Implementation of Lattice Trapdoors on Modules and Applications

Pauline Bert, Gautier Eberhart, Lucas Prabel, Adeline Roux-Langlois, Mohamed Sabt

Implementation of Lattice Trapdoors on Modules and Applications

Pauline Bert, Gautier Eberhart, Lucas Prabel,
Adeline Roux-Langlois, and Mohamed Sabt

Univ Rennes, CNRS, IRISA

Abstract. We develop and implement efficient Gaussian preimage sampling techniques on module lattices, which rely on the works of Micciancio and Peikert in 2012, and Micciancio and Genise in 2018. The main advantage of our implementation is its modularity, which makes it practical to use for signature schemes, but also for more advanced constructions using trapdoors such as identity-based encryption. In particular, it is easy to use in the ring or module setting, and to modify the arithmetic on $\mathcal{R}_q$ (as different schemes have different conditions on q).
Relying on these tools, we also present two instantiations and implementations of proven trapdoor-based signature schemes in the module setting: GPV in the random oracle model and a variant of it in the standard model presented in Bert *et al.* in 2018. For that last scheme, we address a security issue and correct obsolescence problems in their implementation by building ours from scratch. To the best of our knowledge, this is the first efficient implementation of a lattice-based signature scheme in the standard model. Relying on that last signature, we also present the implementation of a standard model IBE in the module setting. We show that while the resulting schemes may not be competitive with the most efficient NIST candidates, they are practical and run on a standard laptop in acceptable time, which paves the way for practical advanced trapdoor-based constructions.

Keywords: Lattice-based cryptography · trapdoors · Gaussian preimage sampling · module lattices · signature scheme · identity-based encryption

1 Introduction

Lattice-based cryptography is a viable candidate for possibly replacing number theoretic cryptography in the future. Hardness assumptions on lattices are conjecturally quantum resistant, whereas the discrete logarithm and factorization problems are known to be solvable in polynomial time in a quantum setting [Sho94]. Worst-case to average-case reductions from fundamental lattice problems (relaxations of NP-hard problems) also provide strong theoretical security guarantees for lattice-based primitives.

Although such constructions were quite inefficient in the early years of the field, the introduction of ideal lattices (or the ring setting) [PR06; SST+09;

LPR10], module lattices [LS15] and NTRU lattices [HPS98; SS13] led to constructions relying on lattices that possess a polynomial structure, effectively speeding up computations and reducing storage costs. On the practical side, much work has been put into improving the efficiency of polynomial multiplication [Sei18; AHH+19], Gaussian sampling over the integers [Kar16; MW17], and Gaussian preimage sampling [MP12; GM18]. Some schemes now have an efficiency comparable to their classical counterparts, but quasilinear in the security parameter, providing much better scalability and long-term security.

The NIST's post-quantum cryptography standardization process, which aims to select public-key encryption (PKE) schemes, key encapsulation mechanisms (KEM), and signatures, is now in its third round. In the PKE/KEM category, 3 out of 4 candidates are lattice-based, and 2 out of 3 for signatures, proving that cryptography based on lattices can be competitive in practice. Additionally, there are many advanced cryptographic constructions built on lattices, such as identity-based encryption [GPV08; ABB10b; BFRS18], attribute-based encryption [GVW13], group signature [LLL+13] or Fully Homomorphic Encryption.

Lattice-based signatures. The first direct constructions for proven lattice-based signatures were given in 2008. Gentry, Peikert, and Vaikuntanathan [GPV08] proposed a hash-and-sign signature scheme and proved its security in the Random Oracle Model (ROM). Lyubashevsky and Micciancio [LM08] constructed a one-time signature scheme in the standard model, and combined it with a tree structure to obtain an unrestricted signature scheme.

The GPV signature scheme [GPV08] was the first of a family of proven trapdoor-based signature schemes. The idea behind it is the following: the public key is a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ that defines the q-ary lattice $\Lambda_q^\perp(\boldsymbol{A}) = \{ \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \bmod q \}$, and the secret key is a trapdoor for $\boldsymbol{A}$ which is a short basis $\boldsymbol{T} \in \mathbb{Z}^{m \times m}$ of this lattice. To sign a message $M \in \{0,1\}^*$, the signer first hashes it to a vector $\boldsymbol{u} = \mathcal{H}(M) \in \mathbb{Z}_q^n$, and then computes a small preimage of $\boldsymbol{u}$ under the function $f_{\boldsymbol{A}} : \boldsymbol{x} \longmapsto \boldsymbol{A}\boldsymbol{x}$. This operation, known as Gaussian preimage sampling, is made possible by knowledge of the trapdoor: using $\boldsymbol{T}$, one can sample a vector $\boldsymbol{\nu} \in \mathbb{Z}^m$ following a narrow discrete Gaussian distribution and is such that $\boldsymbol{A}\boldsymbol{\nu} = \boldsymbol{u} \bmod q$. Verification simply consists in checking that $\boldsymbol{A}\boldsymbol{\nu} = \mathcal{H}(M) \bmod q$ and that $\boldsymbol{\nu}$ is sufficiently short. This scheme admits strong EU-CMA security in the ROM, under the hardness of the SIS problem.

This construction as such was never instantiated in practice because of its inefficiency, but several later improvements led to instantiations and implementations. First, Micciancio and Peikert [MP12] proposed a new notion of trapdoor, which was an improvement on short bases, and efficient associated algorithms in the case where the modulus q is a power of two. In [BB13], the authors implemented these techniques in both the unstructured and the ring settings. Then, Genise and Micciancio [GM18], using the same trapdoors, gave more efficient presampling algorithms in the ring setting and for an arbitrary modulus, which were later implemented in [GPR+18; GPR+19]. Finally, a notion of approximate trapdoors was introduced in [CGM19], enabling the inversion of the

one-way function $f_{\boldsymbol{A}}$ approximately rather than exactly, and leading to smaller parameters in concrete instantiations of signature schemes.

Even with these tools, lattice-based hash-and-sign signatures remain costly in practice, the primary bottlenecks being the generation of the trapdoor and Gaussian preimage sampling. Falcon [FHK+17], a lattice-based NIST candidate, is based on the same paradigm but still efficient in practice. It instantiates the GPV framework over NTRU lattices [SS13], using a Gaussian preimage sampler called fast Fourier sampling, itself derived from the Fast Fourier Orthogonalization algorithm [DP16]. Apart from being used to build signature schemes, lattice trapdoors have shown their utility by enabling many advanced constructions such as identity or attribute-based encryption [GPV08; ABB10b; GVW13] and group signature [LLL+13].

There are several direct constructions of lattice-based signatures in the standard model [CHK+10; Boy10; MP12], which are often similar to identity-based encryption schemes [CHK+10; ABB10b]. In these schemes, a message M is encoded into a lattice $\Lambda_q^\perp(\boldsymbol{A}_M)$, where $\boldsymbol{A}_M$ is a matrix that depends on the public key and M. Signing M then consists in sampling Gaussian preimages on $\Lambda_q^\perp(\boldsymbol{A}_M)$, similarly to [GPV08]. In [Boy10], $\boldsymbol{A}_M = [\boldsymbol{A} \mid \boldsymbol{A}_0 + \sum_i M_i \boldsymbol{A}_i]$, where the M_i are the bits of M, and the $\boldsymbol{A}_i$ are part of the public key. This results in very large public keys. In [BFRS18], $\boldsymbol{A}_M = \boldsymbol{A} + [\boldsymbol{0} \mid H(M)\boldsymbol{G}]$, where H is a function with a strong injectivity property and $\boldsymbol{G}$ the very structured gadget matrix of [MP12]. This yields much lighter public keys, and combines particularly well with the trapdoors from [MP12]. As far as we know, [BFRS18] provides the previously only implementation of a lattice-based standard model signature.

The concept of Identity Based Encryption (IBE) was defined by Shamir in [Sha84]. The first IBE constructions were based respectively on bilinear maps and on quadratic residue assumptions. The first supposedly post-quantum IBE scheme was introduced in [GPV08] and was based on hard lattice problems. It was then followed by many improvements [CHK+10; ABB10a; DLP14; Yam16]. Note that both [DLP14] and more recently [ZMS+21] provide an implementation of an IBE scheme based on NTRU lattices. We notice that a disadvantage of these schemes is that they additionally need the NTRU assumption.

Gaussian preimage sampling. Gaussian preimage sampling is a crucial operation and often the main bottleneck in trapdoor-based schemes, whether it be signature or more advanced constructions. It consists in sampling a vector from a discrete Gaussian distribution on the set $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{ \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q \}$, given $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, $\boldsymbol{u} \in \mathbb{Z}^n$, and a trapdoor $\boldsymbol{T}$ for the matrix $\boldsymbol{A}$. The result is then a preimage of $\boldsymbol{u}$ under the function $f_{\boldsymbol{A}} : \boldsymbol{x} \longmapsto \boldsymbol{A}\boldsymbol{x}$, hence the name.

In early constructions, the trapdoor $\boldsymbol{T} \in \mathbb{Z}^{m \times m}$ was a short basis of $\Lambda_q^\perp(\boldsymbol{A})$, and one would accomplish this task by using Klein's sampler [Kle00; GPV08], with the cost of having to compute the Gram-Schmidt orthogonalization of $\boldsymbol{T}$. Since the introduction of the trapdoors from [MP12], a more efficient method has been combining two complementary operations: G-sampling and perturbation sampling. The problem of efficient G-sampling, which consists in sampling from a spherical Gaussian on a very structured fixed lattice, was solved for a power-

of-two modulus in [MP12] and for an arbitrary modulus in [GM18]. Perturbation sampling, whose goal is to produce vectors following a discrete Gaussian on $\mathbb{Z}^m$ with a covariance that depends on $\boldsymbol{T}$, was made efficient in the ring setting in [GM18], but resorts to the generic Klein sampler in the unstructured setting.

Alternatively, fast Fourier sampling [DP16; FHK+17] follows the same ideas as the generic Klein sampler, but uses the so-called Fast Fourier Orthogonalization, linear algebra that preserves the underlying structure of the matrices in the ring setting, making it much faster in this case.

The module setting. The ring setting and ideal lattices [PR06; SST+09; LPR10], usually based on rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, are often the first choice for efficient lattice-based constructions. Module lattices [LS15], based on modules of the form $\mathcal{R}_q^d$, lie somewhere between ideal lattices and unstructured ones. Constructions in the module setting are (almost) as efficient as ring-based ones, and have other advantages for practical schemes.

Typically, module schemes fix a modulus q and a degree n for all parameter sets, and the security parameter is the rank d of the module. This leads to a more flexible choice of parameters, and potentially easier optimisation (since one only has to optimize arithmetic in the base ring $\mathcal{R}_q$ to obtain a faster arithmetic for all parameter sets). Additionally, fundamental problems on module lattices might not suffer from the same structural weaknesses as on ideal lattices (see [PHS19]). As an example of the interest of module lattices, we note that several NIST candidates at the post-quantum cryptography standardization process rely on them [DKL+18; DKR+18], and that a recent result [CPS+19] proposes a module variant of the Falcon signature scheme [FHK+17].

Our contribution. Our main contribution is the development and the implementation of efficient Gaussian preimage sampling techniques on module lattices. The main advantages of our implementation are its constant-timeness and its modularity, making it practical for both signature schemes and more advanced constructions using trapdoors. For instance, it can be used on rings or modules, with a different arithmetic over $\mathcal{R}_q$ depending on the choice of the parameter q. Relying on this, we also present two instantiations and constant-time implementations of proven signature schemes in the module setting (GPV in the ROM and one of its variant in the standard model) and the instantiation and implementation of a standard model IBE in the module setting. To the best of our knowledge, this is the first implementation of a secure lattice-based signature scheme in the standard model. Our resulting C implementation is public and open-source, available at `https://github.com/lucasprabel/module_gaussian_lattice`.

Preimage sampling. As mentioned above, Gaussian preimage sampling is a very important operation in trapdoor-based schemes, and to the best of our knowledge no methods adapted to module lattices existed previously. We develop efficient algorithms for trapdoor generation and Gaussian preimage sampling in the module setting, by generalizing existing tools in the unstructured and ring

settings [MP12; GM18]. Even if most of this adaptation is quite direct, it has to be done carefully to correctly work over modules. In particular, the perturbation sampling step does not directly adapt, and we resort to our own algorithm, using some subroutines from [GM18]. We also provide a detailed description of those algorithms and of the conditions needed to choose their parameters. This can be used as a building block for advanced trapdoor-based constructions, such as identity-based encryption, attribute-based encryption, or group signature.

Our implementation requires no external dependencies, and is easy to modify if needed. In particular, it is very modular and relies on several basic blocks that can be swapped out, as represented in Figure 1: the arithmetic over $\mathbb{Z}_q$ and $\mathcal{R}_q$, a pseudorandom number generator, and a (constant-time) sampler of discrete Gaussian distributions over $\mathbb{Z}$. For instance, we do not use the same arithmetic over $\mathcal{R}_q$ in our two signature schemes, as they need the ring $\mathcal{R}_q$ to have a different structure.

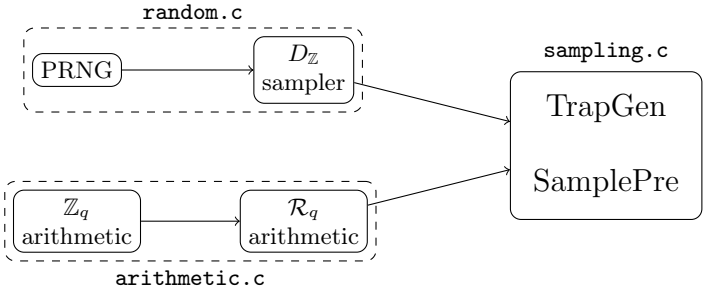

Fig. 1. Basic structure of our implementation and relationships between the blocks.

Table 1. Overview of the performances of our trapdoor tools and cost of sampling scalar Gaussians for $n = 256$ and $d = 4$.

Phase	TrapGen	SamplePre		
		Perturb. sampling	G-sampling	Global
Running time	36.56 ms	5.48 ms	8.28 ms	14.87 ms
Sampling $D_{\mathbb{Z}}$	74%	60%	84%	70%

In Table 1 we give an overview of the running times of our trapdoor algorithms on an Intel i7-8650U CPU running at 1.90 GHz. In particular, we highlight the proportion of time spent sampling Gaussians over $\mathbb{Z}$, and notice that having an efficient sampler is very important, since it makes up the largest part of the running times.

Applications. As an application, we propose an implementation of two trapdoor-based signature schemes and of an identity-based encryption scheme. The GPV signature is the simplest trapdoor-based scheme one can think of, since key generation is exactly the trapdoor generation algorithm, and signing essentially consists in Gaussian preimage sampling. As such, it makes for a natural way of evaluating trapdoor tools and techniques. Our second signature scheme, proven secure in the standard model, is a variation on GPV, and has been constructed by adapting the scheme from [BFRS18] to the module setting. The original construction was using an encoding function which should satisfy a strong injectivity property but does not in practice. We propose a construction for this encoding using a result of [LS18], which allows us to find invertible elements in R_q, and which needs a specific q as a consequence. Relying on this signature scheme, we also implement the standard model IBE scheme from [BFRS18], which was inspired by the IBE [ABB10a], in the module setting.

Our GPV implementation relies on our trapdoor tools, as well as a Number Theoretic Transform for fast multiplication in $\mathcal{R}_q$, adapted from CRYSTALS-Kyber [DKL$^+$18]. In our standard model schemes, the particular structure of the ring, due to the particular choice of q, is incompatible with the NTT. As such, the main difference with GPV in terms of implementation is the use of a partial NTT inspired by [LS18], instead of a full one. An example of performances of our signatures is given in Table 2. For this set of parameters, the public key has size 508kB, the private key 5.06MB and the signature 131kB.

Table 2. Performances of our signatures and comparison with previous GPV implementation (96-bit security parameter sets, lattice dimension 1024, modulus $q \approx 2^{30}$).

Scheme	KeyGen	Sign	Verify
GPV ([GPR$^+$19])	5.86 ms	32.42 ms	0.28 ms
GPV (this work)	8.94 ms	13.08 ms	0.29 ms
BFRS (this work)	9.46 ms	15.66 ms	1.19 ms

Comparison with previous works. From a theoretical point of view, the adaptation of the algorithms from [MP12; GM18] to the module setting is quite direct but has to be done carefully, in particular concerning the perturbation sampling algorithm which is an important step in those algorithms. This algorithm over rings is iteratively sampling vectors with a covariance matrix of dimension 2×2 over $\mathcal{R}$, whereas in our case, the matrix has size $2d \times 2d$, where d is the module rank. As a consequence, we have to decompose the covariance matrix into blocks of different sizes at each iteration instead of simply updating ring elements.

We chose to only compare our GPV implementation with the recent work of Gür *et al.* [GPR$^+$19], as it already outperforms previous implementations of Gaussian preimage sampling [BB13; GPR$^+$18] Again, we stress that one of the main advantages of our implementation compared to [GPR$^+$19] is its modularity rather than its performance.

We provide a new encoding function for the signature and the IBE schemes which allows to correct a security issue in the corresponding schemes in [BFRS18]. Our implementation does not rely on the BFRS one and then does not use the NFLlib library. We do not compare the original implementation of the BFRS scheme [BFRS18] with our corrected version, as the former's limited security would make the comparison irrelevant.

We also present a public and open-source implementation of a standard model IBE scheme in the module setting, relying on our standard model signature scheme, which represents also a contribution, given the low number of existing IBE implementations. In particular, our construction does not rely on the NTRU assumption as both implementations in [DLP14; ZMS+21].

Organization of the paper. This article focuses mainly on our implementation contribution, which we believe is the major contribution of the paper, but we also describe the Gaussian preimage sampling techniques on module lattices in Section 3. In Section 4, we explain our applications with two proven trapdoor-based signature schemes and a standard model IBE in the module setting. The theoretical part which led us to these implementations is presented and detailed in a rigorous way in the appendices.

Conclusion and open problems. Our results show that while the resulting schemes are not competitive with the most efficient NIST candidates (in particular the keys are quite large and probably not fit for embedded platforms), they are practical and run on a standard laptop in acceptable time (see Table 2), paving the way for practical advanced trapdoor-based constructions. Besides, the standard model security of our second scheme comes at a low additional cost compared to the ROM GPV signature.

We believe that our schemes performances can still be improved. In particular, the modularity of our implementation makes it easy to modify if needed. For instance, the use of another Gaussian sampler over integers could reduce our timings. Our results seem to confirm that using NTRU lattices provides much better results even if it requires an additionally NTRU assumption. Finally, an interesting open problem would be to study the impact of approximate trapdoors [CGM19] on IBE schemes, and possibly on more advanced schemes.

2 Preliminaries

Notations. We denote by $\mathbb{Z}$ and $\mathbb{Z}_q$ the rings of integers and integers modulo q, by $\mathbb{R}$ and $\mathbb{C}$ the fields of real and complex numbers. Elements of these sets are denoted by standard lowercase letters, (column) vectors by bold lowercase letters, and matrices by bold uppercase letters. The norm $\|\cdot\|$ denotes the euclidean norm, and the norm of a vector over $\mathbb{Z}_q$ is the norm of the corresponding vector over $\mathbb{Z}$ with entries in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$, the norm of a polynomial $a = \sum_{i=0}^{n-1} a_i X^i$ is the norm of the vector $(a_0, \ldots, a_{n-1})$, and the norm of a

matrix is the maximum norm of its column vectors. If x is sampled from a distribution D, we write $x \leftarrow D$. The uniform distribution over a finite set S is denoted $\mathcal{U}(S)$.

A symmetric matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is said to be positive definite (resp. positive semidefinite) if for all nonzero $\boldsymbol{x} \in \mathbb{R}^n$ we have $\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x} > 0$ (resp. $\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x} \geq 0$), in which case we write $\boldsymbol{M} \succ 0$ (resp. $\boldsymbol{M} \succeq 0$). Positive semidefiniteness induces a partial order on $\mathbb{R}^{n \times n}$, as we say that $\boldsymbol{M} \succeq \boldsymbol{N}$ when $\boldsymbol{M} - \boldsymbol{N} \succeq 0$. For simplicity of notation, we write $\boldsymbol{M} \succeq \eta$ instead of $\boldsymbol{M} \succeq \eta \boldsymbol{I}$, for a real $\eta \geq 0$.

We use the standard Landau notations. A function f is negligible when $f(n) = o(n^{-c})$ for all $c > 0$ as $n \to \infty$. An event happens with overwhelming probability if its probability of not happening is negligible. Two distributions D_0 and D_1 over the same countable domain Ω are said to be statistically indistinguishable if their statistical distance $\Delta(D_0, D_1) = \frac{1}{2} \sum_{\omega \in \Omega} |D_0(\omega) - D_1(\omega)|$ is negligible. They are computationally indistinguishable if no probabilistic polynomial time algorithm can distinguish them with non-negligible advantage.

Lattices and discrete Gaussian distributions. Given a set of linearly independent vectors $\boldsymbol{B} = \{ \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \} \subset \mathbb{R}^m$, the lattice with basis $\boldsymbol{B}$ is the set $\{ \sum_{i=1}^k \lambda_i \boldsymbol{b}_i, \lambda_i \in \mathbb{Z} \}$, and its rank is k. For $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{u} \in \mathbb{Z}_q^n$, we define the following m-dimensional q-ary lattice and its coset $\Lambda_q^{\perp}(\boldsymbol{A}) = \{ \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \bmod q \}$ and $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{ \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q \}$.

Module lattices are particular lattices that have a polynomial structure. We consider the ones that are based on the rings $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1 \rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$, where n is a power of two and q is prime. They are sublattices of the full lattice $\mathcal{R}^m$, itself isomorphic to the integer lattice $\mathbb{Z}^{nm}$.

The discrete Gaussian distribution of center $\boldsymbol{c} \in \mathbb{R}^n$ and parameter $\sigma > 0$ over a full-rank lattice $\Lambda \subset \mathbb{Z}^n$ is denoted $D_{\Lambda, \sigma, \boldsymbol{c}}$. It is the probability distribution over Λ such that each $\boldsymbol{x} \in \Lambda$ is assigned a probability proportional to $\rho_{\sigma, \boldsymbol{c}}(\boldsymbol{x}) = \exp(-\frac{\pi \|\boldsymbol{x} - \boldsymbol{c}\|^2}{\sigma^2})$. For a positive definite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, we also define the (skewed) density $\rho_{c, \sqrt{\boldsymbol{\Sigma}}}(\boldsymbol{x}) = \exp(-\pi(\boldsymbol{x} - \boldsymbol{c})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{c}))$, and the corresponding discrete Gaussian distribution of center $\boldsymbol{c}$ and covariance $\boldsymbol{\Sigma}$ denoted $D_{\Lambda, \sqrt{\boldsymbol{\Sigma}}, \boldsymbol{c}}$.

Smoothing parameter. The smoothing parameter $\eta_\varepsilon(\Lambda)$ of a lattice Λ was introduced in [MR07]. We use the following lemma to find a lower bound for it.

Lemma 1 ([GPV08, Lemma 3.1]). *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\boldsymbol{B}$, and $\tilde{\boldsymbol{B}}$ the Gram-Schmidt orthogonalization of $\boldsymbol{B}$. Then, for any $\varepsilon > 0$, we have $\eta_\varepsilon(\Lambda) \leq \|\tilde{\boldsymbol{B}}\| \cdot \sqrt{\ln(2n(1 + 1/\varepsilon))/\pi}$.*

Gaussian tailcut. We denote by t the tailcut of the discrete Gaussian of parameter σ. It is a positive number such that samples from $D_{\mathbb{Z}, \sigma}$ land outside of $[-t\sigma, t\sigma]$ only with negligible probability. We choose it using the fact that $\Pr_{x \leftarrow D_{\mathbb{Z}, \sigma}}[|x| > t\sigma] \leq \operatorname{erfc}\left(t/\sqrt{2}\right)$, where $\operatorname{erfc}(x) = 1 - \frac{2}{\pi} \int_0^x \exp^{-u^2} \, \mathrm{d}u$. This generalizes to higher dimensions using the following lemma.

Lemma 2 ([MR07, Lemma 4.4]). *For any n-dimensional lattice Λ, vector $\boldsymbol{c} \in \mathbb{R}^n$, reals $0 < \varepsilon < 1$ and $\sigma \geq \eta_\varepsilon(\Lambda)$, if x is distributed according to $D_{\Lambda,\sigma,\boldsymbol{c}}$, then we have* $\Pr\left[\|\boldsymbol{x} - \boldsymbol{c}\| > \sigma\sqrt{n}\right] \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}$.

Module hardness assumptions. As in most practical lattice-based constructions [ABB+19; ADP+16; BFRS18; DKL+18; FHK+17], we consider rings of the form $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, where n is a power of two and q a prime modulus. The polynomial $X^n + 1$ is the cyclotomic polynomial of order $2n$, and $\mathcal{R}$ is the corresponding cyclotomic ring.

The module variants generalizing Ring-SIS and Ring-LWE were introduced in [LS15]. The parameter d corresponds to the rank of the module, and nd is the dimension of the corresponding module lattice ($d = 1$ gives the ring problem). Their difficulty is proven by worst-case to average-case reductions from hard problems on module lattices [LS15].

Definition 1 (Module-SIS$_{n,d,m,q,\beta}$). *Given a uniformly random $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$, find a vector $\boldsymbol{x} \in \mathcal{R}^m$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \mod q$, and $0 < \|\boldsymbol{x}\| \leq \beta$.*

Definition 2 (Decision Module-LWE$_{n,d,q,\sigma}$). *Given a uniform $\boldsymbol{A} \in \mathcal{R}_q^{m\times d}$ and the vector $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e} \mod q$, where $\boldsymbol{s} \leftarrow \mathcal{U}(\mathcal{R}_q^d)$ and $\boldsymbol{e} \leftarrow D_{\mathcal{R}^m,\sigma}$, distinguish the distribution of $(\boldsymbol{A}, \boldsymbol{b})$ from the uniform distribution over $\mathcal{R}_q^{m\times d} \times \mathcal{R}_q^m$.*

3 Gaussian preimage sampling on module lattices

Efficient trapdoor-based schemes, including the two signatures and the IBE we implement, are based on the notion of trapdoor from [MP12]. These trapdoors are an improvement on the short bases of [GPV08], as they are more compact and enjoy faster algorithms, both asymptotically and in practice. They were generalized to ideal lattices in [LCC14], and an efficient instantiation of the associated algorithms was given in [GM18]. To the best of our knowledge, neither the trapdoors nor their algorithms had been adapted yet to the module setting.

In Section A, we generalize in detail these constructions to module lattices, following the ideas from [MP12], by accomplishing two goals:

- We derive an algorithm TrapGen from [MP12, Section 5.2], which is described in Section A.1. It generates a uniform matrix $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$ along with its trapdoor $\boldsymbol{T} \in \mathcal{R}^{2d\times dk}$, where $k = \lceil \log_b q \rceil$ and $m = d(k+2)$. The trapdoor $\boldsymbol{T}$ is sampled from a Gaussian distribution of parameter σ. The matrix $\boldsymbol{A}$ defines hard module SIS and ISIS problems.
- We give an algorithm SamplePre, described in Section A.4, that uses $\boldsymbol{T} \in \mathcal{R}^{2d\times dk}$ to perform efficient Gaussian preimage sampling with parameter ζ, effectively solving the module SIS and ISIS problems.

Gaussian preimage sampling consists in sampling from a spherical discrete Gaussian distribution on cosets of the lattice $\Lambda_q^\perp(\boldsymbol{A})$ (that is, the sets $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ for $\boldsymbol{u} \in \mathcal{R}^d$) using $\boldsymbol{T}$. The standard deviation ζ of this distribution should be small

(so that it is hard to sample from it without knowing $\boldsymbol{T}$), and the produced vectors should not leak any information about $\boldsymbol{T}$. To this end, we follow the method introduced in [MP12] where sampling from $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\zeta}$ is divided into two complementary phases:

- *G-sampling* of parameter α (described in Section A.2), which ensures that our samples actually lie in the good coset.
- *Perturbation sampling* with parameters ζ and α (described in Section 3.1), which conceals the information about $\boldsymbol{T}$ in the output distribution.

Most of these steps are direct adaptations of the original results, except the last one that we now explain more in detail.

3.1 Perturbation sampling

Perturbation sampling aims at sampling vectors following the Gaussian distribution over $\mathcal{R}^m$ of covariance $\boldsymbol{\Sigma}_p = \zeta^2\boldsymbol{I} - \alpha^2\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\left[\boldsymbol{T}^T\ \boldsymbol{I}\right]$. In a way, this covariance matrix is complementary to the one of $\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\boldsymbol{z}$, where $\boldsymbol{z}$ is the output of the G-sampling. This is so that when we sum the perturbation $\boldsymbol{p}$ and $\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\boldsymbol{z}$, the final covariance matrix $\boldsymbol{\Sigma}_p + \alpha^2\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\left[\boldsymbol{T}^T\ \boldsymbol{I}\right] = \zeta^2\boldsymbol{I}$ does not leak any information about the trapdoor $\boldsymbol{T}$.

Internally, perturbation sampling takes place in the ring $\mathcal{P} = \mathbb{R}[X]/\langle X^n + 1\rangle$ rather than the usual ring $\mathcal{R}$. As in most discrete Gaussian sampling algorithms, computations are done with real numbers even if the end result is composed of integers only. Since $\mathcal{R}$ can naturally be embedded in $\mathcal{P}$, we can consider $\boldsymbol{T}$ and covariance matrices to have entries in $\mathcal{P}$.

Genise and Micciancio made this operation efficient in the ring setting [GM18]. In particular, they describe an algorithm SampleFz which takes as input a covariance polynomial f and a center c, and returns a sample from the corresponding Gaussian distribution over $\mathcal{R}$. Their method cannot be applied directly to the module setting because of the additional rank module parameter d. Instead of having to sample vectors with a covariance matrix of dimension 2×2 over $\mathcal{R}$ and with a center $(c_0, c_1) \in \mathcal{R}^2$ as in [GM18], we have to work with a covariance matrix $\boldsymbol{\Sigma} \in \mathcal{P}^{2d\times 2d}$ and a center $\boldsymbol{c} \in \mathcal{P}^{2d}$. However, by using [GM18, Lemma 4.3] and the SampleFz algorithm, we wisely decompose the covariance matrices into blocks of different sizes at each iteration and update our center, allowing us to iteratively sample the perturbations $p_i \in \mathcal{R}$.

An efficient algorithm for sampling perturbations. We now give a description of the algorithm SamplePerturb which, given the trapdoor $\boldsymbol{T}$ and the Gaussian parameters ζ and α, returns a vector $\boldsymbol{p}$ sampled from the centered discrete Gaussian over $\mathcal{R}^m$ of covariance $\boldsymbol{\Sigma}_p = \zeta^2\boldsymbol{I} - \alpha^2\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\left[\boldsymbol{T}^T\ \boldsymbol{I}\right]$. This algorithm does not explicitly use $\boldsymbol{\Sigma}_p \in \mathcal{P}^{m\times m}$, but only a much smaller matrix $\boldsymbol{\Sigma} \in \mathcal{P}^{2d\times 2d}$, which can be computed in advance. It uses the algorithm SampleFz [GM18, Section 4] to sample from discrete Gaussians over $\mathcal{R}$.

Algorithm 1 SamplePerturb($\boldsymbol{T}, \zeta, \alpha$) for sampling a perturbation vector

1: **function** SamplePerturb($\boldsymbol{T} \in \mathcal{P}^{2d \times dk}, \zeta > 0, \alpha > 0$)
2: $\quad \boldsymbol{p}_s \leftarrow D_{\mathcal{R}^{dk}, \zeta^2 - \alpha^2}$ $\quad \triangleright\ \boldsymbol{p}_s \in \mathcal{R}^{dk}$
3: $\quad \boldsymbol{c} \leftarrow -\frac{\alpha^2}{\zeta^2 - \alpha^2} \boldsymbol{T} \boldsymbol{p}_s$ $\quad \triangleright\ \boldsymbol{c} \in \mathcal{P}^{2d}$
4: $\quad \boldsymbol{\Sigma} \leftarrow \zeta^2 \boldsymbol{I} - (\alpha^{-2} - \zeta^{-2})^{-1} \boldsymbol{T} \boldsymbol{T}^T$ $\quad \triangleright\ \boldsymbol{\Sigma} \in \mathcal{P}^{2d \times 2d}$
5: $\quad$ **for** $i = 2d - 1 \dots 0$ **do**
6: $\quad\quad \boldsymbol{\Sigma} = \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{B}^T & f \end{array}\right]$ $\quad \triangleright\ \boldsymbol{A} \in \mathcal{P}^{i \times i}, \boldsymbol{B} \in \mathcal{P}^{i \times 1}, f \in \mathcal{P}$
7: $\quad\quad \boldsymbol{c} = (\boldsymbol{c}', c_i)$ $\quad \triangleright\ \boldsymbol{c}' \in \mathcal{P}^i,\ c_i \in \mathcal{P}$
8: $\quad\quad p_i \leftarrow D_{\mathcal{R}, \sqrt{f}, c_i}$ $\quad \triangleright\ p_i \in \mathcal{R}$
9: $\quad\quad \boldsymbol{c} \leftarrow \boldsymbol{c}' + f^{-1} \boldsymbol{B} (p_i - c_i)$ $\quad \triangleright\ \boldsymbol{c} \in \mathcal{P}^i$
10: $\quad\quad \boldsymbol{\Sigma} \leftarrow \boldsymbol{A} - f^{-1} \boldsymbol{B} \boldsymbol{B}^T$ $\quad \triangleright\ \boldsymbol{\Sigma} \in \mathcal{P}^{i \times i}$
11: $\quad \boldsymbol{p} \leftarrow (p_0, \dots, p_{2d-1}, \boldsymbol{p}_s)$ $\quad \triangleright\ \boldsymbol{p} \in \mathcal{R}^m$
12: $\quad$ **return** $\boldsymbol{p}$

Note that in lines 6 and 7 of Algorithm 1, no computation is actually performed: different parts of the variables $\boldsymbol{\Sigma}$ and $\boldsymbol{c}$ are just given names, for a clearer understanding.

Algorithm 1 has a complexity of $\Theta(d^2 n \log n)$ scalar operations, if we ignore the updates to $\boldsymbol{\Sigma}$ (which only depend on $\boldsymbol{T}$ and can actually be precomputed in $\Theta(d^3 n \log n)$ in the trapdoor generation). This stems from the fact that multiplication in $\mathcal{P}$ and SampleFz both take $\Theta(n \log n)$ time.

The correctness of this algorithm is proven by the following Theorem.

Theorem 1. *Let* $\boldsymbol{T} \in \mathcal{P}^{2d \times dk}$, $\zeta, \alpha > 0$, *and* $\boldsymbol{\Sigma}_p = \zeta^2 \boldsymbol{I} - \alpha^2 \left[\begin{smallmatrix} \boldsymbol{T} \\ \boldsymbol{I} \end{smallmatrix}\right] \left[\begin{smallmatrix} \boldsymbol{T}^T & \boldsymbol{I} \end{smallmatrix}\right] \in \mathcal{P}^{m \times m}$ *be the derived perturbation covariance matrix.*

If $\boldsymbol{\Sigma}_p \succeq \eta_\varepsilon^2(\mathbb{Z}^{nm})$, *then* SamplePerturb($\boldsymbol{T}, \zeta, \alpha$) *returns a vector* $\boldsymbol{p} \in \mathcal{R}^m$ *whose distribution is statistically indistinguishable from* $D_{\mathcal{R}^m, \sqrt{\boldsymbol{\Sigma}_p}}$.

We provide more details about this algorithm (in particular how transposition over $\mathcal{P}$ is defined) and a proof of correctness in Section A.3.

3.2 Implementation

To generate our specific discrete Gaussian distributions, we make use of the following building blocks: the AES-based pseudorandom number generator from [MN17] (implemented using AES-NI instructions for x86 architectures), and a sampler of discrete Gaussians over $\mathbb{Z}$ similar to Karney's sampler [Kar16]. We chose this sampler as it can generate samples in constant time, independently of the center, Gaussian parameter, and output value. All of the computations that deal with non-integers are carried out with floating-point operations that do not involve subnormal numbers.

Our implementation of trapdoor generation and G-sampling are quite direct from the description of the algorithms, and do not have any peculiarities. As such, we will focus our explanations on the techniques used to optimize SamplePerturb.

To obtain an efficient arithmetic in $\mathcal{P} = \mathbb{R}[X]/\langle X^n + 1\rangle$ we used the Chinese Remainder Transform (CRT, as defined in [LPR13]), as in several other works [DP16; GM18; GPR+18]. It is a kind of fast Fourier transform that evaluates a polynomial $f \in \mathcal{P}$ at the complex primitive $2n^{\text{th}}$ roots of unity, the n points of the form $\omega_i = e^{\frac{ki\pi}{n}}$ for $i \in \{1, 3, \ldots, 2n-1\}$, in time $\Theta(n \log n)$. As explained in [GM18, Section 4.1], this CRT transform combines especially well with the algorithm SampleFz whose recursive structure is similar to that of an FFT.

Also, the matrix $\boldsymbol{\Sigma}$ is not actually updated during a run of SamplePerturb. Instead, we precompute (during the trapdoor generation) all of the $2d$ values that it would take during the execution of the algorithm, and store them in a single $2d \times 2d$ triangular matrix by "stacking" them. This is possible because at each iteration of the loop, $\boldsymbol{\Sigma}$ is an $i \times i$ matrix of which we only use the last line and column. This comes with an additional storage cost of $d(2d+1)$ elements from $\mathcal{P}$ in the secret key, and Table 3 quantifies the time gains in practice.

Our implementation is constant-time, assuming the compiler produces constant-time code for reduction modulo q and basic operations such as integer division and multiplication. Indeed, our algorithms do not require branching nor memory access that depend on secret values. In particular, our sampler of discrete Gaussians over $\mathbb{Z}$'s running time is independent of both the input parameters and the output value.

3.3 Performances

We now present running times for our trapdoor generation and preimage sampling algorithms, and the cost of their different components. Our experimentations were carried out with $n = 256$, $k = \lceil \log_b q \rceil = 30$ (the values used in our signature schemes in Sections 4), and values of d up to 10. We ran them on an Intel i7-8650U CPU running at 1.90 GHz.

In Table 3, we see how the trapdoor generation is divided into three main operations: sampling from $D_{\mathbb{Z},\sigma}$ for the construction of $\boldsymbol{T}$, the precomputations concerning the covariance matrices (see Section 3.2), and arithmetic, which is mainly computing the matrix product.

Table 4 concerns the algorithm SamplePre. We also measured that sampling from discrete Gaussians over $\mathbb{Z}$ constitutes 57-64% of the perturbation sampling (decreasing with d) and about 85% of the G-sampling, for a total of 67-72% of the whole presampling. Gaussian sampling over $\mathbb{Z}$ makes up most of the running times of both TrapGen and SamplePre. As such, it is important for efficiency to use a fast sampler of discrete Gaussians over $\mathbb{Z}$ as a building block. We remind the reader that in our implementation, this sampler can easily be swapped out for another if needed.

4 Applications

4.1 The GPV signature scheme on modules

A direct application of our Gaussian preimage sampling techniques on module lattices is the GPV signature [GPV08] in the module setting. It was originally

Table 3. Running time of the TrapGen algorithm.

d	4	6	8	10
$D_{\mathbb{Z},\sigma}$ sampling	27.17 ms (74%)	56.37 ms (72%)	100.22 ms (67%)	159.10 ms (64%)
$\boldsymbol{\Sigma}$ computations	7.03 ms (19%)	17.11 ms (22%)	39.40 ms (26%)	71.92 ms (29%)
Arithmetic	2.34 ms (6%)	1.11 ms (1%)	2.60 ms (2%)	5.25 ms (2%)
Total	36.56 ms	78.52 ms	149.57 ms	248.09 ms

Table 4. Running time of the SamplePre algorithm.

d	4	6	8	10
Perturb. sampling	4.73 ms (36%)	6.63 ms (38%)	9.43 ms (38%)	12.03 ms (39%)
G-sampling	7.48 ms (57%)	9.83 ms (56%)	13.29 ms (54%)	16.43 ms (53%)
Arithmetic	0.82 ms (6%)	1.20 ms (7%)	1.98 ms (8%)	2.64 ms (8%)
Total	13.28 ms	17.66 ms	24.70 ms	31.10 ms

formulated on unstructured lattices, and has previously been implemented using improved trapdoors and algorithms [MP12; GM18] in the ring setting [BB13; GPR$^+$18; GPR$^+$19].

You can refer to the Section B in appendices to see a description of how we instantiate it in the module setting, using the Gaussian preimage sampling tools from [MP12; GM18] that we extended to module lattices. Our goal here is not to obtain a competitive signature scheme, but rather to show the relevance of the tools we developed.

Estimating security and choosing parameters. In Table 5, we propose four parameter sets and corresponding security estimates, taking the prime modulus $q = 1073738753$ of bitsize $k = 30$. The sets I and IV corresponds to the ring setting, where n is a power of two and $d = 1$. The sets II and III are intermediate using the module setting. We describe how we chose those parameters, estimating the difficulty of the underlying lattice problems in Section B.

Performance and comparison with previous work. We now present in Table 6 the running times for our implementation of the GPV signature scheme. While it is practical and runs on a standard laptop in acceptable time, the comparison with lattice-based NIST candidates given in Table 12, Section E shows that it is not competitive.

Comparison between rings and modules. As already explained, one goal of using a module variant instead of a ring variant is to be more flexible in the parameters. The comparison between the different levels of security shows that the running time for signing and verifying is increasing with nd, and then that having intermediate levels allow to be faster to sign and verify.

On the other hand, the KeyGen algorithm does not depend only on nd but is slower for larger d. We give a more concrete example of this in Table 7. When nd

Table 5. Suggested parameter sets for our instantiation of the GPV signature.

Parameter set	I	II	III	IV
nd	1024	1280	1536	2048
n	1024	256	512	2048
k	30	30	30	30
d	1	5	3	1
σ	7.00	5.55	6.15	6.85
α	48.34	54.35	60.50	67.40
ζ	83832	83290	112522	160778
BKZ blocksize b to break LWE	367	478	614	896
Classical security	107	139	179	262
Quantum security	97	126	163	237
BKZ blocksize b to break SIS	364	482	583	792
Classical security	106	140	170	231
Quantum security	96	127	154	210

Table 6. Performances of our GPV signature.

Parameter set	KeyGen	Sign	Verify
I	7.48 ms	11.47 ms	0.73 ms
II	51.34 ms	15.25 ms	1 ms
III	36.49 ms	17.45 ms	1.12 ms
IV	15.55 ms	22.64 ms	1.48 ms

is constant, so is the estimated security provided. With a higher n and a lower d ($d = 1$ being the ring setting), the underlying lattices have a stronger structure and the signature is more efficient. With a lower n and a higher d (the extreme being $n = 1$ in the unstructured setting), the lattices have less structure, leading to increased flexibility at the cost of efficiency.

Table 7. Cost of KeyGen, Sign and Verify depending of the parameter d for $nd = 1024$.

(n, d)	KeyGen	Sign	Verify
$(1024, 1)$	7.62 + 1.32 = 8.94 ms	13.08 ms	0.79 ms
$(512, 2)$	15.32 + 2.81 = 18.13 ms	13.20 ms	0.79 ms
$(256, 4)$	29.53 + 7.03 = 36.56 ms	13.36 ms	0.74 ms

Comparison with [GPR+19]. In Table 8, we compare our timings with the work of [GPR+19]. Their parameter set where $(n, k) = (1024, 27)$ is compared with ours where $(nd, k) = (1024, 30)$, which provide approximately the same security.

Table 8. Comparison of GPV implementations.

Implementation		KeyGen	Sign	Verify
[GPR+19]	$n = 1024$	5.86 ms	32.42 ms	0.28 ms
This work	$(n, d) = (1024, 1)$	7.62 + 1.32 = 8.94 ms	13.08 ms	0.79 ms

4.2 A standard model signature scheme on modules

The second application of our tools that we present is an implementation of a signature scheme that is proven secure in the standard model, as opposed to the GPV signature and the NIST schemes.

This scheme is the signature from [BFRS18], which is a variant of GPV, adapted to the module setting. For the security proof to hold, the encoding must fulfil a strong injectivity property. However, the original encoding described in [BFRS18] did not meet these requirements, leading to a limited security. We propose a modified version of this scheme: we translated it to the module setting, and instantiated it with a correct encoding.

We give a complete description of our scheme and state its correctness and security in Section C.

Encoding messages with full-rank differences. We first describe the notion of an encoding with full-rank differences (FRD) needed in our scheme. Note that this definition of FRD differs from the one used in [ABB10b], which does not use the MP12 trapdoors, and therefore does not need the $H(m)$ to be invertible.

Definition 3 (Adapted from [ABB10b]). *An* encoding with full-rank differences *from the set $\mathcal{M}$ to a ring $\mathcal{R}$ is a map $H : \mathcal{M} \longrightarrow \mathcal{R}$ such that:*

- *for any $m \in \mathcal{M}$, $H(m)$ is invertible,*
- *for any $m_1, m_2 \in \mathcal{M}$ such that $m_1 \neq m_2$, $H(m_1) - H(m_2)$ is invertible,*
- *H is computable in polynomial time.*

Before constructing an FRD encoding in the module setting (that is, taking values in $\mathcal{R}_q^{d\times d}$), we first construct one in the ring setting (taking values in $\mathcal{R}_q$). Our construction is based on the following result of [LS18], which allows us to find invertible elements in R_q.

Theorem 2 ([LS18, Corollary 1.2]). *Let $n \geq r > 1$ be powers of 2, and q a prime such that $q \equiv 2r + 1 \pmod{4r}$. Then the cyclotomic polynomial $X^n + 1$ factors in $\mathbb{Z}_q[X]$ as $X^n + 1 = \prod_{i=1}^{r} \left(X^{n/r} - s_i\right)$, for some distinct $s_i \in \mathbb{Z}_q^*$ such that the $\left(X^{n/r} - s_i\right)$ are irreducible in $\mathbb{Z}_q[X]$. Moreover, any $f \in \mathcal{R}_q$ such that $0 < \|f\|_\infty < q^{1/r}/\sqrt{r}$ or $0 < \|f\| < q^{1/r}$ is invertible.*

This result can be used to build two different types of FRD encodings. One could encode messages as polynomials of l_∞-norm smaller than $\frac{q^{1/r}}{2\sqrt{r}}$ with an injective map and obtain an FRD this way. But we decided to use the "low-degree" FRD

described in Proposition 1 as opposed to a "small-norm" one, as it results in a slightly more efficient implementation.

Proposition 1. *Let $n \geq r > 1$ be powers of 2, q a prime such that $q \equiv 2r + 1 \pmod{4r}$, and $\mathcal{M} = \mathbb{Z}_q^{n/r} \setminus \{\mathbf{0}\}$ the set of messages. Then the following map $H : \mathcal{M} \longrightarrow \mathcal{R}_q$ is an FRD encoding.*

$$(m_0, \ldots, m_{n/r-1}) \longmapsto \sum_{i=0}^{n/r} m_i X^i$$

The proof of this proposition is given in Section C.

FRD on modules. We build an FRD encoding in the module setting using an existing FRD encoding in the ring setting $H_R : \mathcal{M} \longrightarrow \mathcal{R}_q$ by constructing:

$$\begin{aligned} H_M : \mathcal{M} &\longrightarrow \mathcal{R}_q^{d \times d} \\ m &\longmapsto H_R(m) \cdot \boldsymbol{I}_d = \begin{bmatrix} H_R(m) & & \\ & \ddots & \\ & & H_R(m) \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{I}_d \in \mathcal{R}_q^{d \times d}$ is the identity matrix.

Lemma 3. *If H_R is an FRD (in the ring setting) from $\mathcal{M}$ to $\mathcal{R}_q$, then H_M as constructed above is an FRD (in the module setting) from $\mathcal{M}$ to $\mathcal{R}_q^{d \times d}$.*

Implementation and performance. The main point that differs from our ROM scheme in the implementation is the arithmetic over $\mathcal{R}_q$. While without having $q \equiv 1 \pmod{2n}$ one cannot use the NTT, we can still make use of the structure of our ring to speed up the multiplication of polynomials. Described at a high level, what we perform is a "partial NTT". To multiply polynomials, we first reduce them modulo all the $X^{n/r} - s_i$ in $\Theta(n \log r)$ operations (see Section C.1). Then, we multiply them in the smaller rings $\mathbb{Z}_q[X]/\langle X^{n/r} - s_i \rangle$ by using the Karatsuba multiplication algorithm, and reducing them both modulo q and modulo the $X^{n/r} - s_i$. The result can then be mapped back to the ring $\mathcal{R}_q$ in time $\Theta(n \log r)$ using an inverse transform. These ideas were formulated in [LS18].

In Table 9, we present the performance of our implementation of this standard model scheme, and in particular highlight the additional cost compared to our ROM scheme of Section 4.1.

We do not give a comparison with the implementation of [BFRS18] as it would not be relevant, given the limited security provided by their instantiation of the FRD encoding.

4.3 An identity-based encryption scheme on modules

Finally, we built a more advanced construction based on our tools: a standard model identity-based encryption scheme.

We give a complete description of our IBE in Section D.

Table 9. Performances of our standard model signature.

Parameter set	KeyGen	Sign	Verify
I	9.46 ms (+27%)	15.66 ms (+37%)	1.19 ms (+63%)
II	73.41 ms (+43%)	21.92 ms (+44%)	2.23 ms (+123%)
III	51.79 ms (+42%)	29.11 ms (+66%)	2.37 ms (+112%)

Implementation and performance. As in our standard model signature scheme, we make use of our ring to speed up the multiplication of polynomials by performing a partial NTT. We make use of the same encoding as in the previous section, which imposes the condition $q \equiv 2r + 1 \pmod{4r}$ on the modulus, to map identities in $\mathcal{M} = \mathbb{Z}_q^{n/r} \setminus \{\mathbf{0}\}$ to invertible elements in $\mathcal{R}_q^{d \times d}$. In Table 10, we present the performance of our implementation of this standard model IBE scheme.

Table 10. Timings of the different operations of our scheme: Setup, Extract, Encrpt, and Decrypt

Parameter Set	Setup	Extract	Encrypt	Decrypt
I	9.82 ms	16.54 ms	4.87 ms	0.99 ms
II	44.91 ms	18.09 ms	5.48 ms	1.04 ms

In Table 11, we give timings for the different operations of some IBE schemes. Our timings could seem worse than the ones in [BFRS18] but the two implementations cannot be compared as the latter's limited security would make the comparison irrelevant. A part of the difference comes from the arithmetic we need to use in order to build a proper FRD encoding. Moreover, in contrast to [DLP14], we did not use NTRU lattices, which explains the differences in the timings.

Table 11. Timings of the different operations for some IBE schemes.

Scheme	(λ, n)	Setup	Extract	Encrypt	Decrypt
BF-128 [Fou13]	(128, –)	–	0.55 ms	7.51 ms	5.05 ms
DLP-14 [MSO17]	$(80, 512)$	4.034 ms	3.8 ms	0.91 ms	0.62 ms

Acknowledgements. This work was supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). Lucas Prabel is funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER).

References

[ABB+19] E. Alkim, P. S. L. M. Barreto, N. Bindel, P. Longa, and J. E. Ricardini. The lattice-based digital signature scheme qtesla. *IACR Cryptology ePrint Archive*, 2019:85, 2019.

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 553–572. Springer, 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, volume 6223 of *LNCS*, pages 98–115. Springer, 2010.

[ADP+16] E. Alkım, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In *USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[AHH+19] M. R. Albrecht, C. Hanser, A. Höller, T. Pöppelmann, F. Virdia, and A. Wallner. Implementing rlwe-based schemes using an RSA co-processor. *IACR TCHES*, 2019(1):169–208, 2019.

[APS15] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.

[BB13] R. E. Bansarkhani and J. A. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In *SAC*, volume 8282 of *LNCS*, pages 48–67. Springer, 2013.

[BFRS18] P. Bert, P. Fouque, A. Roux-Langlois, and M. Sabt. Practical implementation of ring-sis/lwe based signature and IBE. In *PQCrypto*, volume 10786 of *LNCS*, pages 271–291. Springer, 2018.

[Boy10] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, volume 6056 of *LNCS*, pages 499–517. Springer, 2010.

[CGM19] Y. Chen, N. Genise, and P. Mukherjee. Approximate trapdoors for lattices and smaller hash-and-sign signatures. In *ASIACRYPT (3)*, volume 11923 of *LNCS*, pages 3–32. Springer, 2019.

[CHK+10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010.

[CPS+19] C. Chuengsatiansup, T. Prest, D. Stehlé, A. Wallet, and K. Xagawa. Modfalcon: compact signatures based on module NTRU lattices. *IACR Cryptol. ePrint Arch.*, 2019:1456, 2019.

[DKL+18] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *TCHES*, 2018(1):238–268, 2018.

[DKR+18] J. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren. Saber: module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In *AFRICACRYPT*, volume 10831 of *LNCS*, pages 282–305. Springer, 2018.

[DLP14] L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT (2)*, volume 8874 of *LNCS*, pages 22–41. Springer, 2014.

[DP16] L. Ducas and T. Prest. Fast fourier orthogonalization. In *ISSAC*, pages 191–198. ACM, 2016.

[FHK+17] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON: fast-fourier lattice-based compact signatures over NTRU, 2017. URL: `https://falcon-sign.info/falcon.pdf`.

[Fou13] E. Fouotsa. *Calcul des couplages et arithmetique des courbes elliptiques pour la cryptographie*. PhD thesis, Rennes 1, 2013.

[GM18] N. Genise and D. Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In *EUROCRYPT (1)*, volume 10820 of *LNCS*, pages 174–203. Springer, 2018.

[GPR+18] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savas. Implementation and evaluation of improved gaussian sampling for lattice trapdoors. In *WAHC@CCS*, pages 61–71. ACM, 2018.

[GPR+19] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, H. Sajjadpour, and E. Savas. Practical applications of improved gaussian sampling for trapdoor lattices. *IEEE Trans. Computers*, 68(4):570–584, 2019.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554. ACM, 2013.

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998.

[Kar16] C. F. F. Karney. Sampling exactly from the normal distribution. *ACM Trans. Math. Softw.*, 42(1):3:1–3:14, 2016.

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

[LCC14] R. W. F. Lai, H. K. F. Cheung, and S. S. M. Chow. Trapdoors for ideal lattices with applications. In *Inscrypt*, volume 8957 of *LNCS*, pages 239–256. Springer, 2014.

[LLL+13] F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT (2)*, volume 8270 of *LNCS*, pages 41–61. Springer, 2013.

[LM08] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, volume 4948 of *LNCS*, pages 37–54. Springer, 2008.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, volume 7881 of *LNCS*, pages 35–54. Springer, 2013.

[LS15] A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[LS18] V. Lyubashevsky and G. Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT (1)*, volume 10820 of *LNCS*, pages 204–224. Springer, 2018.

[MN17] B. Mennink and S. Neves. Optimal prfs from blockcipher designs. *IACR ToSC*, 2017(3):228–252, 2017.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

[MR07] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MSO17] S. McCarthy, N. Smyth, and E. O'Sullivan. A practical implementation of identity-based encryption over ntru lattices. In *IMA International Conference on Cryptography and Coding*, pages 227–246. Springer, 2017.

[MW17] D. Micciancio and M. Walter. Gaussian sampling over the integers: efficient, generic, constant-time. In *CRYPTO (2)*, volume 10402 of *LNCS*, pages 455–485. Springer, 2017.

[Pei10] C. Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, volume 6223 of *LNCS*, pages 80–97. Springer, 2010.

[PHS19] A. Pellet-Mary, G. Hanrot, and D. Stehlé. Approx-svp in ideal lattices with pre-processing. *IACR Cryptology ePrint Archive*, 2019:215, 2019.

[PR06] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, volume 3876 of *LNCS*, pages 145–166. Springer, 2006.

[Sei18] G. Seiler. Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography. *IACR Cryptology ePrint Archive*, 2018:39, 2018.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[Sho94] P. W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *ANTS*, volume 877 of *LNCS*, page 289. Springer, 1994.

[SS13] D. Stehlé and R. Steinfeld. Making ntruencrypt and ntrusign as secure as standard worst-case problems over ideal lattices. *IACR Cryptology ePrint Archive*, 2013:4, 2013.

[SST+09] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009.

[Yam16] S. Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In *EUROCRYPT (2)*, volume 9666 of *LNCS*, pages 32–62. Springer, 2016.

[ZMS+21] R. K. Zhao, S. McCarthy, R. Steinfeld, A. Sakzad, and M. O'Neill. Quantum-safe hibe: does it cost a latte? Cryptology ePrint Archive, Report 2021/222, 2021.

A Building trapdoors for module lattices

We construct trapdoors and present their usefulness in Section A.1. Sections A.2, A.3, and A.4 concern G-sampling, perturbation sampling, and the Gaussian preimage sampling procedure. Amongst those, our main contributions are the adaptation to modules of the perturbation sampling algorithm, and an analysis of the involved parameters that is more rigorous than in the previous works. The other notions and algorithms are direct generalizations of their counterparts in the unstructured and ring settings, and are stated for the sake of completeness. We detail our implementation and its performances in Sections 3.2 and 3.3.

A.1 Building trapdoors for module lattices

We first define a specific lattice named the module G-lattice, where the Module-SIS problem is easy. Then, we describe how we can generate simultaneously a random-looking lattice and its trapdoor, which is a way of mapping the module G-lattice to it. Finally, we show how to use this trapdoor to solve Module-SIS on the random lattice.

In the case of ideal lattices, the construction starts with the ring *gadget vector*

$$\boldsymbol{g}^T = \begin{bmatrix} 1 & b & b^2 & \cdots & b^{k-1} \end{bmatrix} \in \mathcal{R}_q^{1\times k}, \quad \text{where } k = \lceil \log_b q \rceil,$$

whose entries are constant power-of-b polynomials, and the ring G-lattice $\Lambda_q^\perp(\boldsymbol{g}^T) \subset \mathcal{R}^k$. Typically, we take $b = 2$ (which allows us to use efficient bitwise operations in our implementation) but choosing larger values for b is possible in our implementation and could bring some interesting trade-offs in terms of efficiency. In the module setting, we use these to create the matrix

$$\boldsymbol{G} = \boldsymbol{I}_d \otimes \boldsymbol{g}^T = \begin{bmatrix} \boldsymbol{g}^T & & & \\ & \boldsymbol{g}^T & & \\ & & \ddots & \\ & & & \boldsymbol{g}^T \end{bmatrix} \in \mathcal{R}^{d\times dk},$$

and the associated module G-lattice $\Lambda_q^\perp(\boldsymbol{G}) \subset \mathcal{R}^{dk}$, which is isomorphic to d copies of $\Lambda_q^\perp(\boldsymbol{g}^T)$. Having introduced the G-lattice, we can now define trapdoors.

Definition 4. *A trapdoor for a matrix $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$ is a matrix $\boldsymbol{T} \in \mathcal{R}^{(m-dk)\times dk}$ such that*

$$\boldsymbol{A} \begin{bmatrix} \boldsymbol{T} \\ \hline \boldsymbol{I}_{dk} \end{bmatrix} = \boldsymbol{H}\boldsymbol{G}$$

for some invertible $\boldsymbol{H} \in \mathcal{R}_q^{d\times d}$, called the tag *of $\boldsymbol{A}$.*

We now describe the algorithm TrapGen, which outputs a matrix $\boldsymbol{A}$ along with its associated trapdoor $\boldsymbol{T}$, given a tag $\boldsymbol{H}$. So that $\boldsymbol{A}$'s uniformity is guaranteed by a Module-LWE instance, we instantiate TrapGen with $m = d(k+2)$.

Algorithm 2 TrapGen($\boldsymbol{H}, \sigma$) for the generation of a matrix $\boldsymbol{A}$ and its trapdoor $\boldsymbol{T}$

function TrapGen($\boldsymbol{H} \in \mathcal{R}_q^{d\times d}, \sigma > 0$)
  $\hat{\boldsymbol{A}} \leftarrow \mathcal{U}(\mathcal{R}_q^{d\times d})$ ▷ $\hat{\boldsymbol{A}} \in \mathcal{R}_q^{d\times d}$
  $\boldsymbol{A'} \leftarrow [\, \boldsymbol{I}_d \;\big|\; \hat{\boldsymbol{A}} \,]$ ▷ $\boldsymbol{A'} \in \mathcal{R}_q^{d\times 2d}$
  $\boldsymbol{T} \leftarrow D_{\mathcal{R}^{2d\times dk},\sigma}$ ▷ $\boldsymbol{T} \in \mathcal{R}^{2d\times dk}$
  $\boldsymbol{A} \leftarrow [\, \boldsymbol{A'} \;\big|\; \boldsymbol{HG} - \boldsymbol{A'T} \,]$ ▷ $\boldsymbol{A} \in \mathcal{R}^{d\times m}$
  return $(\boldsymbol{A}, \boldsymbol{T})$

We state the correctness of the trapdoor generation algorithm but do not give a proof for it, and instead direct the reader towards [MP12, Section 5.2] for a better understanding. Indeed, the principles of the algorithm are the same, and the translation to the module setting is quite direct.

Lemma 4. *Let q, d be positive integers, $k = \lceil \log_b q \rceil$, $m = d(k+2)$, $\boldsymbol{H} \in \mathcal{R}_q^{d\times d}$, $\sigma > 0$, and $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \mathrm{TrapGen}(\boldsymbol{H}, \sigma)$. Then, the following points hold:*

- *$\boldsymbol{T}$ is a trapdoor for $\boldsymbol{A}$ with tag $\boldsymbol{H}$;*
- *$\boldsymbol{A}$ is computationally indistinguishable from uniform (excluding the identity submatrix in the first d columns) based on the hardness of the decision version of Module-LWE$_{n,d,q,\sigma}$.*

These trapdoors are useful as they allow one to compute small vectors on any coset of a random lattice if they can do so on the G-lattice. As we explain in Section A.2, a small vector on the G-lattice can be mapped to a small vector on the random lattice using its trapdoor.

A.2 G-sampling

We now describe one of the two main steps of the Gaussian preimage sampling. Given a target vector $\boldsymbol{v} \in \mathcal{R}^d$, the goal of G-sampling is to compute a vector $\boldsymbol{z}$ following a discrete Gaussian of parameter α over the coset $\Lambda_q^{\boldsymbol{v}}(\boldsymbol{G})$ of the module G-lattice defined by $\boldsymbol{G} \in \mathcal{R}^{d\times dk}$.

How to perform G-sampling. We make use of an efficient algorithm [GM18, Section 3] designed for G-sampling in the unstructured setting, running in $\mathcal{O}(k)$ time. It samples on cosets of the scalar G-lattice $\Lambda_q^{\perp}(\boldsymbol{g}_s^T) \subset \mathbb{Z}^k$, defined by the scalar gadget vector $\boldsymbol{g}_s^T = \begin{bmatrix} 1 & b & b^2 & \cdots & b^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1\times k}$ used in the construction of trapdoors on unstructured lattices. While we do not go into detail about the algorithm, and treat it as a black box, we need to take into account the following constraint for its instantiation. According to [GM18, Corollary 3.1 and Proposition 3.1], we need to take $\alpha \geq \sqrt{2b} \cdot (2b+1) \cdot \sqrt{\log(2k(1+1/\varepsilon))/\pi}$ and $\alpha \geq (b+1)\eta_\varepsilon(\mathbb{Z})$ for the algorithm to be correct.

To sample a vector $\boldsymbol{z}_r \in \mathcal{R}^k$ from a coset of the ring G-lattice, one can do the following: independently sample n column vectors from the scalar G-lattice,

arrange them into an $k \times n$ matrix, and form each of the k entries of $\boldsymbol{z}_r$ by reading the rows in order. In turn, module G-sampling consists in d independent operations of ring G-sampling. In summary, we make nd calls to the scalar G-sampling algorithm, leading to a procedure for module G-sampling running in optimal $\mathcal{O}(ndk)$ time.

What to do with G-samples. Now that we are able to sample such vectors, let us explain how to use them, along with the trapdoor $\boldsymbol{T}$, to compute a Gaussian vector $\boldsymbol{x} \in \Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ for a given $\boldsymbol{u} \in \mathcal{R}^d$. Letting $\boldsymbol{v} = \boldsymbol{H}^{-1}\boldsymbol{u}$, we sample $\boldsymbol{z} \leftarrow D_{\Lambda_q^{\boldsymbol{v}}(\boldsymbol{G}),\alpha}$, and output $\boldsymbol{x} = \left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\boldsymbol{z}$. One can check that such an $\boldsymbol{x}$ then lies in the desired coset. Nevertheless, its distribution is degenerate (the support is not full-rank) and skewed (the covariance matrix is $\alpha^2\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\left[\begin{smallmatrix}\boldsymbol{T}^T & \boldsymbol{I}\end{smallmatrix}\right]$), which leaks information about the trapdoor. To solve this problem, we make use of perturbation vectors, as first described in [Pei10].

A.3 Perturbation sampling

The rest of this section contains more details on Section 3.1 (in particular how transposition over $\mathcal{P}$ is defined), proof of this algorithm, and constraints concerning the involved parameters.

Matrices of polynomials and Schur complements. We denote by ϕ the ring homomorphism from $\mathcal{P} = \mathbb{R}[X]/\langle X^n + 1\rangle$ to $\mathbb{R}^{n\times n}$ that takes a polynomial $a = \sum_{i=0}^{n-1} a_i X^i$ to the matrix

$$\phi(a) = \begin{bmatrix} a_0 & -a_{n-1} & \cdots & -a_1 \\ a_1 & a_0 & & \vdots \\ \vdots & & \ddots & -a_{n-1} \\ a_{n-1} & \cdots & a_1 & a_0 \end{bmatrix},$$

which represents multiplication by a in $\mathcal{P}$ when polynomials are represented as vectors over $\mathbb{R}$. This definition is naturally extended to matrices over $\mathcal{P}$ by component-wise application.

Sampling a vector of m polynomials with covariance $\boldsymbol{\Sigma} \in \mathcal{P}^{m\times m}$ is then equivalent to sampling a vector of nm scalars with covariance $\phi(\boldsymbol{\Sigma}) \in \mathbb{R}^{nm\times nm}$. While our algorithm is described in terms of matrices over $\mathcal{P}$, it is easier to understand it when viewing the covariance matrix as a real matrix.

We remind the reader that for a symmetric matrix $\boldsymbol{M} = \left[\begin{smallmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{B}^T & \boldsymbol{D}\end{smallmatrix}\right]$ over $\mathbb{R}$ where the lower-right block $\boldsymbol{D}$ is invertible, the Schur complement of $\boldsymbol{D}$ is $\boldsymbol{M}/\boldsymbol{D} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^T$. Then, $\boldsymbol{M}$ is positive definite if and only if both $\boldsymbol{D}$ and $\boldsymbol{M}/\boldsymbol{D}$ are positive definite themselves. If we write out the blocks of a structured scalar covariance matrix

$$\phi(\boldsymbol{\Sigma}) = \begin{bmatrix} \phi(\boldsymbol{A}) & \phi(\boldsymbol{B}) \\ \phi(\boldsymbol{B})^T & \phi(\boldsymbol{D}) \end{bmatrix} \in \mathbb{R}^{nm\times nm},$$

then we can consider $\phi(\boldsymbol{\Sigma})/\phi(\boldsymbol{D}) = \phi(\boldsymbol{A}) - \phi(\boldsymbol{B})\phi(\boldsymbol{D}^{-1})\phi(\boldsymbol{B})^T$. For a similar definition to make sense in terms of matrices over $\mathcal{P}$ (that is, we want to define the Schur complement $\boldsymbol{\Sigma}/\boldsymbol{D} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^T$) we first need to construct a matrix $\boldsymbol{B}^T$ such that $\phi(\boldsymbol{B}^T) = \phi(\boldsymbol{B})^T$.

For any polynomial $a = \sum_{i=0}^{n-1} a_i X^i \in \mathcal{P}$, we define its *transpose* to be $a^T = a_0 - \sum_{i=1}^{n-1} a_{n-i} X^i \in \mathcal{P}$. It is the polynomial such that $\phi(a^T) = \phi(a)^T$. The transpose of a matrix $\boldsymbol{B}$ over $\mathcal{P}$ is the matrix $\boldsymbol{B}^T$ such that $\boldsymbol{B}^T[i,j] = \boldsymbol{B}[j,i]^T$. We then have that $\phi(\boldsymbol{B}^T) = \phi(\boldsymbol{B})^T$, which is what we needed to be able to define Schur complements of matrices over $\mathcal{P}$. For a real $\eta \geq 0$, we will write $\boldsymbol{\Sigma} \succeq \eta$ instead of $\phi(\boldsymbol{\Sigma}) \succeq \eta$ for ease of notation.

Proof of Algorithm 1. The correctness of SamplePerturb relies on the two following lemmas. The first one is a convolution lemma, which justifies that the output distribution is correct.

Lemma 5 ([GM18, Lemma 4.3]). *For any real $0 < \varepsilon \leq 1/2$, positive integers r and s, vector $\boldsymbol{c} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathbb{R}^{r+s}$, positive definite $\boldsymbol{\Sigma} = \left[\begin{smallmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{D} \end{smallmatrix}\right] \in \mathbb{R}^{(r+s)\times(r+s)}$ composed of blocks $\boldsymbol{A} \in \mathbb{R}^{r\times r}$, $\boldsymbol{B} \in \mathbb{R}^{r\times s}$, $\boldsymbol{D} \in \mathbb{R}^{s\times s}$, we define the following random process:*

- $\boldsymbol{x}_1 \leftarrow D_{\mathbb{Z}^s, \sqrt{\boldsymbol{D}}, \boldsymbol{c}_1}$;
- $\boldsymbol{x}_0 \leftarrow D_{\mathbb{Z}^r, \sqrt{\boldsymbol{\Sigma}/\boldsymbol{D}}, \boldsymbol{c}_0 + \boldsymbol{B}\boldsymbol{D}^{-1}(\boldsymbol{x}_1 - \boldsymbol{c}_1)}$.

If $\boldsymbol{\Sigma} \succeq \eta_\varepsilon^2(\mathbb{Z}^{r+s})$, then this process outputs a vector $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1) \in \mathbb{Z}^{r+s}$ whose distribution is statistically indistinguishable from $D_{\mathbb{Z}^{r+s}, \sqrt{\boldsymbol{\Sigma}}, \boldsymbol{c}}$.

The second one ensures that all covariance matrices manipulated during our algorithm are "positive definite enough" so that we can rigorously apply Lemma 5 at each step of the algorithm.

Lemma 6 ([GM18, Lemma 4.2]). *Let $\varepsilon > 0$, r and s be positive integers, and $\boldsymbol{\Sigma} = \left[\begin{smallmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{D} \end{smallmatrix}\right] \in \mathbb{R}^{(r+s)\times(r+s)}$ be a positive definite matrix made out of blocks $\boldsymbol{A} \in \mathbb{R}^{r\times r}$, $\boldsymbol{B} \in \mathbb{R}^{r\times s}$, $\boldsymbol{D} \in \mathbb{R}^{s\times s}$.*

If $\boldsymbol{\Sigma} \succeq \eta_\varepsilon^2(\mathbb{Z}^{r+s})$, then $\boldsymbol{D} \succeq \eta_\varepsilon^2(\mathbb{Z}^s)$ and $\boldsymbol{\Sigma}/\boldsymbol{D} \succeq \eta_\varepsilon^2(\mathbb{Z}^r)$.

We now prove Theorem 1, which shows the correctness of our perturbation sampling algorithm.

Theorem 3. *Let $\boldsymbol{T} \in \mathcal{P}^{2d\times dk}$, $\zeta, \alpha > 0$, and $\boldsymbol{\Sigma}_p = \zeta^2 \boldsymbol{I} - \alpha^2 \left[\begin{smallmatrix} \boldsymbol{T} \\ \boldsymbol{I} \end{smallmatrix}\right] \left[\begin{smallmatrix} \boldsymbol{T}^T & \boldsymbol{I} \end{smallmatrix}\right] \in \mathcal{P}^{m\times m}$ be the derived perturbation covariance matrix.*

If $\boldsymbol{\Sigma}_p \succeq \eta_\varepsilon^2(\mathbb{Z}^{nm})$, then SamplePerturb$(\boldsymbol{T}, \zeta, \alpha)$ *returns a vector $\boldsymbol{p} \in \mathcal{R}^m$ whose distribution is statistically indistinguishable from $D_{\mathcal{R}^m, \sqrt{\boldsymbol{\Sigma}_p}}$.*

Proof. First, let us observe that $\boldsymbol{\Sigma}_p$ has a particular structure. Indeed,

$$\boldsymbol{\Sigma}_p = \zeta^2 \boldsymbol{I} - \alpha^2 \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{T}^T & \boldsymbol{I} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{A} & -\alpha^2 \boldsymbol{T} \\ \hline -\alpha^2 \boldsymbol{T}^T & (\zeta^2 - \alpha^2)\boldsymbol{I} \end{array}\right],$$

where $\boldsymbol{A} = \zeta^2 \boldsymbol{I} - \alpha^2 \boldsymbol{T}\boldsymbol{T}^T \in \mathcal{P}^{2d \times 2d}$. This allows us to use Lemma 5 to first sample a centered vector $\boldsymbol{p}_s \in \mathcal{R}^{dk}$ with covariance $(\zeta^2 - \alpha^2)\boldsymbol{I}$ (which is easy since the distribution is spherical), and then sample a vector with covariance $\boldsymbol{\Sigma}_p/(\zeta^2 - \alpha^2)\boldsymbol{I}$ and center $\boldsymbol{c} = -\frac{\alpha^2}{\zeta^2 - \alpha^2}\boldsymbol{T}\boldsymbol{p}_s$.

However, $\zeta^2 \boldsymbol{I} - (\alpha^{-2} - \zeta^{-2})^{-1}\boldsymbol{T}\boldsymbol{T}^T$ is precisely the Schur complement of $(\zeta^2 - \alpha^2)\boldsymbol{I}$ in $\boldsymbol{\Sigma}_p$. The goal of what comes after sampling $\boldsymbol{p}_s$ is sampling a vector $(p_0, \ldots, p_{2d-1}) \in \mathcal{R}^{2d}$ with this covariance and an updated center $\boldsymbol{c}$.

To this end, we make use of Lemma 5 iteratively during the loop. More precisely, at each iteration of the loop, we sample the rightmost remaining entry of $\boldsymbol{p}$ using SampleFz. This entry's covariance is the lower-right entry of $\boldsymbol{\Sigma}$, and its center is the last entry of $\boldsymbol{c}$. We then update both $\boldsymbol{\Sigma}$ and $\boldsymbol{c}$ according to Lemma 5, and proceed with the sampling of the other entries in the same way, until we have sampled all of $\boldsymbol{p}$.

The only missing argument is that we can actually apply Lemma 5 at each step of the algorithm. Lemma 6 takes care of that, assuring that at each iteration of the loop we have $\boldsymbol{\Sigma} \succeq \eta_\varepsilon^2(\mathbb{Z}^{i+1})$. □

Gaussian parameters for perturbation sampling. As seen in Theorem 1, we need to have $\boldsymbol{\Sigma}_p \succeq \eta_\varepsilon^2(\mathbb{Z}^{nm})$ for SamplePerturb to be correct. Let us show how this leads to a lower bound on ζ.

Lemma 7. *Let $\boldsymbol{T} \in \mathcal{P}^{2d \times dk}$, $\zeta, \alpha > 0$, and $\boldsymbol{\Sigma}_p = \zeta^2 \boldsymbol{I} - \alpha^2 \left[\begin{smallmatrix} \boldsymbol{T} \\ \boldsymbol{I} \end{smallmatrix}\right] \left[\begin{smallmatrix} \boldsymbol{T}^T & \boldsymbol{I} \end{smallmatrix}\right]$. For any positive real $0 < \varepsilon \leq 1/2$, if ζ is such that $\zeta > \sqrt{(\alpha^2 + 1)s_1^2(\boldsymbol{T}) + \eta_\varepsilon^2(\mathbb{Z}^{nm})}$ and $\zeta > \sqrt{\alpha^2 + \eta_\varepsilon^2(\mathbb{Z}^{nm})}$, then we have $\boldsymbol{\Sigma}_p \succ \eta_\varepsilon^2(\mathbb{Z}^{nm})$.*

Proof. For the sake of simplicity, we will write $\eta = \eta_\varepsilon(\mathbb{Z}^{nm})$. We want the matrix $\boldsymbol{\Sigma}_p - \eta^2 \boldsymbol{I}$ to be positive definite. Since

$$\boldsymbol{\Sigma}_p - \eta^2 \boldsymbol{I} = \left[\begin{array}{c|c} \boldsymbol{M} & -\alpha^2 \boldsymbol{T} \\ \hline -\alpha^2 \boldsymbol{T}^T & (\zeta^2 - \alpha^2 - \eta^2)\boldsymbol{I} \end{array}\right],$$

where $\boldsymbol{M} = (\zeta^2 - \eta^2)\boldsymbol{I} - \alpha^2 \boldsymbol{T}\boldsymbol{T}^T \in \mathcal{R}^{2d \times 2d}$, this condition is equivalent to having both the block $(\zeta^2 - \alpha^2 - \eta^2)\boldsymbol{I}$ and its Schur complement be positive definite. The former trivially is if $\zeta^2 - \alpha^2 - \eta^2 > 0$; for the latter we need to take a closer look at the Schur complement in question.

This Schur complement is equal to $(\zeta^2 - \eta^2)\boldsymbol{I} - \alpha^2 \frac{\zeta^2 - \eta^2}{\zeta^2 - \alpha^2 - \eta^2}\boldsymbol{T}\boldsymbol{T}^T$. It is positive definite if and only if $\boldsymbol{I} - \frac{\alpha^2}{\zeta^2 - \alpha^2 - \eta^2}\boldsymbol{T}\boldsymbol{T}^T$ is. A sufficient condition for that is having $\frac{\alpha^2}{\zeta^2 - \alpha^2 - \eta^2} s_1^2(\boldsymbol{T}) < 1$, which in turn yields $\zeta^2 > (\alpha^2 + 1)s_1^2(\boldsymbol{T}) + \eta^2$. □

To instantiate concretely SamplePerturb, we then need to determine an upper bound on $s_1(\boldsymbol{T})$, knowing that $\boldsymbol{T}$ follows a Gaussian distribution over $\mathcal{P}^{2d \times dk}$ of parameter σ. If $\boldsymbol{T}$ was a $2nd \times ndk$ unstructured matrix over $\mathbb{R}$, one could apply Lemma 2.9 of [MP12], it would state that $s_1(\boldsymbol{T}) < C\sigma(\sqrt{2nd} + \sqrt{ndk} + c)$ for a certain universal constant $C > 0$, except with probability at most $2\exp(-\pi c^2)$.

While in the case where $\boldsymbol{T}$ is structured, there is no similar result that we know of, we can hypothesize that such a property still holds in our case, which we confirmed empirically.

We choose $c = 4.7$ to ensure that the probability $2\exp(-\pi c^2)$ is less than 2^{-100}. During our experiments, we found the constant C to be less than 1.1. We generated matrices $\boldsymbol{T}$ with the same structure and size as the ones used in our signature schemes (see Table 5 for concrete values for n, k, d and σ), computed their exact spectral norm and their expected spectral norm, and deduced the constant C.

A.4 Sampling Gaussian preimages

Once we are able to perform both G-sampling and perturbation sampling, we combine these two operations to sample from the spherical Gaussian of parameter ζ on a coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$. The method we use is the same as in [MP12], but described in the module setting, yielding the algorithm SamplePre.

We remind the reader of the following conditions on α and ζ for the correctness of the algorithm :

- $\alpha \geq \sqrt{2b} \cdot (2b+1) \cdot \sqrt{\log(2k(1+1/\varepsilon))/\pi}$ (see Section A.2),
- $\zeta > \sqrt{(\alpha^2+1)s_1^2(\boldsymbol{T}) + \eta_\varepsilon^2(\mathbb{Z}^{nm})}$ (see Section A.3),
  knowing that $s_1(\boldsymbol{T}) < 1.1\sigma(\sqrt{2nd} + \sqrt{ndk} + 4.7)$ with high probability.

Concrete values for these parameters can be found in Table 5, Section 4.1.

Algorithm 3 SamplePre$(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{u}, \zeta, \alpha)$ for sampling from a discrete Gaussian of parameter ζ over $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$

function SamplePre$(\boldsymbol{A} \in \mathcal{R}_q^{d\times m}, \boldsymbol{T} \in \mathcal{R}^{2d\times dk}, \boldsymbol{H} \in \mathcal{R}_q^{d\times d}, \boldsymbol{u} \in \mathcal{R}_q^d, \zeta > 0, \alpha > 0)$
  $\boldsymbol{p} \leftarrow$ SamplePerturb$(\boldsymbol{T})$ ▷ $\boldsymbol{p} \in \mathcal{R}^m$
  $\boldsymbol{v} \leftarrow \boldsymbol{H}^{-1}(\boldsymbol{u} - \boldsymbol{A}\boldsymbol{p})$ ▷ $\boldsymbol{v} \in \mathcal{R}_q^d$
  $\boldsymbol{z} \leftarrow D_{\Lambda_q^{\boldsymbol{v}}(\boldsymbol{G}),\alpha}$ ▷ $\boldsymbol{z} \in \mathcal{R}^{dk}$
  $\boldsymbol{x} \leftarrow \boldsymbol{p} + \left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{I}\end{smallmatrix}\right]\boldsymbol{z}$ ▷ $\boldsymbol{x} \in \mathcal{R}^m$
  return $\boldsymbol{x}$

Theorem 4. *Let $\boldsymbol{T} \in \mathcal{R}^{2d\times dk}$ be a trapdoor for the matrix $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$ with tag $\boldsymbol{H} \in \mathcal{R}_q^{d\times d}$, $\boldsymbol{u} \in \mathcal{R}_q^d$ be a target vector, and ζ and α be Gaussian parameters with the above constraints. Then,* SamplePre$(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{u}, \zeta, \alpha)$ *outputs a vector whose distribution is statistically close to $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\zeta}$.*

B The GPV signature scheme on modules

In this hash-and-sign scheme, key generation is simply trapdoor generation, and signing consists in hashing a message to a point in space and then sampling a Gaussian preimage of that point.

B.1 Construction

We base ourselves on the probabilistic GPV signature [GPV08, Section 6.2], where a random salt is appended to a message before hashing it. Then, if the same message M is signed several times, one does not obtain multiple samples from the same coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$, which would compromise the security of the scheme.

Parameters. The parameters are: n a power of two, q a prime modulus such that $q \equiv 1 \pmod{2n}$, $k = \lceil \log_b q \rceil$ its size in the base-b expansion, d a positive integer, $m = d(k+2)$, σ, ζ, and α Gaussian parameters, t the Gaussian tailcut, and s the length of the salt.

Description. The scheme is as follows, where $\mathcal{H} : \{0,1\}^* \longrightarrow \mathcal{R}_q^d$ is a collision-resistant hash function.

- $\text{KeyGen}(1^n) \longrightarrow (vk, sk)$
  $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \text{TrapGen}(\boldsymbol{I}, \sigma)$
  $vk \leftarrow \boldsymbol{A} \in \mathcal{R}_q^{d \times m}$
  $sk \leftarrow \boldsymbol{T} \in \mathcal{R}^{2d \times dk}$
- $\text{Sign}(1^n, sk, M) \longrightarrow \nu$
  $S \leftarrow \mathcal{U}(\{0,1\}^s)$
  $\boldsymbol{u} \leftarrow \mathcal{H}(M \parallel S)$
  $\boldsymbol{x} \leftarrow \text{SamplePre}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{u}, \zeta, \alpha)$
  $\nu \leftarrow (\boldsymbol{x}, S) \in \mathcal{R}^m \times \{0,1\}^s$
- $\text{Verify}(1^n, vk, M, \nu) \longrightarrow \{accept, reject\}$
  $\boldsymbol{u} \leftarrow \mathcal{H}(M \parallel S)$
  Accept if $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q$ and $0 < \|\boldsymbol{x}\| \leq t\zeta\sqrt{mn}$

Theorem 5. *Let $n, q, k, d, m, \sigma, \zeta, \alpha, t, s$ be the parameters of the scheme as above. Then, the scheme is correct, and admits EU-CMA security in the random oracle model under the hardness of Module-LWE$_{n,d,q,\sigma}$ and Module-SIS$_{n,d,m,q,\beta}$, with $\beta = 2\zeta\sqrt{mn}$.*

B.2 Estimating security and choosing parameters

We describe how we chose the parameters for ou GPV signature scheme, estimating the difficulty of the underlying lattice problems. Our Gaussian sampling algorithms also impose constraints on our parameters (summarized in Section A.4).

Length of the salt. To make sure the probability of collision (two messages landing in the same coset) is negligible, we follow [FHK+17] and take the length of the random salt to be $s \geq \lambda + \log q_s$, where λ is the security parameter and q_s the number of allowed signing queries.

Key recovery. The computational instantiation of the trapdoors we use is based on the hardness of Module-LWE$_{n,d,q,\sigma}$, where we choose the Gaussian parameter σ to be at least the smoothing parameter $\eta_\varepsilon(\mathbb{Z}^{nd})$ where ε is taken as $2^{-\lambda}$ where λ is the target security parameter of the scheme. To get an idea of the security provided by such a Module-LWE instance, we use the estimator of [APS15][1] and approximate this instance by an instance of an unstructured LWE problem in dimension nd.

Signature forgery. Forging a signature consists in finding a short vector $\boldsymbol{x} \in \mathcal{R}^m$ of norm less than $\beta = t\zeta\sqrt{mn}$ satisfying the relation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q$ for some $\boldsymbol{u} \in \mathcal{R}_q^n$ that depends on the message. This is exactly an instance of Module-ISIS$_{n,d,m,q,\beta}$, which we approximate by an instance of unstructured ISIS$_{nd,nm,q,\beta}$. This ISIS problem can be solved by finding a short vector in $\Lambda_q^{\mathbf{u}}(\boldsymbol{A}')$ for some sub-matrix $\boldsymbol{A}'$ of $\boldsymbol{A}$. To estimate the cost of computing such a solution, we look at the cost of running the BKZ algorithm to reduce a basis of $\Lambda_q^{\mathbf{u}}(\boldsymbol{A}')$ in order to get a sufficiently short vector.

BKZ reduction cost model. The BKZ algorithm reduces a basis by calling an exact SVP oracle in a smaller dimension b, called the BKZ blocksize. We follow the (very pessimistic) core-SVP hardness introduced in [ADP$^+$16], where the cost of a BKZ algorithm with blocksize b is taken to be the cost of only one call to an SVP oracle in dimension b, rather than a polynomial number of calls. We consider the sieving algorithm as our exact SVP oracle, its complexity in dimension b is estimated to 2^{cb}, where $c = \log\sqrt{3/2} \approx 0.292$ in the classical setting, and $c = \log\sqrt{13/9} \approx 0.265$ in the quantum setting.

C A standard model signature scheme on modules

C.1 Encoding messages with full-rank differences

Proposition 2. *Let $n \geq r > 1$ be powers of 2, q a prime such that $q \equiv 2r + 1 \pmod{4r}$, and $\mathcal{M} = \mathbb{Z}_q^{n/r} \setminus \{\mathbf{0}\}$ the set of messages. Then the map*

$$H : \mathcal{M} \longrightarrow \mathcal{R}_q$$
$$(m_0, \ldots, m_{n/r-1}) \longmapsto \sum_{i=0}^{n/r} m_i X^i$$

is an FRD encoding.

Proof. According to Theorem 2 on the factorisation of $X^n + 1$ and the Chinese remainder theorem, we have that $\mathcal{R}_q = \frac{\mathbb{Z}_q[X]}{\langle X^n+1\rangle} \simeq \prod_{i=1}^{r} \frac{\mathbb{Z}_q[X]}{\langle X^{n/r} - s_i\rangle}$, with the $X^{n/r} - s_i$ being irreducible, meaning that the r rings of the product are actually fields (isomorphic to $\mathbb{F}_{q^{n/r}}$).

[1] `https://bitbucket.org/malb/lwe-estimator` (commit `a2296b8`)

Under the canonical CRT map, a nonzero polynomial f of degree less than n/r is sent to $(f \bmod X^{n/r} - s_1, \ldots, f \bmod X^{n/r} - s_r) = (f, \ldots, f)$. Each coordinate of the image vector is then invertible (since nonzero) in $\mathbb{F}_q^{n/r}$, which makes f itself invertible in $\mathcal{R}_q$. This proves that all the $H(m)$ are invertible.

For distinct $m_1, m_2 \in \mathcal{M}$, $H(m_1) - H(m_2)$ is a nonzero polynomial of degree less than n/r, so it is invertible in $\mathcal{R}_q$ by the previous reasoning. Finally, H is clearly computable in polynomial time. □

C.2 The signature scheme

The proposed signature scheme is a module adaptation of the scheme from [BFRS18] with a different instantiation of the FRD encoding.

Parameters. We have: $n \geq r > 1$ two powers of 2, q a prime modulus such that $q \equiv 2r + 1 \pmod{4r}$, $k = \lceil \log_b q \rceil$, d a positive integer, $m = d(k+2)$, σ, ζ, and α Gaussian parameters, and t the tailcut for the Gaussian $D_{\mathbb{Z},\sigma}$.

Description. The scheme is as follows, where $H : \mathcal{M} \longrightarrow \mathcal{R}_q^{d \times d}$ is an FRD encoding instantiated as explained above. Note that during the trapdoor generation, we use the tag $\boldsymbol{H} = \boldsymbol{0}$, meaning that $\boldsymbol{T}$ is not technically a trapdoor for $\boldsymbol{A}$ since $\boldsymbol{H}$ is singular. Still, the scheme is correct because for any message $M \in \mathcal{M}$, $\boldsymbol{T}$ is a trapdoor for $\boldsymbol{A}_M = \boldsymbol{A} + [\boldsymbol{0}_{d,2d} \mid H(M)\boldsymbol{G}]$ with tag $H(M)$.

- KeyGen$(1^n) \longrightarrow (vk, sk)$
  $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \text{TrapGen}(\boldsymbol{0}, \sigma)$
  $vk \leftarrow \boldsymbol{A} \in \mathcal{R}_q^{d \times m}$
  $sk \leftarrow \boldsymbol{T} \in \mathcal{R}_q^{2d \times dk}$
- Sign$(1^n, sk, M) \longrightarrow \boldsymbol{\nu}$
  $\boldsymbol{H}_M \leftarrow H(M)$
  $\boldsymbol{A}_M \leftarrow \boldsymbol{A} + \left[\boldsymbol{0}_{d,2d} \;\middle|\; \boldsymbol{H}_M\boldsymbol{G} \right]$
  $\boldsymbol{\nu} \leftarrow \text{SamplePre}(\boldsymbol{A}_M, \boldsymbol{T}, \boldsymbol{H}_M, \boldsymbol{0}, \zeta, \alpha)$
- Verify$(1^n, vk, M, \boldsymbol{\nu}) \longrightarrow \{accept, reject\}$
  $\boldsymbol{H}_M \leftarrow H(M)$
  $\boldsymbol{A}_M \leftarrow \boldsymbol{A} + \left[\boldsymbol{0}_{d,2d} \;\middle|\; \boldsymbol{H}_M\boldsymbol{G} \right]$
  Accept if $\boldsymbol{A}_M\boldsymbol{\nu} = \boldsymbol{0} \bmod q$ and $0 < \|\boldsymbol{\nu}\| \leq t\zeta\sqrt{mn}$

We now state the correctness and the security of our scheme, but do not give a proof for them, as they are essentially the same as in [BFRS18].

Theorem 6. *Let $n, r, q, k, d, m, \sigma, \zeta, \alpha, t$ be the parameters of the scheme as above. Then, the scheme is correct, and is SU-CMA secure under the hardness of Module-LWE$_{n,d,q,\sigma}$ and Module-SIS$_{n,d,2d,q,\beta}$, where $\beta = (1 + s_1(\boldsymbol{T}))t\zeta\sqrt{mn}$.*

Concrete parameters. Concerning the choice of parameters, we take into account the best known attacks rather than the proof of security, as is the case in most schemes [ABB+19; DKL+18]. As such, the analysis of security is exactly the same as the one we did for GPV in Section B.2.

The only factor influencing the choice of r is the number of bits of security we aim at. For a given level of security of λ bits, we want to be able to encode at least 2^λ messages to prevent brute-force attacks. Since the message space $\mathcal{M}$ is of size $q^{n/r}$, this yields the relation $r < \frac{n \log q}{\lambda}$. As we explain in Section 4.2, we want to take the biggest r possible to have an efficient arithmetic in $\mathcal{R}_q$.

This scheme's suggested parameter sets are then the same as those of our ROM scheme (described in Table 5), except for r and q. For the parameter sets I and II we take $(r, q) = (64, 1073741441)$, and for set III we take $(r, q) = (32, 1073740609)$.

D An identity-based encryption scheme on modules

The IBE scheme we present now is the identity-based encryption from [ABB10a; BFRS18] but adapted to the module setting, with a different instantiation of the FRD encoding (see Section 4.2). Basically, it is the signature scheme presented in the Section C.2 combined with the Dual-Regev encryption scheme.

Our IBE scheme is composed of 4 algorithms :

> $\underline{\text{Setup}}(1^n) \longrightarrow (mpk, msk)$: takes as input the security parameter and outputs the master public key mpk and the master secret key msk.
> $\underline{\text{Extract}}(1^n, mpk, msk, id) \longrightarrow sk_{id}$: takes as input the security parameter, the master keys mpk and msk and an identity $id \in \mathcal{ID}$ and outputs a private key sk_{id} associated to the identity id.
> $\underline{\text{Encrypt}}(1^n, mpk, id, M) \longrightarrow C$: takes as input the security parameter, the master public key mpk, an identity id and a message M and outputs a cyphertext C.
> $\underline{\text{Decrypt}}(1^n, sk_{id}, C) \longrightarrow \{M, Error\}$: takes as input the security parameter, the master public key mpk, a private key sk_{id} associated to the identity id and a cyphertext C and outputs either a message M or the word "$Error$" if the cyphertext is invalid.

Description. The scheme is as follows, where $H : \mathcal{M} \longrightarrow \mathcal{R}_q^{d \times d}$ is an FRD encoding instantiated as explained in Section 4.2.

- $\text{Setup}(1^n) \longrightarrow (mpk, msk)$
  $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \text{TrapGen}(\boldsymbol{0}, \sigma)$
  $\boldsymbol{u} \leftarrow \mathcal{U}(\mathcal{R}_q^d)$
  $mpk \leftarrow (\boldsymbol{A}, \boldsymbol{u}) \in \mathcal{R}_q^{d \times (m+1)}$
  $msk \leftarrow \boldsymbol{T} \in \mathcal{R}_q^{2d \times dk}$
- $\text{Extract}(1^n, mpk = (\boldsymbol{A}, \boldsymbol{u}), msk = T, id \in \mathcal{I}D) \longrightarrow \boldsymbol{sk}_{id}$
  $\boldsymbol{H}_{id} \leftarrow H(id)$

$\boldsymbol{A}_{id} \leftarrow \boldsymbol{A} + \left[\mathbf{0}_{d,2d} \;\middle|\; \boldsymbol{H}_{id}\boldsymbol{G} \right]$
$\boldsymbol{sk}_{id} \leftarrow \text{SamplePre}(\boldsymbol{A}_{id}, \boldsymbol{T}, \boldsymbol{H}_{id}, \boldsymbol{u}, \zeta, \alpha)$

– Encrypt$(1^n, mpk = (\boldsymbol{A}, \boldsymbol{u}), id, M) \longrightarrow C$
$\boldsymbol{H}_{id} \leftarrow H(id)$
$\boldsymbol{A}_{id} \leftarrow \boldsymbol{A} + \left[\mathbf{0}_{d,2d} \;\middle|\; \boldsymbol{H}_{id}\boldsymbol{G} \right]$
$\boldsymbol{s} \leftarrow \mathcal{U}(\mathcal{R}_q^d)$, $\boldsymbol{e_0} \leftarrow D_{\mathcal{R}^{m-k},\tau}$, $\boldsymbol{e_1} \leftarrow D_{\mathcal{R}^k,\gamma}$, $e' \longleftarrow D_{\mathcal{R},\tau}$.
$\boldsymbol{b} \leftarrow (\boldsymbol{s}^t\boldsymbol{A}_{id})^t + (\boldsymbol{e_0^t} \mid \boldsymbol{e_1^t})^t$ et $c \leftarrow \boldsymbol{s}^t\boldsymbol{u} + e' + \lfloor q/2 \rfloor M$
$\boldsymbol{C} \leftarrow (\boldsymbol{b}, c) \in \mathcal{R}_q^{m+1}$

– Decrypt$(1^n, \boldsymbol{sk}_{id} = \boldsymbol{x}, C = (\boldsymbol{b}, c)) \longrightarrow M$
$res \leftarrow c - \boldsymbol{b}^t\boldsymbol{x}$.
For each res_i, if it is closer to $\lfloor q/2 \rfloor$ than to 0, $M_i = 1$, otherwise $M_i = 0$.

Correctness. Let us note $\boldsymbol{x} = (\boldsymbol{x}_0^t \mid \boldsymbol{x}_1^t)^t$ with $\boldsymbol{x}_0 \in \mathcal{R}_q^{m-k}$ and $\boldsymbol{x}_1 \in \mathcal{R}_q^k$. Decrypting a ciphertext then needs the error term $e' - (\boldsymbol{e}_0^t \mid \boldsymbol{e}_1^t)(\boldsymbol{x}_0 \mid \boldsymbol{x}_1^t)^t = e' - \boldsymbol{e}_0^t\boldsymbol{x}_0 - \boldsymbol{e}_1^t\boldsymbol{x}_1$ to be bounded by $\lfloor q/4 \rfloor$. This imposes the following condition on the parameters γ and τ (see [BFRS18], Section 4.4) : $t\tau\sqrt{n} + 2t^2\tau\zeta n + t^2\gamma\zeta kn < \lfloor q/4 \rfloor$. Moreover, we take $\gamma = 2t\sigma\tau\sqrt{n}$ so that the security proof of the scheme holds.

Theorem 7. *Our IBE construction with parameters $n, m, q, k, \sigma, \alpha, \gamma, \tau$ and ζ is IND-sID-CPA secure in the standard model under the hardness of $Module - LWE_{n,q,D_{\mathcal{R}},\tau}$.*

E Comparison with competitive signature schemes

Here we compare our signature schemes with the three lattice-based signatures of the second round of the NIST standardization process: Dilithium [DKL+18], qTESLA [ABB+19], and Falcon [FHK+17]. The timings are shown in Table 12. They were obtained using an Intel i7-8650U CPU running at 1.90 GHz. We also explain why GPV is much less efficient than those, which is not only because our implementation is not as optimized.

Table 12. Performances of our signatures and comparison with other schemes (128-bit security parameter sets).

Scheme	KeyGen	Sign	Verify
Dilithium (ROM)	0.04 ms	21 530 op/sec	30 709 op/sec
qTESLA (ROM)	0.33 ms	7 213 op/sec	46 570 op/sec
Falcon (ROM)	6.24 ms	7 789 op/sec	38 647 op/sec
This work (ROM)	7.48 ms	87 op/sec	1 370 op/sec
This work	9.46 ms	64 op/sec	840 op/sec

On the one hand, Fiat-Shamir schemes such as Dilithium and qTESLA do not come with the high cost of trapdoors associated to hash-and-sign lattice-based

schemes. On the other hand, Falcon does use lattice trapdoors, but still remains efficient because it uses a Klein-like sampler and relies on NTRU lattices. This leads to smaller parameters for the scheme, which in turn yields better performances. However, this gain in efficiency comes at the expense of an additional assumption due to the class of lattices used, which is not the case of this article. The two schemes that use discrete Gaussian sampling, qTESLA and Falcon, then require many fewer calls to the discrete Gaussian sampler over $\mathbb{Z}$, which is the main practical cost of our scheme (see Table 1).