Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

GGH May Not Be Dead After All

Charles F. de Barros, **L. Menasché Schechter**
Graduate Program in Informatics, Federal University of Rio de Janeiro
21941-590, Rio de Janeiro, RJ
E-mail: charles.barros@ppgi.ufrj.br, luisms@dcc.ufrj.br

Abstract: *In 1997, Goldreich, Goldwasser and Halevi presented the GGH cryptosystem, which is based on hard lattice problems. Only two years later, Nguyen pointed out major flaws on the scheme. From that point on, the system was considered officially dead. However, in 2012, Yoshino and Kunihiro proposed some improvements on the GGH cryptosystem, claiming to have fixed the flaws pointed out by Nguyen. In this paper, we make a thorough analysis of this tweaked GGH scheme, showing that, in practice, it behaves mostly in the same way as the original scheme. We also propose some modifications that can effectively make the new GGH different from the original one.*

Keywords: *Public-Key Cryptography, Lattices, Post-Quantum Cryptosystem*

1 Introduction

The GGH cryptosystem was introduced to the world in 1997 [4]. Its security is based on the assumption that lattice basis reduction is a hard problem. It is considered a *post-quantum* cryptosystem, since not even quantum computers would be able to efficiently break it using known algorithms, unlike RSA [9], which could be broken by a quantum computer capable of running Shor's algorithm [11].

Its main idea is to encode a message into a lattice vector $\mathbf{v}$, add some small perturbation $\mathbf{r}$ and generate the ciphertext $\mathbf{c} = \mathbf{v} + \mathbf{r}$. The norm of the vector $\mathbf{r}$ must be sufficiently small, so that $\mathbf{v}$ is the lattice vector closest to $\mathbf{c}$. Thus, decryption is equivalent to solving an instance of the *closest vector problem* (CVP).

GGH employs a secret key with some good structure and then hides this structure in the public key. In order to recover the secret key from the public key, one has to perform a lattice basis reduction, which is assumed to be hard. Besides that, as will be discussed later on this paper, GGH is susceptible to decryption errors. The probability of these errors can be minimized by properly choosing the parameters of the system.

Two years after GGH was presented, Nguyen showed that it has a major flaw in its design [8] and argued that, even modified, it cannot provide a good level of security without becoming impractical. Since then, GGH has been regarded as a dead system [3]. At least, that seemed to be the case until 2012, when Yoshino and Kunihiro proposed some improvements on the scheme [14]. They claim to have fixed the flaws of GGH, improving its security. According to the authors, this modified GGH, which we call GGH-YK, has no longer CVP as its underlying problem and is no longer susceptible to decryption errors.

Throughout this paper, we discuss the working principles of the GGH-YK scheme. Our goal is to show that, even with the modifications proposed by Yoshino and Kunihiro, nothing seems to have changed: CVP is still the underlying problem of the scheme, which means that, in practice, GGH-YK behaves in the same way as the original GGH. Complementing this analysis, we propose some additional changes in order to make GGH-YK effectively different from the original scheme.

DOI: 10.5540/03.2015.003.01.0095

The rest of this paper is organized as follows. In Section 2, we make a brief review of lattice theory. The original GGH scheme is described in Section 3 and the attack proposed by Nguyen is discussed in Section 4. In Section 5, we describe GGH-YK and, in Section 6, we propose some modifications to it. Finally, in Section 7, we present our results and conclusions.

2 Background on Lattices

In this section, we provide a theoretical background on lattices. Some fundamental concepts are briefly discussed, such as lattice bases, lattice determinants and the orthogonality defect of a basis.

Let $B = [\mathbf{b}_1 \cdots \mathbf{b}_n]$ be an $n \times n$ matrix, where $\mathbf{b}_1, \cdots, \mathbf{b}_n$ are linearly independent *column vectors* of $\mathbb{R}^n$. The *full rank lattice generated by* B is the set

$$L(B) = \{B\mathbf{u} \mid \mathbf{u} \in \mathbb{Z}^n\}. \tag{1}$$

The matrix B is called a *basis* of the lattice. Every lattice has infinitely many bases. Two bases A and B generate the same lattice if $L(A) = L(B)$. In this case, there exists an unimodular matrix (an integer matrix with determinant ± 1) U such that $A = BU$.

The *determinant* of a lattice is the volume of the n-dimensional parallelepiped formed by the vectors of a basis of the lattice. It is given by the absolute value of the determinant of the basis. Since two bases differ by a multiplication with an unimodular matrix, the determinant of a lattice does not depend on the choice of the basis.

The *orthogonality defect* of a basis B is given by

$$\mathcal{D}(B) = \frac{\prod_{i=1}^{n} ||\mathbf{b}_i||}{|\det(B)|}. \tag{2}$$

It measures how close to being orthogonal are the vectors of a basis. Unlike vector spaces, not every lattice has an orthogonal basis. Due to Hadamard's inequality [13], we have $\mathcal{D}(B) \geq 1$, and $\mathcal{D}(B) = 1$ iff B is orthogonal. A basis is considered *good* if its orthogonality defect is small. Thus, an orthogonal basis is considered optimal.

Given a good lattice basis, it is easy to obtain a *bad* basis for the same lattice, but the inverse is hard in large dimensions. This problem of finding a good basis, given an arbitrary and possibly bad basis, is closely related to the following problems:

- *Shortest Vector Problem (SVP):* given a lattice, find a vector $\mathbf{v} \neq \mathbf{0}$ in the lattice, such that $||\mathbf{v}||$ is minimum.
- *Closest Vector Problem (CVP):* given a lattice and a vector $\mathbf{c}$, probably not in the lattice, find the lattice vector $\mathbf{v}$ closest to $\mathbf{c}$.

CVP can be solved using Babai's rounding algorithm [1], provided that a good basis for the lattice is known, or by the *immersion technique* [4], which reduces CVP to SVP. In order to solve SVP approximately, we must find a *reduced* lattice basis, which has the property that its first vector is an approximation for the shortest lattice vector.

Reduced bases have other interesting properties, and there are algorithms to obtain these bases, such as LLL [6] and BKZ [10]. The problem of finding a reduced basis, the so-called *lattice reduction problem*, is also considered hard in arbitrary dimensions.

3 The Original GGH

The scheme proposed by Goldreich, Goldwasser and Halevi [4] employs as secret key an integer $n \times n$ matrix of the form $B = \gamma I + P$, where γ is an integer, I is the identity matrix and $p_{i,j} \in \{-t, \cdots, t\}$, for some small integer t. The public key W is obtained by multiplying the

secret key by a random unimodular matrix U. Thus, we have $W = BU$. The secret key has very small orthogonality defect (it is a good basis), while the public key has a large orthogonality defect.

The encryption of a message $\mathbf{x} \in \mathbb{Z}^n$ is given by $\mathbf{c} = W\mathbf{x} + \mathbf{r}$. Decryption of $\mathbf{c}$ consists of applying Babai's algorithm using the secret key, since we expect that $\mathbf{r}$ is sufficiently small and, consequently, $W\mathbf{x}$ is the vector of $L(W)$ closest to $\mathbf{c}$.

In order for the decryption procedure to work properly, we must make sure that all of the entries of the vector $B^{-1}\mathbf{r}$ have absolute value less than 0.5, which means that the *rounding error vector* $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$ is the null vector. Otherwise, there will be decryption errors. In practice, we only minimize the probability of these errors, by setting a sufficiently small parameter σ such that $r_i = \pm\sigma$, for all $i = 1, \cdots, n$.

4 Cryptanalysis of GGH

The most direct way of attacking GGH is to reduce the public key in order to find a good basis to apply Babai's algorithm. Even if the secret key itself is not found, the message can be recovered, provided that the reduced basis is good enough. The other attack employs the immersion technique, which consists of applying lattice reduction to the basis

$$Q = \begin{pmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_n & \mathbf{c} \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

If the attack is successful, we find the vector $(\mathbf{r}^T, 1)^T$ at the first row of the reduced basis obtained after reducing Q.

In 1999, P. Nguyen pointed out major flaws on the GGH scheme [8]. Using lattice reduction, he solved all the GGH Internet challenges, except in dimension 400, for which he could only recover partial bits of the message.

Nguyen exploited two weaknesses of the scheme. The first one is that perturbation vectors are always much shorter than the lattice vectors. As a result, the CVP instance we need to solve in order to break the system is easier than an arbitrary CVP. The second weakness is related to the special form of the perturbation vectors.

The first weakness is inherent to the scheme and, at that time, there seemed to be no easy way to fix it. The second weakness could be fixed by choosing $r_i \in \{-\sigma, \cdots, \sigma\}$ instead of $r_i \in \{-\sigma, \sigma\}$. In this case, though, the perturbation vector $\mathbf{r}$ would become even smaller, making CVP even easier. Even if we choose $r_i \in \{\pm\sigma, \pm(\sigma - 1)\}$, we still create some vulnerability due to the particular form of the perturbation vector.

In principle, all of these problems could be bypassed by increasing the dimension of the keys. However, according to Nguyen, this would make the system impractical. Thus, he came to the conclusion that GGH cannot offer a good level of security without becoming impractical.

In 2010, Nguyen's attack was extended by Lee and Hahn [5], and the last GGH challenge, in dimension 400, was solved. The scheme really seemed to be dead.

5 The Modified GGH Scheme

In 2012, Yoshino and Kunihiro proposed a modified version of GGH [14]. In this new version, which we call GGH-YK, they try to break the relation between decryption and solving CVP by increasing the length of the perturbation vector $\mathbf{r}$. They set a new parameter k, such that

$$e_i = \begin{cases} 0 & \text{for at least } n-k \text{ values of } i, \\ \pm 1 & \text{for at most } k \text{ values of } i, \end{cases}$$

where $\mathbf{e}$ is, as before, the rounding error vector ($\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$). Since $\mathbf{e} \neq \mathbf{0}$, CVP would no longer be the underlying problem to the system. However, as we will see later, this goal cannot be achieved in practice without some additional changes to the scheme.

As in the original GGH, the secret key of GGH-YK is an $n \times n$ integer matrix of the form $B = \gamma I + P$, where γ is an integer parameter randomly chosen, I is the identity matrix and P is a random integer matrix such that $p_{i,j} \in \{-1, 0, 1\}$. We can achieve very small orthogonality defect by setting $\gamma > n$. Let $A = B^{-1}$. For all i, j, the following properties must hold:

$$|a_{i,j}| \leq \frac{1}{\gamma}, \text{ if } i = j; \tag{3}$$

$$|a_{i,j}| < \frac{2}{\gamma^2}, \text{ if } i \neq j. \tag{4}$$

As recommended by Micciancio in [7], the public key W is obtained by computing the Hermite Normal Form (HNF) of B. A matrix is said to be in the HNF if it is upper triangular, all of its diagonal entries are positive, the other entries are nonnegative and the largest entry of each row is in the main diagonal. It provides both efficiency and security, since bases in the HNF are easier to store and harder to reduce.

Message encoding is the next step. In the GGH-YK scheme, binary messages of length $l \leq n$ are encoded into the perturbation vector $\mathbf{r}$ instead of a lattice point. First, we set public parameters (σ, h, k), with $h > \sigma$. The idea is that the vector $\mathbf{r}$ has small entries, with absolute value no greater than σ, and large entries of absolute value h. The number of large entries is k.

Before encoding the message, two secret sets $S, T \subset I_n = \{1, \cdots, n\}$ are randomly chosen, such that $|S| = k$, $|T| = n - l$ and $S \cap T = \emptyset$. The bits of the message are encoded into the non-zero entries of $\mathbf{r}$, with the 0's encoded as negative entries, and the 1's as positive entries, according to the following rules:

$$\begin{cases} r_i = \pm h & \text{for all } i \in S; \\ r_i \in \{-\sigma, \cdots, -1\} \cup \{1, \cdots, \sigma\} & \text{for all } i \in I_n \setminus (S \cup T); \\ r_i = 0 & \text{for all } i \in T. \end{cases}$$

The encryption of $\mathbf{r}$ is given by $\mathbf{c} = W\mathbf{x} + \mathbf{r}$, where $\mathbf{x} = -\lfloor W^{-1}\mathbf{r} \rfloor$. In order to decrypt $\mathbf{c}$, we compute the vector $\mathbf{u} = B^{-1}\mathbf{c} - \lceil B^{-1}\mathbf{c} \rfloor = B^{-1}\mathbf{r} - \lceil B^{-1}\mathbf{r} \rfloor$ and then obtain $\mathbf{r}' = B\mathbf{u} = \mathbf{r} - B\mathbf{e}$. The final step of decryption is to determine the entries of $\mathbf{e}$. In their paper, Yoshino and Kunihiro prove the correctness of the procedure below:

1. If $r'_i < -h - k$, set $e_i = 1$;
2. If $r'_i > h + k$, set $e_i = -1$;
3. Set $e_i = 0$ otherwise.

After computing the rounding error vector, the original encoded message can be recovered, since $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$. Notice that we are clearly assuming that $\mathbf{e}$ is not necessarily the null vector. Also notice that the decryption process is entirely deterministic, unlike the original GGH, in which there is a probability of decryption errors.

In order to make the system work properly, we must ensure that the parameters n, σ, h, k, γ satisfy the following conditions:

$$\frac{\sigma}{\gamma} + \frac{2kh}{\gamma^2} + \frac{2n\sigma}{\gamma^2} < \frac{1}{2}, \tag{5}$$

$$\frac{h - \sigma}{\gamma} + \frac{2h}{\gamma^2} < 1, \tag{6}$$

$$2k + 2h < \gamma. \tag{7}$$

In short, condition (5) ensures that $e_i = 0$ for all $i \notin S$, condition (6) gives us the hope that $e_i \neq 0$ for at least some index $i \in S$, while (7) is a necessary condition for the correctness of decryption.

6 A tweaked version of GGH-YK

After implementing GGH-YK, we noticed that it had some problems. The first one was that it seemed to be impossible to generate a secret key satisfying condition (3). In order to generate a valid secret key, we relaxed (3) to

$$|a_{i,j}| \leq \frac{2}{\gamma}, \text{ if } i = j, \tag{8}$$

and kept (4). After this change, (5) and (6) had to be modified. Employing the same procedures used to derive (5) and (6), but keeping the inequalities tight, we got the following conditions:

$$\frac{2\sigma}{\gamma} + \frac{2kh}{\gamma^2} + \frac{2n\sigma}{\gamma^2} < \frac{1}{2} + \frac{2\sigma(k+1)}{\gamma^2}, \tag{9}$$

$$2(h-\sigma)\left(\frac{1}{\gamma} - \frac{1}{\gamma^2}\right) < 1. \tag{10}$$

Hence, in our first attempt to fix GGH-YK, inequalities (3), (5) and (6) were replaced by conditions (8) to (10). In this attempt, we decided to keep (7), since decryption depends on it. Condition (4) was also left untouched.

The second problem we found was that, even after the changes we have just described, the rounding error vector $\mathbf{e}$ was always the null vector. Consequently, GGH-YK does not behave differently from the original GGH, since decryption is still equivalent to solving an instance of CVP.

In order to effectively fix GGH-YK, we employed M-matrices as secret keys and created a variant of GGH-YK, which we call GGHYK-M. An M-matrix is a square matrix of the form $B = \gamma I + P$, where $\gamma > \rho(P)$[1] and $p_{i,j} \leq 0$. It can be shown that the inverse of an M-matrix has only positive entries [2].

Hence, we used as secret key a matrix $B = \gamma I + P \in \mathbb{Z}^{n\times n}$, where $\gamma \in \mathbb{Z}$ and $p_{i,j} \in \{-1, 0\}$. In order to ensure that $\gamma > \rho(P)$, it is sufficient to choose $\gamma = \alpha n$, for some small integer α. The public key W is still the Hermite Normal Form of B. Along with (8), we imposed another condition over the diagonal entries of the matrix $A = B^{-1}$, namely

$$|a_{i,j}| > \frac{1}{\gamma}, \text{ for all } i = j. \tag{11}$$

After these modifications, we had to replace condition (7) by the following inequality:

$$\frac{h}{\gamma} > \frac{1}{2}, \tag{12}$$

which means that $2h > \gamma$. We also imposed a new condition, in order to keep the decryption process deterministic:

$$h + k < \gamma. \tag{13}$$

As a consequence of our new conditions, the rounding error vector $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$ now satisfies the following properties:

$$e_i = \begin{cases} 0 & \text{for all } i \notin S; \\ 1 & \text{for all } i \in S. \end{cases} \tag{14}$$

We encode the vector $\mathbf{r}$ only with positive entries. While in GGH-YK the message bits are encoded into non-zero entries (small and large), in GGHYK-M these bits are encoded only into small entries, and the large entries are redundant. As in GGH-YK, the subset S contains the indices of the large entries of $\mathbf{r}$. There are no longer zero entries, which means that the subset T no longer exists.

[1] $\rho(P) = \max\{|\lambda_i| : \lambda_i \text{ is an eigenvalue of } P\}$ is the spectral radius of P.

DOI: 10.5540/03.2015.003.01.0095

Notice that we encode exactly $n-k$ message bits into a vector of length n. The 0 bits are encoded as entries randomly chosen from the set $\{1, \cdots, \sigma/2\}$, and the 1 bits as entries randomly chosen from the set $\{\sigma/2+1, \cdots, \sigma\}$. We only require that an even value for σ is chosen.

Encryption works exactly as in GGH-YK. Decryption is also similar to GGH-YK, but the final step, in which we determine the entries of $\mathbf{e}$ according to the corresponding entries of $\mathbf{r}'$, was slightly modified:

1. If $r'_i < 0$, set $e_i = 1$;
2. Set $e_i = 0$ otherwise.

After computing $\mathbf{e}$, we recover the encoded message vector $\mathbf{r}$ because, as in GGH-YK, we have $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$.

7 Our Results and Conclusions

In our implementation of GGHYK-M, we used the NTL C++ library (version 5.5.1) [12], running on a dual core AMD E-350 processor, with 1.6 GHz and 4GB of RAM. In order to avoid approximation errors, we employed only integer arithmetic. Table 1 shows the key generation, encryption and decryption times, in seconds, for the GGHYK-M scheme with several parameters.

(n, σ, h, k)	Secret key	Public key	Encryption	Decryption
$(128, 32, 193, 16)$	$2,36$	$9,28$	$0,02$	$2,32$
$(200, 64, 301, 32)$	$11,93$	$89,94$	$0,02$	$11,83$
$(256, 64, 385, 32)$	$13,86$	$331,53$	$0,03$	$33,90$
$(300, 128, 451, 50)$	$22,29$	$717,54$	$0,04$	$61,58$
$(350, 256, 526, 64)$	$35,45$	$1627,1$	$0,06$	$115,92$
$(400, 256, 601, 64)$	$51,46$	$3075,71$	$0,07$	$210,41$

Table 1: Key generation, encryption and decryption times, in seconds.

Decryption turned out to be quite slow, since it requires the inversion of the secret basis. In order to speed up decryption, the secret basis inverse can be previously stored, so that it will no longer be necessary to compute it during the decryption process. Decryption times with the secret key inverse previously stored are shown on Table 2. It is clear that the inversion of the secret matrix is what slows down decryption.

(n, σ, h, k)	Decryption
$(128, 32, 193, 16)$	$0,01$
$(200, 64, 301, 32)$	$0,04$
$(256, 64, 385, 32)$	$0,08$
$(300, 128, 451, 50)$	$0,11$
$(350, 256, 526, 64)$	$0,17$
$(400, 256, 601, 64)$	$0,27$

Table 2: Decryption time, in seconds, with the secret key inverse previously stored.

The security of GGHYK-M was tested by employing lattice reduction techniques (LLL and BKZ). None of the attacks were successful in dimensions greater than 300. Notice that, since the usual attacks on GGH (reducing the public basis to apply Babai's algorithm and the immersion technique) are ways of solving CVP and this is no longer the underlying problem of GGHYK-M, these attacks no longer work against the system. Recovering the secret key itself is the only way, up to now, to break the scheme. Hence, if lattice reduction fails to find the secret key, the security of the scheme seems unaffected.

DOI: 10.5540/03.2015.003.01.0095

It is important to emphasize that using M-matrices as secret keys does not affect the security of the scheme at all, because the Hermite Normal Form of M-matrices is indistinguishable from the HNF of general matrices. Thus, if an attacker picks up a public key obtained from an M-matrix, he does not take any advantage from this fact. We also remark that the Hermite Normal Form reduces the size of the keys, so that employing dimension 300 or even larger does not compromise the practical viability of the system.

We come to the conclusion that GGHYK-M provides major changes on the GGH-YK variant, making it effectively different from the original GGH. As far as the results of our experiments show, GGHYK-M is secure and viable. Hence, we highly encourage further research on the feasibility of lattice-based cryptography as an alternative to existing public-key cryptosystems in a post-quantum era.

References

[1] L. Babai, On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6(1)** (1986) 1-13.

[2] A. Berman, R. J. Plemmons, Nonnegative Matrices in the Mathematical Sciences. Classics in Applied Mathematics, SIAM (1987).

[3] O. Goldreich, Private communication (1999).

[4] O. Goldreich, S. Goldwasser, S. Halevi, Public-key cryptosystems from lattice reduction problems. Crypto'97, Lecture Notes in Computer Science **1294** (1997) 112-131.

[5] M. S. Lee, S. G. Hahn, Cryptanalysis of the GGH Cryptosystem. Mathematics in Computer Science **3** (2010) 201-208.

[6] A. K. Lenstra, H. W. Lenstra Jr, L. Lovász, Factoring polynomials with rational coefficients. Mathematische Annalen **261** (1982) 515-534.

[7] D. Micciancio, Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In CaLC, Lecture Notes in Computer Science **2146** (2001) 126-145.

[8] P. Q. Nguyen, Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto'97. In Crypto'99, Lecture Notes in Computer Science **1666** (1999) 288-304.

[9] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, ACM Press **21** (1978) 120–126.

[10] C. P. Schnorr, A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science **53(2-3)** (1987) 201-224.

[11] P. W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, Philadelphia **26(5)** (1997) 1484–1509.

[12] V. Shoup, Number Theory C++ Library (NTL) version 5.5.1. Available at http://www.shoup.net/ntl/. Last accessed on February 26 2014.

[13] F. Voloch, Hadamard's Inequality. Document available online at http://www.ma.utexas.edu/users/voloch/Homework/hadamard.pdf. Last accessed on February 26 2014.

[14] M. Yoshino, N. Kunihiro, Improving GGH Cryptosystem for Large Error Vector. International Symposium on Information Theory and its Applications (2012) 416-420.

