Practical Implementation of Ring-SIS/LWE based Signature and IBE

Pauline Bert, Pierre-Alain Fouque, Adeline Roux-Langlois, and Mohamed Sabt

Univ Rennes, CNRS, IRISA
`first.last@irisa.fr`

Abstract. Lattice-based signature and Identity-Based Encryption are well-known cryptographic schemes, and having both efficient and provable secure schemes in the standard model is still a challenging task in light of the current NIST post-quantum competition. We address this problem in this paper by mixing standard IBE scheme, à la ABB (EUROCRYPT 2010) on Ring-SIS/LWE assumptions with the efficient trapdoor of Peikert and Micciancio (EUROCRYPT 2012) and we provide an efficient implementation. Our IBE scheme is more efficient than the IBE scheme of Ducas, Lyubashevsky and Prest based on NTRU assumption and is based on more standard assumptions. We also describe and implement the underlying signature scheme, which is provably secure in the standard model and efficient.
Keywords. Lattice, Signature, IBE, Software implementation, Ring-LWE/SIS.

1 Introduction

The concept of Identity Based Encryption (IBE) was defined by Shamir [Sha85]. It is considered as an alternative to the classical Public Key Encryption (PKE), often requiring a dedicated infrastructure. Indeed, the public key related to a person is simply its identity, such as her email address or her social security number, and the associated private key is generated by a trusted authority using a master public key. Thus, IBE hugely simplifies keys generation and distribution in a multi-user system. The first IBE constructions appeared in [BF01,Coc01], and were based respectively on bilinear maps and on quadratic residue assumptions.

Since the work of Shor in 1994 [Sho97], the hardness of such number theoretic assumptions is extremely reduced when faced to a quantum computer. This has motivated many research work attempting to achieve quantum security. The first supposedly post-quantum IBE scheme, which was introduced in 2008 by Gentry, Peikert and Vaikuntanathan [GPV08], was based on hard lattice problems and followed by many improvements [CHKP10,ABB10,DLP14,Yam16]. Recently in [GHPT17], the first IBE on a code problem in rank metric was proposed.

Lattice-based cryptography. Lattice-based cryptography starts with the work of Ajtai [Ajt96], and uses hard problems on lattices as the foundation of secure cryptographic constructions. A lattice is an infinite arrangement of regularly

The final publication is available at Springer via
`https://doi.org/10.1007/978-3-319-79063-3_13`

spaced points, and it can be generated as the set of all linear combinations of m independent vectors in $\mathbb{R}^n$, called a basis. One fundamental hard problem on lattices is the Shortest Vector Problem (SVP): given some basis, find the shortest non zero vector of the lattice. Lattice-based cryptography is based on the assumption that this problem and its variants are hard problems even for an approximation factor polynomial in the dimension of the lattice.

Lattice-based cryptographic constructions are mainly based on two well known problems: the Small Integer Solution problem (SIS) and its Inhomogeneous variant (ISIS) [Ajt96], and the Learning With Errors problem (LWE) introduced by Regev [Reg05]. In particular, the ISIS problem consists in finding a short vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{u} \mod q$, given an uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and some $\mathbf{u} \in \mathbb{Z}_q^n$. The LWE problem consists in distinguishing $(\mathbf{A}, \mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$ from $(\mathbf{A}, \mathbf{b}^T)$ where $\mathbf{e}$ is a small error sampled from a probability density function over $\mathbb{Z}^m$ (often a discrete Gaussian distribution), and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{b} \in \mathbb{Z}_q^m$ are uniformly chosen. Ajtai and Regev gave reductions from worst-case lattice problems to the average case LWE and SIS problems.

Few years later, structured variants of the LWE and SIS problems were proposed [Mic07,SSTX09], called Ring-SIS and Ring-LWE. These problems are preferred in practice, since they enjoy smaller storage and faster operations. There also exist reductions from worst-case ideal lattice problems to these structured variants [LPR10,SSTX09,LM06,PR06]. These two problems can be used to construct many basic cryptographic primitives such as PKE (adapting the schemes from [Reg05,GPV08]) and signatures [Lyu12,DDLL13,DM14].

Lattice Trapdoors. To construct IBE or Attribute Based Encryption (ABE), we can use trapdoors for the SIS problem. Initially described by Ajtai [Ajt96,Ajt99], a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a short basis of vectors satisfying $\mathbf{T_A} \cdot \mathbf{A} = \mathbf{0} \mod q$, generated together with $\mathbf{A}$. Given only $\mathbf{A}$, it is hard to find such a short basis but, with the knowledge of $\mathbf{T_A}$ it is easy to invert the SIS (and the ISIS) problem. Trapdoors constructions were improved by [AP09], and then described for ideal lattices in [SSTX09].

Micciancio and Peikert [MP12] proposed a new construction allowing a faster inversion of the ISIS problem, by reducing it to the inversion of a smaller problem for some structured gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times l}$. The matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is generated with its associated short basis $\mathbf{R} \in \mathbb{Z}^{(m-l)\times l}$, as $\mathbf{A} = (\mathbf{A}' \,|\, \mathbf{H}\mathbf{G} - \mathbf{A}'\mathbf{R})$ where $\mathbf{A}' \in \mathbb{Z}_q^{n\times(m-l)}$ is uniformly sampled, and $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ is an invertible element. In [GM18], Micciancio and Genise improved the inversion of the ISIS problem in the ring variant with an arbitrary modulus.

Lattice-based IBE. The first lattice-based IBE [GPV08] relies on the Dual-Regev encryption scheme. In Dual-Regev, a public key consists in a vector $\mathbf{u} = \mathbf{A}\mathbf{x} \mod q$, for a short vector $\mathbf{x} \in \mathbb{Z}^m$ (which is the secret key). To build the IBE, an identity can be mapped to a public key via a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^n$ modeled as a random oracle. The master secret key of the IBE is a short trapdoor $\mathbf{T_A}$, that makes it possible to extract a secret key $\mathbf{x}$ associated to any id by inverting $\mathbf{A}\mathbf{x} = \mathcal{H}(id) \mod q$.

This construction was later improved by removing the Random Oracle Model (ROM) [CHKP10,ABB10]. In ABB, a publicly computable matrix $\mathbf{A}_{id} = (\mathbf{A} \,|\, F(id)) \in \mathbb{Z}_q^{n\times(m+m')}$, is associated to an identity id, where $F(\cdot)$ is mapping identities to matrices in $\mathbb{Z}_q^{n\times m'}$. As a result, the secret key for an identity id is a short vector $\mathbf{x} \in \mathbb{Z}^{m+m'}$, such that $\mathbf{A}_{id}\mathbf{x} = \mathbf{u} \mod q$. To find such an $\mathbf{x} = (\mathbf{x}_1^T|\mathbf{x}_2^T)^T$, it is only required to sample a short $\mathbf{x}_2 \in \mathbb{Z}^{m'}$, and to use $\mathbf{T}_\mathbf{A}$ to invert the following ISIS problem: find a small $\mathbf{x}_1 \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x}_1 = \mathbf{u} - F(id)\mathbf{x}_2$. In 2014, Ducas, Lyubashevsky and Prest [DLP14] gave an NTRU variant of the GPV IBE scheme. In their work, the public key is built using NTRU lattices, which improves the efficiency of the scheme.

Our contributions. In this paper, we provide the first software implementation of a standard model IBE based on the hardness of Ring-SIS/LWE. Our main goal is to show that IBE schemes can, and without sacrificing efficiency, guarantee better security by being on the standard model and relying on a classical assumption on lattice problems. We instantiate our implemented IBE scheme from the selective secure IBE scheme described in [ABB10] as well as from the recent variant of trapdoor described in [MP12,GM18]. We also describe and implement the underlying signature scheme that achieve a selective notion of unforgeability based on the hardness of Ring-SIS/LWE. We choose to implement these selective secure schemes in the standard model due to their efficiency and also their simplicity compared to other adaptive secure variants [ABB10,DM14]. Our constructions work over polynomial rings $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ with n a power of two and q a prime modulus congruent to 1 mod $2n$. This setting is wide-spread in ideal lattice cryptography due to its efficiency thanks to the Number Theoretic Transform (NTT) representation. Note that our complete implementation can be found at `https://github.com/lbibe/code`.

We provide our implementation as a general-purpose thread-safe C++ library. We take much care to write plain C++ and not to explicitly include highly specialized instructions, such as AVX2 and NEON, in order to ensure portability over several hardware architectures. Instead, we rely on the GCC 7.2 compiler to make vectorization, and a set of optimization methods and multi-threading have been introduced using a number of C++11 specific features. More importantly, we design our code to be modular and provide three software layers, each of which is of independent interest. The software layers are (1) the different Gaussian sampling techniques based on [DP16,GM18], (2) the Gaussian sampling for trapdoor lattice with arbitrary modulus using [MP12], and (3) the underlying cryptographic constructions (both IBE and the related signature).

We experimentally evaluate the performance of these two constructions. To our surprise, the obtained runtime is fast and even quite close to that of other schemes built in the ROM or/and using different security assumptions. We remind that our goal is not to provide the most efficient IBE or signature scheme, but to show that "good" security properties can still be practical for many scenarios. In Table 1 and Table 2, we provide a comparison with state-of-the-art

Table 1: Timings for the different operations of IBE schemes: Setup (master key generation), Extract (user private key generation), encryption and decryption. Since Setup is performed only once, we provide the timing of only one single operation. Extract can measure how many users the system can manage. As for Encrypt/Decrypt, we give their throughput in KB per second.

Scheme	(λ, n)	Setup (ms)	Extract (ms)	Encrypt (KB/s)	Decrypt (KB/s)
BF-128 [Fou13]	$(128, -)$	–	0.55	4.10	6.19
DLP-14 [MSO17]	$(80, 512)$	4034	3.8	587	1405
This paper	$(80, 1024)$	1.67	4.02	230	1042

Table 2: Timings for the different operations of signature schemes: KeyGen (key generation), signature and verification. Since KeyGen is performed only once per user, we provide the timing of only one single operation. As for Sign/Verify, we give the timing as the number of sign/verify operations per second.

Scheme	(λ, n)	KeyGen (ms)	Sign (op/s)	Verify (op/s)
Falcon [FHK$^+$18]	(195, 768)[a]	53.48	202	2685
This paper	$(170, 1024)$	0.96	540	21276

[a] corresponds to NIST Security Level 2, since Level 1 only achieves 114 bits of security

implementations of IBE and signature schemes. We note the security parameter as λ (classical bit security in Tables 1-6) and the lattice dimension as n.

We obtain our results using dual-core Intel i7 2.6 GHz CPU with standard CPU benchmarks. Concerning the IBE schemes, we notice that our Setup (master key generation), compared to DLP-14, is (much) faster (we note that the DLP-14 Setup could now be improved using the recent results in Falcon [FHK$^+$18]), while decryption and Extract are of the same order of magnitude. However, our encryption is twice slower. For the sake of completeness, we include the timings of the paring-based IBE scheme of Boneh-Franklin. As for the signature scheme, we compare our implementation with Falcon [FHK$^+$18] that is the underlying signature scheme of [DLP14]. In our timings, we did not consider the phase of precomputations that we detail later in Section 5 (see Tables 5 and 6).

Related work. In [EBB13], the authors gave a software implementation of the trapdoor of Micciancio and Peikert [MP12] in both matrix and ring variants. They also included this trapdoor into the signature in the ROM of [GPV08]. Recently, in a concurrent work, a software implementation of the improvement lattice trapdoor [MP12,GM18] is given in [GPR$^+$17,DDP$^+$17], with application to respectively the GPV signature and the ABE scheme of [BGG$^+$14].

The only lattice based IBE given with an implementation we are aware of, is the one of Ducas, Lyubashevsky and Prest [DLP14]. In [DLP14], the authors gave a proof-of-concept implementation, recently improved by [MSO17]. This IBE scheme is the IBE of [GPV08], working with structured NTRU lattices. The Gram-Schmidt norm of a NTRU basis is quite small and efficiently computed, then the Gaussian sampling using this basis outputs better quality vectors.

Conclusion and Open problems. Our main contribution is a software implementation of a lattice-based IBE and signature scheme. Both constructions are proven secure in the standard model under the hardness of Ring-SIS/LWE. Our IBE is a ring-version of the selective ABE scheme, adapted using the MP12 trapdoor, we provide its underlying signature scheme and both security proofs.

We stress that our implementation has an efficiency comparable to NTRU based schemes in the ROM, even if we thought at first that a scheme on the standard model and based on Ring-SIS/LWE would be much less efficient. Then, we find it interesting to observe that even with the constraint on the choice of parameters and using a Gaussian sampling, those constructions, which are not using NTRU lattices, can also be efficient.

There are several open questions which would be interesting to answer. First, we choose to study the selective secure IBE scheme of ABB, and a signature scheme which achieve a selective notion of unforgeability. Both schemes can be improved to a better security by using the adaptive IBE scheme of ABB, and by looking at the standard secure signature of [DM14]. We also discuss in Section 2.5 the choice of the encoding hash function used in both schemes, which could be improved using the recent results of [LS18]. Finally, we want to modify NFLlib, which for now only allows us to use parameter q of size 30 or 62 bits. As discussed in Section 5, using a parameter q in between would optimize our choice of parameters. Note that using a Module variant, as in [DLL$^+$17], could also be a solution.

2 Preliminaries

Notations. Let D be a distribution over some finite set S, then $x \hookleftarrow D$ means that x is chosen from the distribution D and $x \hookleftarrow U(S)$ denotes the sampling of a uniformly random element x from S. We denote column vectors and matrices in bold, respectively by bold lowercase (e.g. $\mathbf{x}$) and bold uppercase (e.g. $\mathbf{A}$). The euclidean norm of the vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. The norm of a matrix $\|\mathbf{T}\| = \max_i \|\mathbf{t}_i\|$ is the maximum norm of its column vectors.

Lattices. An m-dimensional full-rank lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$. A lattice is the set of all integer combinations of some linearly independent basis vectors, $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_m\} \in \mathbb{R}^{m \times m}$, $\Lambda(\mathbf{B}) = \{\sum_{i=1}^{m} z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$. For n a power of two, the polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$ is isomorphic to the integer lattice $\mathbb{Z}^n$, a polynomial $f = \sum_{i=0}^{n-1} f_i x^i$ in R corresponds to the integer vector of its coefficients $(f_0, \cdots, f_{n-1})$ in $\mathbb{Z}^n$. The norm of a polynomial $\|f\|$ is the norm of its coefficients vector. For the rest of the paper we will work with polynomials over R, or $R_q = R/qR = \mathbb{Z}_q[x]/(x^n + 1)$ where q is a prime such that $q = 1 \mod 2n$.

Gaussian distribution. The Gaussian function of center $\mathbf{c} \in \mathbb{R}^n$ and width parameter σ is defined as $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2})$, for all $\mathbf{x} \in \mathbb{R}^n$. We can extend this definition to a positive definite covariance matrix $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^T$: $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi(\mathbf{x}-\mathbf{c})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c})\right)$. The discrete Gaussian distribution over a lattice Λ is defined as $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho\sigma,\mathbf{c}(\Lambda)}$ where $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$.

Tailcut. To tailcut less than $2^{-\lambda}$ of a one-dimensional Gaussian, we use the fact that $\Pr_{\mathbf{x}\hookleftarrow D_{\mathbb{Z},\sigma}}[|x| > t\sigma] \leq \operatorname{erfc}\left(t/\sqrt{2}\right)$, where $\operatorname{erfc}(x) = 1 - \frac{2}{\pi}\int_0^x \exp^{-t^2} \mathrm{d}t$. For example, $t = 12$ for $\lambda = 100$. Then, a vector $\mathbf{x}$ sampled in $D_{\mathbb{Z}^m,\sigma}$ would have small norm $\|\mathbf{x}\| \leq t\sigma\sqrt{m}$ with overwhelming probability.

2.1 Cryptographic problems on lattices

Ring-SIS/Ring-LWE. We use ring variants of SIS and LWE, proposed by [LM06,PR06] and [SSTX09,LPR10], and proven to be at least as hard as the GapSVP/SIVP problems on ideal lattices.

Definition 1 (Ring-SIS$_{q,m,\beta}$). *Given $\mathbf{a} = (a_1, \cdots, a_m)^T \in R_q^m$ a vector of m uniformly random polynomials, find a non-zero vector of small polynomials $\mathbf{x} = (x_1, \cdots, x_m)^T \in R^m$ such that $\mathbf{a}^T\mathbf{x} = \sum_{i=1}^m a_i \cdot x_i = 0 \mod q$ and $0 < \|\mathbf{x}\| \leq \beta$.*

Definition 2 (Decision Ring-LWE$_{n,q,D_{R,\sigma}}$). *Given $\mathbf{a} = (a_1, \cdots, a_m)^T \in R_q^m$ a vector of m uniformly random polynomials, and $\mathbf{b} = \mathbf{a}s + \mathbf{e}$, where $s \hookleftarrow U(R_q)$ and $\mathbf{e} \hookleftarrow D_{R^m,\sigma}$, distinguish $(\mathbf{a}, \mathbf{b} = \mathbf{a}s + \mathbf{e})$ from $(\mathbf{a}, \mathbf{b})$ drawn from the uniform distribution over $R_q^m \times R_q^m$.*

2.2 Dual-Regev Public Key Encryption

The Dual-Regev PKE, first described in [GPV08, Section 7.1], is the starting point to build an IBE on lattices. We describe its ring variant, with parameters n, m, and q integers and ζ, τ two real numbers.

- Key Generation: The secret key corresponds to a vector of small norm polynomials $\mathbf{x} \in R^m$, sampled from $D_{R^m,\zeta}$. The public key contains a uniformly random chosen vector of polynomials $\mathbf{a} \hookleftarrow U(R_q^m)$ and one more polynomial $u = \mathbf{a}^T\mathbf{x} \in R_q$.
- Encryption: The plaintext message consists in a binary polynomial $M \in R_2$. The ciphertext is composed of $m + 1$ Ring-LWE samples: $(\mathbf{b} = \mathbf{a}s + \mathbf{e}, c = u \cdot s + e' + \lfloor q/2 \rfloor M) \in R_q^m \times R_q$, where s is a uniformly chosen polynomial in R_q, and e', $\mathbf{e}$ follow a discrete Gaussian distribution respectively on R and R^m of parameter τ. The ciphertext is then $(\mathbf{b}, c)$.
- Decryption: The recipient uses his private key $\mathbf{x}$ to compute: $\mu = c - \mathbf{b}^T\mathbf{x} = e' - \mathbf{e}^T\mathbf{x} + \lfloor q/2 \rfloor M$. To recover M, he looks at each coefficient of μ, if μ_i is closer to 0 than to $\lfloor q/2 \rfloor$, the message bit $M_i = 0$, otherwise $M_i = 1$.

Security. The ring-variant of the Dual-Regev scheme is IND-CPA secure under the hardness of Ring-LWE$_{n,q,D_{R,\tau}}$ [LPR13]. The correctness of the decryption holds if the error term $\|e' - \mathbf{e}^T\mathbf{x}\|$ is small enough, less than $\lfloor q/4 \rfloor$.

2.3 Cryptographic definition of a signature

Signature. Let λ be the security parameter of the scheme, $\mathcal{M}$ and $\mathcal{S}$ denote respectively the set of messages and the set of signatures. A signature scheme is given by three probabilistic polynomial time algorithms:

$\mathsf{KeyGen}(1^\lambda) \to (vk, sk)$. Takes as input the security parameter λ and outputs a pair of keys, the verification key vk and the signing key sk.

$\mathsf{Sign}(1^\lambda, sk, M) \to \nu$. Takes as input the security parameter λ, the signing key sk and a message $M \in \mathcal{M}$, and outputs a signature $\nu \in \mathcal{S}$.

$\mathsf{Verify}(1^\lambda, vk, M, \nu) \to \{\text{accept, reject}\}$. Takes as input the security parameter λ, the verification key vk, the message M and a signature ν and either accepts or rejects.

Correctness. The signature is said correct if for all message $M \in \mathcal{M}$, $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ then $\mathsf{Verify}(1^\lambda, vk, M, \mathsf{Sign}(1^\lambda, sk, M)) = \text{accept}$ with overwhelming probability.

Security Game. We define here the selective unforgeability against chosen message attack (SU-CMA). The game proceeds as follows:

Init: The adversary $\mathcal{A}$ chooses the challenge message M^*.

Setup: The challenger runs $\mathsf{KeyGen}(1^\lambda)$ and gives the verification key vk to the adversary $\mathcal{A}$.

$\mathsf{Queries}$: The adversary can make signing queries on messages $M \neq M^*$, and the challenger answers by running $\nu \leftarrow \mathsf{Sign}(1^\lambda, sk, M)$.

$\mathsf{Forgery}$: Eventually, $\mathcal{A}$ outputs ν^* and wins if $\mathsf{Verify}(1^\lambda, vk, M^*, \nu^*) = \text{accept}$.

The advantage of the adversary $\mathcal{A}$ playing the SU-CMA security game is

$$\mathrm{Adv}(\mathcal{A})_{\mathrm{SIG}}^{\text{SU-CMA}} = \left| \Pr\left[\mathsf{Verify}(1^\lambda, vk, M^*, \nu^*) = \text{accept}\right] - \frac{1}{2} \right|.$$

A signature scheme is SU-CMA secure if, for all probabilistic polynomial time adversary $\mathcal{A}$, his advantage $\mathrm{Adv}(\mathcal{A})_{\mathrm{SIG}}^{\text{SU-CMA}}$ is negligible.

2.4 Cryptographic definition of an IBE scheme

Identity Based Encryption. Let λ be the security parameter of the scheme, an Identity Based Encryption (IBE) is composed of four probabilistic polynomial time algorithms:

$\mathsf{Setup}(1^\lambda) \to (mpk, msk)$. The first algorithm takes as input the security parameter λ and outputs a master key pair composed of the master public key mpk and the master secret key msk.

$\mathsf{Extract}(1^\lambda, mpk, msk, id) \to sk_{id}$. Takes as input the security parameter λ, the master key pair (mpk, msk) and an identity $id \in \mathcal{ID}$, and outputs an individual private key sk_{id} for the identity id.

$\mathsf{Encrypt}(1^\lambda, mpk, id, M) \to C$. Takes as input the security parameter λ, the master public key mpk, an identity $id \in \mathcal{ID}$, and a message $M \in \mathcal{M}$ and outputs a ciphertext C.

$\mathsf{Decrypt}(1^\lambda, mpk, sk_{id}, C) \to \{M, \perp\}$. Takes as input the security parameter λ, the master public key mpk, a private key associated to the identity sk_{id}, and a ciphertext $C \in \mathcal{C}$ and outputs a message $M \in \mathcal{M}$ or the symbol $\perp$ if the ciphertext is invalid.

Correctness. An IBE is said correct if for any message $M \in \mathcal{M}$, and identity $id \in \mathcal{ID}$, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $sk_{id} \leftarrow \mathsf{Extract}(1^\lambda, mpk, msk, id)$ yields

$$\mathsf{Decrypt}(1^\lambda, mpk, sk_{id}, \mathsf{Encrypt}(1^\lambda, mpk, id, M)) = M,$$

with overwhelming probability.

Security Game. We describe the ciphertext indistinguishability under a selective-identity chosen-plaintext attack (IND-sID-CPA) as a game between an adversary $\mathcal{A}$ and a challenger. The game proceeds as follows:

Init: The adversary $\mathcal{A}$ chooses the challenge identity id^*.

Setup: The challenger runs $\mathsf{Setup}(1^\lambda)$, gives the master public key msk to $\mathcal{A}$.

$\mathsf{Queries\ 1}$: The adversary can make private-key extraction queries on identities $id \neq id^*$, and the challenger answers by running $sk_{id} \leftarrow \mathsf{Extract}(1^\lambda, mpk, msk, id)$.

$\mathsf{Challenge}$: Adversary $\mathcal{A}$ outputs two plaintexts $M_0, M_1 \in \mathcal{M}$. The challenger chooses a random bit $b^* \hookleftarrow \{0,1\}$ and, sets the challenge ciphertext to $C^* = \mathsf{Encrypt}(1^\lambda, mpk, id^*, M_{b^*})$. The challenge ciphertext C^* is sent to $\mathcal{A}$.

$\mathsf{Queries\ 2}$: Adversary can make additionnal queries (answered as in $\mathsf{Queries\ 1}$).

Guess: Eventually, $\mathcal{A}$ 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 $\mathrm{Adv}(\mathcal{A})^{\text{IND-sID-CPA}}_{\text{IBE}} = \left|\Pr[b = b^*] - \frac{1}{2}\right|$. An IBE scheme is IND-sID-CPA secure if, for all PPT adversary $\mathcal{A}$, his advantage $\mathrm{Adv}(\mathcal{A})^{\text{IND-sID-CPA}}_{\text{IBE}}$ is negligible.

2.5 Hash functions

Our IBE and signature schemes use an encoding hash function $H : \mathbb{Z}_q^n \to R_q$ to map identities in $\mathbb{Z}_q^n$ to invertible elements in R_q. The security proof requires the map H to satisfy an injectivity property: the difference of two elements has to be invertible in R_q. Such hash functions have been called *encoding with Full-Rank Differences* (FRD) in [ABB10] and they must satisfy the following properties:

1. for all distinct $u, v \in \mathbb{Z}_q^n$, the element $H(u) - H(v) \in R_q$ is invertible; and
2. H is computable in polynomial time (in $n \log q$).

Implementation of FRD Hash Functions. An invertible element of R_q in the NTT domain corresponds to n non-zero integers of bit-size k. We implement our encoding in a naive manner, by generating these n integers using an PRNG with the identity id as a seed. In the literature, Ducas and Micciancio [DM14] chose the ring R_q, with q a power of 3 because in such ring any polynomial of degree less than $n/2$ with coefficients in $\{-1, 0, 1\}$ is invertible. Recently, Lyubashevsky and Seiler have proposed in [LS18] a way to construct such encoding using the fact that small non-zero polynomials are invertible in cyclotomic rings. They also use that $x^n + 1$ splits modulo q, and perform half of the FFT recursion tree, depending on the logarithm of the number of splittings, and at the end of the FFT tree, for small degree polynomials, multiplications have to be performed using naive or Karatsuba algorithm.

3 Trapdoors on lattices

The construction of our IBE scheme relies on a kind of trapdoors, introduced by Ajtai [Ajt96,Ajt99], and then improved, in particular in [MP12,GM18]. We define the trapdoor function

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \mod q$$

which represents the Inhomogeneous SIS problem (i.e. find a short non-zero vector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{u} \mod q$).

The trapdoor of Ajtai consists in a short basis $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$ of the m-dimensional integer lattice:

$$\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \text{ such that } \mathbf{A}\mathbf{x} = \mathbf{0} \mod q\}.$$

Thanks to a sufficiently short basis $\mathbf{T}_{\mathbf{A}}$, we can sample from a Gaussian distribution with a small parameter σ to obtain short vectors in $\Lambda_q^{\perp}(\mathbf{A})$. Then, solve the SIS, ISIS or LWE problems.

We use the ring version of a second notion of trapdoors (gadget based trapdoors) introduced by [MP12], and recently improved by [GM18], which are more efficient. In this construction, the matrix $\mathbf{A}$ is constructed by picking the first part uniformly at random, and the second part is almost uniformly random by including a structured gadget, to help the inversion of the SIS problem.

3.1 Gadget-based trapdoor construction

Gadget vector. We use the gadget vector $\mathbf{g} \in R_q^k$ for which the inversion of $f_{\mathbf{g}^T}(\mathbf{z}) = \mathbf{g}^T\mathbf{z} \in R_q$ is easy: it is a vector of constant polynomials, $\mathbf{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$ with $k = \lceil \log_2 q \rceil$. The lattice $\Lambda_q^{\perp}(\mathbf{g}^T)$ has a publicly known basis

$$\mathbf{B}_q = \begin{pmatrix} 2 & & & & q_0 \\ -1 & 2 & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & 2 & q_{k-2} \\ & & & -1 & q_{k-1} \end{pmatrix} \in R^{k\times k}, \text{ where } q = \sum_{i=0}^{k-1} 2^i q_i.$$

The quality of $\mathbf{B}_q$ can be seen as the norm of its Gram-Schmidt orthogonalization, which satisfies $\|\tilde{\mathbf{B}}_q\| \leq \sqrt{5}$.

Trapdoor construction. The trapdoor construction is an almost uniformly random vector of polynomials $\mathbf{a} \in R_q^m$ starting from a uniformly random vector of polynomials $\mathbf{a}' \in R_q^{m-k}$ (Alg. 3.1.1) that hides the structured vector $\mathbf{g}$, together with a trapdoor $\mathbf{T}$ that enables its owner to recover this structure when needed.

Definition 3 (g-trapdoor). *Let $\mathbf{a} \in R_q^m$ and $\mathbf{g} \in R_q^k$ with $k = \lceil \log_2 q \rceil$ and $m > k$. A $\mathbf{g}$-trapdoor for $\mathbf{a}$ consists in a matrix of small polynomials $\mathbf{T} \in R^{(m-k)\times k}$, following a discrete Gaussian distribution of parameter σ, such that $\mathbf{a}^T \binom{\mathbf{T}}{\mathbf{I}_k} = h\mathbf{g}^T$ for some invertible element $h \in R_q$. The polynomial h is called the tag associated to $\mathbf{T}$. The quality of the trapdoor is measured by its largest singular value $s_1(\mathbf{T})$.*

Algorithm 3.1.1 Algorithm $\mathsf{TrapGen}(q, \sigma, \mathbf{a}', h)$

```
Input: $q$ the ring modulus, $\sigma$ a Gaussian parameter. Optional $\mathbf{a}' \in R_q^{m-k}$ and $h \in R_q$.
    If no $\mathbf{a}$, $h$ is given as input, the algorithm chooses $\mathbf{a}' \hookleftarrow U\left(R_q^{m-k}\right)$ and $h = 1$.
Output: $\mathbf{a} \in R_q^m$ with its trapdoor $\mathbf{T} \in R^{(m-k)\times k}$, of norm $\|\mathbf{T}\| \leq t\sigma\sqrt{(m-k)n}$
    associated to the tag $h$.
    $\mathbf{T} \hookleftarrow D_{R^{(m-k)\times k},\sigma}$, $\mathbf{a} = (\mathbf{a}'^T \mid h\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$,
    return $(\mathbf{a}, \mathbf{T})$.
```

3.2 Preimage Sampling

Peikert Sampler. To find $\mathbf{x}$ such that $f_{\mathbf{a}^T}(\mathbf{x}) = u$, using this trapdoor, the idea is to find a $\mathbf{z}$ satisfying $f_{\mathbf{g}^T}(\mathbf{z}) = h^{-1} \cdot (u - \mathbf{a}^T\mathbf{p})$ and following a discrete Gaussian distribution of parameter α, where $\mathbf{p}$ is a perturbation vector with covariance matrix $\Sigma_{\mathbf{p}} = \zeta^2\mathbf{I}_m - \alpha^2 \binom{\mathbf{T}}{\mathbf{I}_k}\left(\mathbf{T}^T\ \mathbf{I}_k\right)$. Then, $\mathbf{x} = \mathbf{p} + \binom{\mathbf{T}}{\mathbf{I}_k}\mathbf{z}$, has covariance matrix $\Sigma_{\mathbf{x}} = \Sigma_{\mathbf{p}} + \alpha^2 \binom{\mathbf{T}}{\mathbf{I}_k}\left(\mathbf{T}^T\ \mathbf{I}_k\right) = \zeta^2\mathbf{I}_m$ and satisfies $\mathbf{a}^T\mathbf{x} = \mathbf{a}^T\mathbf{p} + \mathbf{a}^T\binom{\mathbf{T}}{\mathbf{I}_k}\mathbf{z} = \mathbf{a}^T\mathbf{p} + h\mathbf{g}^T\mathbf{z} = \mathbf{a}^T\mathbf{p} + h \cdot h^{-1}(u - \mathbf{a}^T\mathbf{p}) = u$. This idea is summarized in Alg. 3.2.1 $\mathsf{SamplePre}$.
It uses the two following algorithms:

Algorithm 3.2.1 Algorithm $\mathsf{SamplePre}(\mathbf{T}, \mathbf{a}, h, \zeta, \sigma, \alpha, u)$

```
Input: $\mathbf{a} \in R_q^m$, with its trapdoor $\mathbf{T} \in R^{(m-k)\times k}$ associated to an invertible tag $h \in R_q$,
  $u \in R_q$ and $\zeta$, $\sigma$ and $\alpha$ three Gaussian parameters.
Output: $\mathbf{x} \in R_q^m$ following a discrete Gaussian distribution of parameter $\zeta$ satisfying
  $\mathbf{a}^T\mathbf{x} = u \in R_q$.
  $\mathbf{p} \leftarrow \mathsf{SampleP}(q, \zeta, \alpha, \mathbf{T})$, $v \leftarrow h^{-1} \cdot (u - \mathbf{a}^T\mathbf{p})$,
  $\mathbf{z} \leftarrow \mathsf{SamplePolyG}(\sigma, v)$, $\mathbf{x} \leftarrow \mathbf{p} + \left(\begin{smallmatrix}\mathbf{T}\\ \mathbf{I}_k\end{smallmatrix}\right)\mathbf{z}$,
  return $\mathbf{x}$.
```

– $\mathsf{SamplePolyG}(\sigma, v) \to \mathbf{z}$, takes as input a Gaussian parameter σ and a target $v \in R_q$, outputs $\mathbf{z} \hookleftarrow D_{\Lambda_q^\perp(\mathbf{g}^T),\alpha,v}$, with $\alpha = \sqrt{5}\sigma$,
– $\mathsf{SampleP}(q, \zeta, \alpha, \mathbf{T}) \to \mathbf{p}$, takes as input the ring modulus q, ζ and α two Gaussian parameters and $\mathbf{T} \hookleftarrow D_{R^{(m-k)\times k},\sigma}$, outputs $\mathbf{p} \hookleftarrow D_{R^m,\sqrt{\Sigma_\mathbf{p}}}$, where $\Sigma_\mathbf{p} = \zeta^2\mathbf{I}_m - \alpha^2 \left(\begin{smallmatrix}\mathbf{T}\\ \mathbf{I}_k\end{smallmatrix}\right)\left(\begin{smallmatrix}\mathbf{T}^T & \mathbf{I}_k\end{smallmatrix}\right)$ with $\zeta > s_1(\mathbf{T})\alpha$.

4 Identity-based encryption

4.1 Identity-based encryption on lattices

The first IBE scheme on lattices was described in [GPV08], on the random oracle model. The master public key is a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its associated trapdoor $T_\mathbf{A} \in \mathbb{Z}^{m\times m}$ is the master secret key. A secret key associated to an identity $id \in \{0,1\}^*$ is a short vector $\mathbf{x} \in \mathbb{Z}^m$ which satisfies $\mathbf{A}\mathbf{x} = \mathcal{H}(id)$ mod q where $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^n$ is a hash function modeled as a random oracle. The encryption and decryption correspond to the Dual-Regev PKE with public key $(\mathbf{A}, \mathcal{H}(id))$ and private key $\mathbf{x}$.

In [CHKP10], the authors introduced a way to delegate a basis, i.e. a way to extend a basis: given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and its associated trapdoor $\mathbf{T_A}$, they find a way to construct a trapdoor $\mathbf{T}_{\mathbf{A}''}$ for a matrix $\mathbf{A}'' = (\mathbf{A} \mid \mathbf{A}') \in \mathbb{Z}_q^{n\times(m+m')}$. They use this idea to describe the first hierarchical identity-based encryption (in the random oracle model), and also to construct an IBE selective secure in the standard model. They described an IBE selectively secure in the standard model, where the matrix $\mathbf{A}_{id}$ is constructed as

$$\mathbf{A}_{id} = (\mathbf{A} \mid \mathbf{A}_1^{(id_1)} \mid \cdots \mid \mathbf{A}_l^{(id_l)}) \in \mathbb{Z}_q^{n\times m(l+1)},$$

depending on the bits $id_1, \cdots, id_l$ of the identity $id \in \{0,1\}^l$. The matrices $(\mathbf{A}_i^{(b)})$ have to be public, as well as an uniformly random vector $\mathbf{u} \in \mathbb{Z}_q^n$. By extending $\mathbf{T_A}$, we get a trapdoor for $\mathbf{T}_{\mathbf{A}_{id}}$ and a secret key on this scheme is a short vector $\mathbf{x} \in \mathbb{Z}^{m(l+1)}$ satisfying $\mathbf{A}_{id}\mathbf{x} = \mathbf{u} \mod q$. To encrypt and decrypt in the IBE scheme, we use $(\mathbf{A}, \mathbf{u})$ and $\mathbf{x}$ as public and private keys of the Dual-Regev scheme.

The IBE constructions in [ABB10] improve the IBE construction of [CHKP10] by reducing the size of the matrix $\mathbf{A}_{id}$ from $\mathbb{Z}_q^{n\times m(l+1)}$ to $\mathbb{Z}_q^{n\times 2m}$. We associate each identity $id \in \mathcal{ID}$ to a publicly computable matrix

$$\mathbf{A}_{id} = (\mathbf{A} \,|\, F(id)) \in \mathbb{Z}_q^{n \times 2m},$$

where $F(\cdot)$ is a function mapping identities to matrices in $\mathbb{Z}_q^{n\times m}$, which can be instantiated as:

- $F(id) = \mathbf{B} + H(id)\mathbf{C}$, in the selective secure construction, where $H : \mathcal{ID} \to \mathbb{Z}_q^{n\times n}$ is an encoding with Full-Rank Differences and $\mathbf{B}, \mathbf{C} \in \mathbb{Z}_q^{n\times m}$ are two uniform public matrices,
- $F(id) = \mathbf{B} + \sum_{i=1}^{l} id_i \mathbf{A}_i$, in the adaptive secure construction, where we need $l+1$ random matrices $\mathbf{B}, \mathbf{A}_1, \cdots, \mathbf{A}_l$ in the master public key. The adaptive security is a stronger notion of security where the adversary is allowed to make private key queries of its choice after receiving the master public key. At the end of this first queries phase, he output a target identity id^*, with the restriction that he did not ask a private key query for id^* during the previous phase.

4.2 Intuition

Our IBE construction is a ring-version of the selective IBE of [ABB10], adapted to use the gadget-based trapdoor of Micciancio and Peikert [MP12] (also adapted to rings, described in Section 3). We use the same encoding with Full-Rank Differences H to map identities to invertible elements in R_q (defined in Section 2.5).

The master public key consists in a uniformly random polynomial $u \in R_q$ and a pseudo-random vector of polynomials $\mathbf{a} \in R_q^m$. The master secret key is a $\mathbf{g}$-trapdoor $\mathbf{T} \in R^{(m-k)\times k}$ for $\mathbf{a}$ with associated tag set to zero:

$$\mathbf{a} = (\mathbf{a}'^T \mid -\mathbf{a}'^T\mathbf{T})^T.$$

Then, thanks to $\mathbf{a}$, we can compute a publicly computable vector associated to an identity id:

$$\begin{aligned}\mathbf{a}_{id} &= \mathbf{a}^T + (\mathbf{0} \,|\, H(id)\mathbf{g})^T \\ &= (\mathbf{a}'^T \,|\, H(id)\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T.\end{aligned}$$

where H is encoding with with Full-Rank Differences defined in Section 2.5.

The secret key associated to an identity id is a short vector $\mathbf{x} \in R^m$, computed thanks to the algorithm $\mathsf{SamplePre}$, which satisfies $\mathbf{a}_{id}^T\mathbf{x} = u \in R_q$. We also use the Dual-Regev encryption scheme to encrypt (using $\mathbf{a}_{id}$) and decrypt (using $\mathbf{x}$).

4.3 Our construction

The parameters of the scheme are n, m, q, k, and q integers, and σ, α, γ, τ, and ζ are real numbers, and chosen as described in Section 4.4.

1. $\mathsf{Setup}(1^n) \to (mpk, msk)$:

   (a) Compute $\mathbf{a} \in R_q^m$ associated to its trapdoor $\mathbf{T} \in R^{(m-k)\times k}$, $(\mathbf{a}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(q, \sigma, h = 0)$, i.e. $\mathbf{a} = (\mathbf{a}'^T \mid -\mathbf{a}'^T\mathbf{T})^T$,
   (b) Sample a uniformly random polynomial $u \hookleftarrow U(R_q)$,
   (c) Output $mpk = (\mathbf{a}, u) \in R_q^{m+1}$ and $msk = \mathbf{T} \in R^{(m-k)\times k}$.
2. $\mathsf{Extract}(mpk = (\mathbf{a}, u), msk = \mathbf{T}, id \in \mathcal{ID}) \rightarrow sk_{id}$:
   (a) Compute the tag $h_{id} = H(id)$,
   (b) Compute $\mathbf{a}_{id} = \mathbf{a}^T + (\mathbf{0} \mid h_{id}\mathbf{g})^T = (\mathbf{a}'^T \mid h_{id}\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$,
   (c) Sample a short $\mathbf{x} \leftarrow \mathsf{SamplePre}\,(\mathbf{T}, \mathbf{a}_{id}, h_{id}, \zeta, \sigma, \alpha, u)$, such that $\mathbf{a}_{id}^T\mathbf{x} = u$,
   (d) Output $sk_{id} = \mathbf{x} \in R^m$.
3. $\mathsf{Encrypt}(mpk = (\mathbf{a}, u), id, M \in R_2) \rightarrow C$:
   (a) Compute the tag h_{id}, and $\mathbf{a}_{id}$ like above,
   (b) Sample $s \hookleftarrow U(R_q)$, $\mathbf{e}_0 \hookleftarrow D_{R^{m-k},\tau}$, $\mathbf{e}_1 \hookleftarrow D_{R^k,\gamma}$, and $e' \hookleftarrow D_{R,\tau}$,
   (c) Compute $\mathbf{b} = \mathbf{a}_{id}s + (\mathbf{e}_0^T \mid \mathbf{e}_1^T)^T \in R_q^m$, and $c = u \cdot s + e' + \lfloor q/2 \rfloor M \in R_q$,
   (d) Output $C = (\mathbf{b}, c) \in R_q^{m+1}$.
4. $\mathsf{Decrypt}(sk^{id} = \mathbf{x}, C = (\mathbf{b}, c)) \rightarrow M$:
   (a) Compute $\mathrm{res} = c - \mathbf{b}^T\mathbf{x} = e' - (\mathbf{e}_0^T \mid \mathbf{e}_1^T)\mathbf{x} + \lfloor q/2 \rfloor M \in R_q$,
   (b) For each res_i, if it is closer to $\lfloor q/2 \rfloor$ than to 0, $M_i = 1$, otherwise $M_i = 0$.

Correctness. Let $\mathbf{x} = (\mathbf{x}_0^T | \mathbf{x}_1^T)^T$ with $\mathbf{x}_0 \in R_q^{m-k}$ and $\mathbf{x}_1 \in R_q^k$. To decrypt a ciphertext, we need the error term $e' - (\mathbf{e}_0^T \mid \mathbf{e}_1^T)(\mathbf{x}_0^T | \mathbf{x}_1^T)^T = e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1$ to be bounded by $\lfloor q/4 \rfloor$ (see Section 4.4 on the choice of the parameters).

Theorem 1. *Our IBE construction with parameters n, m, q, k, σ, α, γ, τ and ζ chosen as below is IND-sID-CPA secure in the standard model under the hardness of Ring-LWE$_{n,q,D_{R,\tau}}$.*

Proof. To prove the IND-sID-CPA security of the IBE scheme described above, we use a sequence of games starting from the original IND-sID-CPA game (Section 2.4). In the final game, the adversary has no information left about the initial message and hence has advantage zero. To ensure that the adversary has a negligible advantage in winning the IND-sID-CPA game, we show that a probabilistic polynomial time adversary cannot distinguish between games.

Game 0. The original IND-sID-CPA game between an adversary $\mathcal{A}$ and an IND-sID-CPA challenger.

Game 1. In the *Game 0*, the master public key of the scheme is $mpk = (\mathbf{a}, u)$, with $\mathbf{a}$ generated thanks to $\mathsf{TrapGen}(q, \sigma, h = 0)$ with an associated trapdoor $\mathbf{T}$ (i.e. $\mathbf{a} = (\mathbf{a}'^T \mid -\mathbf{a}'^T\mathbf{T})^T$), and u is a uniform polynomial in R_q.

In *Game 1*, we change the generation of the public vector $\mathbf{a}$ by adding information about the challenge identity id^* targeted by $\mathcal{A}$. The public parameter $\mathbf{a}$ is now generated thanks to $\mathsf{TrapGen}(q, \sigma, \mathbf{a}', -h_{id^*})$, i.e. $\mathbf{a} = (\mathbf{a}'^T \mid -h_{id^*}\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$, where the first part $\mathbf{a}' \in R_q^{m-k}$ is chosen from the uniform distribution. For the second part, $\mathbf{a}'^T\mathbf{T} = \left(\sum_{i=1}^{m-k} a_i t_{i,1}, \cdots, \sum_{i=1}^{m-k} a_i t_{i,k}\right)$, is either computationally or statistically indistinguishable from the uniform distribution depending on the trapdoor instantiation we choose.

For the computational instantiation, we set $m - k = 2$, and $\mathbf{a}' = (1, a)$ with $a \hookleftarrow U(R_q)$ and obtain a public vector

$$\mathbf{a} = (1, a \mid -(a \cdot t_{2,1} + t_{1,1}), \cdots, -(a \cdot t_{2,k} + t_{1,k})) \,.$$

To ensure such $\mathbf{a}$ looks uniform, we use the Ring-LWE assumption in its normal form: where the secret and the error follow the same distribution.

The adversary $\mathcal{A}$ issues private key queries on identities $id \neq id^*$, and the challenger has to answer to these queries. To do that, he computes

$$\begin{aligned}\mathbf{a}_{id} &= \mathbf{a}^T + (\mathbf{0} \mid h_{id}\mathbf{g}) \\ &= (\mathbf{a}'^T \mid (h_{id} - h_{id^*})\mathbf{g} - \mathbf{a}'^T \mathbf{T})^T,\end{aligned}$$

and use $\mathsf{SamplePre}(\mathbf{T}, \mathbf{a}_{id}, h_{id} - h_{id^*}, \zeta, \sigma, \alpha, u)$ to compute a private key associated to id: $\mathbf{x} \in R^m$ satisfying $\mathbf{a}_{id}^T \mathbf{x} = u$, because $h_{id} - h_{id^*}$ is invertible. Note that for $id = id^*$, $\mathbf{a}_{id} = (\mathbf{a}'^T \mid -\mathbf{a}'^T \mathbf{T})^T$, and $\mathcal{B}$ can no longer answer private key queries.

Game 2. In the last game, we change how the challenge ciphertext is build. The ciphertext C^* is now chosen uniformly in $R_q^m \times R_q$. The last step is to show that *Game 1* and *Game 2* are computationally indistinguishable for $\mathcal{A}$ by doing a reduction from the Ring-LWE problem. Suppose that $\mathcal{A}$ has non-negligible advantage in distinguishing these two games. Then a simulator $\mathcal{B}$ can use $\mathcal{A}$ to solve the Ring-LWE problem with non-negligible advantage (Section 2.1).

The simulator $\mathcal{B}$ receives $m - k + 1$ samples $(a_i, b_i)_{0 \leq i \leq m-k}$ as an instance of the decisional Ring-LWE problem. The simulator also receives the challenge identity id^* from the adversary $\mathcal{A}$. Let $\mathbf{a}' = (a_1, \cdots, a_{m-k})^T \in R_q^{m-k}$ and $\mathbf{b}' = (b_1, \cdots, b_{m-k})^T \in R_q^m$. The simulator runs $\mathsf{TrapGen}(q, \sigma, \mathbf{a}', -h_{id^*})$, because $\mathbf{a}'$ is following a uniform distribution thanks to the Ring-LWE assumption, and gets back $\mathbf{a} = (\mathbf{a}'^T \mid -h_{id^*}\mathbf{g} - \mathbf{a}'^T \mathbf{T})^T$, as in *Game 1*. Next, $\mathcal{B}$ sets $u = a_0$ and sends $(\mathbf{a}, u)$ to $\mathcal{A}$ as the master public key of the scheme. The adversary $\mathcal{A}$ issues private key queries, and $\mathcal{B}$ answer to these queries like in *Game 1*.

Then the attacker $\mathcal{A}$ sends two messages M_0, M_1 to $\mathcal{B}$. $\mathcal{B}$ generates a random bit b^*, and generates the challenge ciphertext $C^* = (\mathbf{b}^*, c^*)$ as follows:

$$\mathbf{b}^* = (\mathbf{b}'^T \mid -\mathbf{b}'^T \mathbf{T} + \hat{\mathbf{e}}^T)^T, \quad c^* = b_0 + \lfloor q/2 \rfloor M_{b^*},$$

where $\hat{\mathbf{e}} \hookleftarrow D_{R^k, \mu}$ for some μ real.

– If the Ring-LWE samples are drawn from the Ring-LWE distribution, we have $\mathbf{b}' = \mathbf{a}' s + \mathbf{e}$ and $b_0 = a_0 \cdot s + e_0$, for some $s \in R_q$, $\mathbf{e} \hookleftarrow D_{R^{m-k}, \tau}$ and $e_0 \hookleftarrow D_{R,\tau}$. By substitution,

$$\begin{aligned}\mathbf{b}^* &= (\mathbf{b}'^T \mid -\mathbf{b}'^T \mathbf{T} + \hat{\mathbf{e}}^T)^T)^T \\ &= ((\mathbf{a}' s + \mathbf{e})^T \mid -(\mathbf{a}' s + \mathbf{e})^T \mathbf{T} + \hat{\mathbf{e}}^T)^T)^T \\ &= \mathbf{a}_{id^*}^T s + (\mathbf{e}^T \mid -\mathbf{e}^T \mathbf{T} + \hat{\mathbf{e}}^T)^T)^T\end{aligned}$$

$$\text{and } c^* = b_0 + \lfloor q/2 \rfloor M_{b^*} = a_0 \cdot s + e_0 + \lfloor q/2 \rfloor M_{b^*}.$$

For fixed $\mathbf{e}$, the error term $-\mathbf{e}^T\mathbf{T} + \hat{\mathbf{e}}^T$ is indistinguishable from a sample drawn from the distribution $D_{R^k,\gamma}$ with $\gamma^2 = (\sigma\|\mathbf{e}_0\|)^2 + \mu^2$, for μ well chosen. Then the challenge ciphertext, $(\mathbf{b}^*, c^*)$ follows the same distribution as in the IBE of *Game 2*.

– If the Ring-LWE samples are uniformly random samples, the ciphertext challenge also looks uniform. Then, the challenge ciphertext C^* is always a uniform element in $R_q^m \times R_q$ At the end, the adversary $\mathcal{A}$ outputs a guess b.

If $b = b^*$ with overwhelming probability, the simulator concludes that the Ring-LWE instance was drawn from the Ring-LWE distribution, otherwise $\mathcal{B}$ concludes that the Ring-LWE distribution was drawn from the uniform distribution. □

4.4 Parameter choices

In our construction, the modulus q is chosen to be a prime of size 14, 30 or 62 bits (implementation purpose of NFLlib [AMBG+16]). We use the computational instantiation of the trapdoor with $m - k = 2$, and $\mathbf{a}'$ corresponds to two polynomials, the first sets to one and the second to a uniform polynomial $a \hookleftarrow U(R_q)$, as suggested in [EBB13]. We obtain a public vector, associated to tag h: $\mathbf{a} = (1, a \mid h \cdot g_1 - (a \cdot t_{2,1} + t_{1,1}), \cdots, h \cdot g_k - (a \cdot t_{2,k} + t_{1,k}))$.

– The Gaussian parameter σ for the trapdoor sampling is $\sigma > \sqrt{(\ln(2n/\epsilon)/\pi)}$ [MP12] where n is the maximum length of the ring polynomials, and ϵ is the desired bound on the statistical error introduced by each randomized-rounding operation. This parameter is also chosen to ensure that the Ring-LWE instance of parameters n, q and σ is hard.
– The Gaussian parameter α used for the G-sampling [MP12] is $\alpha = \sqrt{5}\sigma$.
– By [GM18], the parameter ζ satisfies $\zeta > s_1(\mathbf{T})\,\alpha$. The spectral norm of $\mathbf{T}$ is a subgaussian random matrix of parameter σ [MP12, Lemma 2.9]. There exists a universal constant C ($\sim 1/\sqrt{2\pi}$), such that for any $t' \geq 0$, we have $s_1(\mathbf{T}) \leq C\sigma(\sqrt{kn} + \sqrt{2n} + t')$ except with probability at most $2\exp\left(-\pi(t')^2\right)$. Then, we get $\zeta > \sqrt{5}C\sigma^2(\sqrt{kn} + \sqrt{2n} + t')$.
– Finally, to choose the Gaussian parameters γ, τ for the Dual-Regev encryption, we need $\|e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1\| < \lfloor q/4 \rfloor$, which thanks to Section 2 gives:

$$\|e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1\| \leq t\tau\sqrt{n} + 2t^2\tau\zeta n + t^2\gamma\zeta kn < \lfloor q/4 \rfloor.$$

– Moreover, γ needs to satisfy $\gamma = \sqrt{\sigma^2\|\mathbf{e}_0\|^2 + \mu^2}$ so that the security proof holds. We can have an idea of the norm of $\mathbf{e}_0$ by using the Section 2,

$$\gamma^2 = \sigma^2\|\mathbf{e}_0\|^2 + \mu^2 \leq \sigma^2(t\tau\sqrt{2n})^2 + \mu^2,$$

we choose $\mu = t\sigma\tau\sqrt{2n}$, and then $\gamma = 2t\sigma\tau\sqrt{n}$.

Table 3: Parameters set for our IBE construction.

n	k	σ	LWE$_\sigma$	ζ	γ	τ	LWE$_\tau$	λ
512	50	3.3	2^{41}	1935.7	4493.2	3.3	2^{41}	40
1024	51	5	2^{80}	6360.5	14747.7	5	2^{80}	80
2048	62	6.7	2^{197}	16898.5	64541.9	6.7	2^{198}	195

Parameters set. We combine all the conditions to obtain the following set of parameters, used in our implementation. *Remark: The bit security of the underlying Ring-SIS instance does not appear here because it dominates the bit security of the Ring-LWE instances.*

4.5 Underlying signature scheme

Behind the IBE scheme above, there is an underlying natural signature scheme. The key generation in both scheme consists in creating a public vector $\mathbf{a} \in R_q^m$ together with its trapdoor $\mathbf{T} \in R^{(m-k)\times k}$. The signature of a message corresponds to the secret key associated to an identity in the IBE scheme. A signature $\mathbf{x}$ of a message M is a solution of $\mathbf{a}_M^T \mathbf{x} = 0 \mod q$. To check if $\mathbf{x}$ is a valid signature for some message M, we have to check that $\mathbf{x}$ is a non trivial solution to the Ring-SIS instance above ($\mathbf{x} \neq \mathbf{0}$, $\mathbf{a}_M^T \mathbf{x} = 0 \mod q$ and $\|\mathbf{x}\| \leq t\zeta\sqrt{nm}$).

Construction. The parameters of the scheme are n, m, q, k, and q integers, and σ, α, and ζ are reals. Let $H : \mathbb{Z}_q^n \to R_q$ be a FRD map.

1. $\mathsf{KeyGen}(1^n) \to (vk, sk)$:
   (a) Compute $\mathbf{a} \in R_q^m$ associated to its trapdoor $\mathbf{T} \in R^{(m-k)\times k}$, $(\mathbf{a}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(q, \sigma, h = 0)$, i.e. $\mathbf{a} = (\mathbf{a}'^T \mid -\mathbf{a}'^T\mathbf{T})^T$,
   (b) Output $vk = \mathbf{a} \in R_q^m$ and $sk = (\mathbf{a}, \mathbf{T}) \in R_q^m \times R^{(m-k)\times k}$.
2. $\mathsf{Sign}(sk = (\mathbf{a}, \mathbf{T}), M \in R_2) \to \nu$:
   (a) Compute the tag $h_M = H(M)$,
   (b) Compute $\mathbf{a}_M = \mathbf{a}^T + (\mathbf{0} \mid h_M\mathbf{g})^T = (\mathbf{a}'^T \mid h_M\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$,
   (c) Sample a short $\mathbf{x} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{a}_M, h_M, \zeta, \sigma, \alpha, 0)$, with $\mathbf{a}_M^T\mathbf{x} = 0$,
   (d) Output $\nu = \mathbf{x} \in R^m$.
3. $\mathsf{Verify}(vk = \mathbf{a}, M, \nu = \mathbf{x}) \to \{\text{accept, reject}\}$:
   (a) Compute the tag h_M and $\mathbf{a}_M$ like above,
   (b) Accept if, and only if, $\mathbf{a}_M^T\mathbf{x} = 0 \mod q$ and $0 < \|\mathbf{x}\| \leq t\zeta\sqrt{mn}$.

Correctness. Thanks to Section 2, with high probability the norm of a signature outputted by $\mathsf{SamplePre}$ is bounded by $t\zeta\sqrt{nm}$, because it is an integer vector of size nm and of Gaussian parameter ζ.

Theorem 2. *Our signature construction with parameters n, m, q, k, σ, α, and ζ chosen as below is SU-CMA (Selective Unforgeability against Chosen Message Attack) secure in the standard model under the hardness of Ring-SIS$_{q,m-k,\beta}$ with $\beta = (1 + t\sigma\sqrt{(m-k)n})t\zeta\sqrt{mn}$.*

Proof. Let $\mathcal{A}$ be an adversary attacking the signature scheme above through an SU-CMA attack. We build a simulator $\mathcal{B}$ attacking the Ring-SIS problem.

Init. The simulator $\mathcal{B}$ receives $m-k$ uniformly random and independent samples from R_q, $\mathbf{a}' = (a_1, \cdots, a_{m-k})^T \in R_q^{m-k}$ as an Ring-SIS instance, and the challenge message M^* from the adversary $\mathcal{A}$. The simulator runs $\mathsf{TrapGen}(q, \sigma, \mathbf{a}', -h_{M^*})$, and gets back $\mathbf{a} = (\mathbf{a}'^T \mid -h_{M^*}\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$ together with $\mathbf{T} \hookleftarrow D_{R^{(m-k)\times k},\sigma}$. Next, $\mathcal{B}$ sends $vk = \mathbf{a} \in R_q^m$ to $\mathcal{A}$.

Signing queries. The adversary $\mathcal{A}$ issues signing queries on messages $M \neq M^*$, and $\mathcal{B}$ has to answer to these queries. To do that, he computes $\mathbf{a}_M = \mathbf{a}^T + (\mathbf{0} \mid h_M\mathbf{g}) = (\mathbf{a}'^T \mid (h_M - h_{M^*})\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$, and then he can use $\mathsf{SamplePre}(\mathbf{T}, \mathbf{a}_M, h_M - h_{M^*}, \zeta, \sigma, \alpha, 0)$ to find a signature $\mathbf{x} \in R^m$ satisfying $\mathbf{a}_M^T\mathbf{x} = 0 \mod q$.

Forgery. Eventually, $\mathcal{A}$ outputs a forgery ν^* for M^*, satisfying $\mathbf{a}_{M^*}^T\nu^* = 0 \mod q$, which gives $\mathbf{a}'^T \underbrace{(I_{m-k} \mid -\mathbf{T})\nu^*}_{\mathbf{z}} = 0 \mod q$.

The norm of $\|\mathbf{T}\| \leq t\sigma\sqrt{(m-k)n}$ because each of its columns is a Gaussian vector of size k and of parameter σ. Then, the vector $\mathbf{z}$ is a solution of the Ring-SIS instance of norm $\|\mathbf{z}\| = (I_{m-k} \mid -\mathbf{T})\nu^* \leq \beta = (1 + t\sigma\sqrt{(m-k)n}) \cdot t\zeta\sqrt{mn}$. $\square$

Parameters choices. The parameters n, m, q, k, q, σ, α, and ζ follow the same conditions detailed above for the IBE scheme. Moreover, we need to look at the underlying Ring-SIS instance of size $m-k=2$ and of norm $\beta = (1 + t\sigma\sqrt{2n})t\zeta\sqrt{mn}$ which corresponds to a SIS instance of size n times bigger. The two following conditions $\beta \geq \sqrt{2n}q^{n/2n} = \sqrt{2nq}$ and $q \geq \beta\sqrt{n}\omega(\log n)$ ensure that the SIS problem has a solution and that is hard.

To get an idea of the security achieved by a SIS instance, we follow the general framework of [APS15,CN11]. To ensure that the shortest vector outputted by BKZ is a solution of our SIS instance of norm $\beta = (1 + t\sigma\sqrt{2n})t\zeta\sqrt{mn}$, the root Hermite factor δ need to satisfy $\frac{\beta}{q^{n/2n}} = \frac{\beta}{\sqrt{q}} = \delta^{2n}$. To achieve this Root Hermite factor, we need to run BKZ with block size at least b. The estimated cost of running BKZ with block size b, a number of rounds of $N = \frac{(2n)^2}{b^2}\log(2n)$ and on dimension $2n$ is $\mathrm{cost}(\mathrm{BKZ}_{b,N}) \approx \frac{(2n)^3}{b^2}\log(2n) \cdot \mathrm{cost}(\mathrm{SVP\ oracle})$.

Parameters set. We now combine all those conditions to obtain the following set of parameters. *Remark: We can also instantiate this scheme such that the public key is statistically close to uniform by using a Regularity Lemma ([SS11, Theorem 3.1]), but in this case, $m-k$ is slightly larger and σ is much larger.*

Table 4: Parameters set for our signature scheme.

n	k	σ	LWE$_\sigma$	ζ	δ	b	SIS	λ
512	30	4.2	2^{64}	2529.3	1.011380	62	2^{74}	60
1024	24	5.8	2^{378}	6143.8	1.008012	132	2^{156}	140
1024	30	6.3	2^{246}	8023.6	1.007348	154	2^{184}	170

5 Implementation

5.1 General Description

Our implementation was carried out in plain C++11 as a general-purpose library. We now discuss the different principles followed during the design of our code:

Cutting-edge compiler. We have always used the latest version available of the GCC compiler to build our binary. Our final timings have been produced using the GCC 7.2 compiler that allows us to perform various optimization and code sanitization that do not exist in older versions.

Thread-Safety. Lattice-based constructions are known to be highly parallelizable. Therefore, we build our library so that it could be simultaneously called from concurrent threads. To this end, we only use the <*thread*> model of C++11, and not OpenMP or OpenCL, in order to keep the design as simple as possible.

Portability. Modern processors often come with a combination of advanced hardware instructions: SSE 4.1, AVX, AVX2, AVX512+flavors and NEON. These instructions allow developers to boost the performance of their applications. Many developers tend to explicitly include such instructions inside their code to gain in efficiency. However, we do not use this method, since it limits the portability of the code. Instead, we rely on the compiler to insert them depending on the required optimization level and the targeted machine using its own auto-vectorization techniques. Thus, our code can be easily compiled for INTEL and ARM processors without any modification.

Dedicated polynomial ring library. We avoid the use of *generic* number theory libraries, such as NTL and FLINT, in our implementation. Instead, we preferred to use NFLlib library [AMBG$^+$16] which offers fast implementations of arithmetic operations over the ring. However, NFLlib has a primary drawback: it does not allow developers to natively choose a prime modulus q of size between 30 bits and 62 bits. This has resulted in performance penalty on our implementation when q can be of size 50 bits, since we had to use 62 bits.

Double-precision float. Our implementation does not depend on multi-precision floating-point arithmetic. Instead, all floating-point computations are performed using double-precision arithmetic, as in [GM18].

Modularity. We designed our code to be composed of three software layers. In what follows, we describe these layers and discuss some technical choices concerning our implementation.

5.2 Software Layers

Gaussian Preimage Sampling. This layer implements the Peikert Gaussian sampler. As mentioned in section 3.2, this consists of combining two stages: an off-line (target independent) stage SampleP generating perturbation vectors, and an on-line (target dependent) stage SamplePolyG generating samples from a particular lattice. We note that this layer does not regard the actual implementation of SampleP and SamplePolyG. We also note that we never include the runtime execution of SampleP in our timings for any operation that does need preimage sampling. Indeed, we suppose that the trusted party, or the signer, periodically calls SampleP and stores the resulted outputs. Below, we provide more details about the runtime of this off-line operation.

SampleP and SamplePolyG. This layer implements the recent techniques described in [GM18] and [DP16] for Gaussian sampling. We note that SamplePolyG just calls SampleG (Figure 2 in [GM18]) n times to build k polynomials in R_q. For instance, when $n = 1024$ and $k = 30$ bits, SamplePolyG calls the underlying sampler 1024 times, and then builds 30 polynomials in their NTT. This n times sampling constitutes the main bottleneck of our Extract/Sign algorithms. As for SampleP, we use the Fourier representation in the finite field as well as the FFT's butterfly transformation to speed up all the required multiplication/inversion. To this end, we implement our C++ Cooley-Tukey FFT and optimize it using template class recursion (a template class that recursively uses its own definition).

IBE and Signature. Here, we implement the described signature and IBE schemes using mainly our Gaussian Preimage Sampling and NFLlib to perform arithmetic operations. For the IBE scheme, we build two classes in order to simulate the different roles: the trusted party that generates the master keys and capable of extracting users private keys, and a user that is able to encrypt using the identity of another user as well as to decrypt using its own private key. This abstraction allows us to easily set up our test benchmark with one trusted party and several users. We note that we do not include the timings of the instantiation of the different objects, since it is done only once during the setup of the entire environment, and more importantly, it concerns memory allocations and some initial computations that could be performed otherwise.

5.3 Experimental results

Our timings have been obtained on an Intel i7-5600 2.6 GHz CPU. Clock cycles were measured with the *high_resolution_clock* class of C++11. Results are provided in Table 5 (resp. Table 6) for the IBE (resp. signature) scheme.

We underline that we obtained our results using the security parameters in Table 3 and 4 except for k. Indeed, NFLlib limits the choice of k, then we use $k = 30$ if it is smaller or equal to 30, and $k = 62$ otherwise. We note that modifying NFLlib to take into account arbitrary moduli is possible, but here, we did not and thus provide upper bounds to our implementations. This explains our similar results when signing for two different levels of security (with $n = 1024$). Our timings show that performance is still practical for many scenarios. We also notice that timings are almost just multiplied by 2 for twice security.

Table 5: Timings in ms for the different operations of the IBE scheme: Setup, PreCompute, Extract, Encrypt and Decrypt.

(λ, n)	Setup	PreCompute	Extract	Encrypt	Decrypt
$(40, 512)$	0.93	1.32	2.27	0.45	0.0625
$(80, 1024)$	1.67	3.125	4.02	1.0	0.12
$(195, 2048)$	3.125	6.67	8.19	2.44	0.94

Table 6: Timings in ms for the different operations of the Signature scheme: KeyGen, PreCompute, Sign and Verify.

(λ, n)	KeyGen	PreCompute	Sign	Verify
$(60, 512)$	0.52	1.05	1.12	0.025
$(140, 1024)$	0.91	3.44	2.0	0.043
$(170, 1024)$	0.96	3.92	1.85	0.047

Comparison with related work. Now, we provide more insight about the Tables 1 and 2 in the introduction, as well as Table 7 presented below.
IBE Scheme. The closest work to our implemented IBE is the one presented in [DLP14] and re-implemented in [MSO17]. Indeed, both requires Gaussian

Table 7: Timings for the different proposals of the NIST competition. Refer to Table 2 for the notations.

Scheme	(λ, n)	KeyGen (ms)	Sign (op/s)	Verify (op/s)
Dilithium [DLL$^+$17]	(128, 768[a])	0.08	2263	10660
qTesla [BAA$^+$18]	$(128, 1024)$	0.88	1267	5938
Falcon [FHK$^+$18]	(195, 768)[b]	53.48	202	2685
DRS [PSDS18]	$(128, 1024)$	380	50	6
This paper	$(140, 1024)$	0.91	498	23000

[a] based on Module-LWE/SIS with a ring of size 256
[b] corresponds to 172 bits of quantum security

sampling to extract users' private keys, and our encryption/decryption algorithms are very similar. However, DLP14 relies on NTRU lattices, which allow them to tremendously reduce the size of users keys, and therefore the number of the required Gaussian sampling (only one), while we need n calls to $\mathsf{SampleG}$. Our implementation is then slower for the Extract and Encrypt operations, nevertheless, due to our efficient implementation of [GM18], the difference is quite small. However, our Setup is much faster. We recall that the trusted party generates the private key for each user only once. Therefore, this operation is less critical than the encryption/decryption.

Signature Scheme. We compare our implementation with the lattice-based signature proposals of the NIST competition[1]. We compiled and ran the *Reference Implementation* of each scheme that corresponds to the *NIST Security Level 1*, namely 128 quantum bit security. Because of some run-time errors, we did not include the timing for pqNTRUsign. We are aware of how different these schemes are in their design and their choice of security parameters. We emphasize that our evaluation study is not complete, especially that we did not implement the secure hash function (as discussed in Section 2.5) that might slow down our signature scheme. However, our results allow us to consider our performance compared to highly optimized signature schemes. As for signature, our implementation performs three to four times slower than Dilithium and qTesla, while it outperforms DRS. But our verification is much faster than other schemes, this asymmetry could be useful in case signatures are produced by powerful machines (e.g. a server) and to be verified by constrained devices. These results confirm that our approach remains interesting for both security and performance.

Storage requirements. Here, we give an estimation of the storage requirements of our implementation for the different entities. We note that, in the IBE scheme, the trusted party requires the Gaussian Sampler, while the user requires it for the signature scheme. Our estimations are directly based on the different fields of our classes, which reflects the modular structure of our implementation. We note that the precomputed values take much space, and therefore a trade-off between the number of precomputations and the allocated storage must be found. A direct conclusion can be drawn from our Table 8: lattice-based constructions are not yet ready for constrained devices since, for instance when $n = 512$ and $k = 30$, the public key is of size 61.875 KB, which is quite big for some systems.

Acknowledgments. This work has received a French government support managed by the National Research Agency in the "Investing for the Future" program, under the national project RISQ P141580-2660001 / DOS0044216, and under the projet TYREX granted by the CominLabs excellence laboratory with reference ANR-10-LABX-07-01. Pauline Bert is funded by the Direction Générale de l'Armement (Pôle de Recherche Cyber).

[1] We got all the codes from the NIST website `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`

Table 8: Storage Requirements in Bits

(a) Trusted Party

Public Key	$nk^2 + 3nk$
Private Key	$2nk$

(b) Cipher/Signature

Cipher	$3nk + nk^2$
Signature	$2nk + nk^2$

(c) User

Public Key	$3nk + nk^2$
Private Key	$2nk + nk^2$

(d) Gaussian Sampler

SampleP	$384n$
SampleG	$192n$
Precomputations	$2kn + nk^2$

References

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

Ajt96. M. Ajtai. Generating hard instances of lattice problems. In *Proc. of STOC*, pages 99–108. ACM, 1996.

Ajt99. Miklós Ajtai. Generating hard instances of the short basis problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1–9. Springer, 1999.

AMBG$^+$16. C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint. NFLlib: NTT-based Fast Lattice Library. In *RSA Conference Cryptographers' Track* , 2016.

AP09. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *In STACS*. Citeseer, 2009.

APS15. M. R Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, pages 169–203, 2015.

BAA$^+$18. N. Bindel, S. Akleylek, E. Alkim, P. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. Ricardini, and G Zanon. qtesla, January 2018.

BF01. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proc. of CRYPTO 2001*. Springer, 2001.

BGG$^+$14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*, pages 533–556. Springer, 2014.

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *In Proc. of EUROCRYPT 2010*, page 523, 2010.

CN11. Y. Chen and P.Q Nguyen. BKZ 2.0: Better lattice security estimates. In *Proc. of ASIACRYPT 2011*, pages 1–20. Springer, 2011.

Coc01. C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA CC*, pages 360–363. Springer, 2001.

DDLL13. L. Ducas, A. Durmus, T. Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Proc. of CRYPTO*. Springer, 2013.

DDP$^+$17. W. Dai, Y. Doröz, Y. Polyakov, K. Rohloff, H. Sajjadpour, E. Savaş, and B. Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. Cryptology ePrint Archive, Report 2017/601, 2017.

DLL$^+$17. L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle. CRYSTALS – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017.

DLP14. L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In *Proc. of ASIACRYPT 2014*, volume 8874, page 22. Springer, 2014.

DM14. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *International Cryptology Conference*, pages 335–352. Springer, 2014.

DP16. L. Ducas and T. Prest. Fast fourier orthogonalization. ISSAC '16, pages 191–198. ACM, 2016.

EBB13. R. El Bansarkhani and J. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In *SAC*, pages 48–67. Springer, 2013.

FHK$^+$18. P. 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, January 2018.

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

GHPT17. P. Gaborit, A. Hauteville, D. H. Phan, and J.-P Tillich. Identity-based encryption from codes with rank metric. Cryptology ePrint Archive, Report 2017/514, 2017. `http://eprint.iacr.org/2017/514`.

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

GPR$^+$17. K. Doruk Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savaş. Implementation and evaluation of improved gaussian sampling for lattice trapdoors. Cryptology ePrint Archive, Report 2017/285, 2017.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. How to use a short basis: Trapdoors for hard lattices and new cryptographic constructions. *In proc. of STOC*, 2008.

LM06. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. *Automata, Languages and Programming*, pages 144–155, 2006.

LPR10. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *EUROCRYPT 2010*, page 1, 2010.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for Ring-LWE cryptography. *EUROCRYPT 2013*, page 35, 2013.

LS18. V. Lyubashevsky and G. Seiler. Partially splitting rings for faster lattice-based zero-knowledge proofs. In *EUROCRYPT 2018*, 2018. In Press.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. *EUROCRYPT 2012*, page 738, 2012.

Mic07. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Comp. Complexity*, 16(4):365–411, 2007.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *EUROCRYPT 2012*, page 700, 2012.

MSO17. S. McCarthy, N. Smyth, and E. OSullivan. A practical implementation of identity-based encryption over NTRU lattices. Cryptology ePrint Archive, Report 2017/1049, 2017.

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

PSDS18. T. Plantard, A. Sipasseuth, C. Dumondelle, and W. Susilo. Drs : Diagonal dominant reduction for lattice-based signature, January 2018.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. of STOC*, 2005.

Sha85. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of CRYPTO 84*, pages 47–53. Springer-Verlag New York, Inc., 1985.

Sho97. P. W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484, 1997.

SS11. D. Stehlé and R. Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Eurocrypt*, volume 6632, pages 27–47. Springer, 2011.

SSTX09. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public-key encryption based on ideal lattices. In *Asiacrypt 2009*, pages 617–635, 2009.

Yam16. S. Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 32–62. Springer, 2016.