Identity-Based Encryption from Lattices Using Approximate Trapdoors

Malika Izabachène[1], Lucas Prabel[2], and Adeline Roux-Langlois[3]

[1] Independent Scholar
[2] Univ Rennes, CNRS, IRISA, Rennes, France
[3] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

Abstract. Practical implementations of advanced lattice-based constructions have received much attention since the first practical scheme instantiated over NTRU lattices, proposed by Prest et al (Asiacrypt 2014). They are using powerful lattice-based building blocks which allow to build Gaussian preimage sampling and trapdoor generation efficiently. In this paper, we propose two different constructions and implementations of identity-based encryption schemes (IBE) using approximate variants of "gadget-based" trapdoors introduced by Chen et al. (Asiacrypt 2019). Both constructions are proven secure.
Our first IBE scheme is an adaptation of the Bert et al. scheme (PQCrypto 2021) to the approximate setting, relying on the Module-LWE hardness assumption and making use of the Micciancio-Peikert paradigm with approximate trapdoors. The second IBE relies on a variant of the NTRU hardness assumption.
We provide several timings and a comparison analysis to explain our results. The two different instantiations give interesting trade-offs in terms of security and efficiency and both benefit from the use of approximate trapdoors. Though our second IBE construction is less efficient than other NTRU-based IBEs, we believe our work provides useful insights into efficient advanced lattice-based constructions.

Keywords: Lattice-based cryptography · approximate trapdoors · Gaussian preimage sampling · module lattices · IBE

1 Introduction

Identity-based encryption (IBE) is an advanced public key encryption scheme in which an identity, such as a username, email address, or social security number, acts as the public key. In identity-based encryption, the sender encrypts a message with the unique identity of the recipient, and the recipient then decrypts the ciphertext with their private key to obtain the original message. In this way, one party can send an encrypted message to any other party without requesting the recipient's public key beforehand. The pair "identity" and "associated secret key" acts as a classical public key and secret key pair in a classical public key encryption scheme.

The idea was to eliminate the need for a public certificate across email systems. These schemes allowed secure communication without exchanging user keys. In [Sha84] Shamir presented a solution for an identity-based signature scheme but the first IBE constructions appeared only in 2001 in [BF01; Coc01] and were based respectively on bilinear maps and quadratic residue-based assumptions. However, these schemes were vulnerable to quantum attacks due to Shor's algorithm.

Lattice-based IBE constructions. In [GPV08], Gentry, Peikert and Vaikuntanathan described the first lattice-based IBE, relying on the Dual-Regev encryption scheme. An important contribution of their work was a sampling algorithm (known as GPV sampling) which showed how to use a short basis as a trapdoor for generating short lattice vectors. This sampler was then used to construct a lattice-based IBE scheme, proven adaptively secure against chosen-plaintext attack in the random oracle model as defined in [BF01; Coc01]. However, the master public key and user secret keys had large sizes in $O(n^2)$ bits. Later on, a construction of a Hierarchical IBE (HIBE) scheme in the standard model was proposed in [CHK$^+$10] based on a new mechanism for users' keys delegation. This IBE scheme was proven secure in the selective model where the adversary needs to target an identity beforehand. In 2010, Agrawal et al. [ABB10] proposed a Learning With Errors (LWE)-based IBE scheme with a trapdoor structure and with performance comparable to the GPV scheme. Their construction viewed an identity as a sequence of bits and then assigned a matrix to each bit. It used a sampling algorithm to obtain a basis with low Gram-Schmidt norm for the master secret key and formed a lattice family with two associated trapdoors to generate short vectors; one for all lattices in the family and the other one for all but one.

The first Ring-LWE based IBE scheme has been proposed by Ducas, Lyubashevsky and Prest [DLP14] (DLP-IBE), which is still considered the most efficient scheme to date due to smaller key sizes. The use of the ring variant increased efficiency by reducing the public key size and ciphertext size to $O(n)$. The security of their scheme holds in the random oracle model and is related to the NTRU hardness assumption. An efficient C implementation of the DLP-IBE scheme and a detailed performance analysis was provided in [MSO17]. In 2017, Campbell and Grover introduced a HIBE scheme, called LATTE, which can be viewed as a combination of the DLP scheme with the delegation mechanism from [CHK$^+$10]. An optimized implementation and refined analysis of LATTE, has recently been proposed in [ZMS$^+$21].

The work from [BFRS18] constructed an IBE using the notion of gadget-based trapdoors in the ring setting, introduced by [MP12]. Such trapdoors can be seen as linear transformations mapping hard instances of cryptographic problems on some lattices to easy instances on a lattice defined by a public "gadget matrix". The IBE from [BFRS18] also made use of the efficient Gaussian preimage sampling algorithms from [GM18] to propose an implementation of their scheme. In [BEP$^+$21], this IBE and its associated sampling algorithms were

adapted to the module setting and instantiated. The use of module lattices of dimension nd, where d is the rank module, led to a more flexible choice of parameters. In [ZY22], the authors proposed new efficient gadget sampling algorithms which didn't need floating-point arithmetic, and as fast as the original [GM18] sampler.

All those constructions generally make use of dedicated trapdoors, needed by the authority to generate the secret key of a user. In that case, building the trapdoor and sampling particular short vectors are quite costly, and represent the main bottleneck in the efficiency of such schemes. More recently, [CGM19] introduced approximate trapdoors to improve the efficiency of schemes built from lattice trapdoors while keeping the same concrete security. [GL20] showed that those approximate trapdoors, relying on the [MP12] framework, exist on a family of NTRU lattices. Our work explores the application of those approximate trapdoors, and in particular of this family of NTRU lattices, to more advanced schemes where the generation of private keys in a multi-user setting is needed.

Our contributions. Our main contribution is to provide and implement two lattice-based IBE schemes (IBE1 and IBE2), which make use of families of gadget-based approximate trapdoors. Our two constructions rely on the LWE problem over modules (Module-LWE) and the inhomogeneous NTRU problem (iNTRU) respectively. We investigate how to instantiate and parametrize the approximate trapdoor preimage sampling over these two families of trapdoors in a way to obtain provable and efficient quantum-safe IBE schemes. The IBE1 construction is an adaptation of the identity-based encryption scheme from [ABB10; BFRS18] to the module setting, using approximate trapdoors. The IBE2 construction follows the same blueprint as the DLP scheme except that it makes use of iNTRU gadget-based approximate trapdoors. As in previous IBE constructions, encryption is based on the Dual-Regev encryption scheme. We provide a complete public and open-source C implementation[4] with performance benchmarking. The implementation is modular and makes it easy to change the building blocks of our algorithms according to the desired properties that we want to get.

Our work explores the use of approximate trapdoors for the construction of IBE schemes and the potential practical insights we can gain from it. We first adapted the construction of [BEP+21] using approximate trapdoors in our IBE1. The error induced by the approximate setting requires changes at several levels, either for the choice of encoding or for the sampling algorithms. As expected, we obtain better timings for all four algorithms composing our IBE by using approximate trapdoors rather than exact ones.

The second scheme IBE2 makes use of approximate trapdoors relying on a variant of NTRU rather than Module-LWE. Our approach was motivated by the fact that other efficient IBEs, such as DLP and Latte, used the NTRU hardness assumption. However, unlike us, these last two schemes used the GPV paradigm

[4] `https://github.com/lucasprabel/approx_lattice`

to generate trapdoors, which significantly changes the way their schemes are constructed compared to ours.

More details on our implementation choices. Our IBE1 proof relies on a statistical trapdoor instantiation. Although the size of the parameters increases consequently, the use of approximate trapdoors allowed us to mitigate this loss in efficiency induced by the use of a statistical instantiation. In order to ensure decryption correctness, we also need to use a large modulus (see Section 4.1). This leads us to perform calculations carefully on 64-bit integers, so as not to affect our scheme efficiency. The IBE1 also makes use of a small-norm encoding, instead of the low-degree encoding used in [BEP+21] to ensure that the noise is still not too large in order to decrypt. The encoding we use sets constraints on the structure of the ring $\mathcal{R}_q$ which is not compatible with the NTT for polynomial multiplications. Instead, we use a "partial NTT" based on [LS18] results, which reduces multiplication in $\mathcal{R}_q$ to multiplication in smaller rings. We also have optimized the underlying CRT representation algorithm compared to [BEP+21]. Finally, the sampling algorithms have been adapted to the approximate setting. Table 2 shows some applicable parameter sets together with their concrete bit security using the LWE estimator from [APS15] with BKZ as a reduction cost model. All the algorithms comprising the IBE1 over module lattices are more efficient than their exact counterpart at the same security level. This improvement concerns in particular Setup and Extract which are optimized by a factor ≈ 1.5. We give more details in Section 4.2.

The IBE2 scheme is instantiated using gadget-based trapdoors on a family of NTRU lattices. Table 4 provides a set of applicable parameter sets together with their concrete bit security. The computational trapdoors instantiation lowers the bounds on parameters required for the correctness compared to the IBE1 scheme, which allows the use of smaller moduli. For the IBE2 construction, an identity is encoded as $\mathsf{H}(\mathsf{id})$ with H having special properties so that we are able to respond to private key queries except for the target identity chosen by the adversary.

Table 1: Timings comparison in ms of the different operations of the IBE2 scheme in this paper and in Latte [ZMS+21] for different sets of parameters.

Scheme	$(n, \lceil log_2(q) \rceil)$	Security level	Setup	Extract	Encrypt	Decrypt
[ZMS+21]	$(1024, 24)$	128	102	0.82	0.05	0.06
[ZMS+21]	$(2048, 25)$	256	292	2.62	0.10	0.13
This paper	$(1024, 25)$	159	3.32	5.92	1.10	0.07
This paper	$(2048, 25)$	293	10.21	12.79	2.96	0.16

In Table 1, we provide timings comparison with the [ZMS+21] IBE scheme. A more complete and detailed comparison is given in Section 5.2 between our IBE2 scheme and the [DLP14] and [ZMS+21] IBE schemes which also rely on the NTRU assumption. The use of approximate trapdoors and of the iNTRU assumption allows to obtain better timings for the Setup algorithm and for the Extract

algorithm for some sets of parameters. Unfortunately, we obtain an overhead for encryption which essentially comes from the Gaussian sampling phase. We can save a factor 3.5 using binomial samples as in [DLP14] and [ZMS+21] but there is still an overhead due to the extra number of samples we need. We make $n(2+m)$ calls to the integer Gaussian sampler while encryption in [DLP14] and [ZMS+21] makes use of $3n$ binomial sampling calls, where n is the dimension of the underlying polynomial ring and m is the size of one of the vector used in encryption. As an example, we obtain a timing of 0.87ms for one encryption against 0.13 for [ZMS+21] for a security level of 128 bits.

Organization of the paper. The paper is structured as follows: Section 2 reviews the necessary linear algebra and lattice backgrounds. The Section also presents the notions of approximate trapdoor. Then, Section 4 and Section 5 contain a detailed description of the two IBEs, based on the Module-LWE hardness assumption and based on a variant of the NTRU assumption respectively. Both Sections provide a security analysis and detailed performance results together with a comparison of their respective analogue in terms of lattice-based building blocks and assumptions.

2 Preliminaries

Notations. Throughout the paper, the vectors are written in lowercase bold letters (e.g. $\boldsymbol{v}$) and matrices in uppercase bold letters (e.g. $\boldsymbol{A}$). We refer to their entries with a subscript index v_i, $A_{i,j}$. We denote $\|\cdot\|$ and $\|\cdot\|_\infty$ the euclidean norm and the infinity norm respectively. The norm of a vector over $\mathbb{Z}_q$ is the norm of the corresponding vector over $\mathbb{Z}$ obtained by choosing its entries in the set $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. The norm of a polynomial $a = \sum_{i=0}^{n-1} a_i X^i$ is the norm of the coefficients vector $(a_0, \ldots, a_{n-1})$. Finally, the norm of a matrix is the maximum norm of its column vectors. If x is sampled from a distribution D, we write $x \leftarrow D$. We denote $\mathcal{U}(S)$ the uniform distribution over a finite set S.

We say that a function f is negligible when $f(n) = o(n^{-c})$ for all $c > 0$ as $n \to \infty$. An event is said to happen 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. Two distributions are said to be computationally indistinguishable if no probabilistic polynomial time algorithm can distinguish them with non-negligible advantage.

2.1 Lattices background and discrete Gaussian distributions.

Lattices. A lattice Λ in $\mathbb{R}^m$ is the set $\{ \sum_{i=1}^{k} \lambda_i \boldsymbol{b}_i, \lambda_i \in \mathbb{Z} \}$ of all integer linear combinations of some linearly independent vectors $\boldsymbol{B} = \{ \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \} \subset \mathbb{R}^m$. We call k the rank of the lattice and m its dimension. When $k = m$, the lattice is said to be full-rank.

Given $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{u} \in \mathbb{Z}_q^n$, we define the following m-dimensional q-ary lattice $\Lambda_q^\perp(\boldsymbol{A}) = \{\, \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \bmod q \,\}$, and its coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{\, \boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q \,\}$.

Module lattices are particular lattices that have a polynomial structure. We denote d the module rank of those lattices. When $d = 1$, module lattices are in fact ideal lattices. 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 $\mathcal{R}^m$, which is isomorphic to the integer lattice $\mathbb{Z}^{nm}$.

Gaussian distributions. We recall that a symmetric matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is positive definite (resp. positive semidefinite) if $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} > 0$ (resp. $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} \geq 0$) for all nonzero $\boldsymbol{x} \in \mathbb{R}^n$. In this case we write $\boldsymbol{M} \succ 0$ (resp. $\boldsymbol{M} \succeq 0$). We say that $\boldsymbol{M} \succeq \boldsymbol{N}$ when $\boldsymbol{M} - \boldsymbol{N} \succeq 0$, and write $\boldsymbol{M} \succeq \eta$ instead of $\boldsymbol{M} \succeq \eta\boldsymbol{I}_n$ when $\eta \geq 0$ is a real. The spherical continuous Gaussian function of center $\boldsymbol{c} \in \mathbb{R}^n$ and parameter σ is defined on $\mathbb{R}^n$ by $\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x}) = \exp(-\frac{\pi\|\boldsymbol{x}-\boldsymbol{c}\|^2}{\sigma^2})$. For a positive definite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$, we also define the (skewed) Gaussian function $\rho_{c,\sqrt{\boldsymbol{\Sigma}}}(\boldsymbol{x}) = \exp(-\pi(\boldsymbol{x}-\boldsymbol{c})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{c}))$. Then, for a full-rank lattice $\Lambda \subset \mathbb{Z}^n$, we denote $D_{\Lambda,\sigma,\boldsymbol{c}}$ (respectively $D_{\Lambda,\sqrt{\boldsymbol{\Sigma}},\boldsymbol{c}}$) the spherical (resp. ellipsoid) discrete Gaussian distribution of center $\boldsymbol{c} \in \mathbb{R}^n$ and parameter $\sigma > 0$, associated to the density $\rho_{\boldsymbol{c},\sigma}$ (resp. $\rho_{c,\sqrt{\boldsymbol{\Sigma}}}$).

Smoothing parameter. The smoothing parameter of a lattice Λ, denoted $\eta_\varepsilon(\Lambda)$, was first introduced in [MR07]. Lemma 1 gives an upper bound on 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 will use the following result to bound the euclidean norm of vectors that follow a discrete Gaussian distribution.

Lemma 2 ([Lyu12, Lemma 4.4]). *For any integer $n \geq 1$ and reals $\sigma > 0$, $t > 1$, $\Pr[\|\boldsymbol{x}\| > t\sigma\sqrt{n} \mid \boldsymbol{x} \leftarrow D_{\mathbb{Z}^n,\sigma}] < t^n e^{\frac{n}{2}(1-t^2)}$.*

We call t a tailcut of the discrete Gaussian of parameter σ when a vector $\boldsymbol{x}$ sampled from $D_{\mathbb{Z}^n,\sigma}$ verifies the inequality $\|\boldsymbol{x}\| \leq t\sigma\sqrt{n}$ on its euclidean norm with overwhelming probability.

2.2 Cryptographic problems on lattices

We work 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 a prime modulus. Let us now define the hard problems on which our constructions rely. In most practical lattice-based schemes, security is based on structured variants of LWE rather than on LWE itself. In the Module-SIS and Module-LWE variants, the parameter d is the rank of the module, and nd

is the dimension of the corresponding module lattice. The hardness of those problems is proven by worst-case to average-case reductions from hard problems on module lattices (see [LS15]).

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

Definition 2 ($\mathsf{Decision\ Module\text{-}LWE}_{n,d,q,\sigma}$)**.** *Given a uniform* $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$ *and the vector* $\boldsymbol{b}^T = \boldsymbol{s}^T\boldsymbol{A} + \boldsymbol{e}^T \mod q \in \mathcal{R}_q^m$*, 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* $\mathcal{R}_q^{d\times m} \times \mathcal{R}_q^m$*.*

As stated in [BJR+23], the Module-LWE problem remains hard when the short secret $\boldsymbol{s}$ is sampled from the Gaussian distribution $D_{\mathcal{R}^d,\sigma}$.

Note that $\mathsf{Module\text{-}LWE}_{n,d,q,\sigma}$ with rank $d = 1$ corresponds to the ring version of LWE, which will be denoted $\mathsf{Ring\text{-}LWE}_{n,q,\sigma}$. We define $\boldsymbol{G} = \boldsymbol{I}_d \otimes \boldsymbol{g}^T$ and its associated lattice named the module $\boldsymbol{G}$-lattice, for which the Module-SIS problem is easy.

To build our second IBE, we use gadget-based trapdoors whose pseudorandomness is based on the inhomogeneous NTRU problem (iNTRU), a variant of NTRU introduced in [GGH+19].

Definition 3 ($\mathsf{iNTRU}_{q,\chi}$)**.** *Let* k, q *be integers and* χ *a distribution over* $\mathcal{R}$*. The input of the* $\mathsf{iNTRU}_{q,\chi}$ *problem is a vector* $\boldsymbol{a} \in \mathcal{R}_q^k$ *which is either taken uniform in* $\mathcal{R}_q^k$ *or either set as* $\boldsymbol{a} = r^{-1}(\boldsymbol{g} + \boldsymbol{e})$ *where* $(r, \boldsymbol{e})$ *is drawn from* χ^{k+1}*. The goal is to decide which is the case.*

In their paper, the authors from [GGH+19] showed a reduction of a matrix-variant of iNTRU called MiNTRU from the non-standard Ring-LWE with a trapdoor oracle access problem (more details are given in [GGH+19, Section 4.3]). The reduction can be adapted to the iNTRU problem as well.

2.3 Encoding identities with full-rank differences

We use the notion of full-rank differences encoding (FRD) in our first scheme, with different properties that the ones used in [ABB10].

Definition 4 ([BEP+21]). *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.*

As shown in [BEP+21], we construct an FRD encoding in the module setting (i.e. over $\mathcal{R}_q^{d\times d}$), by first constructing one in the ring setting (i.e. over $\mathcal{R}_q$). The encodings used in our IBE scheme impose a structure on the ring $\mathcal{R}_q$ which is not compatible with the Number Theoretic Transform (NTT). To speed up polynomial multiplications and to mitigate the loss of performance due to the inability of using the NTT, we use ideas from [LS18]. This method can be thought of as a "partial NTT". It is based on the following result.

Lemma 3 ([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, if $y \in \mathbb{Z}_q[X]/(X^n + 1)$ satisfies $0 < \|y\|_\infty < \frac{q^{1/r}}{\sqrt{r}}$, then y has an inverse in $\mathbb{Z}_q[X]/(X^n + 1)$.*

This lemma was used in [BEP+21] to build a "low-degree" FRD (the identities were encoded as polynomials of low-degree). In our case, we construct a "small-norm" FRD, encoding polynomials with ℓ_∞-norm smaller than $\frac{q^{1/r}}{2\sqrt{r}}$. We make use of the "small-norm" FRD described in Proposition 1 rather than "low-degree" FRD encoding so that the correctness of the decryption algorithm still holds when using approximate trapdoors, because they introduce an additional error term that must be bounded.

Proposition 1. *Let $n \geq r > 1$ be powers of 2, q a prime such that $q \equiv 2r + 1 \pmod{4r}$ and $1 \leq D \leq \frac{q^{1/r}}{2\sqrt{r}}$ an integer. We define $\mathcal{M} = \{-D, \ldots, D\}^n \setminus \{(0, \ldots, 0)\}$ the set of identities. Then the following map $H_M : \mathcal{M} \longrightarrow \mathcal{R}_q$, such that $H_M(m_0, \ldots, m_{n-1}) = \sum_{i=0}^{n-1} m_i X^i$ is an FRD encoding.*

Proof. We have $\|H_M(m)\|_\infty \leq D < \frac{q^{1/r}}{2\sqrt{r}}$ for all m because of the choice of $\mathcal{M}$. So according to Lemma 3, $H_M(m)$ is invertible. For all $m_1, m_2 \in \mathcal{M}$, we also have $\|H_M(m_1) - H_M(m_2)\|_\infty \leq 2D < \frac{q^{1/r}}{\sqrt{r}}$ so $H_M(m_1) - H_M(m_2)$ is invertible. Finally, H_M is an FRD encoding. □

FRD on modules. As explained in [BEP+21], FRD encoding in the module setting can be built using an existing FRD encoding in the ring setting $H_M : \mathcal{M} \longrightarrow \mathcal{R}_q$ by constructing $H_M(m) \cdot \boldsymbol{I}_d \in R_q^{d \times d}$ for $m \in \mathcal{M}$ and $\boldsymbol{I}_d \in \mathcal{R}_q^{d \times d}$ is the identity matrix.

3 Trapdoors on lattices

The two IBE constructions make use of efficient trapdoors, called gadget-trapdoors, introduced in [MP12]. Such trapdoors were generalized to ideal lattices in [LCC14] and to module lattices in [BEP+21]. An efficient instantiation of the sampling algorithms was given in [GM18]. All those results allow us to efficiently instantiate trapdoor generation and sampling algorithms. Having introduced the $\boldsymbol{G}$-lattice in the previous Section, we can now define the associated $\boldsymbol{G}$-trapdoors.

Definition 5. *A $\boldsymbol{G}$-trapdoor for a matrix $\boldsymbol{A} \in \mathcal{R}_q^{d \times m}$ is a matrix $\boldsymbol{R} \in \mathcal{R}^{(m-dk) \times dk}$ such that*

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

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

The knowledge of the trapdoor $\boldsymbol{R}$ allows one to compute small vectors on any coset of a given lattice, by sampling on the $\boldsymbol{G}$-lattice first, and then by using $\boldsymbol{R}$ to map the sample to a small vector in the given lattice.

3.1 Approximate trapdoors for module lattices.

As shown in [CGM19], the gadget trapdoor proposed by Micciancio and Peikert can be modified to an approximate trapdoor, in a way that further reduces the sizes of the public matrix, the trapdoor and the preimage.

Definition 6 ($\mathsf{AISIS}_{n,d,m,q,\alpha,\beta}$)**.** *For any $n, d, m, q \in \mathbb{N}$ and $\alpha, \beta \in \mathbb{R}$, the approximate inhomogeneous short integer solution problem $\mathsf{AISIS}_{n,d,m,q,\alpha,\beta}$ is defined as follows: given $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}, \boldsymbol{y} \in \mathcal{R}_q^d$, find a vector $\boldsymbol{x} \in \mathcal{R}^m$ such that $||\boldsymbol{x}|| \leq \beta$ and such that there is a vector $\boldsymbol{z} \in \mathcal{R}^d$ satisfying: $||\boldsymbol{z}|| \leq \alpha$ and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{z}$ mod q.*

Definition 7 (Approximate Trapdoor). *A string τ is called an (α, β)-approximate trapdoor for a matrix $\boldsymbol{A} \in \mathcal{R}_q^{d\times m}$ if there is a probabilistic polynomial time algorithm that given $\tau, \boldsymbol{A}$ and any $\boldsymbol{y} \in \mathcal{R}_q^d$, outputs a non-zero vector $\boldsymbol{x} \in \mathcal{R}^m$ such that $||\boldsymbol{x}|| \leq \beta$ and there is a vector $\boldsymbol{z} \in \mathcal{R}^d$ satisfying $||\boldsymbol{z}|| \leq \alpha$ and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{z}$ mod q.*

In practice, we generate approximate trapdoors by dropping ℓ entries corresponding to the small powers of b from the gadget matrix $\boldsymbol{G}$ to get the following matrix: $\boldsymbol{F} = \boldsymbol{I}_d \otimes \boldsymbol{f}^T = \boldsymbol{I}_d \otimes \left[b^\ell\ b^{\ell+1}\ b^{\ell+2}\ \cdots\ b^{k-1}\right] \in \mathcal{R}^{d\times d(k-\ell)}$.

3.2 Module-LWE Approximate trapdoors.

In [CGM19], the authors described the approximate trapdoor generation and the approximate preimage sampling algorithms, adapted from [MP12] by making use of the approximate gadget matrix $\boldsymbol{F}$ instead of $\boldsymbol{G}$. Its trapdoor generation and trapdoor preimage sampling algorithms over modules are recalled in Figure 1 and the result of the approximate sampling is stated in Theorem 1.

Theorem 1 ([CGM19, Theorem 4.1]). *There exists probabilistic, polynomial time algorithms* ApproxTrapGen1$(\cdot)$ *and* ApproxSamplePre1$(\cdot)$ *such that:*

1. ApproxTrapGen1$(\boldsymbol{H}, \sigma)$ *takes as input public parameters, a tag matrix $\boldsymbol{H} \in \mathcal{R}^{d\times d}$ and parameter $\sigma > 0$ and returns a matrix-approximate trapdoor pair $(\boldsymbol{A}, \boldsymbol{R}) \in \mathcal{R}_q^{d\times m} \times \mathcal{R}^{\bar{m}\times\omega}$ where $\boldsymbol{R}$ coefficients are drawn from a Gaussian distribution of parameter σ over $\mathcal{R}$.*
2. *Let $\boldsymbol{A}$ be generated with an approximate trapdoor as above. The following two distributions are statistically indistinguishable:*

$\{(\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) \mid \boldsymbol{u} \leftarrow \mathcal{U}(\mathcal{R}_q^d), \boldsymbol{x} \leftarrow \text{ApproxSamplePre1}(\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{0}, \boldsymbol{u}, \zeta), \boldsymbol{y} = \boldsymbol{u} - \boldsymbol{A}\boldsymbol{x}\ mod\ q\}$
and $\{(\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) \mid \boldsymbol{x} \leftarrow D_{\mathcal{R}^m,\zeta}, \boldsymbol{y} \leftarrow D_{\mathcal{R}^d,\sigma\sqrt{(b^{2\ell}-1)/(b^2-1)}}, \boldsymbol{u} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}\ mod\ q\}$

for any $\sigma \geq \sqrt{b^2+1} \cdot \omega(\sqrt{\log d})$ *and* $\zeta \gtrsim \sqrt{b^2+1}\frac{s_1^2(\boldsymbol{R})}{s_{2d}(\boldsymbol{R})}\eta_\epsilon(\mathbb{Z}^{dk})$.

3.3 iNTRU Approximate trapdoors.

In [GL20], Genise and Li introduced a family of Ring-SIS approximate trapdoors whose pseudorandomness is based on the iNTRU problem and then showed that the efficient gadget-based trapdoor framework of [MP12] exists on a family of NTRU lattices. Their trapdoor scheme enjoys small secret keys and is compatible with applications requiring tag matrices.

In their second trapdoor scheme, the matrix $\begin{bmatrix} -\boldsymbol{e}^T \\ r\boldsymbol{I} \end{bmatrix}$ is used as a $\boldsymbol{f}$-trapdoor for $\begin{bmatrix} 1 & \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} 1 & r^{-1}(\boldsymbol{f}^T + \boldsymbol{e}^T) \end{bmatrix}$ where $(r, \boldsymbol{e}) \longleftarrow \chi^{k-\ell+1}$ is drawn from a distribution with small entries and $\boldsymbol{f}$ is the approximate gadget vector. They got the following results, similar from the ones from [CGM19].

ApproxTrapGen1($\boldsymbol{H}, \sigma$)
1. $\bar{\boldsymbol{A}} \leftarrow \mathcal{U}(\mathcal{R}_q^{d \times \bar{m}})$
2. $\boldsymbol{R} \leftarrow D_{\mathcal{R}^{\bar{m} \times \omega}, \sigma}$
3. set $\boldsymbol{A} = [\, \bar{\boldsymbol{A}} \mid \boldsymbol{HF} - \bar{\boldsymbol{A}}\boldsymbol{R} \,] \in \mathcal{R}_q^{d \times m}$
4. return $(\boldsymbol{A}, \boldsymbol{R})$

ApproxSamplePre1($\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{H}, \boldsymbol{u}, \zeta$)
1. sample perturbation $\boldsymbol{p} \leftarrow D_{\mathcal{R}^m, \sqrt{\boldsymbol{\Sigma_p}}}$
2. set coset $\boldsymbol{v} = \boldsymbol{H}^{-1}(\boldsymbol{u} - \boldsymbol{Ap})$
3. sample $\boldsymbol{z} = (\boldsymbol{z_1}^T, \boldsymbol{z_2}^T)^T \leftarrow D_{\Lambda_q^{\boldsymbol{v}}(\boldsymbol{G}), \sigma_g}$
4. set $\boldsymbol{x} = \boldsymbol{p} + \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{z_2} \in \mathcal{R}^m$
5. return $\boldsymbol{x}$

ApproxTrapGen2(σ)
1. sample $r \longleftarrow \mathcal{D}_{\mathcal{R}, \sigma}$, $\boldsymbol{e} \longleftarrow \mathcal{D}_{\mathcal{R}^m, \sigma}$
2. set $\boldsymbol{a}' = r^{-1}(\boldsymbol{f} + \boldsymbol{e}) \in \mathcal{R}^m$
3. set $\boldsymbol{a} = (1, \boldsymbol{a}') \in \mathcal{R}_q^{m+1}$
4. set $\boldsymbol{R} = \begin{bmatrix} -\boldsymbol{e}^T \\ r\boldsymbol{I}_m \end{bmatrix} \in \mathcal{R}^{(m+1) \times m}$
5. return $(\boldsymbol{a}, \boldsymbol{R})$

ApproxSamplePre2($\boldsymbol{a}, \boldsymbol{R}, u, \zeta$)
1. sample perturbation $\boldsymbol{p} \leftarrow D_{\mathcal{R}^{m+1}, \sqrt{\boldsymbol{\Sigma_p}}}$
2. set coset $v = (u - \boldsymbol{a}^T\boldsymbol{p}) \in \mathcal{R}_q$
3. sample $\boldsymbol{z} = (\boldsymbol{z_1}^T, \boldsymbol{z_2}^T)^T \leftarrow D_{\Lambda_q^v(\boldsymbol{g}^T), \sigma_g}$
4. set $\boldsymbol{x} = \boldsymbol{p} + \boldsymbol{Rz_2} \in \mathcal{R}^{m+1}$
5. return $\boldsymbol{x}$

Fig. 1: Description of the approximate trapdoor generation and preimage sampling algorithms based on [CGM19] (on the left hand side) and [GL20] (on the right hand side) instantiations, where the perturbation is sampled with parameter $\boldsymbol{\Sigma_p} = \zeta^2\boldsymbol{I}_m - \sigma_g^2 \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^T & \boldsymbol{I} \end{bmatrix}$ on the left side and $\boldsymbol{\Sigma_p} = \zeta^2\boldsymbol{I}_{m+1} - \sigma_g^2\boldsymbol{RR}^T$ on the right side. Both are independent of the target $\boldsymbol{u}$. We refer to [CGM19; GL20] for more information on the iNTRU based trapdoor generation.

Theorem 2. *Let $r \leftarrow \chi$ and $\boldsymbol{e}^T \leftarrow \chi^m$ and set the trapdoor function description as $\boldsymbol{a} = \begin{bmatrix} 1 & \boldsymbol{a}' \end{bmatrix} = \begin{bmatrix} 1 & r^{-1}(\boldsymbol{f} + \boldsymbol{e}) \end{bmatrix} \in \mathcal{R}_q^{m+1}$. Let $\eta = \eta_\epsilon(\mathbb{Z}^{n \times m})$ and $\sigma_g = \eta_e(\Lambda_q^\perp(\boldsymbol{g}^T)) \geq \sqrt{b^2+1} \cdot \eta_\epsilon(\mathbb{Z}^{n \times m})$ for some $\epsilon \in (0,1)$ and $\zeta \geq \sqrt{\sigma_g^2\boldsymbol{RR}^T + \eta^2\boldsymbol{I}_{m+1}}$. Then, the following distributions are within a max-log distance $3\log\frac{1+\epsilon}{1-\epsilon} \leq \frac{6\epsilon}{1-\epsilon}$:*

$$\{(\boldsymbol{a}, \boldsymbol{x}, u, y) \mid u \leftarrow \mathcal{U}(\mathcal{R}_q), \boldsymbol{x} \leftarrow \text{ApproxSamplePre2}(\boldsymbol{a}, \boldsymbol{R}, u, \zeta), y = u - \boldsymbol{a}^T\boldsymbol{x} \in \mathcal{R}_q\}$$

and $\{(\boldsymbol{a}, \boldsymbol{x}, u, y) \mid \boldsymbol{x} \leftarrow D_{\mathcal{R}^{m+1}, \zeta}, y \leftarrow D_{\mathcal{R}, \sigma_e} \mod q, u = \boldsymbol{a}^T\boldsymbol{x} + y \in \mathcal{R}_q\}$

for $\sigma_e = \sigma_g\sqrt{(b^{2\ell}-1)/(b^2-1)}$.

4 An Approximate-IBE scheme on modules

The first IBE scheme we present is the IBE scheme from [ABB10; BFRS18] adapted to the module setting, instantiated using the approximate trapdoors from [CGM19]. At a high level, the scheme will make use of the following blocks:

- The master secret key is a $\boldsymbol{F}$-approximate trapdoor $\boldsymbol{R} \in \mathcal{R}^{\bar{m} \times \omega}$ associated to $\boldsymbol{A}$ with tag $\mathbf{0}$, with subgaussian coefficients of parameter σ, with $\omega = d(k-\ell)$. The master public key is a tuple consisting of a uniformly random vector $\boldsymbol{u} \in \mathcal{R}_q^d$ and the matrix $\boldsymbol{A} \in \mathcal{R}_q^{d \times m}$, with $m = \bar{m} + \omega$ chosen as: $\boldsymbol{A} = \left[\bar{\boldsymbol{A}} \;\middle|\; -\bar{\boldsymbol{A}}\boldsymbol{R} \right]$. Taking $\bar{m} = d \log q$, we get that $\bar{\boldsymbol{A}}$ is full rank with high probability according to [BJR+23, Lemma 2.6]. Moreover, by taking $\sigma > 4 \cdot 4^{\frac{1}{nd}} \sqrt{n}$, we obtain that $\boldsymbol{A}$ is statistically close to uniform by Corollary 7.5 from [LPR13].
- A "small-norm" FRD encoding H_M as described in Section 2.3. This allows anyone, with the knowledge of the master public key $\boldsymbol{A}$, to compute a public matrix $\boldsymbol{A}_{\mathsf{id}} = \boldsymbol{A} + \left[\mathbf{0}_{d,\bar{m}} \;\middle|\; \boldsymbol{H}_{\mathsf{id}}\boldsymbol{F} \right]$ associated to the identity id of a user. Then, the secret key for id is an approximate short vector $\boldsymbol{x}_{\mathsf{id}} \in \mathcal{R}^m$ obtained by using the ApproxSamplePre1 algorithm. Such a vector satisfies the relation $\boldsymbol{A}_{\mathsf{id}}\boldsymbol{x}_{\mathsf{id}} \approx \boldsymbol{u} \; mod \; q$. We can bound the approximate error in this relation by using Theorem 1 and the fact that we use a small-norm FRD encoding.
- Finally, we use the Dual-Regev encryption scheme for the encryption and decryption algorithms, taking care of the additional error which appears when using approximate trapdoors.

4.1 Construction

We detail the first IBE where users' keys are defined based on the approximate preimage sampling from Theorem 1 and encryption is based on the Dual-Regev scheme.

- $\text{Setup}(1^n) \longrightarrow (\mathsf{mpk}, \mathsf{msk})$:
  - $(\boldsymbol{A}, \boldsymbol{R}) \leftarrow \text{ApproxTrapGen1}(\mathbf{0}, \sigma) \in \mathcal{R}_q^{d \times m} \times \mathcal{R}^{\bar{m} \times w}$, $\boldsymbol{u} \leftarrow \mathcal{U}(\mathcal{R}_q^d)$;
  - $\mathsf{mpk} = (\boldsymbol{A}, \boldsymbol{u})$, $\mathsf{msk} = \boldsymbol{R}$.
- $\text{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \longrightarrow \mathsf{sk}_{\mathsf{id}}\ \boldsymbol{x} \in \mathcal{R}^m$:
  - $\boldsymbol{H}_{\mathsf{id}} \leftarrow H_M(\mathsf{id})$; $\boldsymbol{A}_{\mathsf{id}} \leftarrow \boldsymbol{A} + \left[\mathbf{0}_{d,\bar{m}} \;\middle|\; \boldsymbol{H}_{\mathsf{id}}\boldsymbol{F} \right] \in \mathcal{R}_q^{d \times m}$;
  - $\boldsymbol{x} \leftarrow \text{ApproxSamplePre1}(\boldsymbol{A}_{\mathsf{id}}, \boldsymbol{R}, \boldsymbol{H}_{\mathsf{id}}, \boldsymbol{u}, \zeta)$;
- $\text{Encrypt}(\mathsf{mpk}, \mathsf{id}, M) \longrightarrow C = (\boldsymbol{b}, c) \in \mathcal{R}_q^{m+1}$:
  - $\boldsymbol{H}_{\mathsf{id}} \leftarrow H_M(\mathsf{id})$; $\boldsymbol{A}_{\mathsf{id}} \leftarrow \boldsymbol{A} + \left[\mathbf{0}_{d,\bar{m}} \;\middle|\; \boldsymbol{H}_{\mathsf{id}}\boldsymbol{F} \right] \in \mathcal{R}_q^{d \times m}$;
  - $\boldsymbol{s} \leftarrow D_{\mathcal{R}_q^d, \tau}$, $\boldsymbol{e_0} \leftarrow D_{\mathcal{R}^{m-w}, \tau}$, $\boldsymbol{e_1} \leftarrow D_{\mathcal{R}^w, \gamma}$, and $e' \longleftarrow D_{\mathcal{R}, \tau}$;
  - $\boldsymbol{b} \leftarrow (\boldsymbol{s}^T\boldsymbol{A}_{\mathsf{id}})^T + (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)^T$ and $c \leftarrow \boldsymbol{s}^T\boldsymbol{u} + e' + \lfloor q/2 \rfloor M$;
- $\text{Decrypt}(\mathsf{sk}_{\mathsf{id}}, C) \rightarrow M$:

- set $\boldsymbol{x} = \mathsf{sk}_{\mathsf{id}}$ and compute $\mathsf{res} \leftarrow c - \boldsymbol{b}^T\boldsymbol{x}$ which has integer coefficients;
- for each i, if the coefficient $\mathsf{res}_i \in \mathbb{Z}$ is closer to $\lfloor q/2 \rfloor$ than to 0, then $M_i = 1$, otherwise $M_i = 0$.

Correctness. To use approximate trapdoors with the Dual-Regev approach, we need to sample the LWE secret term with a small norm instead of sampling from the uniform distribution, in order to maintain the correctness of the schemes. Let's write $\boldsymbol{y} \in \mathcal{R}_q^d$ the additional error we get by using approximate trapdoors instead of exact ones. The correctness of the decryption holds if the error term $\|e' - (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\boldsymbol{x} - \boldsymbol{s}^T\boldsymbol{y}\|$ is small enough, i.e. less than $\lfloor q/4 \rfloor$.

$$\begin{aligned}
\mathsf{res} &= c - \boldsymbol{b}^T\boldsymbol{x} \\
&= \boldsymbol{u}\cdot\boldsymbol{s} + e' + \lfloor q/2 \rfloor M - \left[(\boldsymbol{s}^T\boldsymbol{A}_{id})^T + (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)^T\right]^T\boldsymbol{x} \\
&= \boldsymbol{u}\cdot\boldsymbol{s} + e' + \lfloor q/2 \rfloor M - \boldsymbol{s}^T\boldsymbol{A}_{id}\boldsymbol{x} - (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\boldsymbol{x} \\
&= \boldsymbol{u}\cdot\boldsymbol{s} + e' + \lfloor q/2 \rfloor M - \boldsymbol{s}^T(\boldsymbol{u}+\boldsymbol{y}) - (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\boldsymbol{x} \\
&= \underbrace{e' - (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\boldsymbol{x} - \boldsymbol{s}^T\boldsymbol{y}}_{\text{error term}} + \lfloor q/2 \rfloor M \in \mathcal{R}.
\end{aligned}$$

So we need to choose our parameters properly for the correctness of the Dual-Regev encryption to hold. We can bound as follows the euclidean norms of the quantities that appear in the error term:

- $\|e'\| \leq t\tau\sqrt{n}$ from Lemma 2.
- $\|\boldsymbol{e_0}^T\boldsymbol{x_0}\| \leq 2t^2\tau\zeta nd$ from Lemma 2 and Theorem 1.
- $\|\boldsymbol{e_1}^T\boldsymbol{x_1}\| \leq t^2\gamma\zeta nd(k-\ell)$ from Lemma 2 and Theorem 1.
- $\|\boldsymbol{s}^T\boldsymbol{y}\| \leq t^2\tau n^{5/2}d\sigma_g\frac{q^{1/r}}{\sqrt{r}}\sqrt{(b^{2\ell}-1)/(b^2-1)}$ from Lemma 2 and Theorem 1.

By substituting these bounds, we get the following constraints:

$$\begin{aligned}
\|e' - (\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\boldsymbol{x} - \boldsymbol{s}^T\boldsymbol{y}\| &\leq \|e'\| + \|(\boldsymbol{e_0}^T \mid \boldsymbol{e_1}^T)\| + \|\boldsymbol{s}^T\boldsymbol{y}\| \\
&\leq t\tau\sqrt{n} + t^2nd\left[\zeta\left(2\tau + \gamma(k-\ell)\right) + \tau n^{3/2}\sigma_g\frac{q^{1/r}}{\sqrt{r}}\sqrt{(b^{2\ell}-1)/(b^2-1)}\right] \\
&\leq \lfloor q/4 \rfloor.
\end{aligned}$$

Parameter constraints. The following constraints, combined with the norms bounds above, must be met to ensure correctness:

- The Gaussian parameter σ_g used for the $\boldsymbol{G}$-sampling in the ApproxSamplePre1 algorithm must verify $\sigma_g \geq \sqrt{2b}\cdot(2b+1)\cdot\sqrt{\log(2nw(1+1/\epsilon))/\pi}$ (see [GM18], Corollary 3.1).
- The Gaussian width for preimage sampling ζ must follow the condition $\zeta > \sqrt{(\sigma_g^2+1)s_1^2(\boldsymbol{R}) + \eta_\varepsilon^2(\mathbb{Z}^{nm})}$, knowing that $s_1(\boldsymbol{R}) < 1.1\sigma(\sqrt{2nd} + \sqrt{nw} + 4.7)$ with high probability (see [BEP+21], Section A.3), where $s_1(\boldsymbol{R})$ is the spectral norm of the trapdoor $\boldsymbol{R}$.

– The Gaussian width for approximate trapdoor generation σ must verify $\sigma > 4 \cdot 4^{\frac{1}{nd}}\sqrt{n}$ to ensure the public matrix $\boldsymbol{A}$ we use is statistically close to uniform (see [LPR13], Corollary 7.5).
– We choose to set the Gaussian parameter γ of the Gaussian error $\boldsymbol{e}_1 \in \mathcal{R}^w$ as $\gamma^2 = \sigma^2\|\boldsymbol{e}_0\|^2 + 2nt^2\sigma^2\tau^2$.

The proof of the following Theorem 3 is standard and can be found in the Supplementary materials, Appendix B.

Theorem 3. *The IBE construction with parameters $n, d, m, q, k, \ell, \sigma, \alpha, \zeta, \tau$ and γ, chosen as in the above description, is IND-sID-CPA secure in the standard model under the hardness of* $\mathsf{Module\text{-}LWE}_{n,d,q,\tau}$.

4.2 Implementation and performance.

The scheme is implemented in C, using [BEP+21] libraries, inheriting its modularity. It relies on several basic blocks that can be swapped out: the arithmetic over $\mathbb{Z}_q$ and $\mathcal{R}_q$, a pseudorandom number generator, and a (constant-time) sampler of discrete Gaussian distributions over $\mathbb{Z}$. To generate our specific discrete Gaussian distributions, we make use of the following building blocks: an AES-based pseudorandom number generator (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 the computations that deal with non-integers are carried out with floating-point operations that do not involve subnormal numbers. We rely on results from [LS18, Lemma 3] to reduce multiplications in $\mathcal{R}_q$ to polynomials multiplications in rings of the form $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$. The CRT reduction we used then allowed us to speed up polynomial arithmetic in $\mathcal{R}_q$.

To assess the security of each parameter set, we estimate the pseudorandomness of the public-key (corresponding to the LWE security) and the hardness of breaking AISIS. The estimation of the LWE security is done with the LWE estimator of [APS15] with BKZ as the reduction model. We approximate our instances by an instance of an unstructured LWE problem in dimension nd. We follow a very pessimistic core-SVP hardness, where the cost of a BKZ algorithm with blocksize κ is taken to be the cost of only one call to an SVP oracle in dimension κ. For the AISIS problem, we follow the approach of [CGM19] which consists in computing the smallest blocksize achieving the target root Hermite factor corresponding to forging a signature.

Timings. As expected, the 4 algorithms are more efficient for low value of d. Concerning the Decrypt algorithm, its execution time relies mostly on the value of n rather than d. Our timings have been obtained on an Intel i7-8650U CPU running at 1.9 GHz, and then scaled at 4.2GHz to compare ourselves with other schemes. We provide concrete parameters sets and the associated concrete results in Table 2.

$\lceil \log_2 q \rceil$	(n, d)	ℓ	ζ	Setup	Extract	Encrypt	Decrypt	$\mathsf{M-LWE}_{n,d,q,\tau}$
58	$(256, 4)$	0	1137729	203.52	56.88	17.69	2.72	81
58	$(256, 4)$	8	1068989	174.94	49.82	15.00	2.40	80
58	$(256, 4)$	15	1004226	152.70	41.57	12.79	2.19	78
60	$(512, 3)$	0	1398812	210.85	67.28	19.20	4.06	110
60	$(512, 3)$	8	1315427	189.35	59.33	17.13	3.80	109
60	$(512, 3)$	15	1236981	160.33	53.39	14.00	3.22	107

Table 2: Proposed IBE parameters for our first construction of Section 4 with different pairs of polynomial ring dimension n and rank d, for different moduli sizes and taking $\sigma_g = 54.9$, $\sigma = 64.1$. σ_g is the Gaussian parameter for the $\boldsymbol{G}$-Sampling, σ is the Gaussian width of the trapdoor $\boldsymbol{R}$ used in the Setup algorithm and ζ is the standard deviation for the Gaussian preimage sampling in the Extract algorithm. Timings in columns Setup-to-Encrypt are given in ms. $\mathsf{M-LWE}_{n,d,q,\tau}$ denotes the concrete bit security of the scheme.

Scheme	(n, d)	Setup	Extract	Encrypt	Decrypt
[BEP+21] ($\ell = 0$)	$(512, 2)$	123.62	45.80	13.12	2.91
[BEP+21] ($\ell = 0$)	$(256, 4)$	234.45	67.90	17.88	2.69
This paper ($\ell = 15$)	$(512, 2)$	79.10	29.44	10.09	2.36
This paper ($\ell = 15$)	$(256, 4)$	152.70	41.57	12.79	2.19

Table 3: Comparison of timings in ms of the different operations of the IBE scheme between this paper and [BEP+21] for parameters giving an equivalent level of security.

Comparison with related work. We compare our IBE performance with the IBE from [BEP+21], which corresponds to the case $\ell = 0$, that is to say the use of exact trapdoors instead of approximate ones. We observe that for a fixed pair (n, d), the larger ℓ is, the better timings are. Overall, the use of approximate trapdoors allows to obtain better timings for all the algorithms comprising the IBE scheme.

5 An IBE scheme based on the hardness of iNTRU

In this Section, we introduce the IBE2 scheme, instantiated using gadget-based approximate trapdoors over iNTRU trapdoors combined with the Dual-Regev encryption scheme over modules.

We recall that $m = k - \ell$ where $k = \lceil \log_b q \rceil$ and that the approximate gadget vector $\boldsymbol{f}$ is defined as $\boldsymbol{f}^T = \left[b^\ell \; b^{\ell+1} \; b^{\ell+2} \cdots \; b^{k-1}\right] \in \mathcal{R}^{k-\ell}$. Here, the master public key is a vector $\boldsymbol{a} \in \mathcal{R}_q^{m+1}$ generated with the ApproxTrapGen2 algorithm and whose pseudorandomness is based on the iNTRU problem. The master secret key $r, \boldsymbol{e} \in \mathcal{R}^{m+1}$ defines a $\boldsymbol{f}$-approximate trapdoor associated to $\boldsymbol{a}$. An identity is mapped to an element in $\mathcal{R}_q$ by the use of a hash function modeled as a random oracle in the security proof; the secret key associated to an identity id is an approximate short vector $\boldsymbol{x} \in \mathcal{R}^{m+1}$.

5.1 IBE construction details

We detail below the four algorithms of the IBE2:

- Setup$(1^n) \longrightarrow (\mathsf{mpk}, \mathsf{msk})$:
  - let $\boldsymbol{a} = \left[1 \ \boldsymbol{a_0}^T\right]^T \in \mathcal{R}_q^{m+1}$ and $\boldsymbol{R} = \begin{bmatrix} -\boldsymbol{e}^T \\ r\boldsymbol{I}_m \end{bmatrix} \in \mathcal{R}^{(m+1)\times m}$ output by ApproxTrapGen2(σ); and let $\mathcal{H} : \{0,1\}^\star \to \mathcal{R}_q$ a hash function;
  - output $\mathsf{mpk} = (\boldsymbol{a}, \mathcal{H})$ and $\mathsf{msk} = \boldsymbol{R}$.
- Extract$(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \to \boldsymbol{x}_{\mathsf{id}} = \boldsymbol{x}_2 \in \mathcal{R}^m$:
  - define the tag $h_{\mathsf{id}} = \mathcal{H}(\mathsf{id}) \in \mathcal{R}_q$;
  - sample a short preimage $\boldsymbol{x} = (x_1, {\boldsymbol{x}_2}^T)^T \leftarrow$ ApproxSamplePre2$(\boldsymbol{a}, \boldsymbol{R}, h_{\mathsf{id}}, \zeta)$;
- Encrypt$(\mathsf{mpk}, \mathsf{id}, M) \to \boldsymbol{C} = (\boldsymbol{b}, c) \in \mathcal{R}_q^{m+1}$:
  - compute $h_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$; sample $s \leftarrow D_{\mathcal{R},\tau}$, $\boldsymbol{e_1} \leftarrow D_{\mathcal{R}^m,\tau}, e_2 \leftarrow D_{\mathcal{R},\tau}$;
  - compute $\boldsymbol{b} = s\boldsymbol{a}_0 + \boldsymbol{e}_1 \in \mathcal{R}_q^m$ and $c = h_{\mathsf{id}} \cdot s + e_2 + \lfloor q/2 \rfloor M \in \mathcal{R}_q$, where a message is encoded as $M \in \mathcal{R}_2$;
- Decrypt$(\boldsymbol{x}_{\mathsf{id}}, \boldsymbol{C}) \to M$:
  - parse $\boldsymbol{x}_{\mathsf{id}}$ as $(x_1, \boldsymbol{x}_2)$; and compute $\mathsf{res} \leftarrow c - \boldsymbol{b}^T\boldsymbol{x}_2$;
  - for each i, if the coefficient $\mathsf{res}_i \in \mathbb{Z}$ is closer to $\lfloor q/2 \rfloor$ than to 0, $M_i = 1$, otherwise $M_i = 0$.

Correctness. We have the following equality:

$$\begin{aligned} c - \boldsymbol{b}^T\boldsymbol{x}_2 &= h_{\mathsf{id}} \cdot s + e_2 + \lfloor q/2 \rfloor M - (s\boldsymbol{a}_0 + \boldsymbol{e_1})^T\boldsymbol{x_2} \\ &= s \cdot x_1 + s\boldsymbol{a_0}^T\boldsymbol{x_2} - y \cdot s + e_2 + \lfloor q/2 \rfloor M - s\boldsymbol{a_0}^T\boldsymbol{x_2} - \boldsymbol{e_1}^T\boldsymbol{x_2} \\ &= \lfloor q/2 \rfloor M + s \cdot (x_1 - y) + e_2 - \boldsymbol{e_1}^T\boldsymbol{x_2}. \end{aligned}$$

Furthermore, the following bounds apply:

- $\|s \cdot x_1\| \leq t^2\tau\zeta n$ from Lemma 2 and Theorem 2.
- $\|y \cdot s\| \leq t^2\tau\sigma_g\sqrt{(b^{2\ell}-1)/(b^2-1)}n$ from Lemma 2 and Theorem 2.
- $\|e_2\| \leq t\tau\sqrt{n}$ from Lemma 2.
- $\|\boldsymbol{e_1}^T\boldsymbol{x_2}\| \leq t^2\tau\zeta nm$ from Lemma 2 and Theorem 2.

By substituting these bounds, we obtain:

$$\begin{aligned} \|s \cdot (x_1 - y) + e_2 - \boldsymbol{e_1}^T\boldsymbol{x_2}\| &\leq \|s \cdot x_1\| + \|y \cdot s\| + \|e_2\| + \|\boldsymbol{e_1}^T\boldsymbol{x_2}\| \\ &\leq t^2\tau\left[\zeta\,(m+1) + \sigma_g\sqrt{(b^{2\ell}-1)/(b^2-1)}\right] + t\tau\sqrt{n} \\ &\leq \lfloor q/4 \rfloor. \end{aligned}$$

Parameters constraints. The following constraints combined with the errors norm constraints above should be satisfied:

- The Gaussian parameter σ_g used for the $\boldsymbol{G}$-sampling in the ApproxSamplePre2 algorithm must verify $\sigma_g \geq \sqrt{2b} \cdot (2b+1) \cdot \sqrt{\log(2nw(1+1/\epsilon))/\pi}$ (see [GM18], Corollary 3.1).
- The Gaussian width for preimage sampling ζ must follow the condition $\zeta > \sqrt{(\sigma_g^2+1)s_1^2(\boldsymbol{R}) + \eta_\varepsilon^2(\mathbb{Z}^{nm})}$, where $s_1(\boldsymbol{R})$ is the spectral norm of the trapdoor $\boldsymbol{R}$ (see [BEP+21], Section A.3).

The proof of the following is adapted from the GPV IBE proof [GPV08, Section 7.2] and is given in the Supplementary materials, Appendix C.

Theorem 4. *Our IBE construction with parameters $n, m, q, k, \ell, \sigma, \sigma_g, \zeta, \tau$ and γ is IND-sID-CPA secure in the random oracle model under the hardness of* $\mathsf{iNTRU}_{q,D_{R,\sigma}}$ *and* $\mathsf{Ring\text{-}LWE}_{n,q,\tau}$.

5.2 Implementation and performance

In [GGH+19], the authors only propose a reduction from a non-standard version of LWE to iNTRU. Therefore, in the absence of a thorough study on the asymptotic and practical security of the iNTRU problem, which we leave for future work, we have chosen to estimate the security of our iNTRU instances by relying on the existing cryptanalysis on NTRU. As explained in [GGH+19], there is a syntactic link between NTRU and iNTRU. To the best of our knowledge, there is no known a reduction between the two problems and the analysis of the iNTRU assumption might deserve additionally study. Still, we additionally consider the practical security of NTRU for our target set of parameters and we take into account the known cryptanalysis effort on iNTRU.

For security estimation, we follow the same approach as in [GL20] to assess the concrete security of the IBE2 scheme. We determine the hardness of our underlying lattice problem by computing the root Hermite factor, introduced in [GN08]. Then, we use the following heuristic relation between the blocksize κ and the root Hermite factor δ to find the smallest blocksize which would break our underlying lattice problem:

$$\delta \approx \left(\frac{\kappa}{2\pi e}(\pi\kappa)^{1/\kappa}\right)^{1/2(\kappa-1)}.$$

Finally, our experiments estimate the running time of the BKZ algorithm to analyse the concrete security of the scheme. This algorithm makes use of an oracle to solve the Shortest Vector Problem (SVP) in smaller lattices. We chose the "Core-SVP" model introduced in [ADP+16] in the sieving regime as the SVP oracle for the BKZ algorithm with time complexity $2^{0.292\kappa+16.4}$ in the blocksize κ.

Overstretched parameters. [LW20] adapts the attack from [KF17] on iNTRU, which applies on the parameters originally proposed for the homomorphic encryption scheme from [GGH$^+$19]. The attack can be performed when the modulus q is much larger than the dimension of the associated lattices. The target iNTRU instance has an error and secret that follow a uniform binary distribution. Following [KF17] attack, [LW20] uses the fact that a very dense sublattice can be found in this NTRU-like lattice, because of the overstretched regime. Relying on a lemma from Pataki and Tural [PT08, Lemma 1], they can bound the volume of this sublattice and run a BKZ-reduction which leads to a full recovery of the iNTRU secret. [DW21] improves the asymptotic bound given by Kirchner and Fouque [KF17] by conducting a refined analysis which lowers the overstretched regime for NTRU with ternary distribution to the value $q = n^{2.484+o(1)}$.
They also provide a concrete analysis, computing a bound on the modulus q for which the attacks exploiting the overstretched regime are more efficient than standard secret key recovery attacks. The authors of [DW21] run experiments which allows to detect the fatigue "point", which separates these two regimes.

The cryptanalysis carried out by [LW20] and [DW21] can be adapted for the iNTRU instances we consider, where the secret and the error both follow a Gaussian distribution. [DW21] also consider the case where the NTRU secret distributions are Gaussian. Indeed, in both cases, the cryptanalysis in the overstretched regime requires to perform lattice reductions on sublattices of a NTRU-like lattice, in order to retrieve the very dense sublattice. We leave the detailed adaptation of this attack to iNTRU and its experimental study for a future work. For our concrete parameters analysis, we took care to fall outside the range of parameters affected in the overstretched regime.

Timings. Our timings have been obtained on an Intel i7-8650U CPU running at 1.9 GHz, and then scaled at 4.2GHz to compare ourselves with other schemes. Results are provided in Table 4. The use of the efficient gadget-based approximate trapdoor framework together with the iNTRU hardness assumption allows us to obtain efficient algorithms. The slowest of the 4 algorithms of the IBE2 scheme are the Setup and Extract algorithms, which correspond respectively to the approximate trapdoor generation and preimage sampling phases of the scheme. However, the Setup algorithm is usually not performed often, and the subroutine algorithms used by Extract for sampling are really modular, leaving the way for possible future improvements. Moreover, our Setup and Extract algorithms are competitive with other NTRU-based IBE (see Table 5 and Table 6).

Comparison with related works. We compare the IBE2 timings with the ones of [DLP14] (re-implemented in [MSO17]) and with [ZMS$^+$21], two IBE schemes whose security is based on the NTRU hardness assumption. Our comparison experiments were carried out using equivalent parameters sets between the different scheme. In particular, we have been careful to use equivalent module sizes and equivalent noise rate when performing Gaussian Sampling.

(n, q)	$\lceil log_2(q) \rceil$	Setup	Extract	Encrypt	Decrypt	Bit security
(256, 1073741441)	30	1.01	2.13	0.39	0.03	64
(512, 1073741441)	30	2.12	3.79	0.74	0.06	115
(1024, 1073741441)	30	4.08	7.65	1.49	0.11	223
(256, 16777601)	25	0.78	1.66	0.23	0.02	35
(512, 16777601)	25	1.48	3.00	0.49	0.04	82
(1024, 16777601)	25	3.32	5.92	1.10	0.07	159

Table 4: Timings of the different operations for different values of n, given in ms from Setup to Decrypt.

Scheme	$(n, \lceil log_2(q) \rceil)$	Security level	Setup	Extract	Encrypt	Decrypt
[DLP14]	$(512, 26)$	< 80	3.84s	1.77	0.10	0.05
[DLP14]	$(1024, 26)$	< 192	23.93s	6.95	0.27	0.09
This paper	$(512, 25)$	82	1.48	3.00	0.49	0.04
This paper	$(1024, 25)$	159	3.32	5.92	1.10	0.07

Table 5: Timings comparison of the different operations of the IBE scheme between this paper and [DLP14] (the timings are extracted from the [MSO17] article, scaled up to account for CPU differences) for different parameter sets. The timings are given in ms, except for the Setup algorithm from [DLP14], which is given in seconds.

Scheme	$(n, \lceil log_2(q) \rceil)$	Security level	Setup	Extract	Encrypt	Decrypt
[ZMS$^+$21]	$(1024, 24)$	128	102	0.82	0.05	0.06
[ZMS$^+$21]	$(2048, 25)$	256	292	2.62	0.10	0.13
[ZMS$^+$21]	$(1024, 36)$	80	165	26.4	0.08	0.09
[ZMS$^+$21]	$(2048, 38)$	160	643	57.8	0.16	0.18
This paper	$(1024, 25)$	159	3.32	5.92	1.10	0.07
This paper	$(2048, 25)$	293	10.21	12.79	2.96	0.16
This paper	$(1024, 30)$	191	4.08	7.65	1.49	0.11
This paper	$(2048, 30)$	412	12.51	16.03	3.89	0.23

Table 6: Timings comparison of the different operations of the IBE scheme in this paper and in [ZMS$^+$21] for different sets of parameters in ms.

We observe that we obtain better timings for the Setup algorithm than [DLP14] and [ZMS$^+$21] and for the Extract for some sets of parameters. Furthermore, our Decrypt algorithm is slightly faster than [DLP14]. However, the Encrypt algorithms is less efficient than theirs. The use of binomial distribution improves the timings for encryption. An improvement can be obtained in our case by using binomial distribution, but we need more samples in our case which still affects performance for encryption; we make $n(2 + m)$ calls to the integer Gaussian sampler while encryption in [DLP14] and [ZMS$^+$21] can make use of only $3n$ binomial sampling calls. As in [ZMS$^+$21], our Extract algorithm is slower than the Sign algorithm from the Falcon signature scheme [FHK$^+$17]. Note also that for a similar security level, Falcon can use smaller parameters than us.

We obtain an overhead in terms of parameters sizes compared to [DLP14] and [ZMS+21], but the trapdoor generations rely on a different paradigm. Results on public and secret sizes are provided in Appendix D. Nonetheless, as stated in [CGM19], the use of approximate trapdoors instead of exact ones helps us to reduce the sizes of the public-key and signatures by up to two times. Therefore, as expected, our obtained sizes for the master and the users' private keys and the ciphertexts are close to the ones of [GL20] whose signature scheme relies on the same paradigm as the IBE2 scheme.

Acknowledgements. This work is supported by the ANR ASTRID project AMIRAL (ANR-21-ASTR-0016).

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, LNCS. Springer, 2010.

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

[APS15] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 2015.

[BEP+21] P. Bert, G. Eberhart, L. Prabel, A. Roux-Langlois, and M. Sabt. Implementation of lattice trapdoors on modules and applications. In *PQCrypto*, Lecture Notes in Computer Science. Springer, 2021.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, LNCS. Springer, 2001.

[BFRS18] P. Bert, P. Fouque, A. Roux-Langlois, and M. Sabt. Practical implementation of ring-sis/lwe based signature and IBE. In 2018.

[BJR+23] K. Boudgoust, C. Jeudy, A. Roux-Langlois, and W. Wen. On the hardness of module learning with errors with short distributions. *J. Cryptology*, 28(1), 2023.

[CGM19] Y. Chen, N. Genise, and P. Mukherjee. Approximate trapdoors for lattices and smaller hash-sign signatures. In *ASIACRYPT*, 2019.

[CHK+10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

[Coc01] C. C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMACC*. Springer, 2001.

[DLP14] L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT*, 2014.

[DW21] L. Ducas and W. P. J. van Woerden. NTRU fatigue: how stretched is overstretched? In *ASIACRYPT (4)*, 2021.

[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.

[GGH+19] N. Genise, C. Gentry, S. Halevi, B. Li, and D. Micciancio. Homomorphic encryption for finite automata. In *ASIACRYPT*, 2019.

[GL20] N. Genise and B. Li. Gadget-based intru lattice trapdoors. In *INDOCRYPT*, 2020.

[GM18] N. Genise and D. Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In *EUROCRYPT*, 2018.

[GN08] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, Lecture Notes in Computer Science. Springer, 2008.

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

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

[KF17] P. Kirchner and P. Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 3–26, 2017.

[LCC14] R. W. F. Lai, H. K. F. Cheung, and S. S. M. Chow. Trapdoors for ideal lattices with applications. In *Inscrypt*, 2014.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*. 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*, 2018.

[LW20] C. Lee and A. Wallet. Lattice analysis on mintru problem. Cryptology ePrint Archive, Paper 2020/230, 2020.

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[MR07] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 2007.

[MSO17] S. McCarthy, N. Smyth, and E. O'Sullivan. A practical implementation of identity-based encryption over NTRU lattices. In 2017.

[PT08] G. Pataki and M. Tural. On sublattice determinants in reduced bases. *arXiv preprint arXiv:0804.4014*, 2008.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984.

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

[ZY22] S. Zhang and Y. Yu. Towards a simpler lattice gadget toolkit. In *Public Key Cryptography*, 2022.

Supplementary materials

A Definition of an IBE scheme

An IBE scheme is composed of 4 polynomial time algorithms defined as follows:

Setup$(1^n) \rightarrow (\mathsf{mpk}, \mathsf{msk})$: takes as input a security parameter n and outputs the master public key mpk and the master secret key msk. We assume public parameters which are part of mpk are taken as input to all 3 next algorithms.

Extract$(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{id}}$: takes as input the security parameter, the master keys mpk, msk and an identity $\mathsf{id} \in \mathcal{ID}$ and outputs a private key $\mathsf{sk}_{\mathsf{id}}$ associated to the identity id.

Encrypt$(\mathsf{mpk}, \mathsf{id}, M) \rightarrow C$: takes as input the security parameter, the master public key mpk, an identity id and a message M and outputs a ciphertext C which is the encryption of M for identity id.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, C) \rightarrow M$ or $Error$: takes as input the master public key mpk, a private key $\mathsf{sk}_{\mathsf{id}}$ associated to the identity id and a ciphertext C, it outputs either a message M or an $Error$ message indicating the ciphertext is invalid.

Correctness. An IBE is correct if for any message M in the message space and identity id such that $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \text{Setup}(1^n)$ and $\mathsf{sk}_{\mathsf{id}} \leftarrow \text{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$, $\text{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \text{Encrypt}(\mathsf{mpk}, \mathsf{id}, M)) = M$ holds with overwhelming probability.

Security Game. We describe the notion of ciphertext indistinguishability under a selective-identity chosen-plaintext attack (IND-sID-CPA), meaning that the adversary indicates the identity on which he would like to be challenged. The notion is defined as a game between an adversary $\mathcal{A}$ and a challenger, and works as follows:

- Initialisation: The adversary $\mathcal{A}$ chooses the challenge identity id^*.
- Setup : The challenger runs Setup(1^n) and gives the master public key msk to $\mathcal{A}$.
- Extract queries - Phase 1: The adversary can make private-key extraction queries on identities $\mathsf{id} \neq \mathsf{id}^*$, and the challenger answers by running $\mathsf{sk}_{\mathsf{id}} \leftarrow \text{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$.
- Challenge: The adversary $\mathcal{A}$ outputs two plaintexts M_0, M_1. The challenger chooses a random bit $b^* \leftarrow \{0, 1\}$ and sets the challenge ciphertext to $C^* = \text{Encrypt}(\mathsf{mpk}, \mathsf{id}^*, M_{b^*})$. The challenge ciphertext C^* is sent to $\mathcal{A}$.
- Extract queries - Phase 2: The adversary can make additional queries (answered as in Phase 1).
- Guess: The adversary outputs a bit b and wins if $b^* = b$.

The advantage of the adversary $\mathcal{A}$ playing the IND-sID-CPA security game above is $\text{Adv}(\mathcal{A}^{\mathsf{ind\text{-}sID\text{-}cpa}}_{\mathsf{IBE}}) = \mid Pr(b = b^*) - \frac{1}{2} \mid$. An IBE scheme is IND-sID-CPA secure if, for all PPT adversary $\mathcal{A}$, his advantage $\text{Adv}(\mathcal{A}^{\mathsf{ind\text{-}sID\text{-}cpa}}_{\mathsf{IBE}})$ is negligible.

B Proof of Theorem 3

Proof. The proof proceeds in a sequence of games G_i where the first game G_0 is identical to the original IND–sID-CPA game. In the last game in the sequence, the adversary has no information left about the initial message and hence has advantage zero. We show that a PPT adversary cannot distinguish between the intermediate games.

Game G_0. This is the initial IND-sID-CPA game between an adversary $\mathcal{A}$ and an IND-sID-CPA challenger.

Game G_1. In this game, we change the way the challenger generates the public matrix $\boldsymbol{A}$ by adding information about the identity id^* that $\mathcal{A}$ targets for his attack. In Game 0, $\boldsymbol{A}$ was generated thanks to ApproxTrapGen1$(\boldsymbol{0}, \sigma)$, together with the associated trapdoor $\boldsymbol{R}$. We had $\boldsymbol{A} = [\, \bar{\boldsymbol{A}} \,|\, -\bar{\boldsymbol{A}}\boldsymbol{R} \,]$. The public matrix $\boldsymbol{A}$ is now generated in G_1 thanks to the algorithm ApproxTrapGen1$(-\boldsymbol{H}_{\mathsf{id}^*}, \sigma)$, so that we have $\boldsymbol{A} = [\, \bar{\boldsymbol{A}} \,|\, -\boldsymbol{H}_{\mathsf{id}^*}\boldsymbol{F} - \bar{\boldsymbol{A}}\boldsymbol{R} \,] \in \mathcal{R}_q^{d \times m}$.

This matrix $\boldsymbol{A}$ is statistically indistinguishable from a uniformly random matrix.

The challenger answers $\mathcal{A}$'s private key queries on identities $\mathsf{id} \neq \mathsf{id}^*$ by calling Extract$((\boldsymbol{A}, \boldsymbol{u}), \boldsymbol{R}, \mathsf{id})$ and then using ApproxSamplePre1$(\boldsymbol{A}_{\mathsf{id}}, \boldsymbol{R}, \boldsymbol{H}_{\mathsf{id}}, \boldsymbol{u}, \zeta)$. We have $\boldsymbol{A}_{\mathsf{id}} = [\, \bar{\boldsymbol{A}} \,|\, (\boldsymbol{H}_{\mathsf{id}} - \boldsymbol{H}_{\mathsf{id}^*})\boldsymbol{F} - \bar{\boldsymbol{A}}\boldsymbol{R} \,]$ and $\boldsymbol{H}_{\mathsf{id}} - \boldsymbol{H}_{\mathsf{id}^*}$ is invertible because of the FRD construction, which allows the challenger to get a private key associated to the identity id.

Game G_2. This game is identical to G_1 except that the challenge ciphertext $\boldsymbol{C}^*$ is chosen as a uniformly random element in $\mathcal{R}_q^{m+1}$ instead of being chosen by calling Encrypt$((\boldsymbol{A}, \boldsymbol{u}), \mathsf{id}^*, M_{b^*})$.

Reduction from Module-LWE. We show that G_2 and G_1 are computationally indistinguishable for the adversary $\mathcal{A}$ under the Module-LWE assumption.

Suppose $\mathcal{A}$ has a non-negligible advantage in distinguishing G_2 and G_1. We will use $\mathcal{A}$ to construct a simulator $\mathcal{B}$ who will be able to solve the Module-LWE problem with non-negligible advantage.

The simulator $\mathcal{B}$ receives $\bar{m} + 1$ samples $(\boldsymbol{A}_i, b_i)_{0 \leq i \leq \bar{m}}$ with $\boldsymbol{A}_i \in \mathcal{R}_q^d$ and $b_i \in \mathcal{R}_q$ as an instance of the decisional Module-LWE problem. The simulator also receives the identity id^* targeted by the adversary $\mathcal{A}$. The simulator sets $\boldsymbol{A}' = (\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{\bar{m}}) \in \mathcal{R}_q^{d \times \bar{m}}$ and $\boldsymbol{b}' = (b_1, \ldots, b_{\bar{m}}) \in \mathcal{R}_q^{\bar{m}}$ and then runs ApproxTrapGen1$(-\boldsymbol{H}_{\mathsf{id}^*}, \sigma)$ to get $\boldsymbol{A} = [\, \boldsymbol{A}' \,|\, -\boldsymbol{H}_{\mathsf{id}^*}\boldsymbol{F} - \boldsymbol{A}'\boldsymbol{R} \,]$ together with the trapdoor. Next, $\mathcal{B}$ sets $\boldsymbol{u} = \boldsymbol{A}_0 \in \mathcal{R}_q^d$ and he sends $(\boldsymbol{A}, \boldsymbol{u})$ to $\mathcal{A}$ as the master public key. Then, $\mathcal{B}$ answers $\mathcal{A}$'s private key queries as in G_1.

Afterwards, the adversary outputs two plaintexts M_0, M_1 and sends them to $\mathcal{B}$. The simulator generates a random bit b^*, and the challenge ciphertext $\boldsymbol{C}^* = (\boldsymbol{b}^*, c^*)$ as follows:

$$\boldsymbol{b}^* = (\boldsymbol{b}'^T | - \boldsymbol{b}'^T\boldsymbol{R} + \hat{\boldsymbol{e}}^T)^T, \; c^* = b_0 + \lfloor q/2 \rfloor M_{b^*},$$

where $\hat{e} \leftarrow D_{\mathcal{R}^w,\mu}$ for some real μ, that is explicited below.

Samples from the LWE *distribution.* If the Module-LWE samples are drawn from the Module-LWE distribution, we have:

$$\boldsymbol{b'}^T = \boldsymbol{s}^T \boldsymbol{A'} + \boldsymbol{e'}^T \text{ and } b_0 = \boldsymbol{s}^T \boldsymbol{A}_0 + e_0,$$

for some $\boldsymbol{s} \in \mathcal{R}_q^d$, $\boldsymbol{e'} \leftarrow D_{\mathcal{R}^{\bar{m}},\tau}$ and $e_0 \leftarrow D_{\mathcal{R},\tau}$. Therefore, the first part of the ciphertext is $\boldsymbol{b}^* = {\boldsymbol{A}_{\mathsf{id}^*}}^T \boldsymbol{s} + (\boldsymbol{e'}^T| - \boldsymbol{e'}^T \boldsymbol{R} + \hat{\boldsymbol{e}}^T)^T$ and the second part is $c^* = \boldsymbol{s}^T \boldsymbol{A}_0 + e_0 + \lfloor q/2 \rfloor M_{b^*}$.

Then, for a fixed error $\boldsymbol{e'}$, the term $-\boldsymbol{e'}^T \boldsymbol{R}$ is distributed as $D_{\mathcal{R}^w,\sigma||\boldsymbol{e'}||}$ since $\mathbb{E}[\boldsymbol{e'}^T \boldsymbol{R}] = \boldsymbol{e'}^T \mathbb{E}[\boldsymbol{R}] = \boldsymbol{0}$ and $cov(\boldsymbol{e'}^T \boldsymbol{R}) = \sigma^2 \boldsymbol{e'}^T \boldsymbol{e'} \boldsymbol{I}_w = \sigma^2 ||\boldsymbol{e'}||^2 \boldsymbol{I}_w$ by linearity of expectation and bilinearity of covariance. Moreover, the random variable $\hat{\boldsymbol{e}}$ follows the Gaussian distribution $D_{\mathcal{R}^w,\mu}$ and is independent from the random variable $\boldsymbol{e'}^T \boldsymbol{R}$. Therefore, $-\boldsymbol{e'}^T \boldsymbol{R} + \hat{\boldsymbol{e}}$ is indistinguishable from a sample drawn from the distribution $D_{\mathcal{R}^w,\gamma}$ for μ satisfying $\gamma^2 = (\sigma||\boldsymbol{e'}||)^2 + \mu^2$.

Then, the challenge ciphertext $(\boldsymbol{b}^*, c^*)$ follows the same distribution as in G_1.

Samples from the uniform distribution. If the Module-LWE samples are drawn from the uniform distribution, the ciphertext challenge also looks uniforms as in G_2.

Finally, the adversary $\mathcal{A}$ outputs a guess b. Because we supposed $\mathcal{A}$ has a non-negligible advantage in distinguishing G_1 and G_2, if $b = b'$ with overwhelming probability, the simulator concludes that the challenged instance was drawn from the Module-LWE distribution. Otherwise, $\mathcal{B}$ concludes that the Module-LWE instance was drawn from the uniform distribution.

C Proof of Theorem 4

For the sake of completeness, we give the IND-sID-CPA proof of the IBE2 scheme, which is directly adapted from [GPV08, Theorem 7.2].

Proof. The idea of the proof is to simulate the view of the adversary given a Ring-LWE instance (a'_i, b'_i) for $0 \leq i \leq m$ for which we build an attacker $\mathcal{S}$ who makes $Q_{\mathcal{H}}$ queries to $\mathcal{H}$.

We construct a new simulation game given $\boldsymbol{a_0} = [a'_1 \dots a'_m]$. $\mathcal{S}$ chooses an index i^* at random in $[1, Q_{\mathcal{H}}]$ and sets $h_{\mathsf{id}^\star} = a'_0$.

A hash query on input id is answered as follows: it first checks whether an entry of the form $(\mathsf{id}, *, *, *)$ already exists in the hash table. If it is not the case, it responds with a value u_{id} such that an approximated preimage for $\mathcal{H}(\mathsf{id}) = h_{\mathsf{id}}$ is known i.e: it samples $\boldsymbol{x} \leftarrow D_{\mathcal{R}^{m+1},\zeta}$ and $y_{\mathsf{id}} \leftarrow D_{\mathcal{R},\sigma_e}$ such that $y_{\mathsf{id}} = (x_1^T, \boldsymbol{x_2}^T)^T \left[1 \ \boldsymbol{a_0}\right]$, sets $h_{\mathsf{id}} = (x_1^T, \boldsymbol{x_2}^T)^T \left[1 \ \boldsymbol{a_0}\right] - y_{\mathsf{id}}$ and adds the tuple $(\mathsf{id}, h_{\mathsf{id}}, \boldsymbol{x_2}, y_{\mathsf{id}})$ in the hash table.

An extraction query for an identity id is responded as follows: assuming that an entry for id already exists in the table, the corresponding $\boldsymbol{x_2}$ is output by $\mathcal{S}$.

Note that the response to the extract queries are close to the response provided in the real game by Theorem 2 if $\mathcal{H}$ is modeled as a random oracle and the way the public key is generated is indistinguishable from that in the real game under the iNTRU assumption.

When the attacker outputs two messages m_0, m_1 and id^*, if id^* has not been queried to $\mathcal{H}$, then $\mathcal{S}$ aborts; otherwise the challenger sets the challenger ciphertext as:

$$(\boldsymbol{b}, c) = \left(\left[b'_1 \ ... \ b'_m\right], b'_0 + \lfloor q/2 \rfloor m_b\right) \in \mathcal{R}_q^{m+1}.$$

The attacker $\mathcal{S}$ outputs the same bit as $\mathcal{A}$. Assuming no abort has occurred, they both have the same advantage, which concludes the proof.

D Parameters' sizes comparison

The use of the [MP12] paradigm doesn't make us competitive compared to [DLP14] and [ZMS$^+$21] in term of parameters sizes. As explained in Section 5.2, the trapdoor generations rely on a different paradigm. However, for completeness, we provide comparisons with [DLP14] in Table 7 and with [ZMS$^+$21] in Table 8 in terms of parameters' sizes.

Table 7: Parameters sizes comparison between IBE2 parameters sizes and [DLP14] for different sets of parameters at a similar security level, in Bytes.

Scheme	$(n, \lceil log_2(q) \rceil)$	Security	mpk	msk	sk_{id}	Ciphertext
[DLP14]	$(512, 23)$	< 80	1536	6144	1375	1625
[DLP14]	$(1024, 27)$	< 192	3584	14336	3375	3750
This paper	$(512, 25)$	≈ 80	14720	16192	16192	16192
This paper	$(1024, 25)$	≈ 192	32000	35200	35200	35200

Table 8: Parameters sizes comparison between IBE2 parameters sizes and [ZMS$^+$21] for different sets of parameters at a similar security level, in Bytes.

Scheme	$(n, \lceil log_2(q) \rceil)$	Security	mpk	msk	sk_{id}	Ciphertext
[ZMS$^+$21]	$(1024, 36)$	80	4608	18432	4608	9248
[ZMS$^+$21]	$(1024, 24)$	128	3072	12288	3072	6176
[ZMS$^+$21]	$(2048, 25)$	256	6400	25600	6400	12832
This paper	$(512, 25)$	> 80	14720	16192	16192	16192
This paper	$(1024, 25)$	> 128	32000	35200	35200	35200
This paper	$(2048, 25)$	> 256	64000	70400	70400	70400

We observe that using approximate trapdoors instead of exact ones can help reduce the size of public-keys and signatures by up to two times.