Generalized Low Density Lattices for GGH Cryptosystem

Sarah Kamel, Mireille Sarkiss
CEA, LIST, Communicating Systems Laboratory
Gif-sur-Yvette, France
e-mail: {sarah.kamel, mireille.sarkiss}@cea.fr

Ghaya Rekaya-Ben Othman
Telecom ParisTech
Paris, France
e-mail: ghaya.rekaya@telecom-paristech.fr

***Abstract*—Goldreich-Goldwasser-Halevi (GGH) is a lattice-based public-key cryptosystem which has been widely developed due to its simple encryption and decryption procedures. Despite many existing GGH improvements, its huge public key size remains its main drawback, which prevents the system from being used in practice. In order to reduce the key size, we propose in this paper a new GGH cryptosystem using Generalized low density (GLD) lattices. For this proposed GGH lattice-based scheme, we provide a security analysis by considering all known attacks on GGH systems. We conclude that our scheme is as secure as the existing GGH schemes. We also investigate the complexity of our cryptosystem and prove that GLD lattices reduce significantly the key size and the complexity of the key generation and decryption phases compared to other GGH systems.**

Keywords-generalized low-density lattices; GGH cryptosystem; lattice-based cryptography; public-key cryptography

I. INTRODUCTION

Most of the public-key cryptography schemes used today can be broken by quantum computers using Shor's algorithm [1]. This result motivated many cryptographers to focus on the design of post-quantum cryptography in order to prepare for the time when powerful quantum computers will exist. Lately, lattice-based cryptosystems received wide attention from the cryptography community and are emerging now as promising candidates to post-quantum cryptography [2].

Lattice-based cryptography systems are public-key systems based on some hard lattice problems. Lattices were first used in cryptanalysis to break various cryptographic schemes. The first lattice-based scheme was the Ajtai-Dwork (AD) cryptosystem in 1997 [3]. Its security is related to the hardness of the shortest vector problem (SVP), which 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 GGH scheme was then proposed in [4] as a much more practical lattice-based alternative than the theoretical AD cryptosystem. GGH encryption relies on the hardness of the closest vector problem (CVP) in a lattice, which consists in finding the lattice vector $\mathbf{u}\in\Lambda$ minimizing the distance to a given vector $\mathbf{v}\in\mathbb{R}^n$. Another cryptosystem, NTRU was presented as a ring-based public-key cryptosystem that uses convolution polynomial rings and elementary probability theory to design encryption functions [5]. 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 that can be viewed as a decoding problem from a random linear code [6].

Among the lattice-based cryptography schemes, only GGH works explicitly with lattices. Thus, in the sequel, we consider lattice-based GGH cryptosystem. The main contribution of our work is the investigation of new lattices for the encryption and the analysis of the security and the complexity of the proposed lattice-based cryptosystem. Our most interesting result is the significant reduction of the GGH public key size. In addition, the key generation and the decryption complexities are considerably reduced. We show that these improvements in complexity do not 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 the proposed public-key cryptography scheme based on Generalized low density lattices. The security and complexity of this scheme are then analyzed in Section IV and V respectively and experimental results are provided. Finally, Section VI gives concluding comments.

II. GGH SCHEME AND ITS IMPROVEMENTS

In the original GGH cryptosystem [4], the private key R is chosen as a good basis of an n-dimensional random lattice Λ. R is an n×n non-singular integer matrix defined by

$$R = \sqrt{n}\, I + Q \tag{1}$$

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×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

$$\mathbf{c} = \mathbf{m}B + \mathbf{e} = \mathbf{x} + \mathbf{e} \tag{2}$$

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 [7]. This 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. In [8], Micciancio applied some changes to the public key generation and the noise vector choice. 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 [7]. For the public generation, Micciancio proposed to use the Hermite normal form (HNF) lattice basis as the public key for encryption. In fact, an HNF basis M is a lower triangular matrix with the column elements smaller than the diagonal elements. Its elements verify the following conditions: $m_{i,j} = 0\ \forall i < j$, $m_{i,i} > 0\ \forall i$ and $0 \le m_{i,j} < m_{j,j}\ \forall i > j$. Micciancio proved that this HNF basis reduces key storage size from $O(n^3\log_2(n))$ to $O(n^2\log_2(n))$ while guaranteeing, at least, the same security level as the original GGH [8]. The security and efficiency of Micciancio's cryptosystem were analyzed in [9] showing that the system is secure in practice for a minimal lattice dimension of 782 compared to 500 as presumed by Miccianco dimension. The paper also concluded that Micciancio's cryptosystem is far from being practical due to, on one hand, its slow and instable decryption, which employs Babai's nearest plane algorithm, and on the other hand, the high running time of the public key generation, which is related to the complexity of the HNF matrix computation in high lattice dimensions.

Based on Micciancio's system, [10] focused on proposing a larger noise vector **e** to harden the CVP problem. This was obtained by choosing n – k coordinates of the error vector **e** from two sets of integers in a way to result in a product $\lfloor \mathbf{e}R^{-1} \rceil$ with k zero elements and n – k non-zero elements having value $\{\pm 1\}$.

Recently, low density lattice codes (LDLC) [11] were suggested instead of random lattices as used in the previous GGH systems. In this scheme, the noise **e** follows a Gaussian distribution with zero mean and variance σ^2 upper bounded by the Poltyrev limit $\sigma^2 < 1/(2\pi e)$ [12]. The public key is generated by considering the HNF of the LDLC generator matrix following Micciancio's proposition. It was claimed in [11] that the same upper bound $O(n^2\log_2(n))$ is obtained for the public key size and that the complexity of the ciphertext decryption is reduced with LDLC iterative decoding algorithm.

III. GLD Lattice-Based GGH Cryptosystem

We propose to use GLD lattices to design an improved version of GGH cryptosystem. We start first by introducing the construction of GLD lattices.

A. GLD Lattices

GLD lattices were proposed by Boutros et al. in [13]. 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 that these lattices achieve asymptotically Poltyrev limit [12].

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

$$\Lambda = C_{GLD} + p\mathbb{Z}^n \tag{4}$$

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

This construction considers an elementary small linear code $C_0[n_0, k_0]$ defined over the finite field 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

$$C_1 = C_0^{\oplus L} \tag{5}$$

The parity-check matrix of C_1 is given by

$$H_{C_1} = \begin{bmatrix} H_{C_0} & 0 & \dots & 0 \\ 0 & H_{C_0} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H_{C_0} \end{bmatrix} \tag{6}$$

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

$$C_{GLD} = \bigcap_{j=1}^{J} \pi_j(C_1) = \bigcap_{j=1}^{J} \pi_j(C_0^{\oplus L}) \tag{7}$$

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

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

$$H_C = \begin{bmatrix} H_{C_1} \\ \pi_2(H_{C_1}) \\ \vdots \\ \pi_J(H_{C_1}) \end{bmatrix} \tag{8}$$

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

$$H_{C,syst} = [\, I \mid -B^t \,] \tag{9}$$

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

$$G_C = [\, B \mid I \,] \tag{10}$$

Based on construction A (4), the generator matrix of Λ is given by

$$G_\Lambda = \begin{bmatrix} pI & 0 \\ B & I \end{bmatrix} \tag{11}$$

B is a dense matrix whose elements belong to the finite field 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

Our GLD-based GGH cryptosystem 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

$$\mathbf{c} = \mathbf{m}G_\Lambda + \mathbf{e} \tag{12}$$

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

In our study, we consider GLD lattices with dimension $n \approx 1000$. We analyze below the security and complexity of the proposed GLD lattice-based GGH cryptosystem. We prove through experimental results that our system is both secure and practical for this dimension.

IV. Security Analysis and Experimental Results

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_1} 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_1}. Therefore, the search space for the private key is

$$n! \times (J - 1) \tag{13}$$

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

$$n! \sim \sqrt{2\pi e}\left(\frac{n}{e}\right)^n \tag{14}$$

We can see that it is exponential in n. Hence, for high dimensions $n \geq 1000$, 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. These attacks applied Babai's algorithms, namely round-off and nearest plane algorithms, resulting in the well-known round-off attack and nearest plane attack [14].

Round-off attack: The round-off attack [14] consists in multiplying the ciphertext $\mathbf{c}$ by the inverse of the public basis G_Λ^{-1} which results in a noisy message as follows:

$$\mathbf{c}G_\Lambda^{-1} = \mathbf{m} + \mathbf{e}G_\Lambda^{-1} \tag{15}$$

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 [14] 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 g_i, $i = 1,\ldots,n$, the nearest plane algorithm finds recursively the closest hyperplane to the ciphertext $\mathbf{c}$.

In order to improve these decoders and hence the corresponding attacks, basis reduction techniques can be applied, typically Lenstra-Lenstra-Lovász (LLL) and Block Korkine-Zolotarev (BKZ), before running the decoding algorithms. LLL operates on each basis vector alone, whereas BKZ enhances the reduction by working on blocks of t basis vectors. These 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. Nevertheless, they may fail to produce this basis rendering the decryption unfeasible.

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

a - Public basis: 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. As introduced in [4], it is the product of the basis vector lengths divided by the matrix determinant, defined as

$$\mathrm{OD}(G_\Lambda) = \frac{\prod_{i=1}^{n} \|g_i\|}{|\det(G_\Lambda)|} \tag{16}$$

where $\|g_i\|$ is the Euclidean norm of the i^{th} row in G_Λ. Since G_Λ is given by construction A in (11), the first $n - k$ rows have norm p and the determinant is $\det(G_\Lambda) = p^{n-k}$, then

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

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 F_p. Thus, $\|b_i\|^2 \gg 1 \Rightarrow \|g_i\| \gg \sqrt{2}$, yielding an orthogonality defect $OD(G_\Lambda) \gg (\sqrt{2})^k \gg 1$, showing that G_Λ is a bad basis.

Result 1: We have carried out some experiments to confirm this result. Table I presents the average length of the basis vectors and the orthogonality defect of G_Λ in dimension $n \approx 1000$ for different elementary codes C_0. These values indicate that for this dimension, the orthogonality defect ranges approximately from $O(10^{500})$ to $O(10^{1000})$. Fig. 1 presents the $OD(G_\Lambda)$ in logarithmic scale as function of the GLD lattices dimension n for the elementary code $C_0[8, 6]_{17}$. The curves are illustrated for the initial GLD public basis as well as the reduced bases by LLL and BKZ for several block sizes t. We can see that $OD(G_\Lambda)$ increases significantly with the dimension. Thus, applying either basis reduction is not efficient and cannot yield reduced basis. We also notice that increasing the block size for the BKZ algorithm from t = 10 to 40 does not improve the reduction while it increases considerably its running time.

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

C_0	p	n	k	$\|g_i\|$	$OD(G_\Lambda)$
$C_0[3, 2]$	11	999	333	39.9	$O(10^{531})$
$C_0[3, 2]$	17	999	333	63.2	$O(10^{598})$
$C_0[3, 2]$	29	999	333	109.8	$O(10^{678})$
$C_0[8, 6]$	17	1000	500	116.8	$O(10^{1034})$
$C_0[8, 6]$	53	1000	500	365.3	$O(10^{1281})$
$C_0[16, 12]$	17	992	496	145.6	$O(10^{1073})$

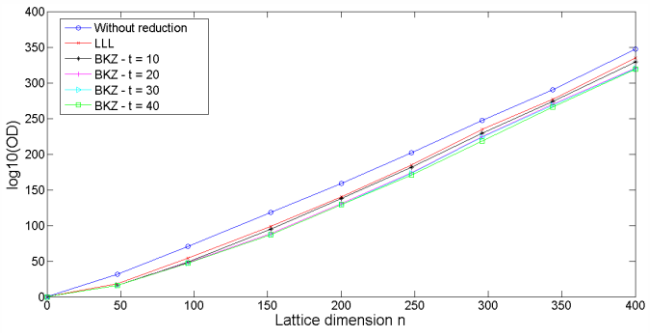

Figure 1. Public key orthogonality defect w/wo reduction for $C_0[8, 6]_{17}$.

b - Noise vector: The second factor that can lead to decryption failure is the added noise vector **e**. If **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, as mentioned in Section II [8], [10], [11].

GLD lattices were first proposed for channel coding and their performance was studied for a Gaussian channel with noise variance smaller than Poltyrev limit. However, this does not prevent us from using these codes with any other noise distribution. In our GLD lattice-based scheme, we consider a uniform real 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.

Result 2: We have determined experimentally the maximum noise variance σ tolerated by the iterative decoding for different elementary codes C_0 with different n_0, k_0 and p. The results of this experimentation are shown in Table II.

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

C_0	p	n	σ
$C_0[3, 2]$	11	999	1
$C_0[3, 2]$	17	999	2
$C_0[3, 2]$	29	999	3
$C_0[8, 6]$	17	1000	1
$C_0[8, 6]$	53	1000	2

c - Security discussion: After identifying these maximal noise variances σ for the considered codes, we have run both the round-off and nearest plane decoding attacks, in dimension $n \approx 1000$, in an attempt to decrypt ciphertexts obtained as $\mathbf{c} = \mathbf{m}G_\Lambda + \mathbf{e}$ for $\mathbf{e} \in [-\sigma, \sigma]$. Before running these decoding algorithms, we have reduced G_Λ using LLL and BKZ with different block sizes. For each code C_0 and its corresponding σ, the decryption attempt fails regardless of the attack scheme and reduction method. We can therefore conclude that these σ values are sufficient to ensure the security of our scheme against previous decoding attacks in dimension $n \approx 1000$.

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 [15].

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 F_p. At the same time, this density should not exceed a given value which enables successful iterative decoding. The codes $C_0[8, 6]_{17}$ and $C_0[16, 12]_{17}$ satisfy this double requirement.

V. Complexity Analysis and Experimental Results

For the complexity analysis of our scheme, we will examine three key features, namely the public key size, the public key generation and the decryption. We compare them to previous GGH schemes.

A. Key Size

Recalling the GLD generator matrix G_Λ defined in (11), we can notice that only part B 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

$$k \times (n-k) \times \log_2(p) \quad \text{bits} \tag{17}$$

Hence, 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 $n \approx 1000$ and we compute their key size.

Result 3: 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 F_{11}. This elementary lattice is given by

$$\Lambda_0 = [3, 2]_{11} + 11\mathbb{Z}^3 \tag{18}$$

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	p	n	Key size (KBs)
$C_0[3, 2]$	17	999	110.66
$C_0[3, 2]$	29	999	131.52
$C_0[8, 6]$	17	1000	124.74
$C_0[8, 6]$	53	1000	174.8
$C_0[16, 12]$	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 [8]. Indeed, the size of the GGH public key is $O(n^3 \log_2(n))$ bits. In Micciancio's cryptosystem, the private key is $O(n^2 \log_2(n))$ bits since the public basis B is in HNF form.

In [11], the same upper bound of $O(n^2 \log_2(n))$ is claimed as an HNF matrix is considered for the public key. However, this later bound was computed by Micciancio based on his specific choice of private key. In addition, Micciancio's cryptosystem works with integer lattices while in [11] real LDLC lattices are used. 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 [11] showed that the public key size in dimension n = 1000 is 43.6 MBs for GGH and 1MB 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 [9]. 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 $H_{C,syst}$. 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 [16]. In our case, this reduces to $O(n \times (n-k)^{\omega-1}) < O(n^3)$.

In Micciancio's system [8], the main problem is the complexity of the HNF algorithm that generates the public key. The most basic algorithm for generating HNF matrices performs 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 in [17] 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 [11], 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.

Result 4: In order to compare the key generation running time of our GLD lattice-based scheme with those of the GGH and Micciancio systems, we experimentally measured the duration of these operations on an Intel i5 3320M (2.6 GHz) platform. The results are presented in Fig. 2. For $n \approx 1000$, Miccianco's key generation procedure lasts more than 5 hours and for the GGH system, it is 10 minutes. On the contrary, our GLD scheme allows to generate the public key in the order of seconds: 1.25s for $C_0[3,2]$, 8.47s for $C_0[8,6]$ and 13.99s for $C_0[16,12]$.

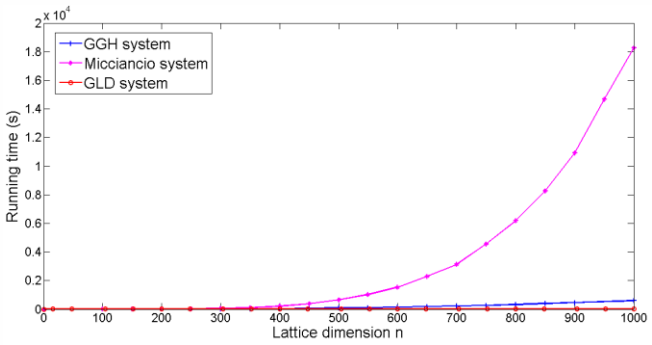

Figure 2. Key generation running time for GGH, Micciancio and GLD cryptosystems.

C. Decryption

As mentioned in [13], GLD lattices are suitable for iterative decoding. Let us analyze its decoding complexity. 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, …, 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 [13] 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 $n = n_0L$ 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 [18]. 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 used in [11] are also suitable for iterative decoding. However, in the LDLC decoding, the messages computed and passed by the check nodes have real values that need to be sampled and quantized into discrete ones. In addition, the number of these values grows exponentially with the number of iterations. 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.

For Micciancio's cryptosystem, Babai's nearest plane algorithm is applied to compute the closest lattice vector to the noisy ciphertext. This sub-optimal algorithm should be repeated many 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. It was shown by simulations that this decryption procedure is very slow in high dimensions [9]. Conversely, GLD iterative decoding provides a performance close to the optimal decoding while keeping the complexity at a very low level.

Result 5: By conducting another experiment on the time needed to decrypt one ciphertext, we have observed that Micciancio scheme requires 2 minutes for $n \approx 1000$. As for the GLD system, this time varies depending on C_0: 17ms for $C_0[3, 2]_{11}$, 0.13s for $C_0[3, 2]_{17}$, 0.4s for $C_0[3, 2]_{29}$, 2.3s for $C_0[8, 6]_{17}$ and 53s for $C_0[8, 6]_{53}$. We can deduce that, in addition to offering better performance, the GLD decryption algorithm is also faster than Micciancio's decryption algorithm.

VI. Conclusion

We have proposed an improvement to GGH cryptosystem using GLD lattices [13]. In this scheme, the private key is the parity-check matrix H_C of the GLD code and the public key is the lattice generator matrix G_Λ which can be obtained by construction A and has a Hermite normal form. We have shown through analysis and experimental results that GLD lattice-based scheme reduces considerably the complexity while being as secure as previous schemes. In fact, the specific form of G_Λ induces a huge reduction in the key size. In dimension $n \approx 1000$, the key can be represented by approximately 100 KBs whereas the key size of previous GGH systems was in the order of MBs. Moreover, since G_Λ is by design in HNF, the public key generation is at least 1300 times faster than Micciancio and LDLC systems. In addition, the GLD lattice iterative decoding offers a better performance than Babai's algorithm and decreases the decryption complexity compared to both Babai's algorithm and LDLCs iterative decoding.

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," Postquantum cryptography (Springer), pp. 1–14, 2009.

[3] 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.

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

[5] 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.

[6] 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.

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

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

[9] C. Ludwig, "The Security and Efficiency of Micciancio Cryptosystem," 2004, Technical Report. Available at http://www.cdc.informatik.tudarmstadt.de/reports/TR/TI-02-07.MiccPaper.pdf.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.