On the implementation of a lattice-based revocable hierarchical Ibe

Mikael Carmona, Doryan Lesaignoux, Antoine Loiseau

On the Implementation of a Lattice-Based Revocable Hierarchical Ibe

Mikael Carmona, Doryan Lesaignoux and Antoine Loiseau
Univ. Grenoble Alpes, CEA, Leti, MINATEC Campus, F-38054 Grenoble, France

Keywords: Revocable Hierarchical Identity-Based Encryption, Lattice-Based Cryptography, Software Implementation, Performances.

Abstract: Identity Based Encryption (IBE) is a serious alternative of Public Key Infrastructure when considering distributed systems such as wireless sensors network, multi-site enterprise, manufacturing sites, and so on. In particular, Revocable Hierarchical IBE (RHIBE) provides all functionalities required for an operational cryptography deployment. This paper proposes a parameter analysis, and a software implementation of one of the most advanced post-quantum RHIBE. The objective is to quantify the performances in software and to provide a concrete set of parameters for a given level of security. For the best of our knowledge, this was not done from previous works that only provide order of magnitudes about parameters and instances sizes. Regarding applications and from today, post-quantum RHIBE lead to very large keys and ciphertext size, letting it difficult to consider such cryptosystems for constraint devices.

1 INTRODUCTION

Identity Based Encryption (IBE) is an alternate to Public Key Infrastructures (PKI) for deploying asymmetric cryptography. Based on centralized master public and master private keys, it avoids the use of certificates by using the identity directly (*i.e.* an identifier) of the recipient as a public key to encrypt messages. Public keys, secret keys and decryption keys are derived from the master keys through a hierarchal structure. Each node of the structure is able to enroll and revoke new devices belonging to its substructure. This appears useful to deploy cryptography in IoT-Cloud context, for example. IBE simplifies the deployment by avoiding need of certificates and by only requiring to the sender the knowledge of the receiver identifier.

IBE state-of-the-art starts in 1984 with Shamir's pioneer article (Shamir, 1984). First practical cryptosystems emerge in 2001 with Weil's pairing (Boneh and Franklin, 2001) and residues (Cocks, 2001). Then, several cryptosystems appear to include more functionalities such as delegation in Hierarchical IBE (HIBE) in (Gentry and Silverberg, 2002) and Revocation (RIBE and RHIBE) in (Boldyreva and *al.*, 2008).

The emergence of the quantum threat, breaking asymmetric cryptography, hurry the NIST to engage a transition to quantum-resistant cryptography (NIST, 2017). Cryptosystems standardized first around 2024 will be key encapsulation mechanisms and the digital signatures. In continuity, other cryptosystems such as IBE are following this transition. However, the post-quantum transition comes with its issues. The sizes of keys and ciphertexts of post-quantum (R)(H)IBE are significantly higher than classical schemes.

From today, it exists quite light ideal-lattice-based IBE (Ducas and *al.*, 2014) compliant with IoT devices. However, such post-quantum IBE does not provide today the functionality of revocation and delegation. On the other hand, post-quantum RHIBE are too heavy for constrained devices but have the required functionalities. However, they are compliant with cloud resources and it is interesting to have a better quantification of their performances, parameters setting and features (sizes of instances). From the authors' knowledge, only orders of magnitude are available regarding such post-quantum RHIBE.

The purpose of this paper is to study one of the most advanced post-quantum RHIBE (Wang and *al.*, 2019) named WZH+ scheme from authors' names. The objective is to provide concrete parameter sets and to deduce the sizes and the performances of the

Carmona, M., Lesaignoux, D. and Loiseau, A.
On the Implementation of a Lattice-Based Revocable Hierarchical Ibe.
DOI: 10.5220/0012047800003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 617-623
ISBN: 978-989-758-666-8; ISSN: 2184-7711

scheme with a real implementation on a target dedicated to embedded applications. The results highlight the *heavy* aspect of the scheme that is today only applicable in a cloud context and cannot be deployed on IoT devices.

The paper is organized in two sections. Section 2 recalls some basics about IBE. A brief survey of post-quantum schemes (R)(H)IBE is presented and highlights the WZH+ as the most efficient post-quantum RHIBE. Section 3 introduces a software implementation of the WZH+ with an analysis of the parameters generation and performances.

2 (R)(H)IBE: CONCEPT AND POST-QUANTUM SCHEMES

A (H)IBE can be seen as a centralized system based on a master public key and a master secret key. PKG (Public Key Generator), also called KGC (Key Generator Center), generates a master key pair. From the master secret key, a secret key is derived level-by-level for each user in the hierarchy. Furthermore, all users can encrypt a message with the identity of the recipient and the master public key. The hierarchical structure is fully dependent on the application (wireless sensor networks, multi-site enterprise, manufacturing site, etc.).

A (H)IBE is composed of four primitives:

- **Setup** for generating the master public key **PP** and the master private key **MK**
- **Derive** (sometimes call **KeyGen** for a simple IBE) for secret key generation **SK** from the parent to the children.
- **Encrypt** for ciphering a message from a user to another by using the identity of the recipient
- **Decrypt** to decrypt a message using the secret key **SK**.

It is important to notice that a (H)IBE does not include the primitive managing the transmission of the derive secret key from the generator (the parent) to the user (the child). The (H)IBE avoids the use of certificates by dealing only with identity to encrypt messages. The constraints are mainly deferring on the PKG, which has to be strongly secure with respect to all known attacks.

As for all centralized systems, a (H)IBE is not flexible when a user/an object has to be revoked. The first approach is to regenerate the entire master and users keys, except for revoking ones. This full regeneration of keys is not applicable in practice. In 2008 (Boldyreva and *al.*, 2008), introduce a mechanism allowing revocation for an IBE. Such cryptosystems are called R(H)IBE for Revocable (H)IBE. In practice, a R(H)IBE is composed of the four primitives composing a (H)IBE and three new primitives:

- **KeyUp** to generate the update of the decryption key when a revocation occurs.
- **GenDK** to update the decryption key which uses the secret key **SK** and the update **KeyUp**.
- **Revoke** to update the revocation list

RHIBE and PKI provide the same hierarchical and revocation features. However, RHIBE gives the simplest key management, which is balanced by key sizes.

The security of a (R)HIBE relies on the following issues: the master public key must not leak information on the master private key, the ciphertext confidentiality, and, the decryption key of a child node shall not leak information on the parent secret key. (Seo and Emura, 2013) introduced a new security notion: the resilience regarding decryption key exposure (DKER). This indicates that if a decryption key is corrupted at a given time and the associated users are revoked, this does not compromise future exchanges between unrevoked users.

The ability for a (R)(H)IBE to satisfies all these conditions depends on assumptions of the underlying problems defining the scheme. There exists two types of models: the Standard model based on NP-Hard problems, and, the Random Oracle Model (ROM) where the security is based on the indistinguability of the output of a given function/oracle (a hash function in many cases) with a random uniform instance.

In (Wang and *al.*, 2019), the coexistence of two models (standard and ROM) is due to the confidence of the Standard Model is higher than the Random Oracle Model but this latter provides more efficient and compact IBE. In particular, for this work, we study the parameters setting and performances in software of the WZH+ in the ROM version.

3 STUDY OF WZH+ SCHEME

3.1 Basics on Lattices for (R)(H)IBE

A lattice Λ of $\mathbb{R}^n$ is a discrete additive subgroup of $\mathbb{R}^n$. Λ is spanned over $\mathbb{Z}$ by a set of m vectors $\boldsymbol{a}_0, \dots, \boldsymbol{a}_{m-1}$ of $\mathbb{R}^n$, where $0 < m \leq n$, and define a basis of Λ. Let $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ be the matrix whose columns are vectors $\boldsymbol{a}_0, \dots, \boldsymbol{a}_{m-1}$. By definition $\Lambda(\boldsymbol{A})$

is the lattice of $\mathbb{R}^n$ spanned by the columns of $\boldsymbol{A}$. Many post-quantum (R)(H)IBE schemes use full rank lattices $\Lambda(\boldsymbol{A})$.

Let q an integer and $\beta > 0$ a real, a trapdoor of a matrix $\boldsymbol{A}$ is a non-zero vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\boldsymbol{Ax} = \boldsymbol{0}_n \bmod q$ and $||\boldsymbol{x}|| < \beta$ where $||\,.\,||$ is the Euclidean norm. For some tuples (n, m, q, β), finding a trapdoor is a NP-complete problem. A strong trapdoor is a matrix $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that $\boldsymbol{AT} = \boldsymbol{0}_{n\times m} \bmod q$ and all vectors of $\boldsymbol{T}$ have a norm lower than β. We denote by $(\boldsymbol{A}, \boldsymbol{T}) = \mathbf{TrapGen}(n, m, q, \beta)$ a primitive generating a random uniform matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a strong trapdoor $\boldsymbol{T}$ of $\boldsymbol{A}$. This primitive is common in lattice-based cryptography for generating a couple of public key ($\boldsymbol{A}$) / private key ($\boldsymbol{T}$).

For a lattice-based HIBE, the delegation relies on generating a random and strong trapdoor $\boldsymbol{T}'$ for a children node from the strong trapdoor $\boldsymbol{T}$ of the parent node. In the ROM, $\boldsymbol{T}$ and $\boldsymbol{T}'$ are related to matrices $\boldsymbol{A}$ and $\boldsymbol{A}.\boldsymbol{R}$, respectively, where $\boldsymbol{A}$ is a random uniform matrix and $\boldsymbol{R}$ is a pseudo-random uniform matrix depending on the identity of the node. The generation of $\boldsymbol{R}$ uses the ROM. The primitive generating $\boldsymbol{T}'$ knowing $\boldsymbol{T}$, $\boldsymbol{A}$ and $\boldsymbol{R}$, is commonly called **BasisDel** (Agrawal and *al.*, 2010).

Two other important notions in lattice-based IBE are the Hermite Normal Form (HNF) (Micciancio and Goldwasser, 2002) and the Gram-Schmidt Orthogonalization (GSO).

For two given matrices $\boldsymbol{B}$ and $\boldsymbol{S}$ it exists an algorithm $\boldsymbol{A} = \mathbf{ToBasis}(\boldsymbol{B}, \boldsymbol{S})$ that computes a basis $\boldsymbol{A}$ of $\Lambda(\boldsymbol{B})$ such that $\|\widetilde{\boldsymbol{A}}\|_{\mathrm{GS}} \leq \|\tilde{\boldsymbol{S}}\|_{\mathrm{GS}}$. This algorithm allows generating basis of a lattice with a controlled Gram-Schmidt norm. This is fundamental for Gaussian sampling as introduced in the next paragraphs.

By definition, the discrete Gaussian distribution $D_{\sigma,\boldsymbol{c},\Lambda}$ on a lattice Λ, with standard deviation $\sigma > 0$ and center $\boldsymbol{c} \in \mathbb{R}^n$ is defined as $D_{\sigma,\boldsymbol{c},\Lambda}(\boldsymbol{x}) := \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,c}(\Lambda)$ for all $\boldsymbol{x} \in \Lambda$ with $\rho_{\sigma,c}(\boldsymbol{x}) := \exp(-\pi\|\boldsymbol{x} - \boldsymbol{c}\|^2/\sigma^2)$, and $\rho_{\sigma,c}(\Lambda) := \sum_{\boldsymbol{x}\in\Lambda}\rho_{\sigma,c}(\boldsymbol{x})$. We will note $\boldsymbol{x} \leftarrow D_{\sigma,\boldsymbol{c},\Lambda}$ a variable of distribution $D_{\sigma,\boldsymbol{c},\Lambda}$. Discrete Gaussian distribution is widely used in lattice-based cryptography to define the LWE.

A strong property of discrete Gaussian vector $\boldsymbol{x} \in \Lambda$ is the control of the norm: the probability that $\|\boldsymbol{x} - \boldsymbol{c}\| < \sigma\sqrt{n}$ writes $1 - \epsilon(n)$ where $\epsilon(n)$ is a negligible function with respect to n.

A Gaussian sampler is a primitive $\boldsymbol{x} = \mathbf{SampleG}(\boldsymbol{B}, \sigma, \boldsymbol{c})$ where $\boldsymbol{x} \leftarrow D_{\sigma,\boldsymbol{c},\Lambda}$ and where $\boldsymbol{B}$ is a basis of Λ. In this work, we use the GPV sampler (Gentry, 2008) that requires a condition on σ and $\|\boldsymbol{B}\|_{\mathrm{GS}}$ to converge: $\sigma \geq \|\boldsymbol{B}\|_{\mathrm{GS}}\,\eta'_\epsilon(\mathbb{Z})$ with: $\varepsilon = 2^{-\lambda}/(2n)$, n the dimension of lattice $\Lambda(\boldsymbol{B})$, $2^{-\lambda}$ the statistical distance between $D_{\sigma,\boldsymbol{c},\Lambda}$ and the distribution obtained from the sampler and $\eta'_\epsilon(\mathbb{Z})$ proportional to the smoothing parameter of $\mathbb{Z}$.

A pre-image of a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ by a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ is a vector $\boldsymbol{e} \in \mathbb{Z}_q^m$ such that $\boldsymbol{Ae} = \boldsymbol{u} \bmod q$. We denote by $\boldsymbol{e} = \mathbf{SamplePre}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{u}, \sigma)$ the primitive generating a random pre-image of $\boldsymbol{A}$ and $\boldsymbol{u}$ with distribution $D_{\sigma,\boldsymbol{0},\Lambda(\boldsymbol{T})}$ and using a trapdoor $\boldsymbol{T}$ of $\boldsymbol{A}$.

Finally, we introduce the LWE (Learning With Error) problem mainly used in IBE for the encryption primitive. A LWE instance is based on an uniform random public matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ where m, n and q are strictly positive public integers, a random uniform secret vector $\boldsymbol{s} \in \mathbb{Z}_q^m$, a random discrete Gaussian (error) vector $\boldsymbol{e} \in \mathbb{Z}_q^n$ of known standard deviation α and the public vector $\boldsymbol{t} = \boldsymbol{A}\,\boldsymbol{s} + \boldsymbol{e} \in \mathbb{Z}_q^n$. LWE is NP-hard if $\alpha q > 2\sqrt{n}$ (Regev, 2009). It is important to know that the concrete security (*i.e.,* computational complexity to solve a LWE instance) of LWE can be evaluated from dedicated programs such as LWE-Estimator (Albrecht, 2015, and dedicated website).

3.2 Parametrization of the WZH+

The WZH+ studied in this paper is a lattice-based, RHIBE in the ROM (Wang and *al.*, 2019).

WZH+ scheme is parameterized by:

- $\boldsymbol{\lambda}$: level of security, specific to the application
- $\boldsymbol{d}$: maximum depth of the hierarchal structure, specific to the application
- $\boldsymbol{n}$: number of rows of the public key, strongly related to the level of security λ.
- $\boldsymbol{q}$: arithmetic modulus
- $\boldsymbol{\alpha}$: standard deviation of discrete Gaussian variables in LWE instances
- $\boldsymbol{m}$: determines the private key size and parameter for TrapGen
- $(\boldsymbol{\sigma_i})_{\boldsymbol{i=0,\ldots d}}$: standard deviations for the discrete Gaussian variables
- $\boldsymbol{N}$: maximum of child nodes for one parent node

In a complement, we denote $\tau_\ell := \sigma_\ell\sqrt{m}\omega(\sqrt{\log(m)})$, the standard deviation used to sample Gaussian secret vectors.

The parameters of a cryptosystem are constrained in four ways: the application, functionality, level of security and performances. For the WZH+ the parameters setting needs a specific care to fulfill all

requirements. Note that (Wang and *al.*, 2019) only gives an order of magnitude to set the parameters. In this section, we provide concrete values for the parameters of WZH+ for two commonly used levels of security. The five conditions driving the parameters constraints are as follows:

1. Achieving a given level of security λ.
2. Be in the conditions where LWE is NP-hard
3. Having a low decryption failure rate
4. Ensuring the convergence of SampleG
5. Ensuring the functionality of TrapGen

About condition 1, the level of security λ is related to parameters n, α and q . λ is the computational complexity to solve LWE instances parametrized by n, α and q . This writes $\lambda = \mathbf{LWE\_Est}(n, \alpha, \alpha q)$ where **LWE_Est** is the best known algorithm for solving LWE. n, α and q has to be to choose to achieve a level of security greater or equal to λ. Furthermore, they are constraints by the inequality $\alpha q \geq 2\sqrt{n}$ ensuring that LWE can be reduced to the NP-hard problem SVP (condition 2). Condition 3 involves parameters used in a Gaussian sampler and writes $\alpha[1 + m(\tau_L + 2\sum_{i=1\ldots L}\tau_{i-1})] < 1/5$ with $\tau_i = \sigma_i\sqrt{m}\log(m)$ In addition, the convergence of Gaussian sampler (condition 4) is verified with $\sigma_\ell = m^{1.5\ell+1.5}\log(n)^{2\ell+2.5}$. Finally, parameters m, n and q are strongly linked by the condition ensuring the functionality of TrapGen from (Micciancio and Peikert, 2012) $m = 2kn$ ceil$[\log(q)]$ where k is an arbitrary strictly positive integer. This corresponds to condition 5.

From the conditions recalled above, it is easy to see that setting all parameters is highly constrained. We develop a configuration setting software providing parameters set for a given level of security λ and depth d. The used degrees of freedom are n, α and δ. This latter, introduced in (Wang and *al.*, 2019) is such that $m' = 6n^{1+\delta}$ and m the closest integer of m' satisfying condition 5 (equation 4) with the same q used for m. Precisely and firstly, q is set at $q = 2\alpha^{-1}\sqrt{n}$ ensuring condition 2. Then, a value of n and δ are chosen arbitrarily (in practice, the use of LWE estimator allows initializing the first values). m is initialized to $6n^{1+\delta}$ and α such that inequality of condition 3 is equality. This is possible because the σ_i only depend on m, n and q which are determined. If the level of security is not achieved, we increase n until achieving it. Then, we decrease δ and α to ensure that condition 3 is still valid. Then, we find integer k verifying that $2kn\log(q)$ is the closest to $m = 6n^{1+\delta}$. Then, we modify $m = 2kn\log(q)$ and check those conditions 3 and 4 are still valid. If it is not the case, variations on δ and α have to be done until satisfying all conditions.

Below, we first give two practical parameters set (*i.e.,* respecting conditions listed above) for two security levels λ. In table 2, we provide instances size. This table reveals that for security level that is commonly used in applications, the sizes of instances are high. They cannot be considered for embedded applications but they are compliant with resources used for cloud computing.

Table 1.

$\boldsymbol{n = 4096}$, $\boldsymbol{L = 2}$, $\boldsymbol{\lambda = 105}$, $\boldsymbol{k = 4}$
m = 2752512
σ_i = [2277964473455.1304,
14979705328642203834122 24,
98505299071939856367805 7032830681088]
τ_i = [80848020546021168,
53164987351470310553961889792,
34960854464866698234190555100447493849088]
α = 4.0108458310149454e − 49
log(q) = 168
q = 319134679797975626692205652596698985542207292834747

$\boldsymbol{n = 8192}$, $\boldsymbol{L = 2}$, $\boldsymbol{\lambda = 299}$, $\boldsymbol{k = 6}$
m = 8847360
σ_i = [16035367938144.068,
71315869687241918377885696,
31717097411587791016552052498579115212 8]
τ_i = [1100681478931855232,
48951827748175769652870124666 88,
21770889087845086141677735978535016818802688]
α = 2.0261448549063876e − 52
log(q) = 180
q = 87162687208864109935674861353029254316316905528151279 3

Table 2: Sizes for a given security level.

	Size in bytes	$L = 2$, $n = 4096$, $\lambda = 105$	$L = 2$, $n = 8192$, $\lambda = 299$
PP	$(2m + 1)n\log(q)$	5.4 e^{11}	3.3 e^{12}
MK	$2m^2$	2.5 e^{12}	2.0 e^{13}
SK_{ID}	$(2m + \log(N))m$	2.5 e^{12}	2.0 e^{13}
$KU_{ID,t}$	$(2d + \log(N))m$	2.0 e^{6}	5.6 e^{6}
$DK_{ID,t}$	$(2d + 1)m$	2.0 e^{6}	5.6 e^{6}
CT	$(2d + 1)m\log(q)$	3.3 e^{8}	1.0 e^{9}

3.3 Implementation

The objective of this implementation of the WZH+ is to quantify the performances in software and to highlight implementation issues.

The implementation is in Python and we are using different libraries:

- Cypari2 for linear algebra, such as the HNF (Cypari2, 2022).

- Numpy for random continuous gaussian sampling (Numpy, 1995).
- Anytree for the revocation module (Anytree, 2020).
- Hashlib for hash primitives (Hashlib, 2001)

Our implementation has 256 bits of precision by default for basic operations such that scalar product or Euclidean norm computation. However, as explained below, the precision needs to be strongly increased within GSO due to numerical instabilities. Table 3 summarizes the main primitives used for the implementation.

3.3.1 SampleGaussian

For Gaussian sampling we use the GPV Sampler (Gentry and *al.*, 2008) instead of other samplers for general lattices (Peikert, 2009) and (Micciancio and Peikert, 2012). This sampler is easy to implement and the convergence condition is part of the parameters setting (Wang and *al.*, 2019). A specific analysis is required for two primitives called by the sampler: GSO and SampleZ. This latter is the Gaussian sampling in one dimension *i.e.,* when the lattice is $\mathbb{Z}$.

To deal with SampleZ and high value of the standard deviations and centers, we used the Peikert sampler (Peikert, 2009) applied in dimension 1. This sampler remains on rounding a continuous Gaussian random variable: if $x \sim N(0,1)$ then $c + \sigma \times \text{round}(x) \sim D_{\mathbb{Z},c,\sigma}$ where $\text{round}_r(x) = \text{round}(x/r) \times r$ with x, r real. The convergence condition of this sampler writes: $\sigma^2 > r^2 = 1$. It is verified thanks to the high standard deviations σ used in the scheme. Using higher $r > 1$ for rounding is possible for an optimized implementation.

The rounding is realized with primitive floor of Cypari requiring to maintain at a high level the accuracy during the computation.

The GSO presents strong numerical instabilities. During implementation tests, it was detected through the Gaussian sampling which provides vectors with norms that do not satisfy the expected bound because of the instability of the GSO. This is a well-known problem and is due to the iterative structure of the computation. It has been shown in (Giraud and *al.*, 2005) that applying twice the GSO to a matrix optimized for the stability of the computation. However, this is not enough to stabilize the GSO computation because of the high modulus q involved in the scheme (the size of the matrix is not a main parameter rather than the size of numbers in the matrix).

A lack of precision in GSO leads to instability in the Gaussian sampler **SampleGaussian**, especially when it is used in a primitive **SamplePre** with a non-zero mean. The stability of primitive **SampleGaussian** depends on precision within the GSO of the use basis. When basis coefficients are bigger, its GSO need to be computed with even more precision to guarantee stability Gaussian sampler. In our implementation of the WZH+ with a depth of 2, we used precision of 2048 bits in the GSO computation for first-level Derive and KeyUp of KGC, 3000 bits for second-level Derive and first-level KeyUp and 4096 bits for second-level DKGen.

3.3.2 BasisDel

For **BasisDel**, we use the procedure described in (Agrawal and *al.*, 2010). Implementation of primitive **ToBasis** follows original construction in lemma 7.1 of (Micciancio and Goldwasser, 2002). This algorithm performs matrix calculations in $\mathbb{Z}$. It uses **SolveRight** primitive that find an integer matrix $\boldsymbol{X}$ satisfying the linear system $\boldsymbol{TX} = \boldsymbol{S}$ where $\boldsymbol{T}$ and $\boldsymbol{S}$ are known square matrix. Cypari didn't provide such a primitive so we implemented **SolveRight** from method written in (Hung and Rom, 1990).

Table 3: Summary of all primitives.

Primitive	Implementation
Linear Algebra	
HNF	Cypari
GSO	Self-implemented
SolveRight mod q	Cypari
SolveRight in $\mathbb{Z}$	Hung and Rom, 1990
MatrixInversion	Cypari
Lattice-Cryptography	
SampleZ	Peikert, 2009
SampleGaussian	Gentry, 2008
TrapGen	Micciancio and Peikert, 2012
BasisDel	Agrawal and *al.*, 2010
ToBasis	Micciancio and Goldwasser, 2002
RandBasis	Cash and *al.*, 2010
SamplePre	Gentry, 2008
General-Cryptography	
Seed generation	TRNG
Seed extension	SHAKE-256
Random Oracle	SHAKE-128
Tree	
CS.Setup	Anytree
CS.Assign	Anytree
CS.Path	Anytree
CS.Cover	Anytree

Primitive **RandBasis**, introduced in (Cash and *al.*, 2010), is used to randomizes basis output by **ToBasis** avoiding deterministic construction of trapdoors. **RandBasis** outputs, from a given basis, a new basis that is random with discrete Gaussian random vectors. It relies on several (but bounded with $\sigma\sqrt{(m)}$) calls to Gaussian sampling until generating a set of linearly independent vectors.

3.4 Performances Analysis

The main feature analyzed for performance analysis is the timing of each primitive. Due to the huge size of instances and the global computational complexity, the performances are computed for a non-practical (*i.e.,* very low) security level (< 20 bits) with: $d = 2$, $n = 2$, $k = 2$, $m = 288$ and $q = 79466497377483581$. Illustrations for security level > 80 bits leads to unpractical computational time and memory sizes. The implementation run on an Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz. To get execution time, we took the average of the times recorded on 50 executions for each primitive. Table 4 below provides the execution time of all primitives of the scheme. The primitives Derive, KeyUp and DKGen are highly time consuming. However, in real deployment, these primitives are called sparsely during the operating lifetime of the application. To highlight the bottlenecks leading to such time performances, a specific look to Derive, KeyUp, DKGen primitives shall be done.

Table 4: Execution of the main primitive of the scheme for the security level of (< 20 bits) and 2 hierarchical levels.

Primitives	Execution time
Setup	5s
Derive $d = 1$	5min17
Derive $d = 2$	10min19
KeyUp KGC	1min50
KeyUp $d = 1$	6min44
DKGen $d = 2$	5min11
Encrypt	15 ms
Decrypt	0.65 ms

For **DKGen** at the second level ($d = 2$), it appears that the main bottlenecks are the following (linear algebra and lattices) primitives: Inverse matrix (25%), GSO (14.3%), HNF (42%) and Trapdoor Generation (15.7%). By doing the same analysis on other primitives such as **KeyUp** and **Derive**, GSO and HNF also appears as the main and common bottlenecks.

4 CONCLUSIONS

Post-quantum RHIBE are still heavy schemes, hard to exploit except is some contexts such as cloud computing. The proposed implementation provides a complete parametrization strategy of a complex scheme, the WZH+ in the ROM model, which is the most efficient and compact post-quantum RHIBE from today. The bottleneck (HNF) and critical operations (such as GSO) require a specific attention for performance issues. It exists acceleration strategy that does not avoid the main issue regarding lattice-based IBE: the size of the instances.

REFERENCES

Agrawal, S., Boneh, D., Boyen., X. (2010). Lattice basis delegation in fixed dimension and shorter ciphertext hierarchical IBE, In *CRYPTO 2010*.

Albrecht, R.M., Player, R., Scott, S. (2015). On the concrete hardness of Learning with Errors, In *Journal of Mathematical Cryptology 2015*. Related website: https://lwe-estimator.readthedocs.io/en/latest/

Boldyreva, A., Goyal, V., Virendra., K. (2008). Identity-based encryption with efficient revocation, In *CCS'08*.

Boneh, D., Franklin M. (2001). Identity-based encryption from the Weil pairing, In *CRYPTO 2001*.

Cash, D., Hofheinz, D., Kiltz, E., Peikert, C. (2010). Bonsai trees or how to delegate a lattice basis, In *EUROCRYPT 2010*.

Cocks, C. (2001). An identity based encryption scheme based on quadratic residues, In *Cryptography and Coding 2001*.

Ducas, L., Lyubachevsky, V., Prest, T.. (2014). Efficient Identity-Based Encryption over NTRU Lattices, In *ASIACRYPT 2014*.

Gentry, C., Silverberg, A. (2002). Hierarchical ID-Based Cryptography, In *ASIACRYPT 2002*.

Gentry, C., Peikert, C., Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions, In *STOC'08*.

Giraud, L., Langou, J., Rozloznik, M. The Loss of Orthogonality in the Gram-Schmidt Orthogonalization Process, In *Computers & Mathematics with Applications*.

Hung, S., Rom, W.O. (1990). An application of the Hermite normal form in integer programming, In *Linear Algebra and Its Applications*.

Micciancio, D., Goldwasser, S. (2002). Complexity of lattice problems: A cryptographic perspectives, In *Book 2002*.

Micciancio, D., Peikert, C. (2012). Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller, In *EUROCRYPT 2012*.

NIST. (2017). Requirements and Evaluation Criteria for the PQC Standardization Process, In *https://csrc.nist.gov/Projects/post-quantum-cryptography*.

O. Regev. (2019). On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM 2009.

Cypari2. (2022). Cypari2 official documentation: https://cypari2.readthedocs.io/_/downloads/en/latest/pdf/

PARI. (2020). PARI/GP official site: https://pari.math.u-bordeaux.fr/.

Numpy. (1995). Numpy official site: https://numpy.org/

Anytree. (2020), Anytree official site: https://anytree.readthedocs.io/en/2.8.0/

Hashlib, (2001), Hashlib official site: https://docs.python.org/3/library/hashlib.html

Peikert, C. (2009). An Efficient and Parallel Gaussian Sampler for Lattices, In *CRYPTO 2010*.

Seo, J.H., Emura., K. (2013). Revocable identity-based encryption revisited: Security model and construction, In *PKC 2013*.

Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In *CRYPTO'84*.

Wang, S., Zhang, J., He, J., Wang, H., Li, C. (2019). Simplified Revocable Hierarchical Identity-Based Encryption from Lattices, In *CANS 2019*.