Improving GGH Cryptosystem Using Generalized Low Density Lattices

Sarah Kamel[1,2], Mireille Sarkiss[1] and Ghaya Rekaya-Ben Othman[2]
[1] CEA, LIST, Communicating Systems Laboratory, BC 173, 91191 Gif-sur-Yvette, France
{sarah.kamel, mireille.sarkiss}@cea.fr
[2] Télécom ParisTech, 46 Rue Barrault, 75013, Paris, France
ghaya.rekaya@telecom-paristech.fr

***Abstract*—A new Goldreich-Goldwasser-Halevi (GGH) cryptosystem is proposed using Generalized Low Density (GLD) lattices. These low density lattices can alleviate a major drawback of the GGH scheme, namely the huge size of its public key. Indeed, we show that the new GGH that we propose in this paper reduces the key size by one order of magnitude. In addition, we show that the key generation complexity as well as those of the encryption and decryption phases are significantly decreased. The security of this new GGH is highlighted through a security analysis that reviews all known attacks on GGH systems. This allows us to conclude that our scheme does not add any new vulnerability as compared with the existing GGH schemes.**

***Index Terms*—Generalized low-density lattices, GGH cryptosystem, Lattice-based cryptography, Public-key cryptography.**

I. Introduction

The advent of quantum computers will undermine the current public-key cryptography paradigm. Indeed, most public-key algorithms will become vulnerable to private-key attacks, carried out through the implementation of quantum algorithms such as Shor's algorithm [1]. The cryptography community is foreseeing this paradigm change by researching into quantum-safe cryptographic primitives. Among these, lattice-based cryptosystems received wide attention and are today emerging as promising candidates to post-quantum cryptography [2].

At first, lattices were used in cryptanalysis as tools to break cryptographic schemes. In 1996, Ajtai's showed a connection between the average-case complexity and the worst-case complexity of some lattice problems [3]. This was the first step towards a different use of lattices in cryptography. From that point, lattices started to be more and more studied in the design of cryptographic systems. Basing on some hard-to-solve lattice problems, lattice-based public-key cryptosystems were designed. Ajtai-Dwork (AD) cryptosystem [4] was the first such lattice-based scheme to be proposed. Its security is related to the hardness of the shortest vector problem (SVP). SVP refers to the computation of a vector $\mathbf{u}$ in a lattice Λ, defined by a basis B, with minimum non-zero Euclidean length $\min\{\ \|\mathbf{u}\|\ ;\ \mathbf{u} \in \Lambda, \mathbf{u} \neq \mathbf{0}\}$. The AD system inspired many cryptosystems, among which GGH [5] and NTRU [6]. As compared with the theoretical AD cryptosystem, GGH emphasized practicality. GGH can be seen as the lattice-based analog of the McEliece cryptosystem, which uses linear codes for encryption [7]. GGH security relies on the hardness of the closest vector problem (CVP) in a lattice, which consists in finding the vector $\mathbf{u} \in \Lambda$ that minimizes the distance to a given vector $\mathbf{v} \in \mathbb{R}^n$. NTRU is a ring-based public-key cryptosystem whose encryption functions are designed using convolution polynomial rings and elementary probability theory. It can be interpreted as SVP or CVP instance. In 2005, Regev introduced the learning with errors (LWE) problem and proposed an LWE-based public-key cryptosystem [8]. This latter can be viewed as a decoding problem from a random linear code.

In this paper, we introduce a new GGH cryptosystem that is based on generalized low density (GLD) lattices. These lattices were recently introduced in coding theory as low density lattices that can achieve the capacity of the Gaussian channel. We analyze the security and complexity of the resulting GGH scheme. We show that our cryptosystem significantly reduces the GGH public key size. Our cryptosystem also reduces the key generation and the decryption complexities. More importantly, we showed that these complexity improvements do no induce any security weaknesses.

The remainder of this paper is organized as follows: in Section II, we review the GGH cryptosystem and some of its improvements. In Section III, we describe our proposed public-key cryptography scheme based on Generalized low density lattices. In Section IV, we carry out a security analysis on this scheme and in Section V, we analyze its complexity. Finally, Section VI gives concluding comments.

II. GGH scheme and its improvements

In the original GGH cryptosystem [5], the private key R is chosen as a good basis of an $n-$dimensional random lattice Λ. R is an $n \times n$ non-singular integer matrix defined by

$$R = \sqrt{n}I + Q, \qquad (1)$$

where I is the identity matrix and Q is a random matrix with elements uniformly chosen from the set $\{-4, \ldots, 4\}$. The public key B is a bad basis of the same lattice Λ, hence, it is also an $n \times n$ integer matrix. B is generated by transforming the good basis of the lattice into a bad one. It can be obtained by applying some elementary linear combinations on the basis vectors of the private key R. In order to encrypt a message $\mathbf{m} \in \mathbb{Z}^n$, $\mathbf{m}$ is multiplied by the public basis B generating a lattice point $\mathbf{x}$. Then, $\mathbf{x}$ is secured by adding an n-dimensional error vector $\mathbf{e}$ chosen uniformly from the set $\{-\sigma, \sigma\}^n$. The ciphertext $\mathbf{c}$ is given by

$$\begin{aligned}\mathbf{c} &= \mathbf{m}B + \mathbf{e} \\ &= \mathbf{x} + \mathbf{e}.\end{aligned} \tag{2}$$

The ciphertext $\mathbf{c}$ can be decrypted and the message $\mathbf{m}$ recovered by computing

$$\mathbf{m} = \lfloor \mathbf{c}R^{-1} \rceil RB^{-1}. \tag{3}$$

The major flaw in this scheme is the particular form of the noise which makes the system vulnerable and hence, easily breakable by the embedding attack [9]. In fact, Nguyen showed in [9] that the CVP decryption is reduced to a simpler CVP instance for which the noise vector is much smaller $\mathbf{e}' \in \{-1/2, 1/2\}$. He concluded that GGH requires working in high lattice dimensions to be secure. However, for high dimensions, the GGH has a huge public key size yielding an impractical cryptosystem.

In order to overcome original GGH drawbacks, many GGH-based improvements were proposed. These works, on one side, focused on reducing the public key size resulting in new designs of public lattice bases. On the other side, they aimed at improving GGH security by proposing new methods of noise generation.

For instance, Micciancio applied some changes to both aspects, public key generation and noise vector choice [10]. He considered two encoding methods. In the first method, the message is encoded in the lattice point and the noise vector is chosen uniformly from an interval $[-\sigma, \sigma]$ whereas in the second one, the lattice point is chosen at random and the message is embedded in the error vector. This uniform noise provides additional robustness against Nguyen's embedding attack [9]. For the public key, Micciancio proposed to use the Hermite normal form (HNF) lattice basis. In fact, an HNF basis M is a lower triangular matrix verifying

- $m_{i,j} = 0 \;\; \forall\, i < j$,
- $m_{i,i} > 0 \;\; \forall\, i$,
- $0 \leq m_{i,j} < m_{j,j} \;\; \forall\, i > j$.

Micciancio proved that this HNF basis requires less storage for the key while it guarantees, at least, the same security level as the original GGH [10]. Indeed, the public key size is reduced from $O(n^3 log_2(n))$ to $O(n^2 log_2(n))$. In addition, he presumed that his system is secure for lattice dimension higher than 500. The security and efficiency of Micciancio's cryptosystem were analyzed by Ludwig in [11]. Ludwig found that the minimal dimension for which the system is secure in practice is 782. He highlighted the major disadvantages of Micciancio's system. On one hand, the decryption is slow and suffers from instability since it employs Babai's nearest plane algorithm. On the other hand, the public key generation process has high running time for the proposed lattice dimensions due to the non-existence of low-complexity algorithms for computing the HNF matrix. Moreover, in Ludwig's results [11], the public keys turned out to have larger size than in Micciancio's experiments. All these findings [11] led to conclude that Micciancio's cryptosystem is far from being practical.

Based on Micciancio's system, [12] focused on proposing a larger noise vector $\mathbf{e}$ to obtain a harder CVP problem. In fact, the original GGH defines $\mathbf{e}$ such that its product by the inverse of the private key generates a null vector, i.e. $\lfloor \mathbf{e}R^{-1} \rceil = \mathbf{0}$. In [12], the error vector $\mathbf{e}$ has $n-k$ coordinates chosen from a set of integers $I_1 = \{-s, \ldots, -1, 1, \ldots s\}$ and the remaining k from a second set of integers $I_2 = \{-h, h\}$ with $h > s$. In this way, the product $\lfloor \mathbf{e}R^{-1} \rceil$ results in k zero elements and $(n-k)$ elements with value $\{\pm 1\}$.

$$\lfloor \mathbf{e}R^{-1} \rceil = (v_1, \ldots, v_n),$$

$$v_i = \begin{cases} 0 & \text{for } \; n-k \; \text{ values of } \; i, \\ \pm 1 & \text{for } \; k \; \text{ values of } \; i. \end{cases} \tag{4}$$

Recently, Hooshmand [13] suggested working with low density lattice codes (LDLC) [14] instead of random lattices as used in the previous GGH systems. In their scheme, the noise $\mathbf{e}$ follows a Gaussian distribution with zero mean and variance σ^2 upper bounded by the Poltyrev limit [15]. For the generation of the public key, they considered the HNF of the LDLC generator matrix following Micciancio's proposition. They claimed to have the same upper bound on the public key size which is $O(n^2 \log_2(n))$ and to reduce the decryption complexity with LDLC iterative decoding algorithm.

III. GLD LATTICE-BASED GGH CRYPTOSYSTEM

A. GLD lattices

GLD lattices were proposed by Boutros *et al.* in [16]. They are integer lattices in $\mathbb{Z}^n$ and have sparse parity-check matrices. These properties enable low complexity iterative decoding, hence allowing the codes to be used in dimensions up to 1 million. It was shown in [17] that these lattices achieve asymptotically Poltyrev limit [15].

GLD lattices can be generated using construction A from linear GLD codes $C_{GLD}[n,k]$ as follows

$$\Lambda = C_{GLD} + p\mathbb{Z}^n, \tag{5}$$

where n is the lattice dimension and p is a prime number.

This method consists in considering a small linear code C_0 of length n_0 and dimension k_0. C_0 is called the elementary code and it is defined over the finite field $\mathbb{F}_p$. Denote by H_{C_0} the parity-check matrix of C_0. A second code C_1 is generated as the direct sum of L copies of C_0

$$C_1 = C_0^{\oplus L}. \tag{6}$$

The parity-check matrix of C_1 is given by

$$H_{C_1} = \begin{bmatrix} H_{C_0} & 0 & \ldots & 0 \\ 0 & H_{C_0} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & H_{C_0} \end{bmatrix}. \tag{7}$$

Now, let $\pi_1 = \text{id}$ and $\pi_2, \ldots, \pi_J$ be J permutations of $\{1, 2, \ldots, n\}$. A GLD code $C_{GLD}[n,k]$ is defined as

$$C_{GLD} = \bigcap_{j=1}^{J} \pi_j(C_1) = \bigcap_{j=1}^{J} \pi_j(C_0^{\oplus L}), \tag{8}$$

where $\pi_j(x_1, x_2, \ldots, x_n) = (x_{\pi_j(1)}, x_{\pi_j(2)}, \ldots, x_{\pi_j(n)})$.

The matrix H_C of the GLD code is a non-square matrix obtained as

$$H_C = \begin{bmatrix} H_{C_1} \\ \pi_2(H_{C_1}) \\ \vdots \\ \pi_J(H_{C_1}) \end{bmatrix}. \quad (9)$$

Note that H_C has $n - k = JL(n_0 - k_0)$ rows and $n = Ln_0$ columns.

In order to compute the generator matrix G_C of the GLD code, the systematic form of the parity-check matrix H_C is computed using the Gaussian elimination algorithm. $H_{C,syst}$ has the following form

$$H_{C,syst} = [\ I \mid -B^t\ ]. \quad (10)$$

Then, the $k \times n$ generator matrix of the code is defined by

$$G_C = [\ B \mid I\ ], \quad (11)$$

Based on construction A (5), the generator matrix of Λ is

$$G_\Lambda = \begin{bmatrix} pI & 0 \\ B & I \end{bmatrix}. \quad (12)$$

B is a dense matrix which elements belong to the finite field $\mathbb{F}_p$. We can see that G_Λ is a lower triangular matrix. Its column elements are modulo the diagonal elements. Therefore, G_Λ is in Hermite normal form.

B. Proposed public-key scheme

We propose to use GLD lattices to design an improved version of GGH cryptosystem. Our scheme is described as follows:

- The private key is the parity-check matrix H_C of a GLD code.
- The public key is the generator matrix G_Λ of the GLD lattice which is an HNF matrix.
- To encrypt a message $\mathbf{m} \in \mathbb{Z}^n$, $\mathbf{m}$ is multiplied by the public key G_Λ and an error vector $\mathbf{e}$ is added to obtain the ciphertext

  $$\mathbf{c} = \mathbf{m}G_\Lambda + \mathbf{e}. \quad (13)$$

  The noise vector $\mathbf{e}$ is a random noise chosen uniformly from an interval $[-\sigma, \sigma]$ as in Micciancio scheme [10].
- To decrypt $\mathbf{c}$ and recover the message $\mathbf{m}$, the GLD iterative decoding is applied.

Since Micciancio cryptosystem was shown to be secure in dimension higher than 782, we will consider GLD lattices with dimension $n \approx 1000$ in our study. We will analyze the security and complexity of the proposed GLD lattice-based GGH cryptosystem in the sequel.

IV. Security analysis

In this section, the security of our proposed scheme is studied from two perspectives. First, we investigate the complexity of the exhaustive attack to recover the private key. Then, we analyze the decoding attacks to decrypt the transmitted ciphertext. Finally, we discuss the dual code attacks to find the private matrix.

A. Brute-force attack

In brute-force attack, the eavesdropper attempts to find the private key by exhaustively testing all the possibilities. Hence, the complexity of this attack depends on the number of private keys that are likely to be used. In our case, the search space of the private key depends on the number of possible matrices H_C. Recall that H_C is generated by the concatenation of $H_{C_0^{\oplus L}}$ with $(J - 1)$ of its permuted versions. The number of possible versions is $n!$ since they are generated by permuting the n columns of $H_{C_0^{\oplus L}}$. Therefore, the search space for the private key is

$$n! \times (J - 1). \quad (14)$$

The formula is dominated by the term $n!$, which can be approximated using Stirling's formula into

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n. \quad (15)$$

Since it is exponential in n, then for $n \geq 1000$, the search space becomes huge making this attack unfeasible.

B. Decoding attacks

In the decoding attacks, the eavesdropper aims at decrypting the ciphertext which is equivalent to finding the closest lattice vector to this ciphertext. In fact, the GGH system is essentially a closest vector problem instance which makes the decoding attacks the most obvious attacks on such cryptosystems. The first approaches to attack GGH applied Babai's round-off and nearest plane algorithms [18].

Round-off attack: The round-off attack [18] consists in multiplying the ciphertext c by the inverse of the public basis G_Λ^{-1} which results in a noisy message as follows:

$$\mathbf{c}G_\Lambda^{-1} = \mathbf{m} + \mathbf{e}G_\Lambda^{-1}. \quad (16)$$

By rounding this result, the message $\mathbf{m}$ can be found only if $\lfloor \mathbf{e}G_\Lambda^{-1} \rceil = 0$. Thus, the feasibility of this attack depends mainly on the orthogonality of the public basis G_Λ and the variance of the noise $\mathbf{e}$.

Nearest plane attack: The nearest plane attack [18] is an improvement of the round-off attack that uses a better approximation for the CVP. The idea is to find the nearest lattice point by considering one dimension after the other. For every basis vector $\mathbf{g}_i$, $i = 1, \ldots, n$, the nearest plane algorithm finds the closest hyperplane to the ciphertext $\mathbf{c}$. It starts by finding the integer multiple k_n of $\mathbf{g}_n$ that minimizes the distance to the hyperplane spanned by the basis vectors $\{\mathbf{g}_1, \ldots, \mathbf{g}_{n-1}\}$ by projecting $\mathbf{c} - k_n\mathbf{g}_n$, $k_n \in \mathbb{Z}$, onto this $(n-1)$-dimensional hyperplane. Then, the algorithm proceeds recursively until dimension 1. Finally, the lattice vector is obtained by the sum of all vectors $k_i\mathbf{g}_i$.

Note that both round-off and nearest plane attacks can be seen as Zero-forcing (ZF) and ZF-decision feedback equalizer (ZF-DFE) decoders, respectively. These decoders are known to be sub-optimal. Though they have low complexity, they are prone to error propagation that can degrade considerably the performance for high dimensions. In order to improve these decoders, Lenstra-Lenstra-Lovász (LLL) and Block Korkine-Zolotarev (BKZ) basis reduction techniques can be applied before running the decoding algorithms.

Security discussion: Lattice reduction approaches are applied to generate good basis with short, nearly orthogonal vectors. They can reduce the decoding errors and thus improve the decryption. In high dimensions, these techniques need a huge running time to provide a shorter basis. Yet, they may fail to produce this basis rendering the decryption unfeasible.

Indeed, the LLL algorithm succeeds to generate a reduced basis until dimension 200. For higher dimensions, BKZ, which is an improved variant of LLL, needs to be used. While LLL operates on each basis vector alone, BKZ works on blocks of t basis vectors. The complexity of BKZ algorithm is $O(n^3t^{t+o(t)} + n^4)$ [11]. For small t, BKZ will not result in a reduced basis for high dimensions. Increasing t will increase the complexity and cannot guarantee a good basis. In our scheme, we consider lattices of dimension $n \approx 1000$. For this dimension, t should be sufficiently large to increase the probability of obtaining a good basis making the BKZ reduction algorithm impractical.

The success of these decoding attacks depends on two factors: the lattice public basis and the noise vector.

a - Public basis: In fact, after applying the reduction algorithms, if the resulting basis is quasi-orthogonal these attacks will be able to decrypt the ciphertext and recover the message. Conversely, if the reduction fails, the attacks have to be applied on a bad basis, and thus, the decryption will certainly fail. Therefore, in order to strengthen the system's security, the public key should be a lattice basis as bad as possible with large dimension in order to prevent reduction success. In our scheme, the generator matrix G_Λ is in HNF. To evaluate its orthogonality, we compute the orthogonality defect of G_Λ, which is the product of the basis vector lengths divided by the matrix determinant, defined by [5]

$$\mathrm{orth-defect}(G_\Lambda) = \frac{\prod_{i=1}^{n} \|g_i\|}{|det(G_\Lambda)|}, \tag{17}$$

where $\|g_i\|$ is the Euclidean norm of the i'th row in G_Λ. Since G_Λ is given by construction A as in (12), the first $(n-k)$ rows have norm p and the determinant is equal to p^{n-k}, then

$$\begin{aligned}\mathrm{orth-defect}(G_\Lambda) &= \frac{p^{n-k}\prod_{i=n-k+1}^{n} \|g_i\|}{p^{n-k}} \\ &= \prod_{i=n-k+1}^{n} \sqrt{1+\|b_i\|^2},\end{aligned}$$

where $\|b_i\|$ is the Euclidean norm of the i'th row in B. B is a dense matrix with elements in the finite field $\mathbb{F}_p$. Thus, $\|b_i\|^2 \gg 1 \;\Rightarrow\; \|g_i\| \gg \sqrt{2}$, yielding an orthogonality defect $\mathrm{orth-defect}(G_\Lambda) \gg (\sqrt{2})^k \gg 1$. Hence, G_Λ is a bad basis. This result can be confirmed by simulating different lattice generator matrices corresponding to different elementary codes C_0 as shown in table I.

b - Noise vector: The second factor that can lead to decryption failure is the added noise vector $\mathbf{e}$. If $\mathbf{e}$ is small, the noisy point will not be translated far from the lattice point which makes the decoding easier. Many methods were proposed to define the error vector and improve the security of the original GGH by defeating its attacks. In [10], Micciancio proposed to use a random noise uniformly chosen from the interval $[-\sigma, \sigma]$. In [13], Hooshmand suggested to choose a

TABLE I. AVERAGE VALUE OF THE NORM OF g_i FOR LATTICE DIMENSION $n \approx 1000$.

C_0	n	k	$\|g_i\|$
$C_0[3,2]_{11}$	999	333	40.06
$C_0[3,2]_{13}$	999	333	47.3
$C_0[4,3]_{11}$	1000	333	31.6
$C_0[8,6]_{11}$	1000	500	76.4
$C_0[16,12]_{17}$	992	496	135.6

Gaussian noise $\mathcal{N}(0, \sigma^2)$, where σ^2 is bounded by Poltyrev limit $\sigma^2 < 1/(2\pi e)$. In [12], a larger noise vector was generated by selecting its elements from two different sets as described in (4).

In our GLD lattice-based scheme, we consider a uniform noise in $[-\sigma, \sigma]$. The value of σ should not exceed the maximal noise variance for which the iterative decoder succeeds. At the same time, it should be large enough to prevent the attacker from decrypting the message. The value of σ tolerated by our system depends on the parameters of the elementary code C_0. Table II shows the maximal noise variance for different elementary codes C_0. These σ values are sufficient to ensure the failure of the decoding attack in dimension $n = 1000$.

TABLE II. NOISE VARIANCE σ FOR LATTICE DIMENSION $n \approx 1000$.

C_0	p	σ
$C_0[3,2]$	11	1
$C_0[3,2]$	17	2
$C_0[3,2]$	29	3
$C_0[8,6]$	17	1
$C_0[8,6]$	53	2

C. Dual code attacks

Dual code attacks are dangerous attacks applied on code-based cryptography systems using low density parity-check (LDPC) codes. These attacks exploit the low density property of the LDPC parity-check matrix H that corresponds to the generator matrix of the dual code. The main idea is to search for low weight codewords belonging to the dual code and recover H. One solution to avoid such attacks consists in increasing the density of the parity check-matrices. In this case, the resulting codes are called moderate density parity-check (MDPC) codes [19].

Similarly, the robustness of the GLD lattice-based scheme against these attacks depends on the density of the lattice. This parameter can be managed by the selection of C_0. To improve the security of our scheme, the density of the considered lattices should be increased by taking a code C_0 of higher length n_0 and working in a larger finite field $\mathbb{F}_p$. At the same time, this density should not exceed a given value which enables successful iterative decoding.

V. COMPLEXITY ANALYSIS

For the complexity analysis, we will examine three key features, namely the public key size, the public key generation and the decryption.

A. Key size

Recall that the GLD lattice generator matrix has the following form

$$G_\Lambda = \left[\begin{array}{cc} pI & 0 \\ B & I \end{array} \right].$$

We can notice that only part B of G_Λ needs to be stored. It is a $k \times (n-k)$ matrix with elements smaller than p by construction. Hence, the space needed to store this matrix is

$$k \times (n-k) \times \log_2(p) \text{ bits.} \tag{18}$$

We notice that, for a fixed lattice dimension, the key size depends on the elementary code's parameters and the finite field over which it is defined. In order to analyze the GLD public key complexity, we consider some examples of GLD lattices Λ of dimension $n = 1000$ and we compute their key size. As a first example, let Λ be generated from the non-binary elementary code C_0 of length $n_0 = 3$ and dimension $k_0 = 2$ defined over the field $\mathbb{F}_{11}$. This elementary lattice is given by

$$\Lambda_0 = [3, 2]_{11} + 11\mathbb{Z}^3. \tag{19}$$

Then, the generated GLD lattice Λ is of dimension $n = n_0 L = 3L$. The matrix B in this case has $k = 333$ rows and $n - k = 666$ columns. Thus, the necessary space to store B is $333 \times 666 \times \log_2(11) = 93.7$ KBs.

In table III, we compute the key size for other examples of elementary codes.

TABLE III. PUBLIC KEY SIZE FOR LATTICE DIMENSION $n \approx 1000$.

C_0	n	Key size (KBs)
$C_0[3,2]_{13}$	999	100.18
$C_0[4,3]_{11}$	1000	105.57
$C_0[8,6]_{11}$	1000	105.57
$C_0[16,12]_{17}$	992	122.75

We can see that for the considered lattice dimension, the key size is always around 100 KBs.

For the previously described GGH based schemes, upper bounds were derived on the key size of the original GGH and Micciancio cryptosystems [10]. Indeed, the size of the GGH public key is upper bounded by $O(n^3 \log_2(n))$ bits. In Micciancio's cryptosystem, the public basis B is in HNF form and can be represented using $O(n^2 \log_2(n))$ bits.

In [13], the author claimed to have the same upper bound of $O(n^2 \log_2(n))$ bits as they considered an HNF matrix for their public key. However, this later bound was computed by Micciancio based on his specific choice of private key which is not the same in [13]. In addition, Micciancio's cryptosystem works with integer lattices while in [13] the used LDLCs are real lattices. In this case, the public key size depends on the precision used to represent the real elements. Therefore, the bound computed by Micciancio does not hold for the LDLC-based system.

The experimental results in [13] showed that the public key size in dimension $n = 1000$ is 43.6 MBs for GGH and 1 MB for LDLC-based systems without specifying the precision chosen for their real numbers. Whereas, Micciancio's system needs 1 MB to represent the public key in dimension $n = 800$ [11]. In our scheme, the key can be represented in dimension $n \approx 1000$ by approximately 100 KBs. Therefore, we can clearly state that the GLD lattice-based cryptosystem reduces at least by a factor of 10 the size of the public key.

B. Key generation

In the proposed GLD lattice-based cryptosystem, the public key is the HNF generator matrix of the GLD lattice. In order to compute this matrix, we apply the Gaussian elimination algorithm to represent the GLD code's parity-check matrix in its systematic form $H_{C,syst} = [\ I \mid -B^t\ ]$. Therefore, the key generation complexity is mainly due to the Gaussian elimination algorithm. This time complexity is in the order of $O(nm^{\omega-1})$ for an $m \times n$ matrix with $m \leq n$ and $\omega < 2.38$ the exponent of matrix multiplication [20]. In our case, this reduces to $O(n \times (n-k)^{\omega-1}) < O(n^3)$.

In Micciancio's system [10], the main problem is the complexity of the HNF algorithm that generates the public key. In fact, there exist different algorithms for generating HNF matrices. The most basic one perfoms a polynomial number of arithmetic operations but has exponential space complexity due to the exponential increase of its matrix elements during the execution. Many improvements were proposed aiming to reduce the space complexity but resulted in increasing the running time. The most space efficient algorithm was designed by Micciancio in [21] with running time $O(n^5 \log_2(M))$ where M is the bound on the matrix entries. Therefore, it can be concluded that the public key generation in our case is at least $O(n^2)$ times faster than Micciancio system.

Moreover, in [13], the LDLC-based cryptosystem used real lattices. However, HNF algorithms cannot operate on real matrices and it is needed to build LDLC which are rational. Before proceeding to HNF, the rational matrix should be transformed into integer matrix by multiplying its elements by the common denominator. This processing increases the running time needed to generate the public key. Hence, for the LDLC-based proposition, generating public key increases the complexity compared to Micciancio system, and hence compared to our proposed scheme.

C. Decryption

As mentioned in [17], GLD lattices are suitable for iterative decoding. In fact, the parity-check matrices H_C of the GLD code can be associated with a Tanner graph with n variable nodes and LJ check nodes. A variable node $x_j, j = 1, \ldots, n$, represents a lattice coordinate and a check node represents one copy of the elementary code C_0 or equivalently the elementary lattice Λ_0. An edge connects a variable node x_j and a check node C_0 if the corresponding position in the parity check matrix H_C has a non zero element. Since in GLD construction [16] all variable nodes have degree $J = 2$, the graph can be converted into a simple generalized Tanner graph with L check nodes on the right and L check nodes on the left each with degree n_0. The total number of edges $n = n_0 L$ represents the lattice dimension thus, one lattice coordinate is assigned to one graph edge.

Iterative decoding of GLD lattices is done via message passing along edges of the generalized Tanner graph. Messages are computed locally by a check node using a soft-input soft-output decoder. This soft decoding is made via the forward-backward algorithm on the syndrome trellis that has $p^{(n_0-k_0+1)}$ transitions where $(n_0 - k_0 + 1)$ is small [22]. For instance, for the codes C_0 in table I, its maximal value is 5. Then, these messages are sent to the n_0 neighboring check nodes. The complexity of iterative message passing is linear in the lattice dimension n and is given by $O(n \cdot t \cdot p^{(n_0-k_0+1)})$ where t is the number of iterations.

The LDLCs are also suitable for iterative decoding that is linear in the dimension n but the constant term multiplying n in $O(\cdot)$ is higher [14]. In fact, in this decoder, the messages computed and passed by the check nodes are continuous Gaussian probability distribution functions (pdfs). These functions are sampled and quantized into discrete vectors which increases the complexity of the decoding. In addition, the number of pdfs grows exponentially with the number of iterations. In order to limit this number, a Gaussian mixture reduction procedure is applied at each iteration, which increases the decryption complexity. Therefore, this decoding results in high computational complexity and large storage requirements. Unlike LDLC decoding, GLD iterative decoding exchanges a fixed number p of discrete messages regardless of the number of iterations. Therefore, we can conclude that GLD iterative decoding is less complex than LDLC decoding.

Moreover, Micciancio applied Babai's nearest plane algorithm. It is a sub-optimal algorithm used to compute the closest lattice vector to the noisy ciphertext. This algorithm should be repeated my times to return the closest lattice point. In order to decrypt correctly without running the algorithm for a long time, the precision used to represent real numbers should be high but this induces large storage requirements. Ludwig showed by simulations that this decryption procedure is very slow in high dimensions [11]. Conversely, GLD iterative decoding provides a performance close to the optimal decoding while keeping the complexity at a very low level.

VI. Conclusion

In this paper, we introduced a new GGH cryptosystem based on GLD lattices [16]. While guarantying the same level of security, our cryptosystem significantly reduces the complexity compared with previous GGH schemes. A large reduction in the key size is induced by the HNF form of the public key G_Λ. The public key in dimension $n \approx 1000$ is represented using 100 KBs approximately, which is one order of magnitude smaller than the public keys of the existing GGH schemes. In addition, the public key is in HNF form by design; hence, no HNF algorithms are applied, which reduces the complexity of the key generation phase by $O(n^2)$. Finally, the iterative decoding of the GLD lattices offers performances close to the optimal decoding algorithms while operating at low complexity. Next steps will consist in investigating other noise models: while uniform noise was considered in this paper, further work will be to define the best noise distribution and find the optimal value of σ in our lattice-based system, through in-depth theoretical and experimental studies of the error vector.

References

[1] P.W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, Nov. 1994.

[2] D. J. Bernstein, "Introduction to post-quantum cryptography," *Post-quantum cryptography (Springer)*, pp. 1–14, 2009.

[3] M. Ajtai, "Generating hard instances of lattice problems," *Proceedings of 28th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 99–108, 1996.

[4] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," *Proceedings of 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 284–293, 1997.

[5] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," *Proceedings of Crypto97, LNCS*, vol. 1294, pp. 112–131, 1997.

[6] J. Hoffstein, J. Pipher, and J.H. Silverman, "NTRU: a ring based public key cryptosystem," *Proceedings of ANTS III, LNCS*, vol. 1423, pp. 267–288, 1998.

[7] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Jet Propulsion Laboratory DSN Progress Report*, pp. 42–44, 1978.

[8] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Proceedings of 37th ACM Symposium on Theory of Computing (STOC)*, pp. 84–93, 2005.

[9] P. Q. Nguyen, "Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto'97," *Proceedings of Crypto99, LNCS*, vol. 1666, pp. 288–304, 1999.

[10] D. Micciancio, "Improving lattice based cryptosystems using the Hermite normal form," *Proceedings Cryptography and Lattices Conference*, pp. 126–145, 2001.

[11] C. Ludwig, "The Security and Efficiency of Micciancios Cryptosystem," 2004, technical Report. Available at http://www.cdc.informatik.tu-darmstadt.de/reports/TR/TI-02-07.MiccPaper.pdf.

[12] M. Yoshino and N. Kunihiro, "Improving GGH cryptosystem for large error vector," *International Symposium on Information Theory and its Applications ISITA*, pp. 416–420, Honolulu, Oct. 2012.

[13] R. Hooshmand, "Improving GGH public key scheme using low density lattice codes," 2015, [Online]. Available: http://eprint.iacr.org/2015/229, http://arxiv.org/abs/1503.03292.

[14] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1561–1585, Apr. 2008.

[15] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 409–417, Mar. 1994.

[16] J.J. Boutros, N. di Pietro, and N. Basha, "Generalized low-density (GLD) lattices," *Proceedings of the 2014 IEEE Information Theory Workshop*, pp. 15–19, Hobart, Nov. 2014.

[17] N. di Pietro, N. Basha and J. J. Boutros, "Non-binary GLD codes and their lattices," *Proceedings of the 2015 IEEE Information Theory Workshop*, pp. 1–5, Jerusalem, Apr. 2015.

[18] L. Babai, "On Lovaśz' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, Mar. 1986.

[19] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2069–2073, Istanbul, Jul. 2013.

[20] C.-P. Jeannerod, C. Pernet, and A. Storjohann, "Rank-profile revealing Gaussian elimination and the CUP matrix decomposition," *Journal of Symbolic Computation*, vol. 56, pp. 46–68, Sept. 2013.

[21] D. Micciancio and B. Warinschi, "A linear space algorithm for computing the Hermite Normal Form," *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pp. 231–236, 2001.

[22] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding for linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.