in \mathcal{R}_n^{2\times 2}$ (by Cholesky decomposition)
2: $\mathbf{\Delta} \leftarrow \mathbf{C}\mathbf{C}^t - (B^2\mathbf{I} - \mathbf{\Sigma})$ $\{\|\mathbf{\Delta}\|_{\max} \leq (n+1)B + \frac{n(n+1)}{8}\}$
3: compute $\mathbf{A}' = \begin{pmatrix} 1 & b & b^2 & \mathbf{d}_0 \\ a_0 & a_1 & a_2 & \mathbf{d}_1 \end{pmatrix} \in \mathcal{R}_n^{2\times(6\ell+5)}$ such that $|a_i|_\infty < b$ and
$\mathbf{A}'\mathbf{A}'^t = ((p^2 s'^2 - p^2) - B^2 - 1)\mathbf{I} - \mathbf{\Delta}$ (see [DGPY20] for details)
4: **return** $\mathbf{A} = (\mathbf{I}\ \mathbf{C}\ \mathbf{A}')$

Algorithm 3 The NTRU Gaussian sampler $\mathsf{NTRUSampler}(sk = (\mathbf{B}, \widetilde{v}, \mathbf{A}), \mathbf{c})$

Input: secret key $sk = (\mathbf{B}, \widetilde{v}, \mathbf{A})$ and $\mathbf{c} \in \mathcal{R}_n^2$
Output: a sample $\mathbf{v}$ from a distribution within max-log distance $10\hat{\epsilon}$ of $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$.
Off-line phase:
1: $\mathbf{p} \leftarrow \mathsf{OfflineSampling}(\mathbf{A})$ $\{\mathbf{p} \sim D_{\mathcal{R}_n^2, r\sqrt{\mathbf{\Sigma}_p}}\}$
On-line phase:
2: $\mathbf{c}^{\text{pert}} \leftarrow \widetilde{\mathbf{B}^*}^{-1}(\mathbf{c} - \mathbf{p})$ $\{\widetilde{\mathbf{B}^*} = \mathbf{B}\mathbf{U}_{-\widetilde{v},n}\}$
3: $\mathbf{v}' \leftarrow \mathsf{OnlineSampling}(\widetilde{v}, \mathbf{c}^{\text{pert}})$ $\{\mathbf{v}' \sim D_{\mathcal{L}(\widetilde{\mathbf{U}}),r,\mathbf{c}'}\}$
4: **return** $\mathbf{v} = \widetilde{\mathbf{B}^*}\mathbf{v}'$

Additionally, $\widetilde{\mathbf{U}} = \begin{pmatrix} 1 & \widetilde{v} \\ & 1 \end{pmatrix}$ has a very simple Gram-Schmidt orthogonalization over $\mathcal{K}_n$, which naturally supports ring-friendly operations and makes all base samplings of the same width. The algorithmic description is shown in Algorithm 5.

Algorithm 5 The on-line sampling $\mathsf{OnlineSampling}(\widetilde{v}, \mathbf{c}^{\text{pert}})$

Input: $\widetilde{v} \in \frac{1}{p} \cdot \mathcal{R}_n$, $\mathbf{c}^{\text{pert}} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \in \frac{1}{pq} \cdot \mathcal{R}_n^2$
Output: a sample $\mathbf{u}$ from a distribution within max-log distance $4\hat{\epsilon}$ of $D_{\mathcal{L}(\mathbf{U}_{\widetilde{v},n}),r,\mathbf{c}}$
1: $u_2' \leftarrow D_{\mathcal{R}_n,r,c_2}$
2: $c_1' \leftarrow c_1 - u_2' \cdot \widetilde{v}$
3: $u_1' \leftarrow D_{\mathcal{R}_n,r,c_1'}$
4: **return** $\mathbf{u} = \begin{pmatrix} 1 & \widetilde{v} \\ & 1 \end{pmatrix} \begin{pmatrix} u_1' \\ u_2' \end{pmatrix}$

Parameter Requirements. Firstly, we present the parameter conditions required by Algorithms 2 and 4. According to Lemmata 6 and 12 in [DGPY20], Algorithm 2 is correct when

$$b^3 \geq (n+1)\left(B + \frac{n}{8}\right) + 6nb^2; \quad p^2 s'^2 - p^2 - B^2 - 1 \geq \frac{b^6 - 1}{b^2 - 1}\ell + b^3.$$

By Theorem 3 in [DGPY20], the correctness of Algorithm 4 is satisfied with

$$r \geq \overline{\eta_\epsilon}(\mathbb{Z}^{2n}); \quad Lr \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A})).$$

Secondly, we estimate $\|\widetilde{\mathbf{B}^*}\|_2$ that determines the minimal Gaussian width $s = s'r$. Let $\mathbf{U} = \mathbf{U}_{v,n}$ where $v = \frac{F\overline{f} + G\overline{g}}{f\overline{f} + g\overline{g}}$, then $\mathbf{B}^* = \begin{pmatrix} g & G^* \\ f & F^* \end{pmatrix} = \mathbf{B}\mathbf{U}^{-1}$. A routine computation shows that $(\mathbf{B}^*)^t\mathbf{B}^* = \begin{pmatrix} d_0 & \\ & d_1 \end{pmatrix}$ where

Algorithm 4 The off-line sampling $\mathsf{OfflineSampling}(\mathbf{A})$

Input: integral Gram root matrix $\mathbf{A} \in \mathcal{R}_n^{2\times m}$ such that $\mathbf{A}\mathbf{A}^t = p^2(\mathbf{\Sigma}_p - \mathbf{I})$
Output: a sample $\mathbf{p}$ from a distribution within max-log distance $8\hat{\epsilon}$ of $D_{\mathcal{R}_n^2, r\sqrt{\mathbf{\Sigma}_p}}$.

1: $\mathbf{p}' \leftarrow \frac{1}{pL} \cdot \mathbf{A} \cdot D_{\mathcal{R}_n^m, Lr}$
2: $\mathbf{p} \leftarrow D_{\mathcal{R}_n^2, r, \mathbf{p}'}$
3: **return** $\mathbf{p}$

$d_0 = f\overline{f} + g\overline{g}$ and $d_1 = F^*\overline{F^*} + G^*\overline{G^*} = \frac{q^2}{d_0}$ are positive definite. Then

$$\begin{aligned}\left\|\widetilde{\mathbf{B}^*}\right\|_2 &\le \left\|\mathbf{B}(\widetilde{\mathbf{U}}^{-1} - \mathbf{U}^{-1})\right\|_2 + \|\mathbf{B}^*\|_2 \\ &\le \left\|\begin{pmatrix} g(v - \lfloor v \rceil_p) \\ f(v - \lfloor v \rceil_p)\end{pmatrix}\right\|_2 + \sqrt{\max\{e_1(d_0), e_1(d_1)\}} \\ &\le \frac{n^{1.5}}{2p}\|(g, f)\| + \sqrt{\max\{e_1(d_0), e_1(d_1)\}}.\end{aligned}$$

We now estimate $\max\{e_1(d_0), e_1(d_1)\}$ that equals $\min_k\left\{\max\left\{e_1\left(d_0^{(k)}\right), e_1\left(d_1^{(k)}\right)\right\}\right\}$ according to our key generation. We suppose (f', g') follows some discrete Gaussian of width $\alpha \cdot \sqrt{2\pi} \cdot \sqrt{\frac{q}{2n}}$ and experimentally searched the optimal α minimizing $\max\{e_1(d_0), e_1(d_1)\}$, and exhibit the experimental data in Figure 1. As a result, we observe that:

- For $n = 512$, the optimum is around $\alpha = 1.36$ leading to (the median of) $\sqrt{\max\{e_1(d_0), e_1(d_1)\}} \approx 282.07$.
- For $n = 1024$, the optimum is around $\alpha = 1.47$ leading to (the median of) $\sqrt{\max\{e_1(d_0), e_1(d_1)\}} \approx 340.38$.

Hence we set $A_{n,q} = 283$ for $n = 512$ and $A_{n,q} = 341$ for $n = 1024$.

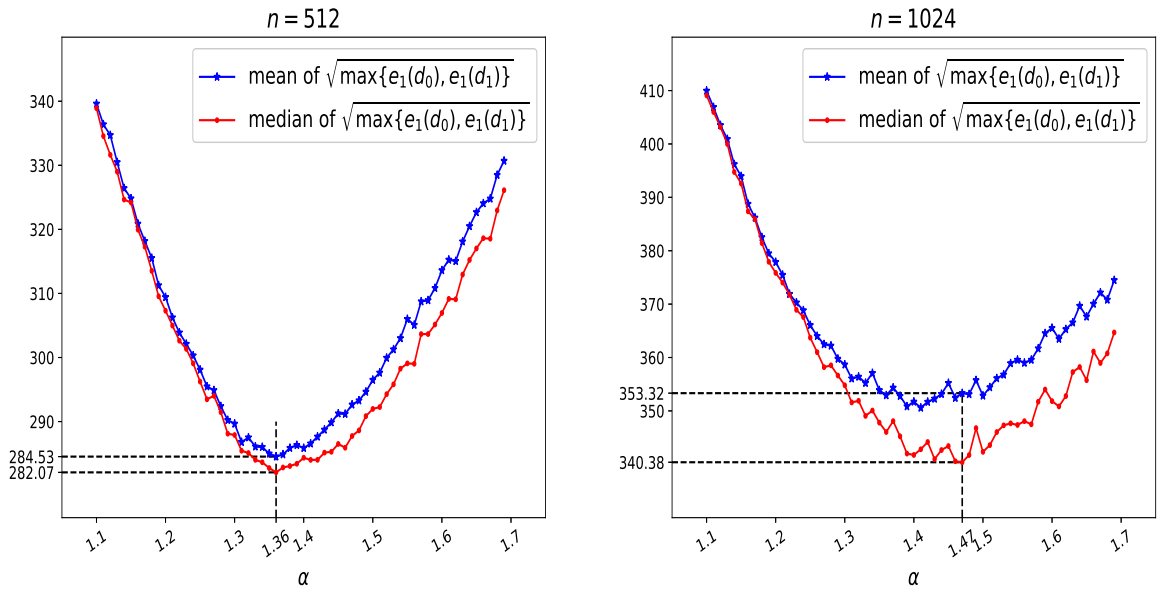

Fig. 1. The left figure is for $(n, q) = (512, 12289)$ and the right one for $(n, q) = (1024, 12289)$. Experimental values measure over 1000 random (f, g) for each α. We plot the curves for both median and mean of $\max\{e_1(d_0), e_1(d_1)\}$.

Remark 5. In [Pre15], Prest estimated $\sqrt{\max\{e_1(d_0), e_1(d_1)\}}$ for f, g drawn from $D_{\mathcal{R},r_{sk}}$. His heuristic analysis assumes that the minimum of $\sqrt{\max\{e_1(d_0), e_1(d_1)\}}$ is achieved when $r_{sk} \approx 1.17\sqrt{2\pi}\sqrt{\frac{q}{2n}}$; the suggested threshold is about $1.265\sqrt{\ell q}$ that ≈ 420.7 (resp. 443.5) for $n = 512$ (resp. 1024) larger than our $A_{n,q}$. This is mainly due to that we select the optimal combination $(f, \sigma_k(g))$ out of n candidates. Note that the reduced threshold $A_{n,q}$ yields smaller signatures and about 10 bits security increase.

Finally, we bound L. Let $M = mn - n$ be the dimension of $\Lambda^{\perp}(\mathbf{A})$. From Lemma 3, we have

$$\eta_{\epsilon}(\Lambda^{\perp}(\mathbf{A})) \leq \overline{\eta_{\epsilon}}(\mathbb{Z}^M) \cdot \lambda_M(\Lambda^{\perp}(\mathbf{A})).$$

By the same argument of [DGPY20], it holds that $\lambda_M(\Lambda^{\perp}(\mathbf{A})) \leq b^2\sqrt{2n}$ when $B = \omega(n^2)$ that coincides our setting. Combining that $\overline{\eta_{\epsilon}}(\mathbb{Z}^M) \leq \sqrt{2}r$, it suffices to choose $L \geq 2b^2\sqrt{n}$.

Correctness of the Sampler. The correctness of Algorithm 3 is given by Lemma 6.

Lemma 6. *Let $n = 2^{\ell} \in \mathbb{N}$, $\mathbf{B} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}_n^{2\times 2}$, $\mathbf{c} \in \mathcal{R}_n^2$, $p \in \mathbb{N}$. Let $\widetilde{\mathbf{B}^*}$ be defined as in Algorithm 3 and $s' \geq \|\widetilde{\mathbf{B}^*}\|_2 + 1$ be an integer. Let $s = s'r$ where $r \geq \overline{\eta_{\epsilon}}(\mathbb{Z}^{2n})$ and $\epsilon' \in (0,1)$. Then the distribution of the output of Algorithm 3 is within max-log distance $14\hat{\epsilon}$ of $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$.*

Proof. We first verify the correctness of three sub-routines. As shown in Section 3.2, Algorithms 2 and 4 are correct under the parameter restriction in Table 1. As for Algorithm 5, we first note that $\|\mathbf{U}_{\widetilde{v},n}\|_{GS} = 1$. By the same proof for Theorem 4.1 of [GPV08], the correctness of Algorithm 5 follows immediately.

We now prove the correctness of Algorithm 3. Let $\mathbf{\Sigma}_1 = \widetilde{\mathbf{B}^*}\widetilde{\mathbf{B}^*}^t$, $\mathbf{\Sigma}_2 = \mathbf{\Sigma}_p$ and $\mathbf{\Sigma}_3 \in \mathcal{K}_n^{2\times 2}$ such that $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$. The probability function can be expressed as

$$\begin{aligned}
&\Pr\left(\mathsf{NTRUSampler}(s, \mathbf{B}, \mathbf{c}, p, \epsilon) = \mathbf{v}\right) \\
\propto &\sum_{\mathbf{p}} \Pr\left(\mathsf{OfflineSampling}(n, m, r', L, \mathbf{\Sigma}_p, p, \mathbf{A}, \epsilon) = \mathbf{p}\right) \cdot \rho_{r,\mathbf{c}'}(\mathbf{v}') \cdot \left[\frac{1-2\hat{\epsilon}}{1+2\hat{\epsilon}}, \frac{1+2\hat{\epsilon}}{1-2\hat{\epsilon}}\right] \\
\propto &\sum_{\mathbf{p}} \rho_{r\sqrt{\mathbf{\Sigma}_2}}(\mathbf{p}) \cdot \rho_{r\sqrt{\mathbf{\Sigma}_1},\mathbf{c}-\mathbf{p}}(\mathbf{v}) \cdot \left[\frac{1-4\epsilon}{1+4\epsilon} \cdot \frac{1-2\hat{\epsilon}}{1+2\hat{\epsilon}}, \frac{1+4\epsilon}{1-4\epsilon} \cdot \frac{1+2\hat{\epsilon}}{1-2\hat{\epsilon}}\right] \\
\propto &\rho_{r\sqrt{\mathbf{\Sigma}_3}}\left(\mathcal{R}_n^2\right) \cdot \rho_{s,\mathbf{c}}(\mathbf{v}) \cdot \left[\frac{1-4\epsilon}{1+4\epsilon} \cdot \frac{1-2\hat{\epsilon}}{1+2\hat{\epsilon}} \cdot \frac{1-\epsilon}{1+\epsilon}, \frac{1+4\epsilon}{1-4\epsilon} \cdot \frac{1+2\hat{\epsilon}}{1-2\hat{\epsilon}} \cdot \frac{1+\epsilon}{1-\epsilon}\right] \\
\propto &\rho_{s,\mathbf{c}}(\mathbf{v}) \cdot \left[\frac{1-7\hat{\epsilon}}{1+7\hat{\epsilon}}, \frac{1+7\hat{\epsilon}}{1-7\hat{\epsilon}}\right]
\end{aligned}$$

The correctness of Algorithms 5 and 4 respectively lead to the first and second equation. The third equation follow by the same argument in [Pei10] and the fact that $r\sqrt{\mathbf{\Sigma}_3} \geq \eta_{\epsilon}\left(\mathcal{R}_n^2\right)$. The last equation comes from a routine computation. So far, the proof is completed. □

Comparison with Other Samplers. We now compare our sampler with other samplers.

1. In [DP16], Ducas and Prest proposed a ring variant of the Klein sampler [Kle00, GPV08]. This is used in Falcon [FHK+] now. This sampler achieves the smallest Gaussian width $\|\mathbf{B}\|_{GS} \cdot \overline{\eta_{\epsilon}}(\mathbb{Z}^{2n})$ and good running time $\widetilde{O}(n)$, but heavily relies on floating-point arithmetic.
2. In [Pei10], Peikert proposed an efficient sampler for q-ary lattices running in $\widetilde{O}(n)$. With the techniques of [DGPY20], the sampler can be fully performed over integers. But the main drawback is that the minimal width becomes $\|\mathbf{B}\|_2 \cdot \overline{\eta_{\epsilon}}(\mathbb{Z}^{2n})$, which leads to a significant quality loss.
3. In [Duc13], Ducas proposed a variant of the Klein sampler. That achieves the same quality as the Klein sampler and does not need FPA in the sampling phase. However, FPA is still used in the final rejection sampling. In addition, the sampler works for generic case and its running time is $O(n^2)$. It remains unclear how to modify the sampler into a ring-efficient variant.

4. In [Pre15], Prest proposed a hybrid sampler that somewhat inherits the merits of both the Klein and Peikert samplers. The minimal Gaussian width is $\|\mathbf{B}^*\|_2 \cdot \overline{\eta_\epsilon}(\mathbb{Z}^{2n})$ and the running time is $\widetilde{O}(n)$. One issue is that removing FPA in this sampler seems complicated. In fact, our sampler achieves the same efficiency and Gaussian quality as the Prest's one but avoids FPA.

According to the analysis in [Pre15], it holds for optimal NTRU bases that

- $\|\mathbf{B}\|_{GS} \approx \|(g, f)\|$
- $\|\mathbf{B}\|_2 \approx 0.82 \cdot n^{\frac{1}{4}}\sqrt{\log n}\|(g, f)\|$
- $\|\mathbf{B}^*\|_2 \approx 1.08\sqrt{\log n}\|(g, f)\|$

Above comparisons are summarized as Table 2. We see that our sampler is both efficient and FPA-free at the cost of a moderate quality loss.

Table 2. Comparison with existing samplers. We omit a factor of $\overline{\eta_\epsilon}(\mathbb{Z}^{2n})$ when comparing the quality.

	quality	running time	need for FPA
[DP16]	$\|\mathbf{B}\|_{GS}$	$\widetilde{O}(n)$	Yes
[Pei10]	$\|\mathbf{B}\|_2$	$\widetilde{O}(n)$	No
[Duc13]	$\|\mathbf{B}\|_{GS}$	$O(n^2)$	Yes
[Pre15]	$\|\mathbf{B}^*\|_2$	$\widetilde{O}(n)$	Yes
This work	$\|\mathbf{B}^*\|_2$	$\widetilde{O}(n)$	No

3.3 Concrete Parameters

We now provide two sets of concrete parameters of ℤalcon in Table 3. For consistency, we fix $r = 3.58$ for two sets of parameters and $r \geq \overline{\eta_\epsilon}(\mathbb{Z})$ with $\epsilon = 2^{-57}$ and $r \geq \overline{\eta_\epsilon}(\mathbb{Z}^{2n})$ for $(\epsilon, n) = (2^{-47}, 512), (2^{-46}, 1024)$. We list (ϵ, n) such that $r \approx \overline{\eta_\epsilon}(\mathbb{Z}^{2n})$ in Table 3.

Table 3. Concrete parameters.

	n	q	ϵ	r	$A_{n,q}$	s'	p	B	b	L
Param-I	512	12289	2^{-47}	3.58	283	300	2^{21}	283×2^{21}	8192	2^{35}
Param-II	1024	12289	2^{-46}	3.58	341	360	2^{24}	341×2^{24}	20480	2^{35}

We estimate the security by lattice attacks based on BKZ reduction. There are two basic attack scenarios for a signature scheme: key recovery attack and forgery attack. Since the trapdoor basis of ℤalcon is generated in the same manner as that of Falcon, ℤalcon achieves the same security against key recovery attack as Falcon: the blocksize required by key recovery attack is at least 482 (resp. 974) for $n = 512$ (resp. 1024). As for forgery attack, the final Gaussian width $s = s'r$ of ℤalcon is larger than that of Falcon due to the larger s'. This weakens the resistance against forgery attack: the minimal blocksize for forgery attack is only 312 (resp. 718) for $n = 512$ (resp. 1024). Given blocksize β, we use the Core-SVP hardness $2^{0.265\beta}$ as the estimated security. This model is quite conservative and takes quantum speedups into account. Detailed numbers are shown in Table 4.

Table 4. Concrete security estimate for ℤalcon based on lattice reduction. The item "A/B" represents the security estimate A and the required blocksize B for BKZ.

	Key Recovery	Forgery
Param-I	134.4 / 507	82.7 / 312
Param-II	274.8 / 1037	190.3 / 718

Remark 7. For the same n, the security and compactness achieved by ℤalcon is worse than those by Falcon, due to larger Gaussian width caused by the new sampler. Nevertheless, compared with Dilithium, ℤalcon is more compact for the equal security level.

4 Integer Gaussian Sampling

The signing procedure requires two types of integer Gaussian samplings as follows:

- Gaussian with arbitrary centers, i.e. $D_{\mathbb{Z},r,c}$ where c is in either $\frac{1}{pq}\mathbb{Z}$ or $\frac{1}{pL}\mathbb{Z}$. This appears in both the on-line sampling (Algorithm 5)) and the off-line sampling (Algorithm 4).
- Gaussian with a large width, i.e. $D_{\mathbb{Z},Lr}$ where L is large. This only appears in the off-line sampling (Algorithm 4).

4.1 Arbitrary Centers

We need to deal with two kinds of Gaussian centers: $c \in \frac{1}{pq}\mathbb{Z}$ and $c \in \frac{1}{pL}\mathbb{Z}$. In ℤalcon, the denominators pq and pL are huge. For more efficient samplings, we exploit the technique introduced in [MW17]. Specifically, we approximate the center c with its k-bit randomized rounding $\mathsf{RR}_k(a) = \lfloor 2^k a \rfloor / 2^k + \mathcal{B}_\alpha$ where $\mathcal{B}_\alpha$ is a Bernoulli variable with parameter $\alpha = 2^k a \bmod 1$, which only incurs a loss of about $\pi^2/2^{2k}$ for the max-log distance [MW17]. Using the improvement of [Pre17, Section 4.1], for 256-bits of security, it suffices to implement $D_{\mathbb{Z},r,c}$ with $c \in 2^{-30}\mathbb{Z}$.

We use the Micciancio-Walter sampler [MW17] to implement $D_{\mathbb{Z},r,\mathsf{RR}_{30}(c)}$ with $\mathsf{RR}_{30}(c) \in 2^{-30}\mathbb{Z}$. This sampler decomposes the sampling with calls to an easier to handle Gaussian $D_{\mathbb{Z},r_1,c'}$ with different centers and standard deviation. More precisely, $r_1 \approx r$ and $c' \in 2^{-k'}\mathbb{Z}$ where $k' \ll 30$.

Let (β, l) be two parameters. The formal description of our adaptation of Micciancio-Walter sampler is given in Algorithm 8. The algorithm proceeds iteratively. First, one need to find $c^{\text{frac}}, c^{\text{int}} \in \mathbb{Z}$ such that

$$\mathsf{RR}_{30}(c) = \underbrace{c^{\text{frac}}/\beta^l}_{<1} + \underbrace{c^{\text{int}}}_{\in\mathbb{Z}}.$$

Drawing from $D_{\mathbb{Z},r,\mathsf{RR}_{30}(c)}$ is the same as drawing from $D_{\mathbb{Z},r_l,c^{\text{frac}}/\beta^l}$ and adding c^{int}. At each level, it reduces the sampling from $D_{\mathbb{Z},r_l,c^{\text{frac}}/\beta^l}$ to the sampling of $D_{\mathbb{Z},r_{l-1},c^{\text{frac}'}/\beta^{l-1}}$.
The underlying idea can be verified thanks to Theorem 4: sampling $D_{\mathbb{Z},r_l,c}$ with $c \in \beta^{-l}\mathbb{Z}$ can be done by first sampling $z_2 \leftarrow \beta^{-l+1} D_{\mathbb{Z},r_1,\beta^{l-1}c}$ and then outputting $z_1 \leftarrow D_{\mathbb{Z},r_{l-1},z_2}$ with $z_2 \in \beta^{-l+1}\mathbb{Z}$.
For the correctness, we refer to [MW17] and make sure that $r_1 = r/\sqrt{\sum_{i=0}^{l-1}\beta^{-2i}}$ and $r_1 \geq \sqrt{\frac{\beta+1}{\beta}}\eta_\epsilon(\mathbb{Z})$.

One interesting advantage of this technique is that we can easily avoid FPA. We just have to implement $\mathsf{SampleC}_1$ with a CDT (cumulative distribution table) approach [Pei10]: we pre-compute the respective CDTs for each $c^{\text{frac}}/\beta \in \left\{\frac{0}{\beta}, \frac{1}{\beta}, \cdots \frac{\beta-1}{\beta}\right\}$. There is a tradeoff between β and l the number of calls of the base sampler $\mathsf{SampleC}_1$ by $\mathsf{SampleC}_l$: larger β means less calls of the base sampler but more storage for CDTs. In our implementation, we choose

$$(\beta, l) = (2^6, 5),$$

for which the width of the base sampler is

$$r_1 = r/\sqrt{\sum_{i=0}^{l-1} \beta^{-2i}} \approx 3.5796.$$

Algorithm 6 $\mathsf{SampleC}_l$

Input: $c \in \frac{1}{pq}\mathbb{Z}$ or $\frac{1}{pL}\mathbb{Z}$
Output: $z \sim D_{\mathbb{Z},r,\mathsf{RR}_{30}(c)} \sim D_{\mathbb{Z},r,c}$
1: $c^{\text{rr}} \leftarrow \mathsf{RR}_{30}(c) \cdot \beta^l$
2: $c^{\text{frac}} \leftarrow c^{\text{rr}} \bmod \beta^l$ {Fractionnal part. Note that $c^{\text{frac}} \in [0, \beta^l - 1]$.}
3: $c^{\text{int}} \leftarrow \lfloor \frac{c^{\text{rr}}}{\beta^l} \rfloor$ {Integral part.}
4: **for** $i := 1$ to l **do**
5: $\quad c^{\text{frac}} \leftarrow \mathsf{SampleC}_1(c^{\text{frac}})$
6: **end for**
7: **return** $c^{\text{int}} + c^{\text{frac}}$

Algorithm 7 $\mathsf{SampleC}_1$

Input: $c^{\text{frac}} \in \mathbb{Z}$
Output: $z \sim D_{\mathbb{Z},r_1,c^{\text{frac}}/\beta}$
1: $c^0 \leftarrow c^{\text{frac}} \bmod \beta$ {Fractional part of c^{frac}/β}
2: $c^1 \leftarrow \lfloor \frac{c^{\text{frac}}}{\beta} \rfloor$ {Integral part of c^{frac}/β}
3: **return** $c^1 + \mathsf{CDT}_{r_1,c^0}()$

Halving the storage of CDTs. Let us denote $\mathsf{CDT}_{r_1,c}$ a CDT-based sampler for the center $c \in \left\{\frac{0}{\beta}, \frac{1}{\beta}, \cdots \frac{\beta-1}{\beta}\right\}$. We observe that if $z \sim D_{\mathbb{Z},r_1,c}$, then $z' = 1 - z \sim D_{\mathbb{Z},r_1,1-c}$. This means that with $\mathsf{CDT}_{r_1,c}$, can be used for both $D_{\mathbb{Z},r_1,c}$ and $D_{\mathbb{Z},r_1,1-c}$ with no additional cost. Therefore, it suffices to store the full CDTs for $c \in \left\{\frac{0}{\beta}, \frac{1}{\beta}, \cdots \frac{\beta/2}{\beta}\right\}$, which halves the storage of CDTs for free.

To estimate the precision of the CDT samplers, we use Renyi divergence results from [BLL+15] that were reused in [Pre17, HPRR20]. Let $\mathtt{Q}_{\mathsf{CDT}}$ denote the number of queries to all the base samplers $(\mathsf{CDT}_{r_1,c})_c$. A sufficient condition for providing $\lambda - O(1)$ bits of security is the following.

$$R_{2\lambda-1}\left(\mathsf{CDT}_{r_1,c}, D_{\mathbb{Z},r_1,c}\right) \leq 1 + \frac{1}{4\mathtt{Q}_{\mathsf{CDT}}}.$$

The number of queries is bounded as follows.

$$\mathtt{Q}_{\mathsf{CDT}_c} \leq m \times n \times \overbrace{l}^{\text{calls}} \times \overbrace{2^{64}}^{\text{max queries}} = (6\ell+1) \times 2^{\ell} \times l \times 2^{64} = 2^{80} \text{ for } \ell \in \{9, 10\} \text{ and } l = 5.$$

Using tools from [HPRR20] we generate all the tables such that $R_{383}\left(\mathsf{CDT}_{r_1,c}, D_{\mathbb{Z},r_1,c}\right) \leq 1 + 2^{-82}$ for all $c \in \left\{\frac{0}{\beta}, \frac{1}{\beta}, \cdots \frac{\beta/2}{\beta}\right\}$. They each contain 15 coefficients of 82 bits. The storing of β tables seems possible even on constraint systems because of their relatively small size.

Remark 8. If an attacker is able to see which table is examined by analyzing the memory management, it may be possible to recover the sensitive center values. Nevertheless, this attack belongs in a very strong side-channel attack model.

Remark 9. Micciancio and Walter also proposed in [MW17] to separate the base samplings (i.e. $\mathsf{CDT}_{r_1,c}$) and the remaining combination operations into two devices with shared buffers. Each buffer corresponds to a coset in $\mathbb{Z}/\beta$. By filling these buffers with a certain number of samples in off-line, $\mathsf{SampleC}_l$ can output the result quickly. One interesting feature is the fact that, depending on the context, this off-line buffer filling technique could be left unprotected against side channels (either not isochronous and/or not masked). Thus, both the on-line sampling and the off-line sampling can be faster for the price of a buffer storing. Nevertheless, it is not beneficial in any contexts: for better protection and general use case, we also present $\mathsf{SampleC}_1$ in a fully side-channel secure manner (see Section 6).

4.2 Large Width

We now move to the sampling of $D_{\mathbb{Z},Lr}$. Let us first clarify that this part is still in progress. The large width samplings only perform in the off-line phase. There exist some controversial views on the efficiency, cost and side-channel security with respect to the off-line sampling. While we have seen several candidate samplers for $D_{\mathbb{Z},Lr}$, more reviews are needed to select a proper one especially considering various restrictions on the off-line sampling.

In the following, we introduce a working approach to implementing $D_{\mathbb{Z},Lr}$ that is introduced in [MW17] as well. The sampling of $D_{\mathbb{Z},Lr}$ is decomposed into l levels; at the i-th level, we sample from $D_{\mathbb{Z},r_i}$. According to [MW17], the sequence $\{r_i\}_i$ satisfies that $r_l = Lr$ and

$$r_i = r_{i-1}\sqrt{z_{i-1}^2 + (z_{i-1}-1)^2} \quad \text{where} \quad z_{i-1} \leq \left\lfloor \frac{r_{i-1}}{\sqrt{2\overline{\eta_\epsilon}(\mathbb{Z})}} \right\rfloor .$$

Algorithm 8 $\mathsf{SampleLW}_l$

Output: $z \sim D_{\mathbb{Z},r_l}$

Pre-computation:

$\mathbf{v} \leftarrow (), N \leftarrow 2^l$
for $i := 0$ to $N-1$ **do**
  $(b_0 \cdots b_{l-1}) \leftarrow$ the binary represent of i
  $v \leftarrow \prod_{j=0}^{l-1}(z_j - b_j)$
  $\mathbf{v} \leftarrow (\mathbf{v}, v)$
end for

Sampling:

$\mathbf{s} \leftarrow D_{\mathbb{Z},r_0}^N$
return $\langle \mathbf{s}, \mathbf{v} \rangle$

In Zalcon, we fix $L = 2^{35}$ and $r = 3.58$. Similar to the case in Section 4.1, one can implement the base sampling $D_{\mathbb{Z},r_0}$ with CDT approach. We suggest to set $l = 4$ and $\{(r_i, z_i)\}_i$ as in Table 4.2. Then the table for $D_{\mathbb{Z},r_0}$ should contain about $\lceil 6r_0 \rceil = 136$ entries. Each call of $D_{\mathbb{Z},Lr}$ requires $2^4 = 16$ calls of $D_{\mathbb{Z},r_0}$ along with a few simple integration operations.

	$i=4$	$i=3$	$i=2$	$i=1$	$i=0$
r_i/r	2^{35}	185364.507	431.869	22.605	6.270
z_i		131072	304	14	3

Table 5. The suggested (r_i, z_i).

Remark 10. The off-line sampling is costly. One important cause is the large column number of the integral Gram root $\mathbf{A}$: the overall calls of the base sampler achieves $n \times m \times 16 = 2^9 \cdot 63 \cdot 2^4 \approx 2^{19}$ for $n = 512$. But unlike the sampling in Section 4.1, all base samplings required by large width samplings are the same, namely $D_{\mathbb{Z},r_0}$, which is more convenient for parallel instructions.

5 Preliminary benchmarks results

Since this work is still ongoing research, providing a solid implementation optimized on a specific platform is not reasonable as it would have to be heavily modified later. However, while an accurate comparison with sibling schemes is only possible once the design is fixed and several optimization passes have been made, it is always nice to have some basic experimental result to ensure that the performances will be reasonable.
We implemented the on-line phase of the algorithm in plain C and ran it on a mid-range laptop with an `Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz`. The code was compiled with the `-O2` optimization flag and implements our first parameter set. We averaged around 400 on-line phases per second. While this performances are not yet at the level of a fully-fledged scheme, these are quite encouraging results for a first unoptimized iteration of the code. The sampling of a full polynomial from $D_{\mathcal{R}_n,r,c}$ costs approximately 1.5 Mcycles. Polynomial multiplication was performed using a textbook version of Karatsuba's algorithm.
Our intuition is that the most important implementation challenge this scheme will face in the signing phase is how to handle the large values ($> 2^{64}$) that appear after multiplications in $\mathcal{R}$.

6 Side-channel protection

First, our sampler is isochronous by design. According to [HPRR20], isochrony ensures independence between the running time of the algorithm and the secret values. Indeed, the use of integers and the absence of conditional branches implies that its timing is independent from the secret key. The only point of attention should be the CDT sampling where the implementation must go through the whole table instead of stopping when the sample is found. In this Section, we turn our sampler into an equivalent one which is protected against more powerful side-channel attacks that exploit the leakage of several executions. We provide here a masked version with a proof in the ISW model [ISW03]. Essentially, we will provide a functionally equivalent alternative algorithm where any set of at most d intermediate variables is independent from $sk = (\mathbf{B}, \widetilde{v}, \mathbf{A})$. The final output of the sampling, $\mathbf{v}$, will part of the signature of the algorithm and thus it will be unmasked before being returned.

In this work in progress, we focus on the most sensitive part: the on-line part of the sampling, namely lines 2 to 4 of Algorithm 3. However, we do not see any immediate difficulty in generalizing the masking to the off-line part (line 1 of Algorithm 3) and we plan on providing a masked off-line part as well in a future version of this paper. Thus, the masked off-line phase, denoted MaskedOfflineSampling, will temporarily be ignored in this document as it is less sensitive and could be performed without masking.

As often in masked lattice-based schemes, some parts of the algorithms are suited for an arithmetical masking and some other ones are suited for Boolean masking. We thus need to perform Boolean to arithmetic conversions (see Section 6.1). We choose an arithmetical masking everywhere in Algorithm 9 except in the function $\mathsf{MaskedSampleC}_1$ that needs Boolean comparisons. In addition, the masking of the on-line sampler presents two unprecedented difficulties in masked lattice-based schemes.

1. 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 maximum size of the manipulated integers. Q^{mask} should be sufficiently large so that no wrap around occurs. We find this bound with a careful analysis of the sizes of the secret key and the samples. It is also possible to use several Q^{mask} with CRT techniques to reduce the size of Q^{mask} if needed as it is proposed in the Falcon description with RNS.

2. 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 denoted SecNTTMult in Section 6.1.

As formally proved in [BBD+16], a proof of masking in the d-probing model can be divided into proofs of subparts, called gadgets. The gadgets should be proved d-NI and d-SNI; we refer to [BBD+16] for the formal description of these properties. In a nutshell, a gadget is d-non-interfering (d-NI) iff any set of at most d observations can be perfectly simulated from at most d shares of each input. A gadget is d-strong non-interfering (d-SNI) iff any set of at most d observations whose d_{int} observations on the internal data and d_{out} observations on the outputs can be perfectly simulated from at most d_{int} shares of each input. It is easy to check that d-SNI implies d-NI which implies d-probing security. In Table 6, we introduce the known and new gadgets necessary for our sampler along with their properties. These properties will be proved in Section 6.1.

Table 6. Masking properties of known and new gadgets

Gadget's name	Security Property	Reference
SecAdd	d-NI	[BBE+18]
SecMult	d-SNI	[ISW03, RP10, BBD+16]
A2B and B2A	d-SNI	[CGV14, Cor17, BBE+18, SPOG19]
MaskedCDT	d-NI	[BBE+19, GR19]
SecNTTMult	d-NI	This work, Lemma 12
MaskedSampleC$_1$	d-NI	This work, Lemma 14
MaskedSampleC$_l$	d-NI	This work, Lemma 15
MaskedOnlineSampling	d-NI	This work, Lemma 11

The overall structure of the sampler is presented in Algorithm 9. It consists in a linear succession of gadgets with no dependency cycle, i.e. each line depends on freshly computed masked inputs. Thus, only a proof for each line is necessary. Line 2 is a linear operation thus it is d-NI. Lines 3 and 5 are proofs d-NI in Section 6.1. Line 6 does not manipulate any sensitive value.

Algorithm 9 MaskedNTRUSampler

Input: . A masked secret key in the following form: $((\widetilde{\mathbf{B}^*}_i)_{0\le i\le d}, (\widetilde{\mathbf{B}^*}_i^{-1})_{0\le i\le d}, (\widetilde{v}_i)_{0\le i\le d}, (\mathbf{A}_i)_{0\le i\le d})$ and a masked vector $(\mathbf{c}_i)_{0\le i\le d} \in \mathcal{R}_n^2$, both arithmetically masked mod Q^{mask}

Output: an umasked sample $\mathbf{v} \sim D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$.

Off-line phase:

1: $(\mathbf{p}_i)_{0\le i\le d} \leftarrow$ MaskedOfflineSampling$((\mathbf{A}_i)_{0\le i\le d})$

On-line phase:

2: $(\mathbf{c}^{\text{pert}}{}_i)_{0\le i\le d} \leftarrow (\mathbf{c}_i)_{0\le i\le d} - (\mathbf{p}_i)_{0\le i\le d}$

3: $(\mathbf{c}^{\text{pert}}{}_i)_{0\le i\le d} \leftarrow$ SecNTTMult$((\widetilde{\mathbf{B}^*}_i^{-1})_{0\le i\le d}, (\mathbf{c}^{\text{pert}}{}_i)_{0\le i\le d})$

4: $(\mathbf{v}'_i)_{0\le i\le d} \leftarrow$ MaskedOnlineSampling$((\widetilde{v}_i)_{0\le i\le d}, (\mathbf{c}^{\text{pert}}{}_i)_{0\le i\le d})$

5: $(\mathbf{v}_i)_{0\le i\le d} \leftarrow$ SecNTTMult$((\widetilde{\mathbf{B}^*}_i)_{0\le i\le d}, (\mathbf{v}'_i)_{0\le i\le d})$

6: **return** $\sum_{i=0}^{d} \mathbf{v}_i \bmod \mathsf{Q}^{\text{mask}}$

We present the masked function MaskedOnlineSampling in line 4 of Algorithm 9 in Algorithm 10. It structure is presented in Figure 2. One can show that this algorithm is d-NI, as proved in Lemma 11 below.

Algorithm 10 MaskedOnlineSampling

Input: Two arithmetically masked mod Q^{mask} values $(\widetilde{v}_i)_{0\le i\le d}$ and $(\mathbf{c}^{\text{pert}}{}_i)_{0\le i\le d} = \begin{pmatrix}(c_{1\,i})_{0\le i\le d}\\(c_{2i})_{0\le i\le d}\end{pmatrix}$

Output: An arithmetically masked $(\mathbf{u}_i)_{0\le i\le d}$

1: $(u'_{2\,i})_{0\le i\le d} \leftarrow \mathsf{MaskedSampleC}_l((c_{2i})_{0\le i\le d})$
2: $(x_i)_{0\le i\le d} \leftarrow \mathsf{SecNTTMult}((u'_{2\,i})_{0\le i\le d}, (\widetilde{v}_i)_{0\le i\le d})$
3: $(c'_{1\,i})_{0\le i\le d} \leftarrow (c_{1\,i})_{0\le i\le d} - (x_i)_{0\le i\le d}$
4: $(u'_{1\,i})_{0\le i\le d} \leftarrow \mathsf{MaskedSampleC}_l((c'_{1\,i})_{0\le i\le d})$
5: $(\mathbf{u}_i)_{0\le i\le d} \leftarrow \begin{pmatrix}(u'_{1\,i})_{0\le i\le d} + (x_i)_{0\le i\le d}\\(u'_{2\,i})_{0\le i\le d}\end{pmatrix}$
6: **return** $(\mathbf{u}_i)_{0\le i\le d}$

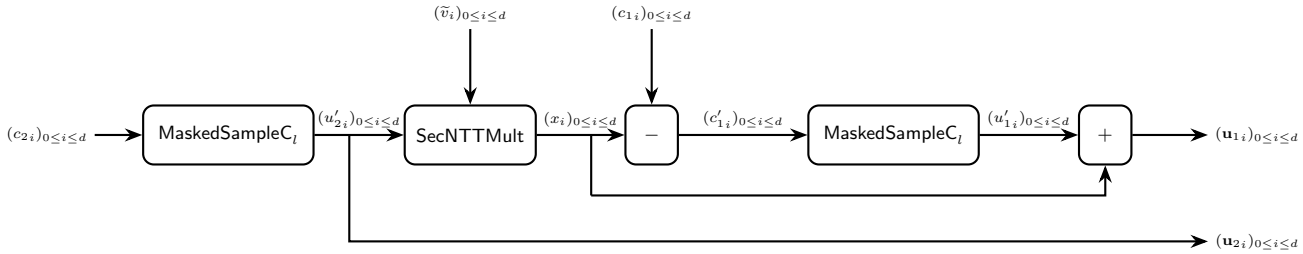

Fig. 2. Structure of the masked on-line sampling.

Lemma 11. *Assuming the properties of Table 6, the algorithm* MaskedOnlineSampling *(Alg. 10) is d-NI.*

Proof. We consider that the attacker made $\delta \le d$ observations during the execution of MaskedOnlineSampling. In the following, we prove that all these δ observations can be perfectly simulated with at most δ shares of $(\widetilde{v}_i)_{0\le i\le d}$, $(c_{1\,i})_{0\le i\le d}$ and $(c_{2i})_{0\le i\le d}$.
We consider the following distribution of the attacker's δ observations: δ_1 made during the first call to $\mathsf{MaskedSampleC}_l$, δ_2 made during the call to SecNTTMult, δ_3 made during the subtraction, δ_4 made during the second call to $\mathsf{MaskedSampleC}_l$, and δ_5 made during the final addition. We have

$$\sum_{i=1}^{5} \delta_i \le \delta \le d.$$

We build the proof classically from right to left. Since the addition is linear with respect to the arithmetical masking type, the final step is d-NI. It is also an affine gadget. In other words, each observation can be simulated with exactly either one share of $(x_i)_{0\le i\le d}$ or one share of $(u_{1\,i})_{0\le i\le d}$. Thus, all the observations from its call can be simulated with at most δ_5 shares among all the shares of $(x_i)_{0\le i\le d}$ and $(u_{1\,i})_{0\le i\le d}$. More precisely, all the observations from its call can be simulated with δ_5^1 shares of $(x_i)_{0\le i\le d}$ and δ_5^2 shares of $(u_{1\,i})_{0\le i\le d}$ such that $\delta_5^1+\delta_5^2 = \delta_5$. By Table 6 properties, $\mathsf{MaskedSampleC}_l$ is d-NI. Hence, all the observations from its call can be simulated with $\delta_5^1 + \delta_4$ shares of $(c'_{1\,i})_{0\le i\le d}$ and δ_5^2 shared of $(x_i)_{0\le i\le d}$. Next, the subtraction is also a linear operation (we can ignore its affine property here). Thus, all the observations from its call can be simulated with $\delta_5^1 + \delta_4 + \delta_3$ shares of $(c_{1\,i})_{0\le i\le d}$ and $\delta_5^1 + \delta_5^2 + \delta_4 + \delta_3 = \delta_5 + \delta_4 + \delta_3$ shares of $(x_i)_{0\le i\le d}$. Still by Table 6 properties, SecNTTMult is d-NI. Hence, the observations from its call can be simulated with $\delta_5 + \delta_4 + \delta_3 + \delta_2$ shares of $(u'_{2\,i})_{0\le i\le d}$ and $(\widetilde{v}_i)_{0\le i\le d}$ (and still $\delta_5^1 + \delta_4 + \delta_3$ shares of $(c_{1\,i})_{0\le i\le d}$). Finally, by the d-NI property of $\mathsf{MaskedSampleC}_l$, the observations from the whole algorithm can be simulated with

- $\delta_5 + \delta_4 + \delta_3 + \delta_2 + \delta_1 \leq d$ shares of $(c_{2i})_{0 \leq i \leq d}$;
- $\delta_5 + \delta_4 + \delta_3 + \delta_2 \leq d$ shares of $(\widetilde{v}_i)_{0 \leq i \leq d}$;
- $\delta_5^1 + \delta_4 + \delta_3 \leq \delta_5 + \delta_4 + \delta_3 \leq d$ shares of $(c_{1i})_{0 \leq i \leq d}$;

which concludes the proof. □

6.1 Gadgets

Known gadgets For the masking of the new gadgets such as $\mathsf{MaskedSampleC}_l$ and $\mathsf{MaskedSampleC}_1$, we use several sub-gadgets used in the literature. We review them below.

- The SecAdd gadget performs an addition in $\mathbb{Z}$ of two Boolean masked inputs. It was inspired from the Kogge-Stone adder of [CGTV15] and proved d-NI secure in [BBE+18].
- The SecMult gadget computes the multiplication of two Boolean masked inputs. It is one of the key building blocks of masking theory and has been introduced in [ISW03, RP10] and proved d-SNI in [BBD+16].
- Masking conversions: A2B and B2A allow to convert an arithmetical masking modulo Q^{mask} to a Boolean masking and vice versa. They were first introduced in [CGV14] in a d-SNI secure way. Then, they were generalized and improved in [Cor17, BBE+18, SPOG19].
- The $\mathsf{MaskedCDT}_{r_1,c}$ gadget generates a sample in a Boolean masked form that follows a tabulated Gaussian distribution of center c and of width r_1. The table values are not sensitive so they are the same as for the unmasked implementation. This masked CDT algorithm was introduced in [BBE+19, GR19] and proved d-NI. In these references, the output is in arithmetical masked form but one can easily change the masking type of the output by removing the last Boolean to arithmetic conversion.
- The $\mathsf{MaskedRR}_k$ corresponds to the masked computation of $\mathsf{RR}_k(a)$ where a is in Boolean masked form and k is a public unmasked value ($k = 30$). Recall that $\mathsf{RR}_k(a) = \lfloor 2^k a \rfloor / 2^k + \mathcal{B}_\alpha$ where $\mathcal{B}_\alpha$ is a Bernoulli variable with parameter $\alpha = 2^k a \bmod 1$. We can define the masking algorithm from known building blocks. First, the rounding operation has been introduced and proved d-NI secure in [GR19]. And, for the Bernouilli, one can refresh the masking of a, compute $2^k (a_i)_{0 \leq i \leq d} \bmod 1$ and compare it to a uniform masked randomness. Depending on the comparison, a sharing of 0 or 1 can be added to obtained a masked value for $\mathsf{RR}_k(a)$.

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 $\{a'_i\}$ and $\{b'_i\} \in \mathcal{R}_n^{d+1}$, we want to compute $\{c'_i\} \in \mathcal{R}_n^{d+1}$ such that $\sum_{i=0}^d c'_i = \left(\sum_{i=0}^d a'_i\right) \cdot \left(\sum_{i=0}^d b'_i\right)$. To perform this masked polynomial multiplication, we propose to rely on an NTT-based[2] multiplication. Using NTT, the product of two polynomials $a, b \in \mathbb{Z}_{\mathsf{Q}^{\text{mask}}}[x]/(x^n + 1)$ is given by

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

with $\circ$ the *coefficient-wise* product between two vectors in $\mathbb{Z}_{\mathsf{Q}^{\text{mask}}}^n$. 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 $\mathbb{Z}_{\mathsf{Q}^{\text{mask}}}$ using the technique of [ISW03]. While a naive multiplication algorithm would require n^2 ISW multiplications, we only need n of them. Since we want to multiply the polynomials in $\mathbb{Z}$ and not in $\mathbb{Z}_{\mathsf{Q}^{\text{mask}}}$, 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 Q^{mask} with CRT techniques to reduce the size.

[2] The reason why we do not use a standard DFT to multiply polynomials with integer coefficients is to avoid reintroducing FPA

Let us define SecNTTMult, the masked product of two polynomials $(a_i)_{0\leq i\leq d}, (b_i)_{0\leq i\leq d}$ arithmetically masked in $\mathbb{Z}_{\mathbf{Q}^{\text{mask}}}[x]/(x^n+1)$ by

$$\mathsf{NTT}^{-1}((\mathsf{SecMult}(\mathsf{NTT}((a_i)_{0\leq i\leq d})_j, \mathsf{NTT}((b_i)_{0\leq i\leq d})_j)_{0\leq j\leq n}).$$

By linearity of the NTT with arithmetical masking, we can easily conclude with the following Lemma.

Algorithm 11 SecNTTMult

Input: Arithmetical maskings $(a_i)_{0\leq i\leq d}$ and $(b_i)_{0\leq i\leq d}$ of $a, b \in \mathbb{Z}_q[x]/(x^n+1)$
Output: An arithmetical masking $(c_i)_{0\leq i\leq d}$ of $c \in \mathbb{Z}_q[x]/(x^n+1)$ such that $c = a \cdot b$.
1: $(\hat{a}_i)_{0\leq i\leq d} \leftarrow \mathsf{NTT}((a_i)_{0\leq i\leq d})$ {Apply NTT on each share independently}
2: $(\hat{b}_i)_{0\leq i\leq d} \leftarrow \mathsf{NTT}((b_i)_{0\leq i\leq d})$
3: **for** $j := 1$ to n **do**
4: $\quad(\hat{c}_i)_{0\leq i\leq d} \leftarrow \mathsf{SecMult}((\hat{a}[j]_i)_{0\leq i\leq d}, (\hat{b}[j]_i)_{0\leq i\leq d})$
5: **end for**
6: $(c_i)_{0\leq i\leq d} \leftarrow \mathsf{NTT}^{-1}((\hat{c}_i)_{0\leq i\leq d})$
7: **return** $(c_i)_{0\leq i\leq d}$

Lemma 12. SecNTTMult *is d-NI secure.*

Gaussian sampling We present the masked version of Algorithms 8 and 7 below in Algorithms 12 and 13. Algorithm 13 is inspired from the masked CDT sampling from [BBE+19, GR19]. Both d-NI securities are proved in Lemmas 15 and 14.

Remark 13. In repercussion of Remark 9, the same can be performed in a masked form. Instead of computing line 5 of Algorithm 13 on the fly, one can generate many samples for each center off-line and store them in a buffer. That way, depending on the context, this off-line generation may even be performed in an unmasked way. The only drawback of this method in a masked way is the fact that instead of consuming l elements of the buffer, the masked version needs to consume $l \cdot \beta$ masked elements of the buffer.

Algorithm 12 MaskedSampleC$_l$

Input: An arithmetical masking $(c_i)_{0\leq i\leq d}$ with $c \in \frac{1}{pq}\mathbb{Z}$ or $\frac{1}{pL}\mathbb{Z}$
Output: An arithmetical masking of an element $(z_i)_{0\leq i\leq d}$ following $D_{\mathbb{Z},r,c^{\text{frac}}/\beta^l}$.
1: $(c_i)_{0\leq i\leq d} \leftarrow \mathsf{A2B}((c_i)_{0\leq i\leq d})$ {We perform all this algorithm with Boolean masking}
2: $({c^{\text{rr}}}_i)_{0\leq i\leq d} \leftarrow \mathsf{MaskedRR}_{30}((c_i)_{0\leq i\leq d}) \cdot \beta^l$
3: $c^{\text{frac}} \leftarrow ((\beta^l - 1) \wedge {c^{\text{rr}}}_i)_{0\leq i\leq d}$
4: $c^{\text{int}} \leftarrow ({c^{\text{rr}}}_i)_{0\leq i\leq d} \gg \log_2(\beta^l)$
5: **for** $j := 1$ to l **do**
6: $\quad({c^{\text{frac}}}_i)_{0\leq i\leq d} \leftarrow \mathsf{MaskedSampleC}_1(({c^{\text{frac}}}_i)_{0\leq i\leq d})$
7: **end for**
8: $(z_i)_{0\leq i\leq d} \leftarrow \mathsf{B2A}(({c^{\text{frac}}}_i)_{0\leq i\leq d})$
9: **return** $(z_i)_{0\leq i\leq d}$ {Going back to arithmetical masking}

Lemma 14. MaskedSampleC$_1$ *is d-NI secure.*

Proof. A graphical representation of MaskedSampleC$_1$ is presented in Figure 3.

Algorithm 13 $\mathsf{MaskedSampleC}_1(c)$

Input: A Boolean masking $({c^{\text{frac}}}_i)_{0\le i\le d}$ of $c^{\text{frac}} \in \mathbb{Z}$.
Output: A Boolean of an element $(z_i)_{0\le i\le d}$ following $D_{\mathbb{Z},r_1,c^{\text{frac}}/\beta}$.

1: $(z_i)_{0\le i\le d} := (0,\dots,0)$
2: $({c^0}_i)_{0\le i\le d} := ((\beta-1) \wedge {c^{\text{frac}}}_i)_{0\le i\le d}$
3: $({c^1}_i)_{0\le i\le d} := ({c^{\text{frac}}}_i)_{0\le i\le d} \gg \log_2(\beta)$ $\{c^{\text{frac}} = c^0 + \beta \times c^1\ \}$
4: **for** $0 \le j < \beta$ **do**
5: $(y_i)_{0\le i\le d} \leftarrow \mathsf{MaskedCDT}_{r_1,j/\beta}()$
6: initialize $(J_i)_{0\le i\le d}$ as a $(\log_2(\beta)+1)$-bit Boolean masking of $-j$
7: $(\delta_i)_{0\le i\le d} \leftarrow \mathsf{SecAdd}\big(({c^0}_i)_{0\le i\le d}, (J_i)_{0\le i\le d}\big)$ $\{\delta = 0 \iff c = j$ and 1 otherwise$\}$
8: $(b_i)_{0\le i\le d} \leftarrow \mathsf{SecMult}\,(\neg(\delta_i)_{0\le i\le d}, (y_i)_{0\le i\le d})$ $\{b = y \iff c = j$ and 0 otherwise$\}$
9: $(z_i)_{0\le i\le d} := (b_i)_{0\le i\le d} \oplus (z_i)_{0\le i\le d}$
10: **end for**
11: $(z_i)_{0\le i\le d} \leftarrow \mathsf{SecAdd}\big((z_i)_{0\le i\le d}, ({c^1}_i)_{0\le i\le d}\big)$
12: **return** $(z_i)_{0\le i\le d}$

– First, let us prove that each iteration has a "weak" d-SNI property: any set of at most d observations whose d_{int} observations on the internal data and d_{out} observations on the outputs can be perfectly simulated from at most d_{int} shares of the input $({c^0}_i)_{0\le i\le d}$ and $d_{int}+d_{out}$ shares of $(z_i)_{0\le i\le d}$. The result comes from the structure of this sub-gadget: the dependencies of the observations on the returned values are stopped by the d-SNI property of SecMult.
– Similarly to the proof of Lemma 11, we consider that the attacker made $\delta \le d$ observations during the execution of $\mathsf{MaskedSampleC}_1$. We aim at proving that all these δ observations can be perfectly simulated with at most δ shares of $({c^{\text{frac}}}_i)_{0\le i\le d}$.
We consider the following distribution of the attacker's δ observations: δ_1 made during the shift $\gg$ in line 3, δ_2 made during the $\wedge$ in line 2, δ_3^0 made during the first iteration (considered as a black box), δ_3^j made during the j-th iteration, and δ_4 made during the call to SecAdd. We have

$$\delta_1 + \delta_2 + \sum_{0=1}^{\beta} \delta_3^j + \delta_4 \le \delta \le d.$$

Let us build the proof from right to left. Since the SecAdd gadget is d-NI, all the observations from its call can be perfectly simulated with at most δ_4 shares of the last computed $(z_i)_{0\le i\le d}$ and $({c^1}_i)_{0\le i\le d}$. Next, the last iteration has the "weak" d-SNI property showed in the first item, thus, all the observations from its call can be simulated with at most $\delta_3^\beta + \delta_4$ shares of $(z_i)_{0\le i\le d}$ and δ_3^β shares of $({c^0}_i)_{0\le i\le d}$. Recursively, all the observations from the first call to iter can be simulated with at most $\sum_{0=1}^{\beta}\delta_3^j$ shares of $({c^0}_i)_{0\le i\le d}$ and $\delta_4 + \sum_{0=1}^{\beta}\delta_3^j$ shares of the first $(z_i)_{0\le i\le d}$, namely $(0,...,0)$. The linearity and affine properties of the $\gg$ and $\wedge$ gadgets allow to conclude that all the adversary's observations can be perfectly simulated with at most $(\delta_1+\delta_4) + (\delta_2 + \sum_{0=1}^{\beta}\delta_3^j) \le \delta \le d$ shares of $({c^{\text{frac}}}_i)_{0\le i\le d}$. This is enough to prove that $\mathsf{MaskedSampleC}_1$ is d-NI secure. □

Lemma 15. $\mathsf{MaskedSampleC}_l$ *is* d*-NI secure.*

Proof. We first note that the linearity and affine properties of the $\gg$ and $\wedge$ gadgets break the dependency cycle on $({c^{\text{rr}}}_i)_{0\le i\le d}$. Thus, this algorithm is a linear succession of d-NI gadgets without dependency cycle. This is enough to prove that $\mathsf{MaskedSampleC}_l$ is d-NI secure. □

References

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the sta-

tistical distance. Cryptology ePrint Archive, Report 2015/483, 2015. https://eprint.iacr.org/2015/483.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. pages 116–129, 2016.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. pages 354–384, 2018.

[BBE+19] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. pages 2147–2164, 2019.

[BAA+19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[BDE+18] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. pages 494–524, 2018.

[CPS+20] Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. Modfalcon: Compact signatures based on module-ntru lattices. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020*, pages 853–866. ACM, 2020.

[Cor17] Jean-Sébastien Coron. High-order conversion from Boolean to arithmetic masking. pages 93–114, 2017.

[CGTV15] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala. Conversion from arithmetic to Boolean masking with logarithmic complexity. pages 130–149, 2015.

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between Boolean and arithmetic masking of any order. pages 188–205, 2014.

[Duc13] Léo Ducas. *Signatures Fondées sur les Réseaux Euclidiens: Attaques, Analyses et Optimisations.* PhD thesis, PhD thesis, École Normale Supérieure Paris, 2013., 2013.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. pages 40–56, 2013.

[DGPY20] Léo Ducas, Steven Galbraith, Thomas Prest, and Yang Yu. Integral matrix gram root and lattice gaussian sampling without floats. pages 608–637, 2020.

[DLP14a] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. pages 22–41, 2014.

[DLP14b] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient Identity-Based Encryption over NTRU Lattices. In *ASIACRYPT 2014*, pages 22–41, 2014.

[DP16] Léo Ducas and Thomas Prest. Fast Fourier Orthogonalization. In *ISSAC 2016*, pages 191–198, 2016.

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. pages 1857–1874, 2017.

[FHK+] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. https://falcon-sign.info/.

[FKT+20] Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Key recovery from Gram-Schmidt norm leakage in hash-and-sign signatures over NTRU lattices. pages 34–63, 2020.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. pages 197–206, 2008.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. pages 530–547, 2012.

[GR19] François Gérard and Mélissa Rossi. *An Efficient and Provable Masked Implementation of qTESLA*, pages 74–91. 2019.

[HPRR20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. pages 53–71, 2020.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. pages 463–481, 2003.

[Kar16] Charles F. F. Karney. Sampling Exactly from the Normal Distribution. *ACM Trans. Math. Softw.*, 42(1):3:1–3:14, 2016.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA 2000*, pages 937–941, 2000.

[MHS+19] Sarah McCarthy, James Howe, Neil Smyth, Séamus Brannigan, and Máire O'Neill. BEARZ attack FALCON: implementation attacks with countermeasures on the FALCON signature scheme. In Mohammad S. Obaidat and Pierangela Samarati, editors, *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2: SECRYPT, Prague, Czech Republic, July 26-28, 2019*, pages 61–71. SciTePress, 2019.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MW17] Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. pages 455–485, 2017.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking Dilithium - efficient implementation and side-channel evaluation. pages 344–362, 2019.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. pages 271–288, 2006.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. pages 80–97, 2010.

[PP19] Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. In *PKC 2019*, pages 504–533, 2019.

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

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. pages 347–374, 2017.

[RBRC20a] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. Drop by drop you break the rock - exploiting generic vulnerabilities in lattice-based pke/kems using em-based physical attacks. *IACR Cryptol. ePrint Arch.*, 2020:549, 2020.

[RBRC20b] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery and key recovery attacks. *IACR Cryptol. ePrint Arch.*, 2020:1559, 2020.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. pages 413–427, 2010.

[SPOG19] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. pages 534–564, 2019.

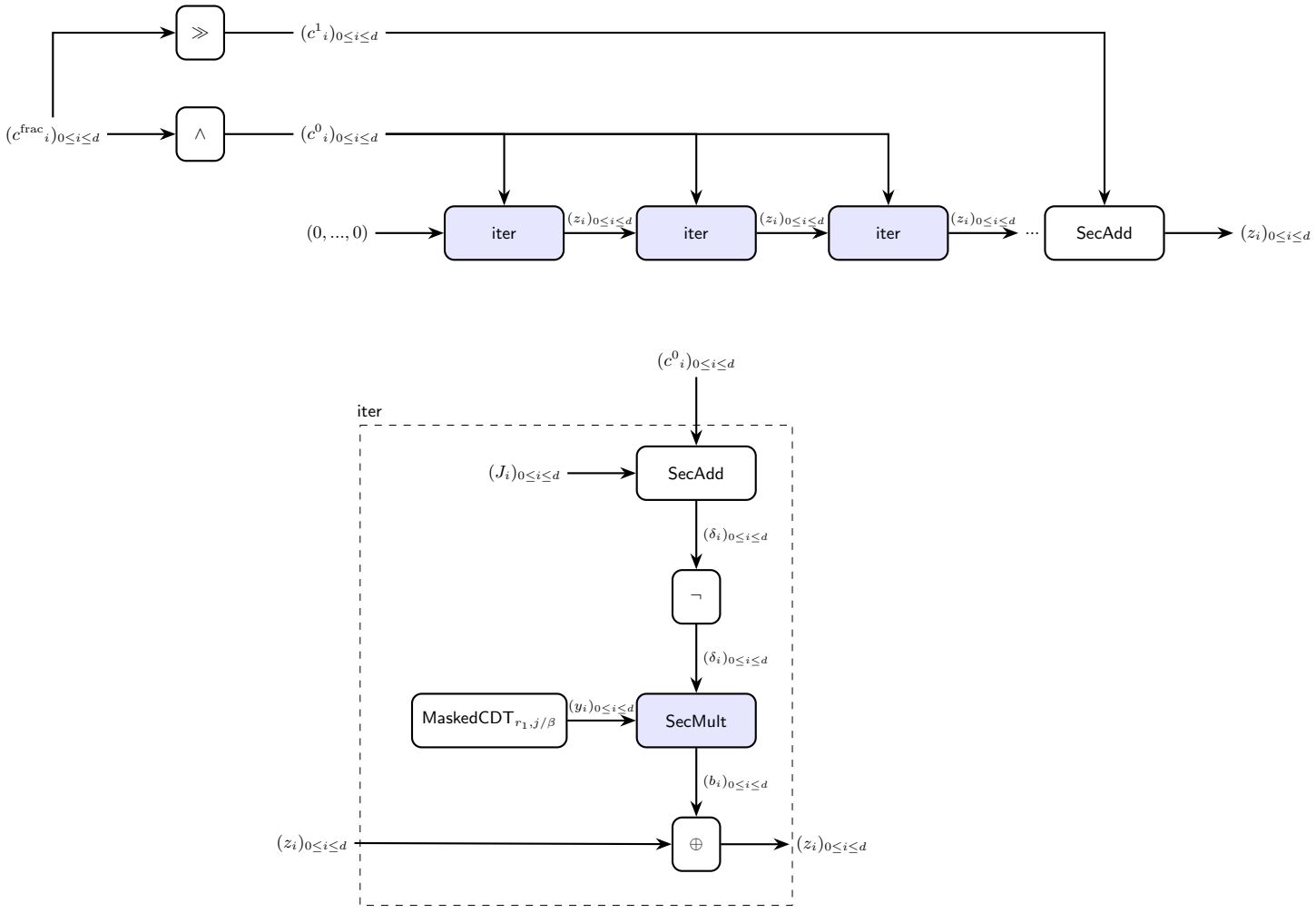

Fig. 3. Structure of MaskedSampleC$_1$. In blue, we represent the d-SNI gadgets while the white ones are the d-NI secure one.