Research Article

Arnaud Sipasseuth*, Thomas Plantard and Willy Susilo

Enhancing Goldreich, Goldwasser and Halevi's scheme with intersecting lattices

https://doi.org/10.1515/jmc-2016-0066
Received November 21, 2016; revised May 25, 2019; accepted June 7, 2019

Abstract: We present a technique to enhance the security of the Goldreich, Goldwasser and Halevi (GGH) scheme. The security of GGH has practically been broken by lattice reduction techniques. Those attacks are successful due to the structure of the basis used in the secret key. In this work, we aim to present a new technique to alleviate this problem by modifying the public key which hides the structure of the corresponding private key. We intersect the initial lattice with a random one while keeping the initial lattice as our secret key and use the corresponding result of the intersection as the public key. We show sufficient evidence that this technique will make GGH implementations secure against the aforementioned attacks.

Keywords: Lattice-based cryptosystem, lattice intersections, lattice reduction, GGH

MSC 2010: 94B75,94A60,11H99

Communicated by: Tran van Trung

1 Introduction

The popularity of post-quantum cryptography has increased significantly after the formal announcement by the National Institute of Standards and Technology (NIST) to move away from classical cryptography [76]. This is due to the potential threat that will be brought by the upcoming large scale quantum computers, which theoretically break the underlying traditional hard problem by using Shor's algorithm [70]. There are currently three main families in post-quantum cryptology, namely, code-based cryptography, multivariate cryptography, and lattice-based cryptography. This work primarily concerns with lattice-based cryptography. First introduced by Minkowski in a pioneering work [53] to solve various number problems, lattices have the advantage to often base their security on worst-case assumptions [1] rather than the average case, and to be highly parallelizable and algorithmically simple enough to compete with traditional schemes in terms of computing speed. Inspired by this, Goldreich, Goldwasser and Halevi (GGH) [27] proposed an efficient way to use lattices to build a public-key encryption scheme. Their practical scheme has been broken using lattice reduction techniques [55]; however, the central idea is still viable and gave birth to a wide array of applications and improvements using tensor products [19], Hermite normal forms [50], polynomial representations [59], rotations [72], etc. The idea of intersecting lattices for cryptography first appeared in a broadcast attack on lattices [64], and since then, it has been used in various cryptanalytic efforts. Nevertheless, there is no attempt that makes use of this technique in a positive way, i.e., for building a secure cryptosystem. This paper aims to fill this gap by exploring this possibility.

***Corresponding author: Arnaud Sipasseuth,** Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, Australia, e-mail: as447@uowmail.edu.au
Thomas Plantard, Willy Susilo, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, Australia, e-mail: thomaspl@uow.edu.au, wsusilo@uow.edu.au

Our work does not follow the previous works on the LWE or SIS type problems, which are currently very popular. Those lattices have a particular form and are called q-ary lattices, which only represent a small fraction of all lattices. They do benefit from Ajtai's worst-case to average-case security reduction [1], but so do most randomly taken lattices as shown recently by Gama et al. [23]. We will moreover focus on lattices with a Hermite normal form that only admits a single dense column which belongs to the most dense set of all lattices [57]. Therefore, our work can be seen as a continuation of the work on non-q-ary following the line of the initial proposal of GGH [27] and Micciancio's improvement of GGH with Hermite normal forms [50], where the latter has yet to be broken asymptotically. Furthermore, a lot of schemes based on LWE require Gaussian sampling for encryption and key generation [42, 46], which is very costly and impractical [61, 75] either in memory or in speed compared to the efficiency of more classical lattice-based schemes or other specialized versions as NTRU [31], whose security is also based on q-ary lattice problems. This problem is big enough to motivate many research directions to either tackle the issue [12, 16, 17] or avoid it using LWR (learning with rounding) [9] or LWE with uniform distribution [45]. In this paper, we attempt to use a more radical option by moving away from LWE-based cryptosystems and q-ary lattices and hope for a gain in encryption speed and key-size efficiency. On the other hand, we do not have provable security on our particular structure, but neither does NTRU; although NTRU has been researched intensively for quite some time now. Another motivation to follow an alternative path of study from LWE-based cryptosystems (and q-ary lattices) is the recent progress of attacks on such cryptosystems [3, 5, 13, 38, 39, 41] and the recent talk given by Lyubashevsky at PKC'16 in which he also qualifies LWE-based problems as particular instances of "basic" lattice problems and advised "to understand the underlying knapsack problems" to build practical schemes [47].

Our contribution and paper organization

The initial practical schemes of GGH have been broken mostly because of the very specific structure of the keys used. The main idea developed in this paper is to intersect the public key with a random lattice of the same rank to hide the previously exploited specific structure. However, we also show that improving security using intersections is not as simple as just intersecting any key to any random lattice, and therefore we present specific choice of keys based on intersection properties.

The rest of the paper is organized as follows. We first review the basics of lattice theory and discuss the initial GGH scheme in Section 2. Then, in Section 3, we present our idea to enhance its security and show that this extra layer of security does not affect our capacity to encrypt or decrypt messages. We also explain how we compute keys in practice as a lot of properties arise from intersecting lattices, which can further be exploited to either enforce or weaken the security of the resulting system, and more importantly, we demonstrate how it hides the initial "weak" structure. We give a short comparison of efficiency and key sizes compared to other schemes. In Section 4, we discuss potential security concerns given by the stated properties. Finally, in Section 5, we discuss further applications of intersecting lattices.

2 Background

In this section, we briefly recall the basics of lattice theory.

2.1 Lattice theory

Definition 1. We call lattice a discrete subgroup of $\mathbb{R}^n$, where n is a positive integer. We say a lattice is an integer lattice when it is a subgroup of $\mathbb{Z}^n$. A basis of the lattice is a basis as a $\mathbb{Z}$-module. If M is a matrix, we define $\mathcal{L}(M)$ as the lattice generated by the rows of M.

In this work, we only consider full-rank integer lattices, i.e., such that their basis can be represented by an $n \times n$ non-singular integer matrix.

Theorem 1 (Determinant). *For any lattice $\mathcal{L}$, there exists a real value we call* determinant, *denoted* $\det(\mathcal{L})$, *such that* $\det(\mathcal{L}) = \sqrt{\det(BB^T)}$ *for any basis B.*

The literature sometimes call $\det(\mathcal{L})$ the *volume* of $\mathcal{L}$ (see [53]).

Definition 2 (Hermite normal form (HNF)). Let $\mathcal{L}$ be an integer lattice of dimension d and $H \in \mathbb{Z}^{d,n}$ a basis of $\mathcal{L}$. Then H is said to be of *Hermite normal form* if and only if

$$H_{i,j} \begin{cases} = 0 & \text{if } i > j, \\ \geq 0 & \text{if } i \leq j, \\ < H_{j,j} & \text{if } i < j \end{cases} \qquad \text{for all } 1 \leq i, j \leq d.$$

The HNF can be computed from any basis in polynomial time [36], is unique [15], and has a very compressed form, and thus is an ideal form for public keys [50]. We denote $\mathrm{HNF}(M)$ the HNF of a matrix M. One of the easiest forms to work with when the HNF is computed is when we obtain a "perfect" HNF.

Definition 3. Let M be the basis of an integer lattice $\mathcal{L}$ of dimension n in HNF form. As the HNF form is unique per lattice, we can write $\mathrm{HNF}(M) = \mathrm{HNF}(\mathcal{L})$. We say $\mathrm{HNF}(\mathcal{L})$ has pseudo-perfect form when only one column differs from Id_n and perfect form when only the first column differs.

Example 1. A has perfect form, B pseudo-perfect, and C neither of them.

$$A = \begin{bmatrix} 34 & 0 & 0 & 0 \\ 27 & 1 & 0 & 0 \\ 32 & 0 & 1 & 0 \\ 13 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 34 & 0 & 0 \\ 0 & 25 & 1 & 0 \\ 0 & 18 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \end{bmatrix}.$$

For consistency with the rest of the paper, we assume Id_n is a perfect form HNF.

Definition 4. Let $\mathcal{L}$ be a full-rank integer lattice of dimension n. We say $\mathcal{L}$ is co-cyclic when $\mathbb{Z}/\mathcal{L}$ is cyclic.

According to Nguyen and Shparlinski [57], the set of co-cyclic lattices represents 85 % of full-rank integer lattices. Note that if $\mathrm{HNF}(\mathcal{L})$ has perfect or pseudo-perfect form, then $\mathcal{L}$ is co-cyclic.

Definition 5. We say a lattice $\mathcal{L}_1$ is an overlattice of $\mathcal{L}_2$, and $\mathcal{L}_2$ is a sublattice of $\mathcal{L}_1$, when $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

Thus, if $\mathcal{L}_3 = \mathcal{L}_1 \cap \mathcal{L}_2$, then $\mathcal{L}_3$ is a sublattice to both $\mathcal{L}_1$ and $\mathcal{L}_2$.

Definition 6. We say a lattice is a diagonally dominant (DD) type lattice if it admits a basis of the form $D + R$ where $D = d \times \mathrm{Id}$, $d \in \mathbb{Z}$, and R is a "noise" matrix whose entries lie in $[-\mu, \mu]$, where $d \gg \mu$, and μ is called the bound of the noise.

We note that the definition is a bit different to the one which can be found in fundamental mathematics books [11]. The other definition requires the sum of norms of non-diagonal entries in a row to be lower or equal to the norm of its diagonal entry in that same row (or line, depending how you consider the vectors). Here the definition of "low" is arbitrary, but the overall idea is the same. In our following experimentations, we will restrict ourselves to $d = \lceil \sqrt{n} \rceil$, where n is the dimension, and $\mu = 1$.

Definition 7 (Minima). We denote by $\lambda_i(\mathcal{L})$ the i-th minimum of a lattice $\mathcal{L}$. It is the radius of the smallest zero-centered ball containing at least i linearly independent elements of $\mathcal{L}$.

Definition 8 (Lattice gap). We denote by $\delta_i(\mathcal{L})$ the ratio $\frac{\lambda_{i+1}(\mathcal{L})}{\lambda_i(\mathcal{L})}$ and call that a lattice gap. When mentioned without index and called "the" gap, the index is implied to be $i = 1$.

We also define the "root lattice gap", i.e., elevated to the power $\frac{1}{n}$, where n is the dimension of the lattice.

2.2 Lattice problems

The most famous problems on lattices are the shortest vector problem (SVP) and the closest vector problem (CVP). We tend to approximately solve CVP by solving heuristically SVP in an expanded lattice [27].

Definition 9 (CVP: closest vector problem). Given a basis B of a lattice $\mathcal{L}$ of dimension n and $t \in \mathbb{R}^n$, find $v \in \mathcal{L}$ such that $\|t - v\| \leq \|t - w\|$ for all $w \in \mathcal{L}$.

Definition 10 (SVP: shortest vector problem). Given a basis B of a lattice $\mathcal{L}$ of dimension n, find $v \in \mathcal{L}$ such that $v \neq w$, $\|v\| \leq \|w - v\|$, i.e., $\|v\| = \lambda_1(B)$ for all $w \in \mathcal{L}$.

In cryptography, we rely on the "easier" versions of those problems.

Definition 11 (uSVP$_\delta$: δ-unique shortest vector problem). Given a basis of a lattice $\mathcal{L}$ with its lattice gap $\delta > 1$, solve SVP.

Since $\lambda_1(\mathcal{L})$ is also hard to determine (it is indeed another lattice problem we do not state here), measuring the efficiency of an algorithm is another challenge by itself. Therefore, to measure algorithm efficiency, we must be able to define a problem with easily computable parameters, which is where the Hermite factor is originated from.

Definition 12 (HSVP$_\gamma$: γ-Hermite shortest vector problem). Given a basis B of a lattice $\mathcal{L}$ of dimension n and a factor γ we call Hermite Factor, find $y \in \mathcal{L}$ such that $\|y\| \leq \gamma \det(\mathcal{L})^{1/n}$.

Some cryptosystems are based on worst-case hardness on uSVP with polynomial gap, such as [2, 66]. The practical hardness of uSVP depends on its gap compared to a fraction of the Hermite factor, where the constant in front of the factor depends on the lattice and the algorithm used [24]. There exists an attack that was specifically built to exploit high gaps [44].

Definition 13 (BDD$_\gamma$: γ-bounded distance decoding). Given a basis B of a lattice $\mathcal{L}$, a point x and an approximation factor γ such that there exists $v \in \mathcal{L}$ such that $\|x - v\| < \gamma\lambda_1(B)$, find $v \in \mathcal{L}$ such that $\|x - v\| \leq \|x - w\|$ for all $w \in \mathcal{L}$.

It has been proved that BDD$_{1/(2\gamma)}$ reduces to uSVP$_\gamma$ in polynomial time, and the same goes for uSVP$_\gamma$ to BDD$_{1/\gamma}$ when γ is polynomially bounded by n [48]; in cryptography, the gap is polynomial, so the target point x must be polynomially bounded, and therefore solving one or the other is relatively the same in our case.

2.3 The GGH cryptosystem

GGH is a public key cryptosystem named after its creators (Goldreich, Goldwasser and Halevi). Like every public key encryption scheme, GGH is composed of three algorithms, namely, a key generator, an encryption algorithm, and a decryption algorithm.

- KeyGen(n). Take a "good" basis Sk of a lattice $\mathcal{L}$ of dimension n; compute a "bad" basis Pk from $\mathcal{L}$(Sk); provide Pk as the public key, and keep Sk as the secret key.
- Encrypt(Pk, m). Use Pk to encrypt a message m, which is a "small" vector, less than half the size of the smallest vector of Sk, by adding a random vector v of $\mathcal{L}$; output $c = m + v$.
- Decrypt(Sk, c). Use Sk to decrypt a message, solving the BDD instance of c on $\mathcal{L}$(Pk), thus separating m from v and thus recovering m.

To be able to decrypt m, it has to be relatively short, say, $\|m\| \lessapprox \gamma$, thus solving special instances of BDD$_\gamma$. For this cryptosystem to be relevant, it must rely on three important points.

- It is easy to encrypt a message with a public key (i.e., generate c given (Pk, m)).
- It is easy to decrypt a message with a secret key (i.e., solve BDD$_\gamma$ with Sk).
- It is hard to recover the secret key/original message from the public key.

The first point is pretty straightforward, and the second is guaranteed only by a proper key generation, which is where the concepts of "good basis" and "bad basis" are originated from. Sk is typically a diagonal dominant matrix and Pk its Hermite normal form. Because v is generated from Pk and $\mathcal{L}(\mathrm{Pk}) = \mathcal{L}(\mathrm{Sk})$, we can then recover v from c and thus m by solving BDD_γ with the "good" basis Sk (typically composed of short and nearly orthogonal vectors). The problem stems from the third point.

2.4 Attacks on classical GGH

2.4.1 Message attacks and basis reduction on semi-orthogonal basis

The original GGH challenges [28] used an error vector whose entries are drawn in $\{-\sigma, \sigma\}$, which allowed Nguyen to easily retrieve messages in high dimension [55]. Using entries from $[-\sigma, \sigma]$ fixes the problem, but the structure present in secret keys were still exploitable. Basis reduction techniques allowed Gama and Nguyen to recover very good bases from structural weaknesses within $\mathcal{L}(\mathrm{Pk})$ (see [24]), comparing them to semi-orthogonal bases.

2.4.2 Special key recovery attack on diagonal dominant keys

For diagonally dominant keys of the form $\mathrm{Sk} = D + R$, where $D = d \times \mathrm{Id}$, it is easier to attack the basis structure rather than the message itself.

Solving BDD_γ for vectors

$$(d, 0, \ldots, 0), (0, d, 0, \ldots, 0), \ldots, (0, \ldots, 0, d) \quad \text{in} \quad \mathcal{L}(d \times \mathrm{Id} + R) = \mathcal{L}(\mathrm{Sk}) = \mathcal{L}(\mathrm{Pk})$$

to recover R yields very short vectors $R[1], \ldots, R[n]$ and thus recovers the secret key: the difference between the i-th vector of the matrix $d \times \mathrm{Id}$ and $\mathcal{L}(d \times \mathrm{Id} + R)$ is exactly the i-th vector of R, which holds small values within $[-\mu, \mu]$ and is much shorter than a message. We will denote ϕ the BDD_γ solving algorithm. It is not an oracle, but rather an algorithm that anybody could choose to solve the problem. Our suggestion for ϕ would be to use Kannan's embedding technique [35], which transforms the BDD instance into a uSVP_γ instance, and then apply lattice reduction techniques on the extended basis, using Klein's algorithm [40] or more recent works such as Liu and Nguyen enumeration-based solver [43].

Note that on the above algorithm, each iteration on a position of the diagonal makes the next iteration easier: since $\mathcal{L}(\mathrm{Pk}) = \mathcal{L}(\mathrm{Sk})$, each short vector found of Sk decreases the complexity of ϕ. In fact, like most lattice-based cryptosystems, finding one shortest vector is enough to break GGH.

Input: the public key Pk of full rank n, the diagonal coefficient d, a BDD_γ solver ϕ
Output: the secret key Sk
$\mathrm{Sk} \leftarrow d \times \mathrm{Id}_n$;
`// Loop on every position of the diagonal`
foreach $\{i \in [1, \ldots, n]\}$ **do**
  `// find the difference` r `between` $(0, \ldots, d_i, \ldots, 0)$ `and` $\mathcal{L}(\mathrm{Pk})$
  $r \leftarrow \phi(\mathcal{L}(\mathrm{Pk}), \mathrm{Sk}[i])$;
  $\mathrm{Sk}[i] \leftarrow \mathrm{Sk}[i] + r$;
end
return Sk

Algorithm 1: Diagonal dominant key recovery attack.

3 Enhanced GGH cryptosystem

3.1 Modification of GGH using an intersecting lattice

If we denote Pk the public key and Sk our private key, our modification is to go from $\mathcal{L}(\mathrm{Pk}) = \mathcal{L}(\mathrm{Sk})$ to $\mathcal{L}(\mathrm{Pk}) = \mathcal{L}(\mathrm{Sk}) \cap \mathcal{L}(R)$, where R is a random non-singular integer matrix of dimension n. The modified GGH scheme, informally, is the following:

- $\mathrm{KeyGen}_2(n)$. Take a "good" basis $\mathrm{Sk} = (D \times \mathrm{Id}_n) + ([-\mu, \mu]^{n\times n})$ of dimension n; compute the HNF basis Pk of $\mathcal{L}(\mathrm{Sk}) \cap \mathcal{L}(R)$, where R is a random integral matrix of dimension n with a perfect HNF with a determinant co-prime to Sk; provide Pk as the public key, and keep Sk as the secret key.
- $\mathrm{Encrypt}_2(\mathrm{Pk}, m)$. Use Pk to encrypt a message m encoded in a small vector by adding a random vector v of $\mathcal{L}(\mathrm{Pk})$; output $c = m + v$.
- $\mathrm{Decrypt}_2(\mathrm{Sk}, c)$. Use Sk to decrypt a message the same way as in a classical GGH, separating m from v by solving the corresponding BDD instance and thus recovering m.

The secret key in our experiments used a diagonal coefficient $D = \sqrt{n}$ and a low noise of random values in $\mu = 1$, and our security analysis will be based on those parameters. However, we do not see a problem in taking noise within $\mu = 4$ as in the original GGH proposal [27] or as it was the case when Micciancio applied the use of a HNF [50]. Following the work in [27, 50], we can also choose our messages such that $\|m\|_2 \leq \frac{1}{2} \min \|s_i^*\|_2$, where s_i^* is the i-th vector of the orthogonalized basis obtained from the secret key using the *Gram–Schmidt orthogonalization* process, or simply encode it as a vector in $[1 - \mu, \mu - 1]^n$. However, the message space we actually use in this paper will be different as we resort to a padding technique to attempt to reach IND-CCA security. The decryption works as $v \in \mathcal{L}(\mathrm{Pk}) \subset \mathcal{L}(\mathrm{Sk})$, thus separating m from v as before. We will discuss security concerns related to this scheme in the next section (and the appendix), especially why $\det(R)$ has to be co-prime to Sk and has to have a perfect HNF. In particular, we will show that structural attacks are no longer effective when R is sufficiently large. Then $\mathcal{L}(\mathrm{Pk})$ no longer admits a diagonally dominant basis $D + R$; therefore, the BDD_γ key recovery attack on $(d, 0, \ldots, 0), (0, d, 0, \ldots, 0), \ldots (0, \ldots, 0, d)$ is no longer applicable. Nevertheless, the ratio between the size of the messages we can decrypt and the size of the public key will decrease. We will explicitly express the factor later. The question is now whether we solve the problem with the previous third point "it is hard to recover the secret key from the public key" in our modified scheme.

3.2 Modified attack on the intersected GGH key

As stated earlier, the structural key recovery attack using BDD_γ on $(d, 0, \ldots, 0), (0, d, 0, \ldots, 0), \ldots, (0, \ldots, 0, d)$ in $\mathcal{L}(\mathrm{Pk})$ does not work since $\mathcal{L}(d \times \mathrm{Id} + R) \neq \mathcal{L}(\mathrm{Pk})$, but $\mathcal{L}(d \times \mathrm{Id} + R) = \mathcal{L}(\mathrm{Sk}) \subset \mathcal{L}(\mathrm{Pk})$. Adapting this attack to $\mathcal{L}(\mathrm{Pk})$ requires finding $\mathcal{L}(\mathrm{Sk})$ in $\mathcal{L}(\mathrm{Pk})$, and then using the structural key recovery attack. We assume the existence of a function Δ which finds the "optimal" overlattice $\mathcal{L}(\mathrm{Sk})$ given $\mathcal{L}(\mathrm{Pk})$ (the overlattice that admits a diagonal basis; experimental data suggests it is indeed the "weakest" overlattice; see the next section).

To the best of our knowledge, the difficulty of recovering $\mathcal{L}(\mathrm{Sk})$ in $\mathcal{L}(\mathrm{Pk})$ is mostly dependent on the values of $\det(\mathrm{Pk})$ and $\det(\mathrm{Sk})$. The efficiency of the modified attack is dependent on the efficiency of the overlattice recovery function Δ.

In this case, we will consider Pk and R to have a perfect HNF and Sk to be able to be reduced to a perfect HNF as we believe it offers the best security assumptions. As a bonus, it also allows a fair comparison with random matrices as randomly selected matrices from an increasingly big random determinant tend to have a perfect HNF [29]. Experimental results also suggest that most lattices are "equivalent" to a lattice admitting a perfect HNF, whose easily computable "equivalency" relation might cover most of co-cyclic lattices (see Appendix A.2) and allow us to use Gama's work on their structural worst-case to average-case reduction [23]. We state the following theorem, which partly solve the problem of requiring perfect HNFs for intersections.

Input: the public key Pk of full rank n, the diagonal coefficient d, a BDD_γ solver ϕ, an "optimal" overlattice Δ
Output: the secret key Sk
// find $\mathcal{L} = \mathcal{L}(\mathrm{Sk})$ among all integer overlattices of $\mathcal{L}(\mathrm{Pk})$
$\mathcal{L} \leftarrow \Delta(\mathcal{L}(\mathrm{Pk}))$;
$\mathrm{Sk} \leftarrow d \times \mathrm{Id}_n$;
// loop on every position of the diagonal
foreach $\{i \in [1, \ldots, n]\}$ **do**
  // find the difference r between $(0, \ldots, d_i, \ldots, 0)$ and $\mathcal{L}$
  $r \leftarrow \phi(\mathcal{L}, \mathrm{Sk}[i])$;
  $\mathrm{Sk}[i] \leftarrow \mathrm{Sk}[i] + r$;
end
return Sk

Algorithm 2: Modified diagonal dominant key recovery attack.

Theorem 2. *Let A and B be perfect HNF bases of lattices $\mathcal{L}(A)$ and $\mathcal{L}(B)$ of full rank n, where $\det(A)$, $\det(B)$ are co-prime. Let C be the HNF basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$. Then $\det(C) = \det(A)\det(B)$; C has perfect HNF, and*

$$C_{i,1} = \begin{cases} A_{i,1} \bmod \det(A), \\ B_{i,1} \bmod \det(B) \end{cases} \qquad \text{for all } i \in [2, n].$$

Example 2. A, B have perfect HNF, and $\mathcal{L} = \mathcal{L}(A) \cap \mathcal{L}(B)$.

$$A = \begin{bmatrix} 17 & 0 & 0 & 0 \\ 12 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}; \quad \text{therefore,} \quad \mathrm{HNF}(\mathcal{L}) = \begin{bmatrix} 51 & 0 & 0 & 0 \\ 46 & 2 & 0 & 0 \\ 23 & 1 & 1 & 0 \\ 35 & 0 & 0 & 1 \end{bmatrix}.$$

The proof of this theorem can be found in Appendix A.1. If we intersect two lattices whose HNF bases are not perfect or whose determinant are not co-prime, the result will not be perfect, and common factors might appear (see Appendix A.1).

In our case, considering $C = \mathrm{Pk}$, $A = \mathrm{Sk}$ and $B = R$, we have to avoid $\mathcal{L}(\mathrm{Sk})$ to be easily recovered. Theorem 2 is also interesting for an attacker as it reveals one of the main issues of our approach, which we discuss in the next subsection. Let $\omega(M)$ be the number of prime factors counted without multiplicity in the decomposition of $\det(M)$. Then

$$\det(C) = \prod_{i=1}^{\omega(A)} p_i, \quad \det(B) = \prod_{i=0}^{\omega(B)} q_i, \quad \det(C) = \left(\prod_{i=1}^{\omega(A)} p_i\right)\left(\prod_{i=1}^{\omega(B)} q_i\right), \tag{3.1}$$

which means $\omega(C) = \omega(A) + \omega(B)$ and very easily leads to the following property.

Property 1. Let C be a perfect HNF square matrix of dimension n. The couples $(\mathcal{L}(A), \mathcal{L}(B))$, where A and B are perfect HNFs of the same dimension with $\det(A)$ and $\det(B)$ co-prime such that $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$ and $\det(A)\det(B) = \det(C)$, correspond exactly to the couples $(a > 0,\ b > 0)$, where a and b are co-primes such that $ab = \det(C)$; there are exactly $2^{\omega(C)}$ possibilities.

Each possible solution $(\mathcal{L}(A), \mathcal{L}(B))$ corresponds exactly to $(\det(A), \det(B))$.

Therefore, recovering $\mathcal{L}(A)$ from $\mathcal{L}(C)$, i.e., the complexity of the overlattice distinguisher Δ, is assumed to be at least polynomially equivalent to distinguishing p_i from q_i in equation (3.1). As we work in post-quantum cryptography, we assume that the factorization problem can be solved in polynomial time [70]. As we explain later, we will purposely choose keys with very smooth determinants, which make factorization easy even in the non-quantum case.

As $\omega(C)$ is lower-bounded by $\omega(A)$ and has no upper bound limit (as we have complete control over $\det(B)$), one might first think that the number of combinations to search is $2^{\omega(C)}$. However, if we assume an

oracle that knows what type of lattice to search for, then it is unsafe to assume the attacker has no knowledge of $\omega(A)$. We then assume an attacker has possession of exactly $\omega(A)$ and $\omega(B)$; the number of combinations[1] he has to try is then $\binom{\omega(C)}{\omega(A)} = \binom{\omega(C)}{\omega(B)}$. In practice, the attacker will probably only have an approximation, which increases the number of combinations by quite a lot and is a much bigger number depending on how precise the approximation is, but for simplicity, we assume he has the exact value.

Hence, if $\omega(C)$ is small or if $\omega(A)$ or $\omega(B)$ is too small relative to the number of possibilities, it might be too easy to recover $\mathcal{L}(A)$, which would nullify the point of our modification. In practice, if we let the scheme untouched as it is, then $\omega(A)$ will most often be very low, as illustrated by the following theorem.

Theorem 3 (Erdős–Kac theorem [18]). *Let $\rho(n)$ be the number of prime factors of the integer n. Then the probability distribution of $\frac{\rho(n)-\log\log n}{\sqrt{\log\log n}}$ is the standard normal distribution.*

Experimental data on low dimensions suggest that $\omega(\mathrm{Pk})$ is indeed too low to ensure a reasonable number of combinations. To deal with that problem, we first optimistically assume that, without the knowledge of det(Sk), an attacker will have no choice rather than trying every possible combination stated by Property 1. Our proposed solution to this apparent weakness is to ensure the number of combinations is sufficiently large to make sure it is not feasible to recover $\mathcal{L}(\mathrm{Sk})$. Therefore, our target public key should have the form

$$\mathrm{Pk} = \left[\begin{array}{c|ccc} \det(\mathrm{Pk}) & 0 & \dots & 0 \\ \hline \mathrm{Pk}[2,1] & & & \\ \vdots & & \mathrm{Id}_{n-1} & \\ \mathrm{Pk}[n,1] & & & \end{array}\right], \qquad \det(\mathrm{Pk}) = \prod_{i=1}^{\omega(\mathrm{Pk})} p_i \quad \text{for all } i \neq j,\ \gcd(p_i, p_j) = 1,$$

where $\omega(\mathrm{Sk}) + \omega(R) = \omega(\mathrm{Pk})$ is large enough to allow a large number of combinations. To achieve this, we must generate Sk and R such that HNF(Sk) and R have the same form as Pk (Theorem 2) while controlling $\omega(\mathrm{Sk})$ and $\omega(R)$.

3.3 Countermeasure by controlling $\omega(\mathrm{Sk})$, $\omega(R)$ on key generation

From the last subsection, two properties must arise from our keys if we want our modification to be effective: one is the perfectness of our HNF basis; the other the smoothness of our keys' determinants.

First of all, we begin by showing the generation algorithm of the random matrix R. We first choose the determinant $\det(R)$ we want to obtain, and create

$$R = \left[\begin{array}{c|ccc} \det(R) & 0 & \dots & 0 \\ \hline R[2,1] & & & \\ \vdots & & \mathrm{Id}_{n-1} & \\ R[n,1] & & & \end{array}\right]$$

such that $R[i,1]$ are random integer values in $[0, \det(R) - 1]$ for all $i \in [2, n]$.

With the knowledge of how to generate R, one simple way to maximize $\omega(\mathrm{Pk})$, and thus the number of possibilities, is to increase $\omega(R)$ until we reach the number of required combinations without caring about $\omega(\mathrm{Sk})$. However, this is not wise; we mentioned before that the ratio of message size over size of the public key is lower in our modified GGH than in the original GGH. As, in both schemes, we use the same private key, the size of the message m we decrypt remains unchanged. Let $\mathrm{Pk}_{\mathrm{org}}$ be the public key of the original GGH scheme (i.e., HNF(Sk)). Note that $\mathrm{Size}(\mathrm{Pk}) \approx \mathrm{Size}(\mathrm{HNF}(\mathrm{Sk})) + \mathrm{Size}(R) = \mathrm{Size}(\mathrm{Pk}_{\mathrm{org}}) + \mathrm{Size}(R) \approx \mathrm{Size}(\mathrm{Pk}_{\mathrm{org}}) + n\log(\det(R))$. Let c be the factor determining the ratio decrease. Then we have

$$c \approx \frac{\mathrm{Size}(\mathrm{Pk}_{\mathrm{org}})}{\mathrm{Size}(\mathrm{Pk}_{\mathrm{org}}) + n\log(\det(R))} = \frac{\mathrm{Size}(\mathrm{HNF}(\mathrm{Sk}))}{\mathrm{Size}(\mathrm{HNF}(\mathrm{Sk})) + n\log(\det(R))}.$$

1 We note that the notation C_r^n is often used instead of $\binom{n}{r}$. They both mean "n choose r" and equal $\frac{n!}{r!(n-r)!}$

Therefore, compared to the original GGH, the size increase of the public key is solely determined by $\det(R)$. Thus, for the benefit of key size, we choose to have $\omega(R) < \omega(\mathrm{Sk})$. We will discuss later what this implies at the end of this subsection.

We know have to generate Sk. It is a bit harder to obtain a perfect HNF of a diagonal dominant matrix with a chosen determinant $\det(\mathrm{Sk})$. Our targeted end result is

$$\mathrm{HNF}(\mathrm{Sk}) = \left[\begin{array}{c|ccc} \det(\mathrm{Sk}) & 0 & \dots & 0 \\ \hline \mathrm{HNF}(\mathrm{Sk})[2,1] & & & \\ \vdots & & \mathrm{Id}_{n-1} & \\ \mathrm{HNF}(\mathrm{Sk})[n,1] & & & \end{array}\right]$$

such that $\mathrm{HNF}(\mathrm{Sk})[i,1]$ are integer values in $[0, \det(\mathrm{Sk}) - 1]$ for all $i \in [2, n]$ and $\det(\mathrm{Sk}) = \prod_{i=1}^{\omega(\mathrm{Sk})} p_i$ and $\gcd(p_i, p_j) = 1$ for all $i \neq j$.

In the following, we will illustrate and then explain the way to achieve this. Let D_d be a diagonal dominant matrix of dimension $n-1$ with diagonal coefficient d whose HNF is perfect, and let c be a column of $n-1$ entries within $[-\mu, \mu]$, where μ is also the bound of the noise of D_d. We concatenate c on the left of D_d and compute the HNF of the result to obtain

$$D'_d = \mathrm{HNF}\left(\left[\begin{array}{c|c} c & D_d \end{array}\right]\right) = \left[\begin{array}{cc|ccc} a & b & 0 & \dots & 0 \\ \hline \vdots & \vdots & & \mathrm{Id}_{n-2} & \end{array}\right].$$

If $a = D'_d[1,1]$ and $b = D'_d[1,2]$ are not co-prime, we retry with a different column c until those two values are co-primes. Once they are co-prime, we compute u, v such that $|ua - bv| = \det(\mathrm{Sk})$, ensuring $\det(\mathrm{Sk})$ is co-prime with either a or b (for simplicity, we assume b is co-prime with u; the algorithm in Appendix A.3 provides a more general form), such that

$$2d \times \det(D_d) > \det(\mathrm{Sk}) > d \times \det(D_d), \quad \det(\mathrm{Sk}) \nmid \det(D_d). \tag{3.2}$$

We create a line l of n entries with $l[1] = v$ and $l[2] = u$, reduce l with Babai's nearest plane algorithm and concatenate the result l' to c and D_d as shown below to obtain Sk as we wanted, which is still a diagonal dominant matrix relatively close of parameters d and μ and possess a perfect Hermite normal form.

$$l = [v, u, 0, \dots, 0], \quad l' = \mathrm{NearestPlane}(l, [c|Dd]), \quad \mathrm{Sk} = \left[\begin{array}{c|ccc} l'[1] & l'[2] & \dots & l'[n] \\ \hline c & & D_d & \end{array}\right].$$

The Hermite normal form of Sk is perfect since

$$\mathrm{HNF}(\mathrm{Sk}) = \mathrm{HNF}\left(\left[\begin{array}{c|ccc} l'[1] & l'[2] & \dots & l'[n] \\ \hline c & & D_d & \end{array}\right]\right) = \mathrm{HNF}\left(\left[\begin{array}{cc|ccc} v & u & 0 & \dots & 0 \\ \hline a & b & 0 & \dots & 0 \\ \hline \vdots & \vdots & & \mathrm{Id}_{n-2} & \end{array}\right]\right),$$

and because we ensured a and b co-prime, $|ua - bv| = \det(\mathrm{Sk})$ and u and b co-primes,

$$\mathrm{HNF}(\mathrm{Sk}) = \left[\begin{array}{c|ccc} \det(\mathrm{Sk}) & 0 & \dots & 0 \\ \hline \vdots & & \mathrm{Id}_{n-1} & \end{array}\right].$$

Now that we know how to create Sk and R with chosen determinants, we must decide how to choose det(Sk) and det(R) to obtain secure values of ω(Sk) and ω(Pk). This is how we suggest to do it and use in most experimentations. We first generate D_d as we see fit (the same matrix used when generating Sk), and we fix a prime we call a "scaling factor" that must be prime to det(D_d), then enumerate all primes from a certain point (at least strictly greater than the "scaling factor") and choose to pick them randomly until their product is bigger than D_d (and prime to det(D_d)). We denote that set of primes S_p. We then take as much factors of S_p as we see fit to construct det(R), keep the rest and multiply the remaining product by a power of the scaling factor to respect the bound given by equation (3.2). Suppose that the scaling factor is given away for free; the number of combinations an attacker has to search is indeed $\binom{|S_p|}{\omega(R)}$. Note that, given S_p, the maximum number of combinations is achievable if $\omega(R) = \frac{|S_p|}{2}$.

Overall, the generation of keys is done in polynomial time; the computations of the HNF and Babai's nearest plane algorithm are the most time consuming operations, and they both run in polynomial time. To study the complexity and how this can be generated in practice, we refer the readers to Appendix A.3.

We present in this table the minimum ratio $\omega(R)/\omega$(Pk) required to go over some 2^λ combinations in total using the algorithm we just described, and to ensure $\omega R < \omega$ Sk, we use a scaling factor of 2, enumerating and choosing all primes from 3 until the product S_p is bigger than det(D_d)d, assuming a diagonal coefficient of $d = \sqrt{n}$, where n is the dimension, and getting an average determinant of $n^{n/2}$ for Sk. When S_p is not sufficiently large, then we put the symbol "—".

λ	300	400	500	600	700	800
80	34/87	24/115	21/142	19/168	18/195	17/222
100	—	37/115	29/142	26/168	24/195	23/222
120	—	—	42/142	35/168	32/195	29/222
140	—	—	—	48/168	41/195	37/222
160	—	—	—	69/168	53/195	47/222

However, if we start enumerating from 2741, which is the 400th prime, and then its successive primes, we have the following table.

λ	200	300	400	500	600	700	800
80	—	—	—	28/96	23/119	21/141	19/164
100	—	—	—	—	35/119	30/141	27/164
120	—	—	—	—	—	42/141	36/164
140	—	—	—	—	—	—	49/164
160	—	—	—	—	—	—	—

We put a blank entry at $\lambda = 160$ for dimension $n = 800$, but we are actually very close with

$$\log_2\left(\binom{164}{82}\right) > 159.99.$$

If we want to enforce $\omega(R) < \omega$(Sk) while using the same algorithm, we only need to increase d to a bigger value, increasing det(Sk) and thus ω(Pk). The next table shows the minimum amount of primes to put in S_p, which means the minimum value of ω(Pk) (+1 if we count the scaling factor) to achieve a number of combinations strictly superior to 2^λ.

λ	80	100	120	140	160	180	200	220	240
ω**(Pk)**	84	104	124	144	165	185	205	225	245

As a "rule of thumb", to achieve a number of combinations of 2^λ, we require $|S_p| = \omega(\mathrm{Pk}) + 1 > \lambda + 4$ for $\lambda \in [1, 250]$. We can deduce that, in practice, the number of combinations will always be as high as we need it to be, either by increasing the diagonal coefficient d of our secret key Sk or by adding factors to R. Therefore, security concerns do not lie in the number of combinations $(\mathcal{L}(\mathrm{Sk}), \mathcal{L}(R))$ any longer.

3.4 Key size comparisons and perfect HNFs

First of all, we would like to stress that this comparison only aims to show that our obfuscation technique applied to GGH is not impractical as far as storage is concerned. LWE-based cryptosystems usually rely on stronger security assumptions than our proposal, and other cryptosystems, like NTRU, have been studied for a much longer time; the lack of literature on lattice intersections does not let us claim the same level of confidence in security assumptions. Here we compare in Table 1 our public key sizes to the public key sizes (in bits) presented in Lindner and Peikert's work for LWE-based encryption [42] and the key sizes we obtain with intersections (in average). We only compare with the size of their keys per user, and not their full key which is already much bigger. Note that, unlike q-ary lattices, the values of the determinant are not set at the key generation, but usually do not stray away from each other by too much bits. To compute the key size of a perfect HNF, one just has to look at the number of bits of the determinant and multiply it by the lattice dimension. For dimension n, we use $\sqrt{n}$ as the diagonal coefficient and $[-1, 1]$ as the noise interval.

As we see, their partial public key is only smaller than our full public keys for higher dimensions. Furthermore, the technique used in their scheme is a very clever way to delegate part of the key to a trusted source or to the user that is an instance from an abstract system presented by Micciancio [51], while our scheme has mostly kept the basic setup from the GGH cryptosystem [27]. It might be possible that, in the future, such techniques become available for classical random lattices (and most of them admitting a perfect HNF), leading to better key sizes per user for higher dimensions.

Dimension	q	Public key size per user
128	2053	1.8×10^5
192	4093	2.9×10^5
256	4093	4.0×10^5
320	4093	4.9×10^5

	n					
s	125	150	175	200	225	250
20	6.1×10^4	8.8×10^4	1.2×10^5	1.6×10^5	2.1×10^5	2.6×10^5
40	6.5×10^4	9.4×10^4	1.3×10^5	1.7×10^5	2.2×10^5	2.7×10^5
60	7.7×10^4	1.0×10^5	1.4×10^5	1.8×10^5	2.3×10^5	2.8×10^5
80	—	—	1.6×10^5	2.0×10^5	2.4×10^5	2.9×10^5
100	—	—	—	2.2×10^5	2.6×10^5	3.1×10^5

	n					
s	275	300	325	350	375	400
20	3.2×10^5	3.9×10^5	4.6×10^5	5.3×10^5	6.2×10^5	7.1×10^5
40	3.3×10^5	4.0×10^5	4.7×10^5	5.5×10^5	6.3×10^5	7.3×10^5
60	3.4×10^5	4.1×10^5	4.9×10^5	5.6×10^5	6.5×10^5	7.5×10^5
80	3.6×10^5	4.3×10^5	5.0×10^5	5.8×10^5	6.7×10^5	7.6×10^5
100	3.8×10^5	4.4×10^5	5.2×10^5	6.0×10^5	6.9×10^5	7.8×10^5

Table 1: LWE key sizes from Lindner and Peikert on top, key sizes from perfect HNFs and intersections at the bottom (n = dimension, $s = \log_2 \binom{\omega(\mathrm{Pk})}{\omega(\mathrm{Sk})}$)

Perfect Hermite normal forms also allow us to improve the encryption scheme. Instead of sending a vector $c = m + v$, where $v \in \mathcal{L}$ is random, we can choose v such as $v = (c_1, 0, \ldots, 0)$, i.e., only a single integer of size $\det(\mathcal{L})$ in average will be sent. This is done by reducing the message m by a Gaussian elimination with the rows of the public key, where they all have 1 in their diagonal. Furthermore, the security is not affected. Since the transformation will give the same result as m for any $m' = m + v'$ with any $v' \in \mathcal{L}$, breaking this transformation will break all schemes which use BDD_γ as a security assumption. The decryption is left unchanged since the only difference is that $c = m + v'$, where $v' \in \mathcal{L}$ is not random anymore. This is actually convenient from a practical point of view, where encryption can be reduced to n additions and $n - 1$ multiplications (by relatively low integers on one side and big ones on the other side) modulo the determinant; this is comparable to a modular knapsack. Lindner and Peikert also presented in their paper their different ciphertext size for messages of 128 bits. The smallest ciphertext size they presented was for $n = 128$ and $q = 2053$, and it has the same size of our average determinant size (hence, in our case, ciphertext size) for a lattice of dimension $n = 475$ and combinations security $s = 100$.

Generating public keys, however, is slow as it involves computing Hermite normal forms. On the other hand, Gaussian sampling, which is required for a lot of LWE-based cryptosystems, was also far from fast and memory efficient and led to alternatives like LWR [9] or uniform distribution [45]. Weiden et al. reported that Gaussian sampling takes 50 % on the running time of their implementation of Lyubashevsky's signature scheme [46], and Dwarakanath and Galbraith [17] reported that Peikert's sampler can take up to 12 MB for some parameters [61]. Nevertheless, the research is still very active on that matter and has seen various improvements [12, 16, 17] mostly targeted at lattice-based cryptography thanks to the popularity of that matter in the community but not to a point where sampling is both very fast and memory efficient. On the other hand, research to compute Hermite normal forms nowadays is mostly done for random matrices [60, 63] and not targeted at structured matrices, especially ones that arise in cryptography. A rebound of interest towards Hermite normal forms in the community might lead to similar improvements in the future.

4 Security

The security assumptions rely on three important points.

- *Assumption* 1: recovering the "optimal" integer overlattice is hard. (Δ is polynomial on the number of combinations.)
- *Assumption* 2: the underlying BDD_γ problem is hard (on our specific keys).
- *Assumption* 3: the underlying modular knapsack problem is hard (on our specific keys and messages).

Like every cryptosystem, even when based on a hard problem, what we use are specific instances of a hard problem, which might not be as hard as the original problem. Therefore, we discuss in the following the special structure which arise from our scheme. Note that, because the main idea is to change the public key without changing the secret key while keeping the encryption/decryption process unchanged, the message space is left unchanged along with the BDD_γ problem from the previous iteration from Micciancio [50], except the lattice has a slightly bigger determinant.

Furthermore, our first assumption is actually very pessimistic. Micciancio's scheme has not been broken asymptotically, and up to this day, basis reduction techniques are still not well enough understood to predict, given a HNF, how easy it would be to reduce the corresponding lattice. It is therefore possible that, in the future, further theoretical studies will allow us to strongly reinforce our first assumption. Our particular structure (perfect HNF) also allows to convert our instance of the BDD_γ problem to an instance of a single general modular knapsack problem with very smooth moduli; therefore, our third assumption is that our instances of modular knapsack with smooth determinant are hard, which can be seen as a multiple modular knapsack with co-prime moduli.

4.1 Smoothness of determinant

To discuss the problem of the smoothness of the determinant, we assume the existence of a polynomial time solving oracle which can determine if a combination of primes lattice is correct and output a weak basis out of it if yes. We also assume that the attacker does not have an algorithm that permits to eliminate prime overlattices one by one until only factors of Sk remain, or something easy to decipher, which would lead to an easier solution than searching through every possibility, which we think is reasonable as our experimentations could not distinguish any overlattice from random lattices. If there was such an algorithm, then this might apply to any random lattice and lowers the overall security of lattice problems. However, it is possible that the problem is much easier depending on the kind of secret keys we actually use and what we intersect it with.

The only attacks based on overlattices according to the best of our knowledge were one from Becker, Gama and Joux [22] and one from Gama et al. [23]; even then, the way their overlattices are generated is very different and does not search for integer overlattices specifically. Aside from the overlattices consideration, there is also no attack to the best of our knowledge that makes use of a smooth determinant on random lattices, and other popular schemes such as NTRU [31] and Ring-LWE [49] also rely on lattices with smooth determinants (q-ary lattices have naturally smooth determinants, but their factorization differ).

Under all of those assumptions, the total security provided by our enhancement is either solving the problem directly on the public key, finding the right combination multiplied by the time of running a detection oracle (find which integer overlattice is weak), or working in a properly chosen non-integer overlattice of a large enough volume. The latter possibility, which could seem the most efficient, is in our opinion not-effective; to the best of our knowledge, our system does not hold a particular weakness towards this approach compared to any other random lattice as our public key seems to hold the same structure as far as our heuristic experimentations on HKZ-reduced bases are concerned (see the next subsection) Assuming the "weak integer overlattice" detection oracle runs in polynomial time (for simplification, we will consider it constant, but in practice, we do not know how to even create a practical one; the latter security is bounded by $\binom{\omega(\mathrm{Pk})}{\lceil \omega(\mathrm{Pk})/2 \rceil}$, which, as we discussed in the previous section, is not a problem in high dimensions as $\omega(\mathrm{Pk})$ grow big, as, in our tests, we never reach that bound (Figure 2)).

We stress that finding the shortest vector for a small prime overlattice does not help solving the problem in the ring product in general. It is, in fact, still a research problem to be able to compare two numbers given their decompositions over a ring product without computing them back as it is the main issue with residue number systems (RNS for cryptography is an old and still active research topic [6–8, 25]). This is even more problematic when comparing vectors.

4.2 Perfectness of basis and primality between factors

Due to Goldstein and Mayer [29], taking a perfect Hermite normal form matrix with a random prime determinant can be considered as taking a random lattice. As we are intersecting a lot of lattices of this type, with different prime determinants which result in a perfect HNF, we are comparing our results with other perfect HNF bases with the same determinant and random entries. Since our experimental results show that 80 % of the bases generated with coefficients from bounded entries admits a perfect HNF or are equivalent to one by permutation of columns (see Appendix A.2), we believe the comparison to be fair.

This is further reinforced due to recent works from Nguyen and Shparlinski [57], used shortly after by Gama et al. [23] to make their generalized worst-case to average-case reduction which allows us to use a much more general lattice form and is therefore very different from those q-ary lattices first introduced by Ajtai [1]. In practice, Chen and Nguyen's work [14] on Darmstadt's lattice challenges lead us to think it is easier to solve problems (find short vectors) in a q-ary lattice than in a random lattice of large volume; therefore, in terms of basis recovery attacks, the perfect Hermite normal form could be actually more desirable.

On top of that, there is no currently known algorithm to our knowledge that will provide efficient secret keys with an imperfect HNF on purpose (one that cannot be converted to a perfect HNF; see Appendix A.2),

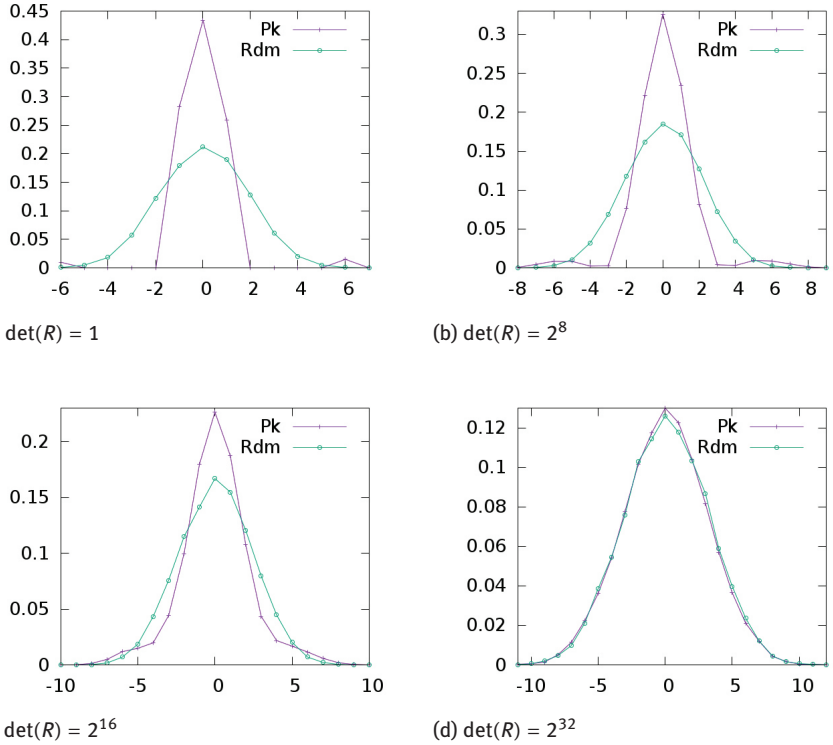
where the imperfect coefficients do not leak information on the first element of the diagonal (they very often have common divisors for example). We also keep in mind that if $\mathcal{L}(M)$ is a lattice that admits a diagonal dominant basis, then so do $\mathcal{L}(\sigma(M))$, where σ is a column permutation, retaining the exact same values, measuring basis quality in every criteria known (see Appendix A.2).

Furthermore, given a finite set of prime factors on the diagonal, the perfect form gives the hardest challenge as it is harder to guess a large number of factors in a single position rather than a small fixed amount in multiple positions (given the same total amount of primes), provided we could make sure it could not be transformed into a perfect HNF by permutation (if having a perfect HNF becomes a weakness).

Having factors prime to each other not only ensures an easier perfect HNF, but also avoids giving information beneficial to the attacker (see Appendix A.1). As stated before, having a perfect HNF is also desirable when managing keys.

4.3 Shortest vector and basis structure

We present the result of experimentations for intersecting diagonal dominant type matrices with a random one with a perfect HNF form below. We chose 3 as our scaling factor, allowing our perturbation matrix to have a measurable determinant as a power of 2. To determine the impact of R over Pk's resistance against enumeration and classical lattice reduction techniques compared to random lattices [69], we also observe the distribution of coefficients using SVP on small dimension (40), comparing them to the random case (every time with the exact same determinant and dimension). It seems like, after reaching a size of 32 bits, there is almost no difference between Pk or random lattices of the same determinant (Figure 1). As the difference tends to decrease very rapidly, we scale the graphs to the extrema.



(a) $\det(R) = 1$ (b) $\det(R) = 2^8$

(c) $\det(R) = 2^{16}$ (d) $\det(R) = 2^{32}$

Figure 1: Distribution of the shortest vector's coefficients.

However, the obfuscation of $\mathcal{L}(\mathrm{Sk})$'s structure is not exclusive to $\mathcal{L}(\mathrm{Sk})$' shortest vectors. We show with the following tests that the obfuscation works on the whole HKZ-reduced basis of $\mathcal{L}(\mathrm{Pk})$. In that regard, we compare condition numbers (CN) with the maximum norm. The test is done in dimensions 30, 40 and 50 with over 20 matrices per dimension and 20 witnesses per new determinant, the diagonal dominant type having noise in $[-1, 1]$ with the diagonal being $\lceil \sqrt{\dim} \rceil$. We only choose matrices with a perfect form. Every matrix computed has been HKZ-reduced.

First, with dimension 30, det(Sk) has on average 79 bits.

det(R)	1	2^4	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}
Avg CN (inter)	55.33	158.48	199.38	234.25	260.06	263.49	262.40	268.40	263.10
Avg CN (rdm)	269.41	273.39	272.88	273.99	273.74	267.13	267.31	271.71	272.65

With dimension 40, det(Sk) has on average 113 bits.

det(R)	1	2^4	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}
Avg CN (inter)	69.52	304.18	400.67	529.02	620.70	691.42	709.63	748.11	769.18
Avg CN (rdm)	755.89	760.90	805.09	755.15	789.46	786.06	770.28	760.09	786.72

With dimension 50, det(Sk) has on average 151 bits.

det(R)	1	2^4	2^8	2^{12}	2^{16}	2^{20}
Avg CN (inter)	78.98	524.19	636.06	837.49	1087.47	1426.19
Avg CN (rdm)	2038.68	1976.51	1993.44	1952.17	1944.57	2021.48

det(R)	2^{24}	2^{28}	2^{32}	2^{36}	2^{40}	2^{44}
Avg CN (inter)	1616.31	1682.42	1795.90	1830.14	1870.03	1871.08
Avg CN (rdm)	2074.12	1991.93	1964.16	1979.45	2000.98	1998.67

According to our experimental results, there is little influence on increasing the size of the perturbation over 32 bits as we get very close to the same condition number as a random HKZ-reduced matrix with the same determinant. This reflects the results we have when measuring the distribution of shortest vectors' coefficients values. This means that obtaining a good basis of the public key will allow to decrypt nearly as well as with a good basis of a random matrix, being very sensitive to noise. Therefore, the only factor to consider is the number of primes, and as we grow larger in dimension, we will obviously take much bigger primes, ending up in a noise with a determinant size of over 32 bits. We can then take small primes to minimize the lattice gap (as we will measure below).

What we need to consider is the possibility to distinguish the different overlattices very easily (which would nullify our improvement of GGH), or the possibility to know if, by intersecting different overlattices, we could determine if we are getting closer to the right combination or not. Therefore, we compare condition numbers of Sk's and R's overlattices, the intersection of Sk's and R's overlattices separately or mixed together (with all matrices being HKZ-reduced) to HKZ-reduced bases of random lattices of the same respective determinant for every test. The result is that the overlattices themselves seem to be indistinguishable from random cases. Since removing several factors of Sk randomly does seem to make the problem harder (in the sense that the resulting lattice looks more random than keeping all factors), this observation should also hold for non-integer overlattices. For now, it does not seem that an attack on a random overlattice would be more effective.

4.4 Message recovery attacks

We now measure the influence of R on Pk against message recovery attacks, and we will use the lattice gap in that regard. We assume the attack used is the popular heuristic transformation from CVP to SVP by attacking an extended basis B. What we need then is closely enough approximate values of $\lambda_1(B)$ and $\lambda_2(B)$. In that regard, we consider $\lambda_1(B)$ the size of our message m. With maximum decryption capacity, we have

$$\|mS_k^{-1}\|_\infty \leq \|m\|_\infty \|S_k^{-1}\|_\infty \leq \frac{1}{2}.$$

As we take the best value possible,

$$\|m\|_\infty = \frac{1}{2\|S_k^{-1}\|_\infty}, \quad \lambda_1(B) = \|m\|_2 \approx \frac{\sqrt{n}}{2\|S_k^{-1}\|_\infty},$$

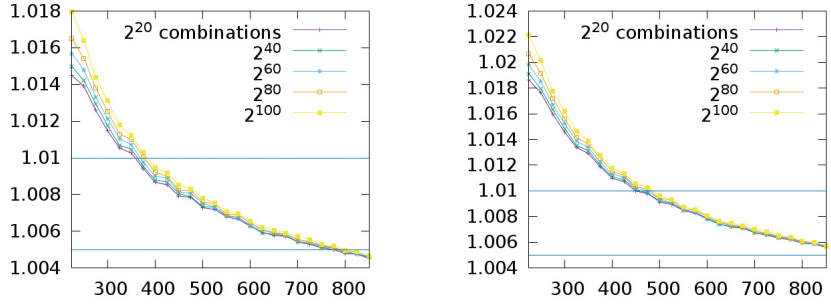
and we assume $\lambda_2(B) \approx \sqrt{\frac{n}{2\pi e}}\,\mathrm{Det}(\mathrm{Pk})^{1/n}$ with the Gaussian heuristic. It is the best approximation we can use as we do not know the length of the shortest vector in the public key[2]. Therefore, the lattice gap is

$$\delta(\mathrm{Pk}, \mathrm{Sk}) = \left(\frac{\mathrm{Det}(\mathrm{Pk})^{1/n} \times 2\|\mathrm{Sk}^{-1}\|_\infty}{\sqrt{2\pi e}}\right).$$

It is a bit different than most schemes as our public key does not represent the same lattice as our private key. The first remark we do when looking at the formula is that the gap increases as $\det(R)$ increases, which might give us a good reason to carefully manage $\mathrm{Size}(\det(R))$ besides key size considerations.

Here, we present our experimental results for the simulations on computing the root lattice gap (with primes starting from 3, scaling factor 2). From the earlier work of Gama and Nguyen [24], the problem is solvable as soon as the lattice gap is lower than a fraction of the Hermite factor. We observe the evolution of the lattice gap for different numbers of combinations over increasing dimensions, and it appears that increasing the number of combinations has a lower impact as we increase dimensions; thus a high value for $\det(R)$ is not a problem as dimensions increase.

As the constants mentioned in Gama and Nguyen's work [24] depend on the algorithm used and the lattice structure, we scale the curve with factors 0.50 and 0.20. We can observe that, under the pessimistic assumption of a constant of 0.20 (which is not the worst since [24] mentions a factor of 0.18 for BKZ-20), we do not reach the optimal factor of 1.005 (considered by many to be impossible to reach without proper structure in dimension 500, see [24]) even past dimension $n = 850$.



(a) $C = 1/2$ (b) $C = 1/5$

Figure 2: Evolution of the root lattice gap from dimension 225 to 850.

2 We note that other competitive approximations for random lattices [37, 73] do exist, but, as those results are relatively recent, they seem to be widely ignored by the cryptography community; thus we use more "popular" analysis tools for now.

4.5 IND-CCA security and message space

Suppose the private key has been generated by a diagonal coefficient D and a noise bound μ for a full-rank lattice of dimension; a first suggestion would be to use messages in the space $[-M, M]^n$, where $M = \frac{D-\mu}{2}$. This allows us to decrypt a number of bits which would be a multiple of the lattice dimension. However, the scheme does not achieve IND-CCA security that way, and it is advised to sacrifice the number of bits and restrict the message space to achieve IND-CCA security.

To achieve IND-CCA security, one can set the ciphertext to encrypt only a single bit with a technique similar to the one proposed in Regev's first LWE-based public-key encryption scheme [67], where the decryption is a test whether an integer is closer to 0 or to a certain bound, which, from a lattice-based point of view, just consists of set bounds for a norm on deciphered vectors m (original messages); for all deciphered plaintexts m and $\|m\| \in]a, b[$, where $a > 0$ and b is within our decryption capacity, if $\|m\| > \frac{a+b}{2}$, then return the bit 1 as the message, or 0 otherwise.

4.5.1 Using padding techniques

Alternatively, we can also use techniques which are already used for number theoretical schemes. Suppose our message is composed of bits; then we can apply padding techniques such as OAEP [10] to make the scheme more secure (and its further improvements such as [71]), or more general ones such as REACT [58] or the Fujisaki–Okamoto transformation [21]. Those techniques work assuming the one-wayness of our initial problem as it is done with NTRU. This seems to be the most straightforward method to achieve IND-CCA security in the random oracle model, and we will briefly present the Fujisaki–Okamoto transformation as it was recently suggested by Peikert for his suggestion of lattice-based cryptography for the internet [62].

The Fujisaki–Okamoto transformation [21] is to transform the encryption of the message m to

$$\mathcal{E}^{\text{hy}}_{\text{pk}}[m, c] = \mathcal{E}^{\text{asy}}_{\text{pk}}[c, H(c, \mathcal{E}^{\text{sy}}_{G(c)}[m])] \parallel \mathcal{E}^{\text{sy}}_{G(c)}[m],$$

where

- $\mathcal{E}^{\text{asy}}_{\text{pk}}[m, c]$ is the asymmetric encryption of the indicated message m using the indicated coins c as random bits and the public key pk,
- $\mathcal{E}^{\text{sy}}_{\text{sk}}[m]$ is the symmetric encryption of the indicated message m using the private key sk,
- σ is a random string chosen from an appropriate domain,
- G, H are hash functions, where $G(\sigma), H(\sigma)$ are the hashes of σ by G, H, respectively.

We need to choose a symmetric encryption scheme to apply this transformation, and depending on the symmetric scheme chosen, one might be interested to look at the earlier, simpler and more specialized proposition first given in [20]. The same can be said for the choices of the hash functions, and such considerations are also important if using OAEP or REACT (which, on the other hand, do not require an additional symmetric scheme and seem computationally less complex).

NTRU, on the other hand, uses padding schemes such as NAEP [33, 34] that are specific to NTRU due to its usage of polynomials and cannot be applied to other lattice schemes. We stress that a precise choice of a padding scheme is not trivial, as earlier choices for NTRU have been proven insecure, due to the particular nature of NTRU rather than the padding itself [32, 56].

REACT is also a good candidate for padding, due to its efficiency. The encryption of the message m is transformed to

$$\mathcal{E}_{\text{pk}}(m; R, r) = \mathcal{E}^{\text{asy}}_{\text{pk}}(R; r) \parallel G(R) \oplus m \parallel H(m, R, \mathcal{E}^{\text{asy}}_{\text{pk}}(R; r), G(R) \oplus m),$$

where

- $\mathcal{E}^{\text{asy}}_{\text{pk}}(m, c)$ is the asymmetric encryption of the message m using the coins c as random bits and the public key pk,
- R is a random element from the message space,
- r is a random element from the random coin space,
- G, H are hash functions.

Here both encryption and decryption only require two more hashings compared to the original scheme and do not require an additional symmetric encryption scheme unlike the Fujisaki–Okamoto transformation.

Since we have to use a padding scheme for real life applications and padding schemes are mostly made for binary written messages (XOR operations are almost always involved), we define our message space as $[-b, b]^{n-k}$, where $b = \lfloor \log_2(\frac{D-\mu}{2}) \rfloor - 1$ (that way, each cell can be XOR-ed with any binary string and remain in the decryption space), D being the diagonal coefficient and μ the noise bound of the secret key, n the lattice dimension and $0 \leq k \leq n$ the number of dimensions we sacrifice in the message space to add the padding and extra surjections (to make the original scheme pre-padding surjective and not bijective, especially, to assign a space for random coins). Values for D, μ are suggested in the next section.

5 Discussions

5.1 Parameter choices

To achieve a reasonably improved security (from lattice reduction techniques) without drastically increasing the key size, we suggest to use a matrix of dimension at least 400, with a diagonal coefficient of 20 and a noise bound of 1. Suppose we sacrifice half of the dimension for padding; this yields 4 bits per remaining dimension, which will be 800 bits. We suggest using any random family of primes whose total products slightly exceed the size of the determinant of the secret key, and furthermore, whenever possible, we should avoid very small primes to avoid any information potentially recovered by the scaling factor (even if unrealistic). The increase in key size will depend on the size of the random matrix we use to hide the secret lattice (as shown in Section 2). Moreover, taking too big primes will reduce the number of combinations, hence lowering the security in terms of discovering the secret lattice. We then recommend using primes between 541 (the 100th prime) and 2741 (the 400th prime); those are arbitrary choices but offer enough combinations for whatever matrix of such dimension, or even much higher, would need from what we see in the previous tables, and the size of the noise generated is, in our opinion, enough to protect against basis recovery attacks. The scaling factor can be as small as 2. Depending on the security level required (for various applications, as mentioned by NIST [54]), we can let $\lambda > 64, 80, 96, 128$ as far as the number of combinations 2^λ is concerned. We note that intersecting a small random lattice is already improving the security; therefore, even if we do not reach the same average as random matrices, the public key is still more secure.

Intersecting lattices is a technique that can be used with any kind of lattice. This technique may give rise to other useful applications, which remain unknown.

5.2 Alternate secret keys

Our approach is not only limited to a diagonal dominant matrix. As any message generated using Pk is equal to a message generated using HNF(Sk), any decryption process using HNF(Sk) as a secret key remains unchanged. The main problem in our opinion is always how to hide the overlattice $\mathcal{L}(\mathrm{Sk})$. As an example, using tensor products as secret keys [19] would leave a slightly different issue in hiding det(Sk). Other improvements on GGH type cryptosystems could be compatible with ours as we leave the decryption and message space unchanged; therefore, security improvements may become cumulative (a case-by-case study and adaptation might be necessary). An example of such a possible cumulative improvement to study can be found in [68].

Additionally, our approach is not limited to a perfect HNF either. A lot of properties do indeed arise when intersecting lattices which determinants are not co-prime which could lead to security issues when not kept in mind, as information on the factorization of det(Sk) is leaked (see Appendix A.1). However, if we carefully generate non-perfect HNF matrices on purpose from intersections it might be possible to ensure a reasonable security without increasing the key size too much.

Another idea is to use weaker secret keys. In our experiments, we randomized the occurrence of the noise values in $[-1, 1]$. However, seeing how intersections allow us to hide the secret basis' structure, we can make it so in a line vector of the secret key (or even in both lines and columns) the number of times a non-zero value is used is limited. For example, what if we set 1 to appear α times and -1 to appear β times, which will leave χ zeroes, where $\chi = n - 1 - (\alpha + \beta)$? If we allow a big χ, this will lead to a bigger decryption capacity. We do not know if such a prefixed setting would create visible weaknesses in the result of the intersection, or the overlattices, and how much impact it will have on the root lattice gap.

Another type of weaker secret key is using ideal lattices. Ideal lattices are a powerful tool to compress keys and to enable fully homomorphic encryption [26]. The research development in ideal lattices has been very popular in the last years; hence we are providing some remarks on the intersection of ideal lattices. The advantage of ideal lattices is allowing a stable multiplication and offering very compressed keys. Some interesting properties arise when intersecting ideal lattices (see Appendices A.1 and A.2), but, to the best of our knowledge, there is no method to control the determinant of an ideal lattice, which is how we make intersections secure in our paper. The possibility of reinforcing the security of ideal lattice based schemes with intersections also remains an open question.

A Appendix

A.1 Arithmetic of HNF basis from intersections

In this section, we present all the proofs and technical details concerning intersections of HNF. We do not focus on the perfect case, as it is just seen as a particular case, but it allows to understand how an intersection of lattices can break the perfectness of a HNF and what phenomenons appear when randomly intersecting lattices successively.

We will start with a few definitions. We consider line vectors with M_i being the i-th line vector and $M_{i,j}$ being the coefficient in the i-th line and j-th column in the matrix M, and we denote $\mathcal{L}(M)$ the lattice generated by M. If v is a vector, we denote v_i its i-th coefficient.

Definition 14. We denote $\mathrm{Sm}(M, (i, j))$ the square submatrix of M of size $(j - i + 1)^2$ extracted from the h-th lines and columns $i \le h \le j$ with $1 \le i \le j \le n$. We denote $\mathrm{Sm}(M, (1, i)) = \mathrm{Sm}(M, i)$ the square submatrix with only the first i lines and columns.

Definition 15. Let $\mathcal{L}(A)$ be a full-rank lattice with A in HNF. We denote $\mathcal{L}_i(A)$ the sublattice of $\mathcal{L}(A)$ such that $\mathcal{L}_i(A)$ is trivially isomorphic to $\mathcal{L}(\mathrm{Sm}(A, i))$, i.e., $\mathcal{L}_i(A)$ is generated by the first i vectors of A.

Definition 16. A vector v is a "standard" vector if it is also a vector of the Id matrix basis (i.e., 1 in one position, 0 in all the others) and "p-standard" if it is a vector of a $p \times$ Id matrix, and we say it is the i-th p-standard when the non-zero entry is at the i-th position.

Definition 17. For a triangular matrix M of dimension n and $i \le n$, we denote

$$\mathrm{dg}(M, i) = \prod_{1}^{i} M_{i,i} = \det(\mathrm{Sm}(M, i)) \quad \text{and} \quad \mathrm{Diag}(M, i) = \{M_{j,j}, 1 \le j \le i\}$$

Definition 18. Let M be the basis of an integer lattice $\mathcal{L}$ of dimension n in HNF form. Then we define by $\Phi(\mathcal{L})$ the number of non-1 values in the diagonal of M (which is also the number of non-standard columns). As the HNF form is unique, we can write $\Phi(B) = \Phi(\mathcal{L})$ for any basis B of $\mathcal{L}$. We say M is of pseudo-perfect form when $\Phi(M) = 1$ and of perfect form if $\mathrm{Sm}(M, n - 1) = \mathrm{Id}_{n-1}$.

Definition 19. We say a lattice is prime if it is an integer lattice whose determinant is the power of a prime number and it is pseudo-perfect.

Property 2. Suppose $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$, all lattices are of dimension $n > i$, and A, B, C in HNF; then every vector C_i is uniquely determined by $\mathrm{Sm}(A, i)$ and $\mathrm{Sm}(B, i)$ and $C_{1,1} = \mathrm{lcm}(A_{1,1}, B_{1,1})$. In particular,

$$\mathcal{L}(\mathrm{Sm}(C, i)) = \mathcal{L}(\mathrm{Sm}(A, i)) \cap \mathcal{L}(\mathrm{Sm}(B, i)).$$

Therefore, modifying $\mathrm{Sm}(A, (i+1, n))$ and $\mathrm{Sm}(B, (i+1, n))$ has no influence on $\mathrm{Sm}(C, i)$.

Proof. If M is in HNF, the HNF basis of $\mathcal{L}_i(M)$ is the first i lines of M. Apply this directly to $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$ to obtain the above property. □

Property 3. Let A be the HNF basis of a full-rank lattice of dimension $n \geq i \geq 1$, and denote $g = \mathrm{dg}(A, i)$. The i-th g-standard vector always belongs to $\mathcal{L}(A)$.

Proof. We know that, for any vector $v \in \mathcal{L}_i(A)$, we have $A_{i,i} \mid v_i$.

Let us put $\alpha \leftarrow \frac{g}{A_{i,i}}$ and $v' \leftarrow \alpha A_i$. Note that α is divisible by every coefficient in $\mathrm{Diag}(A, i-1)$. Then v' has g as its head coefficient, and all of its other coefficients are multiples of α.

Let us eliminate v'_{i+1} by subtraction. First set $\alpha \leftarrow \frac{\alpha}{A_{i+1,i+1}}$; then $v' \leftarrow v' - \alpha A_{i+1}$. We note that all coefficients of v' are still multiples of α. Rinse and repeat for coefficients $(v'_{i+2}, \ldots, v'_n)$ until the end result is $v' = v$, i.e., $v \in \mathcal{L}(A)$. □

Property 4. Suppose A, B, C are in HNF. If $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$, then

$$\mathrm{lcm}(A_{i,i}, B_{i,i}) \mid C_{i,i} \mid \mathrm{lcm}(\mathrm{dg}(A, i), \mathrm{dg}(B, i)).$$

Proof. Using Property 2, it is sufficient to work on $\mathcal{L}_i(C)$, and we know that $C_i = \alpha A_i + a' = \beta B_i + b'$ with $a' \in \mathcal{L}_{i-1}(A)$ and $b' \in \mathcal{L}_{i-1}(B)$, for some $\alpha, \beta \in \mathbb{Z}$. Therefore, $C_{i,i} = \alpha A_{i,i} = \beta B_{i,i}$, hence the left part of the expression.

For the right part, just apply the reasoning of the previous property to the i-th g-standard vector with $g = \mathrm{lcm}(\mathrm{dg}(A, i), \mathrm{dg}(B, i))$ and find it belongs to $\mathcal{L}(A) \cap \mathcal{L}(B)$. □

We note that those bounds are reached in practice and are therefore tight. What we will be showing next is how the head coefficients are actually affected by the big non-standard column of the HNF.

Property 5. Suppose A and B are perfect-form HNF bases of lattices $\mathcal{L}(A)$ and $\mathcal{L}(B)$ of full rank n, with $a = \det(A) = A_{1,1}$ and $b = \det(B) = B_{1,1}$. Then $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$ admits a perfect HNF basis C if and only if

$$\text{for all } i, \quad \text{there exists } k_1 < b,\ k_2 < a \quad \text{such that} \quad A_{1,i} + k_1 a \equiv B_{1,i} + k_2 b \mod \mathrm{lcm}(a, b) = C_{1,1}.$$

If the condition is respected, then

$$C_{1,i} = A_{1,i} + k_1 a = B_{1,i} + k_2 b \mod C_{1,1}.$$

Proof. If the condition is respected for a certain i, then there exists a common vector in both $\mathcal{L}(A)$ and $\mathcal{L}(B)$ that have 1 in its i-th position, $A_{1,i} + k_1 a = A_{1,i} + k_2 b$ in its first position, and 0 everywhere else. Therefore, it must be representable by the basis C, and since $C_{i,i} \mid v_{i,i}$ for all i, perfect form or not, thus enforces $C_{i,i} = 1$.

Now suppose the condition is not respected for a certain i such that $\mathrm{Sm}(C, i-1)$ already has perfect form. Since $\mathrm{Sm}(C, i-1)$ has perfect form, then every coefficient that is neither the first nor the i-th one is reduced to 0 by definition of the HNF. If $C_{i,i}$ was equal to 1, then A being perfect would mean $C_{1,i} \equiv A_{1,i} \pmod a$ since $C_{i,i} \in \mathcal{L}(A)$ and the same reasoning for B. That would respect the condition, which is contradictory; hence $C_{i,i} \neq 1$. □

Property 6. Let A, B be two perfect HNF bases of lattices such that $\det(A) = \det(B) = p$ is a prime number. Then either $\det(C) = \det(\mathcal{A} \cap \mathcal{B}) = p^2$ and $\Phi(C) = 2$ with two p-standard vectors, or $A = B$.

Proof. If $A = B$, then it is obvious. Otherwise, Property 5 tells us it is not possible to have a perfect HNF. Using the same reasoning, the first i such that $A_{1,i} \neq B_{1,i}$ will have C_i as the i-th p-standard vector, as Property 4 leaves us only p as a diagonal coefficient. Since $\mathrm{Sm}(C, i-1)$ is a perfect HNF, the only solution for C_i to be a vector of $\mathcal{L}(A) \cap \mathcal{L}(B)$ is to have $C_i = pA_i - A_{1,i}A_1 = pB_i - B_{1,i}B_1$, i.e., the i-th p-standard vector; since p is prime, there is no lower integer that would make this work.

Now that we have to prove that $\mathrm{Sm}(C, (i, n))$ is a perfect form matrix. For that, in $\mathbb{Z}_p$, we just have to prove that,

$$\text{for all } j \in]i, n], \quad \text{there exists } x, y \in \mathbb{Z}_p \quad \text{such that} \quad xA_i + A_j = yB_i + B_j,$$

which is always true in our case as it corresponds to finding a solution (x, y) to $yB_{i,1} + B_{j,1} = xA_{i,1} + A_{j,1}$ and $yB_{i,i} + B_{j,i} = xA_{i,i} + A_{j,i}$. □

Theorem 2 is a direct consequence of Property 5 (and to a lesser extent of Property 6). Applying the same reasoning on more than two columns, if we carefully choose $i < n$ lattices all with the same prime determinant p, then it is possible to make the intersection of those i lattices contain i p-standard vectors in their HNF basis. For $i = n$, the HNF of their intersection can be the orthogonal matrix $p \times \mathrm{Id}_n$. Generally, the more lattices of determinant d (prime or not) we get, the closer we get to $d \times \mathrm{Id}_n$.

It is actually possible to list all possibilities leading to the result of a prime collision, as we show in this example where p "moves" to the second column (i.e., $a_{2,1} \neq b_{2,1}$):

$$\mathcal{L}\left(\begin{bmatrix} p & 0 & 0 \\ a_{2,1} & 1 & 0 \\ a_{3,1} & 0 & 1 \end{bmatrix}\right) \cap \mathcal{L}\left(\begin{bmatrix} p & 0 & 0 \\ b_{2,1} & 1 & 0 \\ b_{3,1} & 0 & 1 \end{bmatrix}\right) = \mathcal{L}\left(\begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ c_{3,1} & c_{3,2} & 1 \end{bmatrix}\right),$$

which leads to $c_{3,1} \equiv c_{3,2}b_{2,1} + b_{3,1} \equiv c_{3,2}a_{2,1} + a_{3,1} \bmod p$.

And supposing we have higher dimension $n > 3$, then, for all $i \in [3, n]$,

$$c_{i,1} \equiv c_{i,2}b_{2,1} + b_{i,1} \equiv c_{i,2}a_{2,1} + a_{i,1} \mod p.$$

Hence, a prime collision has lots of easily computable solutions, and anybody having to solve this can choose the solution he sees fit (just choose $a_{2,1}$ and $b_{2,1}$, $p(p-1)$ possibilities). For the general case ("collision" at column j), just change 2 by j and 3 by $j+1$. The generalization can go even further with determinants $\det(A) = p^x$ and $\det(B) = p^y$ with $x = y$. When $x \neq y$, the result is still predictable but requires a little more work.

Property 7. The following statements hold.

- If a lattice is prime, every possible decomposition into an intersection of prime lattices must include itself.
- If a lattice has prime determinant, it has no integer overlattice (except itself).
- For any integer lattice with diagonal coefficients prime with each other (in its HNF form), there is a unique decomposition into an intersection of prime lattices with determinants that are prime with each other. (every condition is important as it becomes false otherwise)

The three properties, directly deduced from the previous ones, are important to understand the recovery of overlattices. To obtain a prime lattice decomposition, you just have to use the Chinese remainder theorem on every prime factor.

Property 8. Suppose $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$, and A, B, C are integer matrices. Then we can recover A, B from C with only the knowledge of $\mathrm{Diag}(A, n)$ or $\mathrm{Diag}(B, n)$ in polynomial time on $\det(C)$ and n.

Proof. The decomposition into prime lattices allow us to choose which prime lattices are part of A or B. Obtaining the decomposition into prime lattices is done in polynomial time with the Chinese remainder theorem, and then knowledge of the diagonal coefficients allow us to recover exactly A and B. □

Note that if $\gcd(\mathrm{Diag}(A, n), \mathrm{Diag}(B, n)) = 1$, then the solution is unique.

This recovery assumes the knowledge about the positions of diagonal coefficients. We do not need to consider the case when both determinants are co-prime; however, it can become tricky when determinants are not co-prime and positions are unknown. In that case, we will have to use the results from Property 6. Supposedly, 80 % of random lattices with randomly chosen bounded entries have perfect HNF or are equivalent to one by permutation, and we do want to be as close as possible to the random case (see Appendix A.2); the bigger the primes are, the more likely a random matrix of the same determinant is going to have an perfect form [29].

Furthermore, having an imperfect HNF means we have certain coefficients on the diagonal which potentially reveals some information about the secret key (whether by their value or by their position), and thus having transforming into a perfect HNF allows us to hide that information. Therefore, even if we do not obtain an equivalent perfect HNF for a basis M, minimizing $\Phi(M)$ is still useful.

Properties 4 and 5 ensure that if a lattice has perfect HNF, then prime lattices from its decomposition all have perfect HNF. In fact, Property 4 makes it unwise to use non-perfect HNF as secret keys (at least without prior study), as the information would be leaked instantly and cannot be hidden with intersections or permutations as we explain in the next subsection.

Another basis type to consider, besides diagonal dominant matrices for the secret key, are ideal lattices.

Definition 20. Let $R_f = \mathbb{Z}[X]/\langle f\rangle$ be a ring of polynomials over integers, where f is a monic polynomial of degree n, and consider the map $\Phi_f : X^{(i-1)} \to v_i$, where v_i is the i-th standard vector of $\mathbb{Z}^n$, and X^i is the monomial of degree i over R_f. We can then define a non-trivial ring structure over a particular set of sublattices of $\mathbb{Z}^n$ by mapping R_f to $\mathbb{Z}^n$ via Φ. Suppose we take an ideal of R_f, namely, $R_f(p) = \langle p\rangle$ with $p \in R_f$. We denote $\mathcal{L}(p, f)$ the lattice $\mathrm{Im}_{\Phi_f}(R_f(p))$.

For any lattice $\mathcal{L}$, if there exists p, f such that $\mathcal{L} = \mathcal{L}(p, f)$, we say $\mathcal{L}$ is an ideal lattice.

Since we have an isomorphism of rings between $\mathcal{L}(p, f)$ and $R_f(p)$, working on one or the other is equivalent. In the literature, we usually see $f = X^n - 1$ or $f = X^n + 1$.

Property 9. $\mathcal{L}(\mathrm{lcm}(p, q) \bmod f, f) \subseteq \mathcal{L}(p, f) \cap \mathcal{L}(q, f)$.

Proof. Let $r = \mathrm{lcm}(p, q) \bmod f$. Hence $r \in \langle p\rangle$ and $r \in \langle q\rangle$; therefore, $\Phi_f(r) \in \mathcal{L}(p, f)$ and $\Phi_f(r) \in \mathcal{L}(q, f)$; thus $\mathcal{L}(r, f) \subseteq \mathcal{L}(p, f) \cap \mathcal{L}(q, f)$. □

The equality happens quite often in experimentations. Intersecting two ideal lattices with two different quotients bring results which are, in general, unknown to the best of our knowledge.

Property 10. Let $\mathcal{L}_1, \mathcal{L}_2$ be two integer lattices with co-prime determinants and admitting a basis with perfect HNF. Then $\mathcal{L}_1 \cap \mathcal{L}_2$ is an ideal lattice if and only if $\mathcal{L}_1$ and $\mathcal{L}_2$ are.

Proof. Recall the perfect HNF of an ideal lattice, which only uses one root and a determinant (one visual representation can be seen in [65, p. 4]). If the HNF of both $\mathcal{L}_1$ and $\mathcal{L}_2$ respects the root exponentiation structure of ideal lattices, the same goes for $\mathcal{L}_1 \cap \mathcal{L}_2$ as Theorem 2 shows. If not, then the root exponentiation structure in the intersection is not respected either, while being a perfect HNF. □

The intersection of an ideal lattice and a random lattice is therefore not an ideal lattice.

A.2 Permutations and perfectness

There is a way to greatly increase the chance to obtain a perfect HNF basis, and it comes with a column permutation. The advantage of using permutations is to be able to obtain perfect HNFs without changing the norms or the orthogonality of any basis. If one needs a perfect HNF, it also drastically increases the chance to get a secret key with perfect HNF rather than discarding it and generating a new one. It is also used to hide various structures, but this is not the main point here.

We note that Tourloupis [74] had a way to select matrices with a perfect HNF with a 99 % chance of success; however, our aim is different here as we are trying to transform a secret key into another one rather than discarding it completely.

Property 11. If M is a diagonal dominant matrix, then $\mathcal{L}(\sigma(M))$ for $\sigma \in S_n$ still admits a diagonal dominant basis with the same parameters as far as noise bound and diagonal value are concerned.

This is trivial to see if we just permute the lines of the permuted columns on M.

Property 12. Let M be an integer matrix which admits a perfect HNF. Then if a permutation $\sigma \in S_n$ leaves the first column untouched, $\mathrm{HNF}(\sigma(M))$ still has perfect form.

What happens is just a permutation of the coefficients in the first column. It allows us to reorder them as much as we see fit, which might lead one day to interesting properties on any type of lattice. More noteworthy is the following property.

Theorem 4. *Let A be an integer matrix. If there exists $\sigma' \in S_n$ such that $\mathrm{HNF}(\sigma'(A))$ has perfect form, then there exists a column swap $\sigma = (1\ i) \in S_n$ with $i \in [1, n]$ such that $\mathrm{HNF}(\sigma(A))$ has perfect form.*

Proof. Suppose there is such a permutation σ'. Then we denote $\sigma'(A) = A'$ with $\mathrm{HNF}(A')$ being perfect. Now, σ' can be decomposed to $X \circ \sigma$, where $\sigma = (1\ i)$, $i \in [1, n]$, and X leaves the first column untouched. Also, X^{-1} leaves the first column untouched, and therefore $X^{-1}(\sigma'(A)) = \sigma(A)$ also has perfect form. □

This reduces the number of combinations to obtain perfect HNFs from $n!$ to 1. The single permutation $(1\ n)$ is sufficient to check since if $(1\ i)$ transforms M into a basis M' of an equivalent lattice $\mathcal{L}(M')$ admitting a perfect HNF basis, then $(i\ n)$ transforms M' into M'' still admitting a perfect HNF. However, for hiding a certain type of basis, any element $\sigma \in S_n$ can be used as a part of the secret key.

Also, we can easily identify some automatic failure cases, which allows us to avoid computations of HNFs in guaranteed case of failure.

Property 13. The following statements hold.

- Let A, B, C denote three lattices such that $A = B \cap C$. Then $\sigma(A) = \sigma(B) \cap \sigma(C)$ for any column permutation $\sigma \in S_n$.
- Let $\mathcal{L}$ be an imperfect lattice, and let us take any decomposition into prime lattices (as it might not be unique). Then $\mathcal{L}$ is equivalent by permutation to a perfect lattice if and only if there exists $\sigma \in S_n$ such that every integer overlattice in that decomposition is made perfect by σ, and they all respect towards each other Property 5.
- Let M be an integer matrix in HNF which is not perfect. Let $i < n$ be the biggest integer such that $\mathrm{Sm}(M, (i, n))$ has perfect form. Then $\mathrm{HNF}(\sigma(M))$ for every $\sigma = (1\ a)$, $a \in [1, i-1]$ does not have perfect form.

The following test is made with random matrices with coefficients that vary between −7 and 7. We then transform them into HNF and exclude every matrix which already has a perfect HNF, only keeping the imperfect ones to try conversion on. This is a test with 1000 matrices non-perfect HNF matrices on various dimensions.

Dimension	50	80	100	120	150	180	200	300
Success rate	0.756	0.726	0.741	0.748	0.744	0.749	0.728	0.737

Counting the ones that have already a perfect HNF, we believe the probability of a random matrix with bounded entries to have a perfect HNF or being equivalent to one by column permutation raises to be over 80 % (up from at most 40 % without permutation [68]). This is actually incredibly close to the 85 % presence of co-cyclic lattices among all lattices counted by Nguyen and Shparlinski [57] and used by Gama et al. [23] for their average-case to worst-case proof for most randomly chosen lattices.

The following property gives us an important case of non-convertible non-perfect HNFs.

Property 14. Denote by A, B matrices in perfect Hermite normal form. Let C be the Hermite normal form basis of the lattice $\mathcal{L}(A) \cap \mathcal{L}B$. If C does not have a perfect form (i.e., A and B do not respect Properties 5 and 6 relative to each other), then $\Phi(\sigma(C)) > 1$ for any $\sigma \in S_n$.

This is noticeable by looking at the prime decomposition and looking at the prime collision. The colliding primes will never intersect into a matching lattice, and despite the high success rate of permutations in transforming a non-perfect lattice in non-perfect form, no permutation (acting on the two matrices simultaneously) would change that.

Property 15. Let M be a perfect HNF basis of an ideal lattice. In general, with $\sigma \in S_n$, $\mathcal{L}(\sigma(M))$ is not an ideal lattice.

This is because permuting columns change the root exponentiation structure present in the perfect HNF of an ideal lattice basis. This could prove useful to hide ideal lattices structure, but we do not know whether the inverse permutation is easily recoverable or not.

A.3 Generation algorithm and complexity

To make the following easier to read, we introduce the following functions.

- HNF(M): given a matrix M, returns its Hermite normal form,
- IsCoprime(a, b): returns true if a, b are co-prime, false otherwise,
- Prod(S): given a set S, returns the product of all elements of S,
- Det(M): returns the determinant of M,
- M_1 cat M_2: given M_1, M_2, concatenates them into a new matrix,
- Babai(M, v): applies Babai's nearest plane algorithm to v with the lattice $\mathcal{L}(M)$. If M is HKZ-reduced, it is incredibly effective, especially, with a diagonal dominant lattice.

The parameters are a family of primes S_p, a scaling factor f, a security parameter λ for the number of combinations, a dimension, and parameters (D, b) for generating the diagonal dominant matrix. Of course, we have to check beforehand that S_p is large enough to meet 2^λ combinations but not too big to have a high chance to respect equation (3.2).

Note that, following Theorem 2, computing the HNF basis of the intersection of two full rank n lattices $\mathcal{L}_1$, $\mathcal{L}_2$ with perfect HNF basis of co-prime determinants d_1 and d_2 has only complexity $O(n \log(d_1 d_2))$ in time and in memory, as it is nothing more than using the Chinese remainder theorem to reconstruct n integers. We believe this is faster in practice than computing $\text{Dual}(\text{Dual}(\mathcal{L}_1) \cup \text{Dual}(\mathcal{L}_2))$ to obtain $\mathcal{L}_1 \cap \mathcal{L}_2$ in our particular case.

The complexity is mostly the computation of the HNF and Babai's nearest plane algorithm; the rest are manipulations of integers that are linear in the dimension and polynomial in the determinant (GCD computations, multiplications). Using Micciancio and Warinschi's algorithm for obtaining HNF [52], the complexity is $O(n^4 \text{ polylog}(M, n))$ arithmetical operations, which can be improved in practice using later works like [63], and Babai's nearest plane algorithm is also polynomial [4]. Trying to get a perfect HNF can appear to be costly in the while loop, but, in experimentations, we have in average only one retry when using permutations.

This goes in line with the probability of having two random numbers being co-prime which is $\frac{6}{\pi^2} \approx 61\,\%$ (first solved by Euler in 1735; see [30, Theorem 332, p. 269 in the 4th edition] for proof) and our experimental results with obtaining a perfect HNF from permutations with over 80 % ($0.8 \times 0.6 = 0.48$). The probability is slightly higher if we use the concatenated column c as a permutable column, and we might even gain more in efficiency by reinitializing the permutable column c instead of Sk_, even though it does not change the overall complexity much. Another way to heuristically improve speed at the cost of memory in the generation of Pk^c is to save the transformation matrix T such that $T \times \text{Sk\_} = \text{HNF}(\text{Sk\_})$, and using the first of row of T, one can quickly find c such that $(Tc)[1]$ is prime to det(Sk_). In that case, the extra computation due to $\text{Pk}^*[1][1]$, $\text{Pk}^*[1][2]$ not being co-prime becomes negligible.

Now we have to explain why this algorithm works and outputs the correct result. The first part is the part of the algorithm which has a note above "ensuring the end result is in perfect form". This is to ensure that the small square we complete with Bézout coefficients can be reduced in perfect form. If those two columns are not co-prime with det(Sk), then the final result cannot be reduced to a perfect form (as reducing is using linear combinations); therefore, we choose a column we will ensure to be co-prime with det(Sk), and this guarantees the result.

The second part is to explain why the end result is still a diagonal dominant matrix (at least with overwhelming probability). Suppose we have not used any permutation (if we used one, the result is the same according to a property in the previous section), and for simplicity, we denote $\text{Pk}^c[1][2] = |\det(\text{Sk\_})| = a$,

Input: S_p, $\min(S_p) > f > 1$, λ, $n > 1$, (D, b)
Output: Sk, Pk
NbPrimes $\leftarrow 0$;
while $2^\lambda > \binom{|S_p|}{\text{NbPrimes}}$ **do**
  NbPrimes $\leftarrow$ NbPrimes + 1;
end
$\text{Pk}^* = [2, 2]$;
while IsCoprime($\text{Pk}^*[1][1]$, $\text{Pk}^*[1][2]$) **do**
  Sk_- $\leftarrow$ DD matrix Sk_- of dimension $n - 1$ with diagonal coefficient D and noise b;
  Sk^c $\leftarrow$ Sk_- cat (c, a column of $n - 1$ values bounded by b);
  // new column position is 1
  $\text{Pk}^c = \text{HNF}(\text{Sk}^*)$;
  // use permutations to raise chance of perfect HNF on $\text{Sm}(\text{Pk}^c, (2, n))$
  if Pk^c *amputated of its first column is not a perfect HNF* **then**
    continue;
  end
end
find coefficients u, v such that $u\text{Pk}^c[1][2] + v\text{Pk}^c[1][1] = 1$;
// ensuring the end result is in perfect form
if IsCoprime(f, $\text{Pk}^c[1][1]$) **then**
  Id $\leftarrow 1$
else
  Id $\leftarrow 2$
end
$D_R \leftarrow 1$; $D_{S_k} \leftarrow 1$;
// choosing $\det(R)$
foreach $\{p \in S_p, p|\text{Pk}^c[1][Id]\}$ **do**
  remove p from S_p; NbPrimes $\leftarrow$ NbPrimes $- 1$; $D_R \leftarrow D_R p$;
end
while NbPrimes > 0 **do**
  Remove p randomly from S_p;
  $D_R \leftarrow D_R p$;
  NbPrimes $\leftarrow$ NbPrimes $- 1$;
end
// choosing $\det(\text{Sk})$
$D_{S_k} \leftarrow \text{Prod}(S_p)$;
// using the scaling factor f to respect equation (3.2)
while $D_{S_k} < D \det(\text{Sk}_-)$ **do**
  $D_{S_k} \leftarrow D_{S_k} f$;
end
$Y \leftarrow$ line of n zeroes; $Y[1] \leftarrow uD_{S_k}$; $Y[2] \leftarrow -vD_{S_k}$;
$\text{Pk} \leftarrow \text{HNF}(\text{Pk}^c \text{ cat } Y) \cap$ (random perfect HNF of determinant D_R);
$\text{Sk} \leftarrow \text{Sk}^c$ cat $\text{Babai}(\text{Sk}^c, Y)$;
return Pk, Sk

Algorithm 3: Generating new GGH keys using diagonal dominant matrices.

$\mathrm{Pk}^c[1][1] = b$. Subtracting a vector from the other ones does not change the determinant. Let us reduce l by the vectors of Sk^c (with Babai's nearest plane algorithm, which is the final step of the algorithm to compute Sk), and denote that reduced vector l'. Because Sk_- is diagonal dominant with diagonal coefficient d, reducing the line l with Sk^c will reduce all its coefficients except the first to integers below d; therefore, $l'[2] < d$. We denote $\mathrm{Pk}^{l'}$ the square matrix (l' cat Pk^c). We know $\det(\mathrm{Pk}^{l'}) = \det(\mathrm{Pk})$. Now we look at the computation of $\det(\mathrm{Pk}^{l'})$ developing over the first row l' and naturally obtain $\det(\mathrm{Pk}) = l'[1]a - l'[2]b$. Since a and $\det(\mathrm{Pk})$ are positive, $l'[2]$ and b have the same sign if and only if $l'[1]$ is positive. Now, because of equation (3.2), $da < l'[1]a - l'[2]b < 2da$; therefore, $d < l'[1] - \frac{l'[2]b}{a} < 2d$, and since $|b| < a$ with overwhelming probability (the opposite has never occurred in experimentations) and $|l'[2]| < d$, this forces $l'[1]$ to be positive and reasonably close to $[d, 2d]$. In practice, it seems closer to $2d$ than d.

References

[1] M. Ajtai, Generating hard instances of lattice problems (extended abstract), in: *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing*, ACM, New York (1996), 99–108.

[2] M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, in: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing—STOC '97*, ACM, New York (1999), 284–293.

[3] M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick and L. Perret, Algebraic algorithms for lwe problems, *ACM Commun. Comput. Algebra* **49** (2015), no. 2, 62–62.

[4] L. Babai, On Lovász' lattice reduction and the nearest lattice point problem, *Combinatorica* **6** (1986), no. 1, 1–13.

[5] S. Bai and S. D. Galbraith, Lattice decoding attacks on binary LWE, in: *Information Security and Privacy—ACISP 2014*, Lecture Notes in Comput. Sci. 8544, Springer, Berlin (2014), 322–337.

[6] J.-C. Bajard, J. Eynard and N. Merkiche, Multi-fault attack detection for RNS cryptographic architecture, in: *IEEE 23nd Symposium on Computer Arithmetic*, IEEE Press, Piscataway (2016), 16–23.

[7] J.-C. Bajard and L. Imbert, A full RNS implementation of RSA, *IEEE Trans. Comput.* **53** (2004), no. 6, 769–774.

[8] J.-C. Bajard and T. Plantard, Rns bases and conversions, in: *Optical Science and Technology, the SPIE 49th Annual Meeting*, SPIE Press, Bellingham (2004), 60–69.

[9] A. Banerjee, C. Peikert and A. Rosen, Pseudorandom functions and lattices, in: *Advances in Cryptology—EUROCRYPT 2012*, Lecture Notes in Comput. Sci. 7237, Springer, Heidelberg (2012), 719–737.

[10] M. Bellare and P. Rogaway, Optimal asymmetric encryption, in: *Advances in Cryptology—EUROCRYPT '94*, Lecture Notes in Comput. Sci. 950, Springer, Berlin (1995), 92–111.

[11] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Encyclopedia Math. Appl. 39, Cambridge University, Cambridge, 1991.

[12] J. Buchmann, D. Cabarcas, F. Göpfert, A. Hülsing and P. Weiden, Discrete ziggurat: A time-memory trade-off for sampling from a gaussian distribution over the integers, in: *Selected Areas in Cryptography—SAC 2013*, Lecture Notes in Comput. Sci. 8282, Springer, Berlin (2013), 402–417.

[13] J. Buchmann, F. Göpfert, R. Player and T. Wunderer, On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack, in: *Progress in Cryptology—AFRICACRYPT 2016*, Lecture Notes in Comput. Sci. 9646, Springer, Cham (2016), 24–43.

[14] Y. Chen and P. Q. Nguyen, BKZ 2.0: better lattice security estimates, in: *Advances in Cryptology—ASIACRYPT 2011*, Lecture Notes in Comput. Sci. 7073, Springer, Heidelberg (2011), 1–20.

[15] H. Cohen, *A Course in Computational Algebraic Number Theory*, Grad. Texts in Math. 138, Springer, Berlin, 1993.

[16] L. Ducas and P. Q. Nguyen, Faster Gaussian lattice sampling using lazy floating-point arithmetic, in: *Advances in Cryptology—ASIACRYPT 2012*, Lecture Notes in Comput. Sci. 7658, Springer, Heidelberg (2012), 415–432.

[17] N. C. Dwarakanath and S. D. Galbraith, Sampling from discrete Gaussians for lattice-based cryptography on a constrained device, *Appl. Algebra Engrg. Comm. Comput.* **25** (2014), no. 3, 159–180.

[18] P. Erdős and M. Kac, The Gaussian law of errors in the theory of additive number theoretic functions, *Amer. J. Math.* **62** (1940), 738–742.

[19] R. Fischlin and J.-P. Seifert, Tensor-based trapdoors for CVP and their application to public key cryptography (extended abstract), in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 1746, Springer, Berlin (1999), 244–257.

[20] E. Fujisaki and T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: *Advances in Cryptology—CRYPTO'99*, Lecture Notes in Comput. Sci. 1666, Springer, Berlin (1999), 537–554.

[21] E. Fujisaki and T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, *J. Cryptology* **26** (2013), no. 1, 80–101.

[22] N. Gama, A. Becker and A. Joux, Solving shortest and closest vector problems: The decomposition approach, Cryptology ePrint Archive (2013), https://eprint.iacr.org/2013/685.pdf.

[23] N. Gama, M. Izabachène, P. Q. Nguyen and X. Xie, Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems, in: *Advances in Cryptology—EUROCRYPT 2016. Part II*, Lecture Notes in Comput. Sci. 9666, Springer, Berlin (2016), 528–558.
[24] N. Gama and P. Q. Nguyen, Predicting lattice reduction, in: *Advances in Cryptology—EUROCRYPT 2008*, Lecture Notes in Comput. Sci. 4965, Springer, Berlin (2008), 31–51.
[25] F. Gandino, F. Lamberti, G. Paravati, J.-C. Bajard and P. Montuschi, An algorithmic and architectural study on Montgomery exponentiation in RNS, *IEEE Trans. Comput.* **61** (2012), no. 8, 1071–1083.
[26] C. Gentry, *A fully homomorphic encryption scheme*, PhD thesis, Stanford University, 2009.
[27] O. Goldreich, S. Goldwasser and S. Halevi, Public-key cryptosystems from lattice reduction problems, in: *Advances in Cryptology—CRYPTO '97*, Lecture Notes in Comput. Sci. 1294, Springer, Berlin (1997), 112–131.
[28] O. Goldreich, S. Goldwasser and S. Halevi, The GGH challenges.
[29] D. Goldstein and A. Mayer, On the equidistribution of Hecke points, *Forum Math.* **15** (2003), no. 2, 165–189.
[30] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University, London, 1938.
[31] J. Hoffstein, J. Pipher and J. H. Silverman, NTRU: A ring-based public key cryptosystem, in: *Algorithmic Number Theory*, Lecture Notes in Comput. Sci. 1423, Springer, Berlin (1998), 267–288.
[32] N. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer and W. Whyte, The impact of decryption failures on the security of NTRU encryption, in: *Advances in Cryptology—CRYPTO 2003*, Lecture Notes in Comput. Sci. 2729, Springer, Berlin (2003), 226–246.
[33] N. Howgrave-Graham, J. H. Silverman, A. Singer and W. Whyte, NAEP: Provable security in the presence of decryption failures, IACR Eprint archive (2003), http://eprint.iacr.org/2003/172.
[34] N. Howgrave-Graham, J. H. Silverman and W. Whyte, Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3, in: *Topics in Cryptology—CT-RSA 2005*, Lecture Notes in Comput. Sci. 3376, Springer, Berlin (2005), 118–135.
[35] R. Kannan, Minkowski's convex body theorem and integer programming, *Math. Oper. Res.* **12** (1987), no. 3, 415–440.
[36] R. Kannan and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8** (1979), no. 4, 499–507.
[37] S. Kim, On the distribution of lengths of short vectors in a random lattice, *Math. Z.* **282** (2016), no. 3–4, 1117–1126.
[38] P. Kirchner and P.-A. Fouque, An improved BKW algorithm for LWE with applications to cryptography and lattices, in: *Advances in Cryptology—CRYPTO 2015. Part I*, Lecture Notes in Comput. Sci. 9215, Springer, Heidelberg (2015), 43–62.
[39] E. Kirshanova, A. May and F. Wiemer, Parallel implementation of BDD enumeration for LWE, in: *Applied Cryptography and Network Security*, Lecture Notes in Comput. Sci. 9696, Springer, Cham (2016), 580–591.
[40] P. Klein, Finding the closest lattice vector when it's unusually close, in: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM, New York (2000), 937–941.
[41] K. Lauter, H. Chen and K. E. Stange, Attacks on search RLWE, Cryptology ePrint Archive (2015), http://eprint.iacr.org/2015/971.
[42] R. Lindner and C. Peikert, Better key sizes (and attacks) for LWE-based encryption, in: *Topics in Cryptology—CT-RSA 2011*, Lecture Notes in Comput. Sci. 6558, Springer, Heidelberg (2011), 319–339.
[43] M. Liu and P. Q. Nguyen, Solving BDD by enumeration: an update, in: *Topics in Cryptology—CT-RSA 2013*, Lecture Notes in Comput. Sci. 7779, Springer, Heidelberg (2013), 293–309.
[44] M. Liu, X. Wang, G. Xu and X. Zheng, Shortest lattice vectors in the presence of gaps, IACR Cryptology ePrint Archive (2011), https://eprint.iacr.org/2011/139.pdf.
[45] V. Lyubashevsky, Fiat-shamir with aborts: Applications to lattice and factoring-based signatures, in: *Advances in Cryptology—ASIACRYPT 2009*, Lecture Notes in Comput. Sci. 5912, Springer, Berlin (2009), 598–616.
[46] V. Lyubashevsky, Lattice signatures without trapdoors, in: *Advances in Cryptology—EUROCRYPT 2012*, Lecture Notes in Comput. Sci. 7237, Springer, Berlin (2012), 738–755.
[47] V. Lyubashevsky, Future directions in lattice cryptography, Public Key Cryptography 2016, invited talk.
[48] V. Lyubashevsky and D. Micciancio, On bounded distance decoding, unique shortest vectors, and the minimum distance problem, in: *Advances in Cryptology—CRYPTO 2009*, Lecture Notes in Comput. Sci. 5677, Springer, Berlin (2009), 577–594.
[49] V. Lyubashevsky, C. Peikert and O. Regev, On ideal lattices and learning with errors over rings, *J. ACM* **60** (2013), no. 6, Article ID 43.
[50] D. Micciancio, Improving lattice based cryptosystems using the Hermite normal form, in: *Cryptography and Lattices*, Lecture Notes in Comput. Sci. 2146, Springer, Berlin (2001), 126–145.
[51] D. Micciancio, Duality in lattice cryptography, Public Key Cryptography 2010, invited talk.
[52] D. Micciancio and B. Warinschi, A linear space algorithm for computing the hermite normal form, in: *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, ACM, New York (2001), 231–236.
[53] H. Minkowski, *Geometrie der Zahlen*, B. G. Teubner, Leipzig, 1896.
[54] D. Moody, Post-quantum cryptography: Nist's plan for the future, Public Key Cryptography 2016, Invited Talk.
[55] P. Nguyen, Cryptanalysis of the Goldreich–Goldwasser–Halevi cryptosystem from crypto'97, in: *Advances in Cryptology—CRYPTO' 99*, Lecture Notes in Comput. Sci. 1666, Springer, Berlin (1999), 288–304.

[56] P. Q. Nguyen and D. Pointcheval, Analysis and improvements of NTRU encryption paddings, in: *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Comput. Sci. 2442, Springer, Berlin (2002), 210–225.
[57] P. Q. Nguyen and I. E. Shparlinski, Counting co-cyclic lattices, preprint (2015), https://arxiv.org/abs/1505.06429.
[58] T. Okamoto and D. Pointcheval, React: Rapid enhanced-security asymmetric cryptosystem transform, in: *Topics in Cryptology—CT-RSA 2001*, Lecture Notes in Comput. Sci. 2020, Springer, Berlin (2001), 159–174.
[59] S.-H. Paeng, B. E. Jung and K.-C. Ha, A lattice based public key cryptosystem using polynomial representations, in: *Public Key Cryptography—PKC 2003*, Lecture Notes in Comput. Sci. 2567, Springer, Berlin (2003), 292–308.
[60] C. Pauderis and A. Storjohann, Computing the invariant structure of integer matrices: Fast algorithms into practice, in: *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ACM, New York (2013), 307–314.
[61] C. Peikert, An efficient and parallel Gaussian sampler for lattices, in: *Advances in Cryptology—CRYPTO 2010*, Lecture Notes in Comput. Sci. 6223, Springer, Berlin (2010), 80–97.
[62] C. Peikert, Lattice cryptography for the internet, in: *Post-Quantum Cryptography*, Lecture Notes in Comput. Sci. 8772, Springer, Berlin (2014), 197–219.
[63] C. Pernet and W. Stein, Fast computation of hermite normal forms of random integer matrices, *J. Number Theory* **130** (2010), no. 7, 1675–1683.
[64] T. Plantard and W. Susilo, Broadcast attacks against lattice-based cryptosystems, in: *Applied Cryptography and Network Security—ACNS 2009*, Lecture Notes in Comput. Sci. 5536, Springer, Berlin (2009), 456–472.
[65] T. Plantard, W. Susilo and Z. Zhang, LLL for ideal lattices: Re-evaluation of the security of Gentry–Halevi's fhe scheme, *Des. Codes Cryptogr.* **76** (2015), no. 2, 325–344.
[66] O. Regev, New lattice-based cryptographic constructions, *J. ACM* **51** (2004), no. 6, 899–942.
[67] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, *J. ACM* **56** (2009), no. 6, Article No. 34.
[68] M. Rose, T. Plantard and W. Susilo, Improving BDD cryptosystems in general lattices, in: *Information Security Practice and Experience*, Lecture Notes in Comput. Sci. 6672, Springer, Berlin (2011), 152–167.
[69] C. P. Schnorr, Average time fast svp and cvp algorithms for low density lattices and the factorization of integers, Technical report, Goethe Universität Frankfurt, 2010.
[70] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26** (1997), no. 5, 1484–1509.
[71] V. Shoup, Oaep reconsidered, in: *Advances in Cryptology—CRYPTO 2001*, Lecture Notes in Comput. Sci. 2139, Springer, Berlin (2001), 239–259.
[72] N. J. A. Sloane, Encrypting by random rotations, in: *Cryptography—EUROCRYPT'82*, Lecture Notes in Comput. Sci. 149, Springer, Berlin (1983), 71–128.
[73] A. Södergren, On the Poisson distribution of lengths of lattice vectors in a random lattice, *Math. Z.* **269** (2011), no. 3–4, 945–954.
[74] V. E. Tourloupis, *Hermite normal forms and its cryptographic applications*, Master thesis, University of Wollongong, 2013.
[75] P. Weiden, A. Hülsing, D. Cabarcas and J. Buchmann, Instantiating treeless signature schemes, Cryptology ePrint Archive (2013), https://eprint.iacr.org/2013/065.pdf.
[76] NIST, Nist kicks off effort to defend encrypted data from quantum computer threat, 28/04/2016.