Implementation of Lattice Trapdoors on Modules and Applications

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

Univ Rennes, CNRS, IRISA, Rennes, France
lucas.prabel@irisa.fr

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.

J. H. Cheon and J.-P. Tillich (Eds.): PQCrypto 2021, LNCS 12841, pp. 195–214, 2021.
https://doi.org/10.1007/978-3-030-81293-5_11

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 Fig. 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.

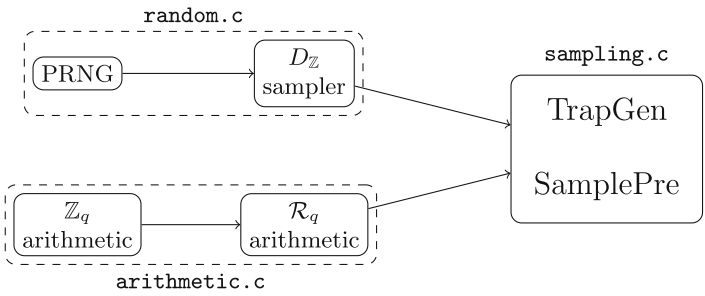

Fig. 1. Basic structure of our implementation and relationships between the blocks.

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 Sect. 3. In Sect. 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 of the full version of the paper.

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 (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 $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. 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. $\boldsymbol{M} \succeq 0$).

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 $\Pr[\|\boldsymbol{x} - \boldsymbol{c}\| > \sigma\sqrt{n}] \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}$.*

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 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \bmod q$, 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* $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e} \bmod q$*, 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 the full version of this article, 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 the full version of the paper. 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 the full version of the paper, 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 of the full version of the paper), which ensures that our samples actually lie in the good coset.
- *Perturbation sampling* with parameters ζ and α (described in Sect. 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 $\boldsymbol{\Sigma}_p = \zeta^2\boldsymbol{I} - \alpha^2 \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{T}^T & \boldsymbol{I} \end{bmatrix}$. 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

function SamplePerturb($\boldsymbol{T} \in \mathcal{P}^{2d\times dk}, \zeta > 0, \alpha > 0$)
  $\boldsymbol{p}_s \leftarrow D_{\mathcal{R}^{dk}, \zeta^2-\alpha^2}$ ▷ $\boldsymbol{p}_s \in \mathcal{R}^{dk}$
  $\boldsymbol{c} \leftarrow -\frac{\alpha^2}{\zeta^2-\alpha^2}\boldsymbol{T}\boldsymbol{p}_s$ ▷ $\boldsymbol{c} \in \mathcal{P}^{2d}$
  $\boldsymbol{\Sigma} \leftarrow \zeta^2\boldsymbol{I} - (\alpha^{-2} - \zeta^{-2})^{-1}\boldsymbol{T}\boldsymbol{T}^T$ ▷ $\boldsymbol{\Sigma} \in \mathcal{P}^{2d\times 2d}$
  for $i = 2d - 1 \ldots 0$ **do**
    $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & f \end{bmatrix}$ ▷ $\boldsymbol{A} \in \mathcal{P}^{i\times i}, \boldsymbol{B} \in \mathcal{P}^{i\times 1}, f \in \mathcal{P}$
    $\boldsymbol{c} = (\boldsymbol{c}', c_i)$ ▷ $\boldsymbol{c}' \in \mathcal{P}^i$, $c_i \in \mathcal{P}$
    $p_i \leftarrow D_{\mathcal{R}, \sqrt{f}, c_i}$ ▷ $p_i \in \mathcal{R}$
    $\boldsymbol{c} \leftarrow \boldsymbol{c}' + f^{-1}\boldsymbol{B}(p_i - c_i)$ ▷ $\boldsymbol{c} \in \mathcal{P}^i$
    $\boldsymbol{\Sigma} \leftarrow \boldsymbol{A} - f^{-1}\boldsymbol{B}\boldsymbol{B}^T$ ▷ $\boldsymbol{\Sigma} \in \mathcal{P}^{i\times i}$
  $\boldsymbol{p} \leftarrow (p_0, \ldots, p_{2d-1}, \boldsymbol{p}_s)$ ▷ $\boldsymbol{p} \in \mathcal{R}^m$
  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 Appendix A.3 of the full version of this paper.

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 Sects. 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 Sect. 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.

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

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 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 full version of this article 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 the full version of the paper.

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

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, Appendix E of the full version of the paper shows that it is not competitive.

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

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 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 the full version of the paper, in Appendix 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 the full version of the paper.

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:

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

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. 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 Sect. 4.1.

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%)

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 the full version of our paper.

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] Alkim, E., Barreto, P.S.L.M., Bindel, N., Longa, P., Ricardini, J.E.: The lattice-based digital signature scheme qtesla. IACR Cryptology ePrint Archive 2019:85 (2019)

[ABB10a] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28

[ABB10b] Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_6

[ADP+16] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. In: USENIX Security Symposium, pp. 327–343. USENIX Association (2016)

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

[BB13] El Bansarkhani, R., Buchmann, J.: Improvement and efficient implementation of a lattice-based signature scheme. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 48–67. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_3

[BFRS18] Bert, P., Fouque, P.-A., Roux-Langlois, A., Sabt, M.: Practical implementation of ring-SIS/LWE based signature and IBE. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 271–291. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_13

[Boy10] Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_29

[CGM19] Chen, Y., Genise, N., Mukherjee, P.: Approximate trapdoors for lattices and smaller hash-and-sign signatures. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_1

[CHK+10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27

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

[DKL+18] Ducas, L., et al.: Crystals-dilithium: a lattice-based digital signature scheme. TCHES **2018**(1), 238–268 (2018)

[DKR+18] D'Anvers, J.-P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2018. LNCS, vol. 10831, pp. 282–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89339-6_16

[DLP14] Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 22–41. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_2

[DP16] Ducas, L., Prest, T.: Fast fourier orthogonalization. In: ISSAC, pp. 191–198. ACM (2016)

[FHK+17] Fouque, P.-A., et al.: Falcon: fast-fourier lattice-based compact signatures over NTRU (2017). https://falcon-sign.info/falcon.pdf

[Fou13] Fouotsa, E.: Calcul des couplages et arithmetique des courbes elliptiques pour la cryptographie. Ph.D. thesis, Rennes 1 (2013)

[GM18] Genise, N., Micciancio, D.: Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7

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

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

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

[GVW13] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC, pp. 545–554. ACM (2013)

[HPS98] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054868

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

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

[LCC14] Lai, R.W.F., Cheung, H.K.F., Chow, S.S.M.: Trapdoors for ideal lattices with applications. In: Lin, D., Yung, M., Zhou, J. (eds.) Inscrypt 2014. LNCS, vol. 8957, pp. 239–256. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16745-9_14

[LLL+13] Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 41–61. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_3

[LM08] Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_3

[LPR10] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1

[LPR13] Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_3

[LS15] Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr. **75**(3), 565–599 (2014). https://doi.org/10.1007/s10623-014-9938-4

[LS18] Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 204–224. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_8

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

[MP12] Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41

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

[MSO17] McCarthy, S., Smyth, N., O'Sullivan, E.: A practical implementation of identity-based encryption over NTRU lattices. In: O'Neill, M. (ed.) IMACC 2017. LNCS, vol. 10655, pp. 227–246. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71045-7_12

[MW17] Micciancio, D., Walter, M.: Gaussian sampling over the integers: efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 455–485. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_16

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

[PR06] Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_8

[Sei18] Seiler, G.: Faster AVX2 optimized NTT multiplication for ringlwe lattice cryptography. IACR Cryptology ePrint Archive 2018:39 (2018)

[Sha84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5

[Sho94] Shor, P.W.: Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In: Adleman, L.M., Huang, M.-D. (eds.) ANTS 1994. LNCS, vol. 877, pp. 289–289. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58691-1_68

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

[SST+09] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_36

[Yam16] Yamada, S.: Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 32–62. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_2

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