Improving Lattice Based Cryptosystems Using the Hermite Normal Form

Daniele Micciancio[1]*

Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
daniele@cs.ucsd.edu

Abstract. We describe a simple technique that can be used to substantially reduce the key and ciphertext size of various lattice based cryptosystems and trapdoor functions of the kind proposed by Goldreich, Goldwasser and Halevi (GGH). The improvement is significant both from the theoretical and practical point of view, reducing the size of both key and ciphertext by a factor n equal to the dimension of the lattice (i.e., several hundreds for typical values of the security parameter.) The efficiency improvement is obtained without decreasing the security of the functions: we formally prove that the new functions are at least as secure as the original ones, and possibly even better as the adversary gets less information in a strong information theoretical sense. The increased efficiency of the new cryptosystems allows the use of bigger values for the security parameter, making the functions secure against the best cryptanalytic attacks, while keeping the size of the key even below the smallest key size for which lattice cryptosystems were ever conjectured to be hard to break.

Keywords: Lattices, trapdoor functions, public-key encryption

1 Introduction

Recent results on the complexity of lattices [1] have drawn considerable attention to lattice problems as potential candidates to design cryptographic primitives, and encryption schemes in particular. The two most notable proposals are the Ajtai-Dwork cryptosystem (AD, [2]) and the Goldreich-Goldwasser-Halevi cryptosystem (GGH, [14]). Other lattice based cryptosystems designed along the lines of [2, 14], were subsequently proposed by Fischlin and Seifert [10] and Cai and Cusick [6]. While the AD cryptosystem is mainly of theoretical interest, the GGH cryptosystem was suggested as a practical alternative to number theory based schemes currently in use (e.g., the RSA cryptosystem [28]). Two other related cryptosystems are McEliece's [21] and NTRU [17]. Neither of them is a lattice based cryptosystem in the strict meaning of the term, as they uses ideas

* Research supported in part by NSF Career Award CCR-0093029

from other areas of mathematics (polynomial ring and finite field arithmetics respectively). However, the security of NTRU is related to the hardness of certain lattice problems (see Sect. 7 for further discussion), and it definitely deserves a mention as the most practical of the above proposals, yielding keys of size $O(n \log n)$ instead of the $\Omega(n^2)$ key size of McEliece, GGH and related schemes. (In fact, we'll see that GGH keys can be as big as $O(n^3 \log n)$.) McEliece's scheme is based on the hardness of coding theoretic problems, rather than lattices, but it bears much resemblance to the GGH cryptosystem, and we will further discuss this scheme in Sect. 7, after introducing our new lattice based trapdoor function. As it is the case for RSA, no proof of security for the GGH cryptosystem (or any of the afore mentioned schemes with the only exception of Ajtai and Dwork's theoretical proposal) is currently known, and its conjectured security is based on the empirical evidence that certain lattice problems are hard. In the attempt of stimulating further cryptanalytic efforts against their system and determining appropriate key size, the authors of GGH published 5 numerical challenges [13] corresponding to increasing values of the security parameter $n = 200, 250, 300, 350, 400$, resulting in public key sizes ranging from 330 KB to over 2 MB. Despite the big key size even for the smallest dimension (330 KB), this cryptosystem was still competitive with number theoretic cryptosystems because the encryption time is essentially linear in the size of the key, while modular exponentiation typically requires $O(n^3)$ operations.

At the time the GGH cryptosystem was presented at Crypto'97, challenges in dimension $n = 150$ were known to be breakable [31] using lattice reduction techniques. Still these techniques didn't seem to apply to lattices in higher dimension $n > 200$. At Crypto'99 Nguyen [24] showed how to exploit a weakness specific to the way GGH challenges were chosen and break the first four of the five GGH challenges. As the only unbroken challenge had key size over 2 MB, the practical value of the GGH cryptosystem (and variants) seemed quite questionable and [24] concluded that "unless major improvements are found, lattice-based cryptography cannot provide a serious alternative to existing public-key encryption algorithms". In this paper we present one such improvement.

Although quite simple, the techniques described in this paper can be used to significantly reduce the key size of GGH-like cryptosystems (e.g., those presented in [14, 10]). In particular, we show how to build lattice based trapdoor functions with public keys of size below 330 KB, and still secure against the best known lattice attacks. The improvement gets even better as the dimension of the lattice grows: our techniques can be used to asymptotically reduce the key size from $O(n^3 \log n)$ to $O(n^2 \log n)$. The improvement in the encryption time is analogous, being roughly proportional to the size of the public key, and the size of the ciphertext is also significantly reduced going from $O(n^2 \log n)$ to $O(n \log n)$. Surprisingly, we can achieve these efficiency improvements without decreasing the security of the scheme: any attack to the new scheme can be provably transformed into an (at least) equally effective attack against the original GGH (and variant) schemes. The increased efficiency allows one to use greater values of the security parameter than considered in [14] and [24], while maintaining the scheme reasonably practical. In particular, our techniques give encryption schemes that resist the best known lattice attacks and still have public keys even smaller than the weakest of the GGH challenges, bringing the feasibility of lattice based cryptography back into discussion.

The rest of the paper is organized as follows. In Sect. 2 we recall some basic definitions and properties of lattices. In Sect. 3 we describe the original GGH scheme and discuss its weaknesses. In Sect. 4 we define a new cryptographic function that significantly improves the GGH scheme both in terms of security and efficiency. The new scheme is analyzed in Sect. 5. To be precise, the GGH scheme, as well as the new one we are proposing in this paper, should be considered as *trapdoor functions* instead of ready-to-use *encryption schemes*. The reason is explained in Sect. 6, where we also discuss how to transform these functions into encryption schemes. In Sect. 6 we also describe some cryptanalytic experiments that we performed to validate our theoretical results. The McEliece and NTRU cryptosystems are discussed in Sect. 7. Section 8 concludes with final remarks and open problems.

2 Preliminaries

Let $B = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n}\}$ be a set of n linearly independent vectors in $\mathbb{R}^m$. The *lattice* generated by B is the set $\mathcal{L}(B) = \{\sum_i x_i \boldsymbol{b_i} \mid x_i \in \mathbb{Z}\}$ of all *integer* linear combinations of the vectors in B. The set B is called *basis* and it is usually identified with the matrix $\boldsymbol{B} = [\boldsymbol{b_1}, \ldots, \boldsymbol{b_n}] \in \mathbb{R}^{m \times n}$ having the vectors $\boldsymbol{b_i}$ as columns. In matrix notation $\mathcal{L}(\boldsymbol{B}) = \{\boldsymbol{Bx} \mid \boldsymbol{x} \in \mathbb{Z}^n\}$. The lattice $\mathcal{L}(\boldsymbol{B})$ is *full rank* if $n = m$, i.e. if $\boldsymbol{B}$ spans the entire vector space $\mathbb{R}^m$ over the reals. For simplicity, in the rest of this paper we will consider only full rank lattices. Moreover, for computational purposes, we will assume the basis vectors $\boldsymbol{b_i} \in \mathbb{Z}^n$ have integer entries. Then, a basis is just a non-singular integer matrix $\boldsymbol{B} \in \mathbb{Z}^{n \times n}$.

The basis of a lattice is not unique. A particularly convenient basis for some applications is the *Hermite Normal Form* (HNF). A basis $\boldsymbol{B}$ is in HNF if it is upper triangular, all elements on the diagonal are strictly positive, and any other element $b_{i,j}$ satisfies $0 \le b_{i,j} < b_{i,i}$. It is easy to see that every integer lattice $\mathcal{L}(\boldsymbol{B})$ has a unique basis in Hermite Normal Form, denoted HNF($\boldsymbol{B}$). Moreover, given any basis $\boldsymbol{B}$ for the lattice, HNF($\boldsymbol{B}$) can be efficiently computed (see [23] for a recent algorithm and a survey of previous results). Notice that HNF($\boldsymbol{B}$) does not depend on the particular basis $\boldsymbol{B}$ with started from, and it is uniquely defined by the lattice $\mathcal{L}(\boldsymbol{B})$ generated by $\boldsymbol{B}$.

For every basis $\boldsymbol{B}$, the *orthogonalized basis* $\boldsymbol{B}^* = [\boldsymbol{b_1^*}, \ldots, \boldsymbol{b_n^*}]$ is defined by the usual Gram-Schmidt orthogonalization process:

$$\begin{aligned} \boldsymbol{b_1^*} &= \boldsymbol{b_1} \\ \boldsymbol{b_i^*} &= \boldsymbol{b_i} - \sum_{j<i} \frac{\langle \boldsymbol{b_i}, \boldsymbol{b_j^*} \rangle}{\langle \boldsymbol{b_j^*}, \boldsymbol{b_j^*} \rangle} \boldsymbol{b_j^*} \quad . \end{aligned}$$

Notice that $\boldsymbol{B}^*$ is a basis for the vector space $\mathbb{R}^n$, but is not in general a lattice basis for $\mathcal{L}(\boldsymbol{B})$. If $\boldsymbol{B}$ is in HNF, then $\boldsymbol{B}^*$ is simply the diagonal matrix with $b_{1,1}, \ldots, b_{n,n}$ on the diagonal. The determinant of a lattice $\mathcal{L}(\boldsymbol{B})$ is the absolute value of the determinant of the matrix $\boldsymbol{B}$. The determinant is a lattice invariant, i.e. it does not depend on the particular choice of the basis $\boldsymbol{B}$. If $\boldsymbol{B}$ is in HNF, then the determinant $\det(\boldsymbol{B})$ is just the product of the elements on the diagonal $\prod_{i=1}^{n} b_{i,i}$. An important property of integer lattices is that if two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ are congruent modulo $\det(\boldsymbol{B})$ (i.e. $\det(\boldsymbol{B})$ divides $v_i - w_i$ for all i), then $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ if and only if $\boldsymbol{w} \in \mathcal{L}(\boldsymbol{B})$. In other words, the lattice repeats identically if translated by multiples of $\det(L)$ along the direction of any of the main axes.

The distance between two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ is defined by

$$\operatorname{dist}(\boldsymbol{v}, \boldsymbol{w}) = \|\boldsymbol{v} - \boldsymbol{w}\| = \sqrt{\sum_i (v_i - w_i)^2} \ .$$

The distance function is extended to sets of vectors as usual

$$\operatorname{dist}(S_1, S_2) = \inf\{\|\boldsymbol{v} - \boldsymbol{w}\| : \boldsymbol{v} \in S_1, \boldsymbol{w} \in S_2\} \ .$$

In particular the distance of a vector from a lattice is given by $\operatorname{dist}(\boldsymbol{v}, \mathcal{L}(B)) = \min\{\|\boldsymbol{v} - \boldsymbol{w}\| : \boldsymbol{w} \in \mathcal{L}(B)\}$. In the *closest vector problem* (CVP), one is given a basis $\boldsymbol{B}$ and a target vector $\boldsymbol{v}$ (usually not in the lattice) and must find the lattice vector in $\mathcal{L}(\boldsymbol{B})$ closest to $\boldsymbol{v}$. CVP was proved NP-hard in [36], and it remains hard even if the lattice basis can be arbitrarily preprocessed [22], or one allows for approximate solutions with approximation factor $2^{\lg^{1-\epsilon} n}$ [3, 9]. To date, the best polynomial time algorithm to approximate CVP achieves only a worst case approximation factor which is almost exponential in the dimension of the lattice [19, 4, 29].

A closely related problem is the shortest vector problem (SVP): given a lattice $L = \mathcal{L}(\boldsymbol{B})$, find the length $\lambda(L)$ of the shortest non-zero vector in $\mathcal{L}(\boldsymbol{B})$. By linearity, $\lambda(\boldsymbol{B})$ equals the minimum distance between any two lattice points $\min\{\|\boldsymbol{v} - \boldsymbol{w}\| : \boldsymbol{v}, \boldsymbol{w} \in \mathcal{L}(\boldsymbol{B}), \boldsymbol{v} \neq \boldsymbol{w}\}$. It is easy to see that for any vector $\boldsymbol{v}$ (not necessarily in the lattice) there exists at most one lattice point within distance $\lambda/2$ from $\boldsymbol{v}$. One can easily show that the length of the shortest vector in a lattice $L = \mathcal{L}(\boldsymbol{B})$ satisfies $\lambda(L) \geq \min_i \|\boldsymbol{b}_i^*\|$. Moreover, given a vector $\boldsymbol{v}$ within distance $\rho = \frac{1}{2} \min_i \|\boldsymbol{b}_i^*\|$ from the lattice, the (unique) lattice vector within distance ρ from $\boldsymbol{v}$ can be efficiently computed from $\boldsymbol{B}$ and $\boldsymbol{v}$ using Babai's *nearest plane* algorithm [4]. (See also [18].)

3 The GGH Encryption Scheme

The GGH encryption scheme [14] works essentially as follows. The private and public keys of the scheme are two bases $\boldsymbol{B}, \boldsymbol{R}$ of the same lattice $L = \mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{R})$. The private key $\boldsymbol{R}$ is an exceptionally good basis. In particular, $\boldsymbol{R}$ is chosen in such a way that the quantity $\rho = \frac{1}{2} \min \|\boldsymbol{r}_i^*\|$ is relatively large, so that all errors of length less than ρ can be efficiently corrected using $\boldsymbol{R}$. However, given the public basis this same task should be computationally hard. In particular, $\frac{1}{2}\min\|\boldsymbol{b}_i^*\|$ is much smaller than ρ.

The two bases are used to define a trapdoor function that takes as input an integer vector $\boldsymbol{x}$ and an error vector $\boldsymbol{r}$ of length at most ρ, and returns the vector $\boldsymbol{c} = \boldsymbol{Bx} + \boldsymbol{r}$, i.e. the lattice vector with public coefficients $\boldsymbol{x}$ perturbed by a small additive error $\boldsymbol{r}$. Notice that $\boldsymbol{Bx}$ can be recovered from $\boldsymbol{c}$ using the private basis $\boldsymbol{R}$. Once $\boldsymbol{Bx}$ is recovered, one can easily compute $\boldsymbol{x}$ and $\boldsymbol{r}$ using simple linear algebra, therefore inverting the function.

A message m is "encrypted" by first encoding m in the input $(\boldsymbol{x}, \boldsymbol{r})$, and then applying the trapdoor function to $(\boldsymbol{x}, \boldsymbol{r})$. Two encoding method are considered in [14]:

1. In the first method, the message m is encoded in the error vector $\boldsymbol{r}$, and $\boldsymbol{x}$ is chosen at random
2. In the second method, the message m is encoded in the coefficients $\boldsymbol{x}$, and $\boldsymbol{r}$ is chosen at random.

For concreteness, in the rest of the paper we will assume the first encoding method, but most of the techniques we describe can be easily adapted to the second method as well.

In order to fully specify the trapdoor function the following questions must be answered:

1. How is the private basis $\boldsymbol{R}$ chosen?
2. How is the public basis $\boldsymbol{B}$ obtained from $\boldsymbol{R}$?
3. How is the random vector $\boldsymbol{x}$ chosen?
4. How is the error vector $\boldsymbol{r}$ chosen?

The authors of [14] suggest[1] to take $\boldsymbol{R} = \sqrt{n} \cdot \boldsymbol{I} + \boldsymbol{Q}$, where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{Q}$ is a random perturbation matrix with entries in $\{-4, \ldots, +4\}$. Then obtain the public basis $\boldsymbol{B}$ applying a sufficiently long sequence of random elementary column operations to $\boldsymbol{R}$. Then the message m is "encrypted" by encoding it in an error vector $\boldsymbol{r}$ with entries $r_i = \pm 3$, and adding it to a lattice vector $\boldsymbol{Bx}$ chosen at random from a sufficiently large region of space.

Notice that in order to avoid attacks based on exhaustive search, the sequence of operations applied to $\boldsymbol{R}$ to obtain the public basis, and the region of space from which the lattice vector $\boldsymbol{Bx}$ is chosen must be sufficiently large. Since the lattice repeats identically if translated by $\det(L)$ along any of the main axes, we can always assume that the entries of $\boldsymbol{B}$ and $\boldsymbol{x}$ are reduced modulo $\det(L)$ without decreasing the security of the scheme. We can use this observation to estimate the proper size of the public key $\boldsymbol{B}$ and the ciphertext $\boldsymbol{c} = \boldsymbol{Bx} + \boldsymbol{r}$ as $O(n^2 \cdot \lg(\det(L)))$ and $O(n \cdot \lg(\det(L)))$. The determinant $\det(L)$ can also be estimated to be $2^{O(n \lg n)}$ applying Hadamard inequality to the private basis,

[1] These are the choices used to generate the challenges in [13] and considered in cryptanalytic attacks [24]. In fact a broader range of possible choices is considered in the paper [14].

resulting in $O(n^3 \lg n)$ and $O(n^2 \lg n)$ estimates for the key and ciphertext size. At this point one can point out how the particular way questions 1-4 were answered in [13] introduced some serious weakness in the system. In fact:

1. The only property of $\boldsymbol{R}$ required for the decryption algorithm to work is that the orthogonalized vectors $\boldsymbol{r}_{\boldsymbol{i}}^*$ are sufficiently long. Choosing them to be close to the main axes $\sqrt{n}\cdot\boldsymbol{e}_{\boldsymbol{i}}$ seems a quite peculiar restriction that might make the scheme much weaker. Other work in the design of lattice based encryption schemes [35, 10] suggests that rotations might play a fundamental role in making these schemes secure.
2. The size of the GGH challenges published at [13] are about an order of magnitude smaller than what we estimated to be the proper size. This suggests that the public key of the GGH challenges had not been "randomized" enough, making them particularly easy to break.
3. The same arguments apply to the choice of the lattice vector $\boldsymbol{Bx}$.
4. Choosing error vectors whose entries have all the same absolute value also introduces serious weaknesses, as shown in [24].

Now it shouldn't be a surprise that the numerical challenges in [13] have been broken, but it should also be clear that known attacks say very little about the general methodology used by the GGH cryptosystem. In particular, Nguyen attack [24] relies, in an essential way, on the property that all the entries in the error vector have the same absolute value and is based on the following observation: if all entries of $\boldsymbol{r}$ have absolute value σ, then $\boldsymbol{c} + \boldsymbol{s} \equiv \boldsymbol{Bx} \pmod{2\sigma}$, where $\boldsymbol{s}$ is the vector $[\sigma, \ldots, \sigma]^T$. This allows to find $\boldsymbol{x}$ modulo 2σ and reduce the problem of finding a lattice vector within distance $\|\boldsymbol{r}\|$ from $\boldsymbol{c}$ to the problem of finding a lattice vector within smaller distance $\|\boldsymbol{r}\|/(2\sigma)$ from $(\boldsymbol{c} - \boldsymbol{B}(\boldsymbol{x} \bmod 2\sigma))/(2\sigma)$. (In the GGH challenges $\sigma = 3$, reducing the length of the error by a factor 6.) As already noted in [24], removing the restriction that all entries in the error vector have the same modulus, immediately fixes the problem, and the main question is whether the security parameter in the GGH scheme can be made sufficiently large to achieve security, and maintaining the scheme practical at the same time.

4 An Optimal GGH-like Trapdoor Function

We first describe a general scheme to define GGH-like trapdoor functions, of which GGH is a special case. Then we show how to instantiate the scheme in an essentially optimal way, defining a specific trapdoor function that is both more secure and efficient than any other function from the scheme. In particular, the new trapdoor function is considerably more efficient than the original GGH.

Fix a probability distribution on the set of private bases $\boldsymbol{R}$ and let ρ be a correction radius such that using $\boldsymbol{R}$ one can correct any error of length less than ρ. (e.g. $\rho = \frac{1}{2}\min_i \|\boldsymbol{r}_{\boldsymbol{i}}^*\|$.) We define a family of functions $f_{\beta,\gamma}$ parametrized by two algorithms β and γ as follows.

1. Let β be a (possibly randomized) function that on input a matrix $\boldsymbol{R}$ outputs another basis $\boldsymbol{B}$ for the same lattice that will be used as a public key.

2. γ is a (possibly randomized) function that on input the public basis $\boldsymbol{B}$ and an error vector $\boldsymbol{r}$, outputs the coefficients $\boldsymbol{x}$ of a lattice point $\boldsymbol{Bx}$ to be added to the error vector.
3. Let $f_{\beta,\gamma}$ be the (possibly randomized) function with domain the set of vectors shorter than ρ defined as follows:

$$f_{\beta,\gamma}(\boldsymbol{r}) = \boldsymbol{Bx} + \boldsymbol{r}$$

   where $\boldsymbol{B} = \beta(\boldsymbol{R})$ and $\boldsymbol{x} = \gamma(\boldsymbol{B}, \boldsymbol{r})$.

Notice that if the input $\boldsymbol{r}$ has length less then ρ, then the basis $\boldsymbol{R}$ can be used to find the lattice point $\boldsymbol{Bx}$ closest to $f_{\beta,\gamma}(\boldsymbol{r})$ and recover $\boldsymbol{r}$.

Therefore, for any fixed β and γ, the probability distribution on $\boldsymbol{R}$ defines a family of trapdoor functions $f_{\beta,\gamma}$ with public key $(\boldsymbol{B}, \rho)$ and trapdoor information $\boldsymbol{R}$. The GGH trapdoor function can be defined as a special case of this scheme when $\beta(\boldsymbol{R})$ applies a sequence of elementary random operations to $\boldsymbol{R}$, and $\gamma(\boldsymbol{B}, \boldsymbol{r})$ outputs an integer vector $\boldsymbol{x}$ chosen at random from a sufficiently large region of space.

We are now ready to define different β and γ, that greatly increase both the security and the efficiency of the scheme. The idea is to replace the random choice of the public basis $\boldsymbol{B}$ and vector $\boldsymbol{x}$ with deterministic choices that can be formally proved to be optimal from the security point of view. In the following subsections we first give some definitions that will be useful in the sequel, then show how to compute the public basis, and finally define the new trapdoor function.

4.1 Reducing Vectors Modulo a Basis

Every lattice $L = \mathcal{L}(\boldsymbol{B})$ induces an equivalence relation over $\mathbb{Z}^n$ defined as follows: $\boldsymbol{v} \equiv_L \boldsymbol{w}$ if and only if $\boldsymbol{v} - \boldsymbol{w} \in L$. It is easy to see that for every point $\boldsymbol{v} \in \mathbb{Z}^n$, there exists a unique point $\boldsymbol{w}$ in the *orthogonalized parallelepiped* $\mathcal{P}(\boldsymbol{B}^*) = \{\sum_i x_i \boldsymbol{b}_i^* \mid 0 \leq x_i < 1\}$ such that $\boldsymbol{w} \equiv_L \boldsymbol{v}$. Vector $\boldsymbol{w}$ can be easily computed from $\boldsymbol{v}$ and $\boldsymbol{B}$ as follows. For all $i = n, \ldots, 1$ (in decreasing order), let $\alpha_i = \frac{\langle \boldsymbol{v}, \boldsymbol{b}_i^* \rangle}{\langle \boldsymbol{b}_i^*, \boldsymbol{b}_i^* \rangle}$ the component of $\boldsymbol{v}$ along $\boldsymbol{b}_i^*$ and subtract $\lfloor \alpha \rfloor \boldsymbol{b}_i$ from $\boldsymbol{v}$. Let $\boldsymbol{w}$ the final result. Since $\boldsymbol{w}$ and $\boldsymbol{v}$ differ only by integer multiples of basis vectors, we have $\boldsymbol{w} \equiv_L \boldsymbol{v}$. Moreover, it is easy to check that $\frac{\langle \boldsymbol{w}, \boldsymbol{b}_i^* \rangle}{\langle \boldsymbol{b}_i^*, \boldsymbol{b}_i^* \rangle} \in [0, 1)$ for all i, and therefore $\boldsymbol{w} \in \mathcal{P}(\boldsymbol{B}^*)$. The unique element of $\mathcal{P}(\boldsymbol{B}^*)$ congruent to $\boldsymbol{v}$ modulo L is denoted $\boldsymbol{v} \bmod \boldsymbol{B}$. Notice that although the equivalence relation $\boldsymbol{v} \equiv_L \boldsymbol{w}$ does not depend on the particular choice of the basis $\boldsymbol{B}$ for the lattice $L = \mathcal{L}(\boldsymbol{B})$, the definition of the reduced vector $(\boldsymbol{v} \bmod \boldsymbol{B})$ is basis dependent.

Pictorially, we can think the vector space $\mathbb{R}^n$ as partitioned into parallelepipeds $\{\mathcal{P}(\boldsymbol{B}^*) + \boldsymbol{z} \mid \boldsymbol{z} \in \mathcal{L}(\boldsymbol{B})\}$. Then, the reduced vector $\boldsymbol{v} \bmod \boldsymbol{B}$ is the relative position of $\boldsymbol{v}$ in the parallelepiped $\mathcal{P}(\boldsymbol{B}^*) + \boldsymbol{z}$ it belongs to. Notice that if $\boldsymbol{v}$ is an integer vector, then also $\boldsymbol{v} \bmod \boldsymbol{B}$ is an integer vector. Therefore the function $\boldsymbol{x} \mapsto (\boldsymbol{x} \bmod \boldsymbol{B})$ defines a function from $\mathbb{Z}^n$ to $\mathcal{P}(\boldsymbol{B}^*) \cap \mathbb{Z}^n$.

If $\boldsymbol{B}$ is in Hermite Normal Form, then $\boldsymbol{w} = \boldsymbol{v} \bmod \boldsymbol{B}$ is an integer vector satisfying $0 \leq w_i < b_{i,i}$. In particular $\boldsymbol{w}$ can be represented using roughly $\sum_{i=1}^{n} \lg b_{i,i} = \lg(\det(L))$ bits. This representation is essentially optimal because $\equiv_L$ induces exactly $\det(L)$ congruence classes on $\mathbb{Z}^n$.

4.2 Choosing the Public Basis

Let's consider the choice of the public basis. The private basis $\boldsymbol{R}$ we start from is an exceptionally good basis that allows to solve the closest vector problem in the lattice $\mathcal{L}(\boldsymbol{R})$ and consequently decrypt messages. We would like to transform it into another basis for the same lattice L that gives the least possible amount of information about $\boldsymbol{R}$. Instead of computing $\boldsymbol{B}$ by applying a complex random transformation to $\boldsymbol{R}$, we set the public key of the new cryptosystem to be the Hermite Normal Form $\boldsymbol{B} = \mathrm{HNF}(\boldsymbol{R})$ of $\boldsymbol{R}$. Since the $\mathrm{HNF}(\boldsymbol{R})$ depends solely on the lattice $\mathcal{L}(\boldsymbol{R})$ generated by $\boldsymbol{R}$ (and not on the the particular basis $\boldsymbol{R}$ we used to compute it), the new public key gives *no* information about the private key $\boldsymbol{R}$, other then the lattice L it generates. More formally, one can prove that any information about $\boldsymbol{R}$ that can be efficiently computed from $\boldsymbol{B} = \mathrm{HNF}(\boldsymbol{R})$ can also be efficiently computed starting from any other (possibly random) basis $\boldsymbol{B'}$. This is because if $\boldsymbol{B'}$ and $\boldsymbol{R}$ generate the same lattice $L = \mathcal{L}(\boldsymbol{B'}) = \mathcal{L}(\boldsymbol{R})$ then $\boldsymbol{B} = \mathrm{HNF}(\boldsymbol{B'}) = \mathrm{HNF}(\boldsymbol{R})$ and $\boldsymbol{B}$ can be efficiently computed from $\boldsymbol{B'}$.

4.3 Adding a "Random" Lattice Point

Let's now look at how to simulate the addition of a "random" lattice vector $\boldsymbol{Bx}$ to the error vector $\boldsymbol{r}$. Ideally, we would like $\boldsymbol{Bx}$ to be a uniformly chosen vector from L. Unfortunately this is neither a computationally feasible nor a mathematically well defined operation. However, we notice that we can achieve exactly the same result by mapping the error vector $\boldsymbol{r}$ to its equivalence class $[\boldsymbol{r}]_L$ modulo $\equiv_L$. An efficient way to do this is to use the reduced vector $\boldsymbol{r} \bmod \boldsymbol{B}$ as a representative for this class. So, instead of adding to $\boldsymbol{r}$ a random lattice point $\boldsymbol{Bx}$, we reduce $\boldsymbol{r}$ modulo the public basis $\boldsymbol{B}$ to obtain the ciphertext $\boldsymbol{c} \in \mathcal{P}(\boldsymbol{B}^*)$.

The new trapdoor function is then defined as follows:

$$f(\boldsymbol{r}) = \boldsymbol{r} \bmod \boldsymbol{B}$$

where $\boldsymbol{B} = \mathrm{HNF}(\boldsymbol{R})$ is the Hermite Normal Form of the trapdoor information $\boldsymbol{R}$. The triangular form of $\boldsymbol{B}$ also makes the trapdoor function (i.e., the reduction modulo $\boldsymbol{B}$) extremely simple. Given $\boldsymbol{r}$, the reduced vector $\boldsymbol{r} \bmod \boldsymbol{B}$ can be easily determined as follow. Compute the integer vector $\boldsymbol{x}$ one coordinate at a time (starting from x_n) using the formula

$$x_i = \left\lfloor \frac{r_i - \sum_{j>i} b_{i,j} x_j}{b_{i,i}} \right\rfloor \quad .$$

The output of the trapdoor function is $\boldsymbol{c} = \boldsymbol{r} - \boldsymbol{Bx} \equiv \boldsymbol{r} \bmod \boldsymbol{B}$. The reader can easily check that for every i, $0 \leq c_i < b_{i,i}$, i.e., the result is the unique point in the parallelepiped $\mathcal{P}(\boldsymbol{B}^*) = \{\boldsymbol{w} \mid 0 \leq w_i < b_{i,i}\}$ which is congruent to $\boldsymbol{r}$ modulo $\mathcal{L}(B)$.

4.4 The New Trapdoor Function

We now put all pieces together and define the new trapdoor function. Let $\boldsymbol{R}$ be a private basis chosen in such a way that $\rho = \frac{1}{2}\min_i \|\boldsymbol{r}_i^*\|$ is relatively big (see Fig. 1). The public basis $\boldsymbol{B}$ is the Hermite Normal Form of $\boldsymbol{R}$ (see Fig. 2). One can see that the public basis $\boldsymbol{B}$ and the corresponding orthogonalized parallelepiped $\mathcal{P}(\boldsymbol{B}^*)$ are very skewed. The public basis $\boldsymbol{B}$ defines a function with domain the set of vectors of length at most ρ (the shaded circle in Fig. 2). The result of applying the function to vector $\boldsymbol{r}$ is the point $(\boldsymbol{r} \bmod \boldsymbol{B})$ in the parallelepiped $\mathcal{P}(\boldsymbol{B}^*)$ congruent to $\boldsymbol{r}$ modulo the lattice. Notice that even if we always start from a vector $\boldsymbol{r}$ close to the origin, the result of performing the reduction operation is a point of $\mathcal{P}(\boldsymbol{B}^*)$ possibly closest to some other lattice point (see black regions in Fig. 2 and the corresponding closest lattice points). Notice that recovering the input vector $\boldsymbol{r}$ from $f(\boldsymbol{r})$ involves finding the lattice point closest to $f(\boldsymbol{r})$, which is conjectured to be infeasible using only the public basis $\boldsymbol{B}$. However the lattice vector closest to $\boldsymbol{r} \bmod \boldsymbol{B}$ can be computed using the private basis $\boldsymbol{R}$ as discussed in Sect. 2 because $\operatorname{dist}(f(\boldsymbol{r}), L) = \operatorname{dist}(\boldsymbol{r}, L) \leq \rho$. Fig. 3 shows the orthogonalized parallelepipeds $\mathcal{P}(\boldsymbol{R}^*)$ centered at every lattice point. Notice that the lattice point closest to $f(\boldsymbol{r})$ (i.e., any point of the black regions in the picture) is just the center of the parallelepiped $\mathcal{P}(\boldsymbol{R}^*)$ containing $f(\boldsymbol{r})$, which can be found using the private basis $\boldsymbol{R}$.

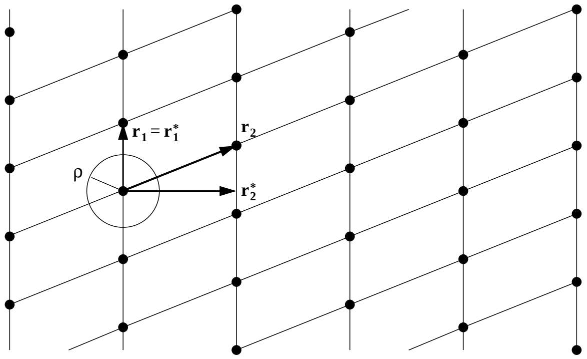

Fig. 1. A good lattice basis and the corresponding correction radius

5 Analysis

In this section we discuss the security and performance of the new scheme.

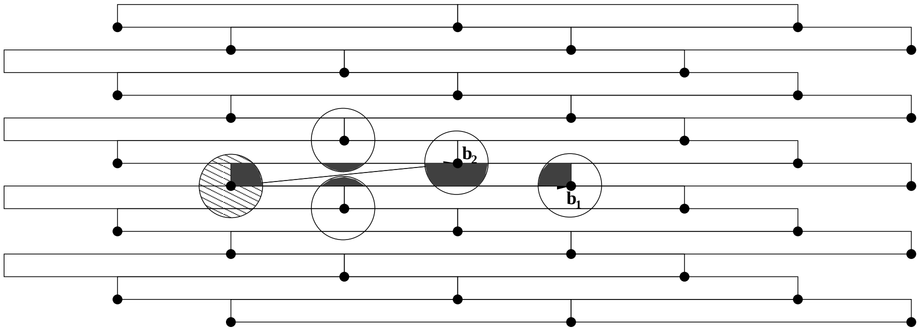

Fig. 2. HNF basis and corresponding orthogonalized parallelepiped

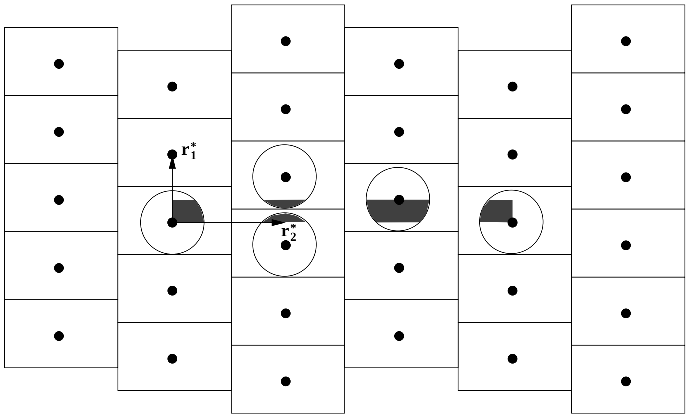

Fig. 3. Correcting small errors using the private basis

5.1 Security

We want to prove that the new trapdoor function $f(\boldsymbol{r})$ is at least as secure as the original GGH function. We actually prove that $f(\boldsymbol{r})$ is at least as secure as *any* GGH-like function $f_{\beta,\gamma}$ as defined in the previous section.

Theorem 1. *For any (efficiently computable) functions β, γ, and for any (efficient) algorithm that on input $f(\boldsymbol{r})$ finds some partial information about $\boldsymbol{r}$ with non-trivial probability, there exists an efficient algorithm that on input $f_{\beta,\gamma}(\boldsymbol{r})$ finds the same partial information with the same success probability.*

Proof. The proof is a simple reduction argument. Assume A is an algorithm that breaks f. We show how to attack $f_{\beta,\gamma}$ using A as a subroutine. We are given public basis $\boldsymbol{B} = \beta(\boldsymbol{R})$ and a ciphertext $\boldsymbol{c} = \boldsymbol{B} \cdot \boldsymbol{\gamma} + \boldsymbol{r}$. The task is to find (some partial information about) $\boldsymbol{r}$. We first compute $\boldsymbol{B'} = \mathrm{HNF}(\boldsymbol{B})$ and $\boldsymbol{c'} = \boldsymbol{c} \bmod \boldsymbol{B'}$. Notice that $\boldsymbol{B'} = \mathrm{HNF}(\boldsymbol{R})$ and $\boldsymbol{c'} = (\boldsymbol{B}\boldsymbol{\gamma} + \boldsymbol{r}) \bmod \boldsymbol{B'} = \boldsymbol{r} \bmod \boldsymbol{B'}$. Therefore $\boldsymbol{B'}$ and $\boldsymbol{c'}$ have the right distribution for algorithm A. Running A on $\boldsymbol{B'}$ and $\boldsymbol{c'}$ we will recover (the partial information about) $\boldsymbol{r}$ with the same success probability as A. □

5.2 Space Efficiency

We now analyze the size of the keys and the ciphertext of the new encryption algorithm. We assume that the private key satisfies $|r_{i,j}| < \mathit{poly}(n)$. Therefore the size of the private key can be bounded by $O(n^2 \lg n)$. Using the Hadamard inequality we can also bound the size of the determinant by $O(n \lg n)$, and using the bounds proved in Sect. 3, we get that also the public basis is $O(n^2 \lg n)$ and the ciphertext has size $O(n \lg n)$.

Estimates of the key and ciphertext sizes for the GGH and the modified scheme are shown in Fig. 1. The estimates are based on the GGH challenges published at [13]. Notice that the size of the GGH challenges is much smaller than it should have been to assure adequate randomization. This discrepancy might be explained noticing the the authors of GGH applied the LLL reduction algorithm to the public basis to somehow reduce their size. Nevertheless, the new scheme results in keys and ciphertexts more than an order of magnitude smaller than GGH. We remark that the sizes relative to the modified scheme are only upper bounds obtained using the Hadamard inequality to estimate the determinant of the lattice, and the actual sizes of the keys and ciphertexts of the modified cryptosystem can be even smaller than shown in the table. On the other hand, public-key and ciphertext size might be bigger than the estimate shown in the table if the secret key is generated differently than the GGH challenges.

5.3 Running Time

The experiments described in the next section have been performed using a highly unoptimized prototype implementation, so we do not have meaningful

Table 1. Comparison of the key and ciphertext sizes in the GGH scheme and the modified scheme. All sizes are in kilobytes (KB).

	Basis Size		Ciphertext	
dimension	GGH	New scheme	GGH	New scheme
200	330	32	2	0.16
250	620	50	3	0.20
300	990	75	4	0.25
350	1630	100	5	0.30
400	2370	140	6	0.35

experimental data on the running time of the new lattice scheme. However a few remarks regarding the running time are due.

Encryption time for GGH-like schemes is roughly proportional to the size of the public key. So, if the original GGH cryptosystem was competitive with RSA (see [14]), we should expect the new scheme to outperform RSA in terms of encryption speed because of the reduced key size.

Key Generation was one of the major problems in GGH, requiring the application of LLL on a high dimensional matrix with very large entries. Here, key generation essentially consists of a Hermite Normal Form computation, a much simpler task than lattice reduction. Moreover, recent progress on the design of HNF algorithms [23] might lead to efficient key generation procedures.

The most critical part of lattice based encryption schemes at this point is probably *decryption*. In this paper, and in our experiments we have used Babai's nearest plane algorithm [4], as we believe this is a quite natural choice. Although polynomial time, this algorithm is very slow compared to the linear time encryption procedure, and decrypting lattices in dimension 400 (using the private basis) can take several minutes with a straightforward implementation. In fact, [14] suggested to use the simpler (but less accurate) rounding off algorithm (also from [4]) instead of nearest plane. It should be noted that the nearest plane algorithm can be made considerably faster if the orthogonalized basis needed by the nearest plane algorithm is precomputed and stored as part of the secret key. Of course, this would increase the size of the secret key, but a relatively poor approximation of the orthogonalized basis should be enough to achieve reasonable correction radius. Decryption methods based on probabilistic rounding procedures as described in [18] are also an interesting alternative to be explored, and much work is still to be done. However, since the decryption procedure is strongly related to the choice of the secret basis, it is probably not worth focusing on the optimization of the decryption algorithm until we get more confidence on what's a good way to generate the private basis for the lattice. (See Sect. 6 for further discussion about the choice of the private key.)

6 Discussion and Experiments

We defined a trapdoor function f that is at least as hard to break as the GGH "encryption" scheme. Notice that although the original GGH function was a randomized one, the new function is deterministic. Since any semantically secure encryption scheme must be probabilistic [16], the new function f cannot be a secure encryption scheme in the sense of [16]. It is now clear that also GGH (which is less secure than our new function) cannot be a semantically secure encryption scheme, although it is randomized. The situation is similar to other popular "encryption" functions, like RSA: also RSA is a deterministic function, and therefore cannot be a semantically secure encryption scheme if directly applied to the message. In fact, encrypting with RSA usually involves padding the message with some random bits, or applying some other randomized procedure.

In fact, standard techniques can be used to turn trapdoor functions into semantically secure encryption schemes. In the seminal paper [16] Goldwasser and Micali showed that if h is a hard-core predicate for a trapdoor function f, then $E(b,r) = (f(r), h(r) \oplus b)$ is a semantically secure encryption function for one bit messages $b \in \{0,1\}$. Hard-core predicates from any one-way (or trapdoor) function can be easily constructed following [37, 15], giving a first theoretical construction of a semantically secure cryptosystem. The construction is easily generalized to hard-core functions. Namely, if h is a hard-core function for f, then $E(m,r) = (f(r), h(r) \oplus m)$ is a semantically secure encryption scheme for messages of length equal to the output size of h. Practical instantiations of this scheme can be obtained in the random oracle model [5], simply observing that random oracles are hard-core functions for any one-way function. See [5] also for simple constructions (still in the random oracle model) achieving security against chosen ciphertext attacks [27]. The constructions in [5] have also been recently extended in [26, 11, 12] to allow for probabilistic trapdoor functions, but these extensions are not required because our trapdoor function is deterministic.

A problem that needs further investigation is how to choose the private basis $\boldsymbol{R}$. In the GGH cryptosystem $\boldsymbol{R}$ was chosen as the sum of a multiple of the identity matrix $\sqrt{n}\boldsymbol{I}$ plus a perturbation matrix $\boldsymbol{Q}$ with small entries $q_{i,j} \in \{-4,\ldots,+4\}$. It is not clear why one should prefer this probability distribution to other distributions, and in fact we think that disclosing the approximate direction of the vectors $\boldsymbol{b_i}$ might actually weaken the system. At this point of our research, we believe it is better to leave the set of matrices from which $\boldsymbol{R}$ is chosen as big as possible. A possible way to choose $\boldsymbol{R}$ is[2] to choose each entry at random in the interval $\{-n,\ldots,n\}$. It turns out that random matrices are pretty good on average [8] and running the LLL algorithm [19] on them rapidly yields matrices $\boldsymbol{R}$ with relatively large correction radius $\rho = \min_i \|\boldsymbol{r}_{\boldsymbol{i}}^*\| = O(n)$. On the other hand, if one applies the LLL basis reduction algorithm to the public basis $\boldsymbol{B} = \mathrm{HNF}(\boldsymbol{R})$, although this time LLL takes a much longer time to terminate, the correction radius obtained is much smaller even in relatively low dimension.

[2] Similar distributions on $\boldsymbol{R}$ were already considered in [14], thought not used in the construction of the challenges [13].

In Fig. 4 we show some preliminary experimental results obtained running the LLL algorithm on random matrices, and their Hermite normal forms. We observe that random matrices give a correction radius approximately equal to $n/2$, while running the LLL algorithm on the Hermite Normal Forms of the same matrices results in a correction radius that approaches zero as the dimension of the lattice grows.

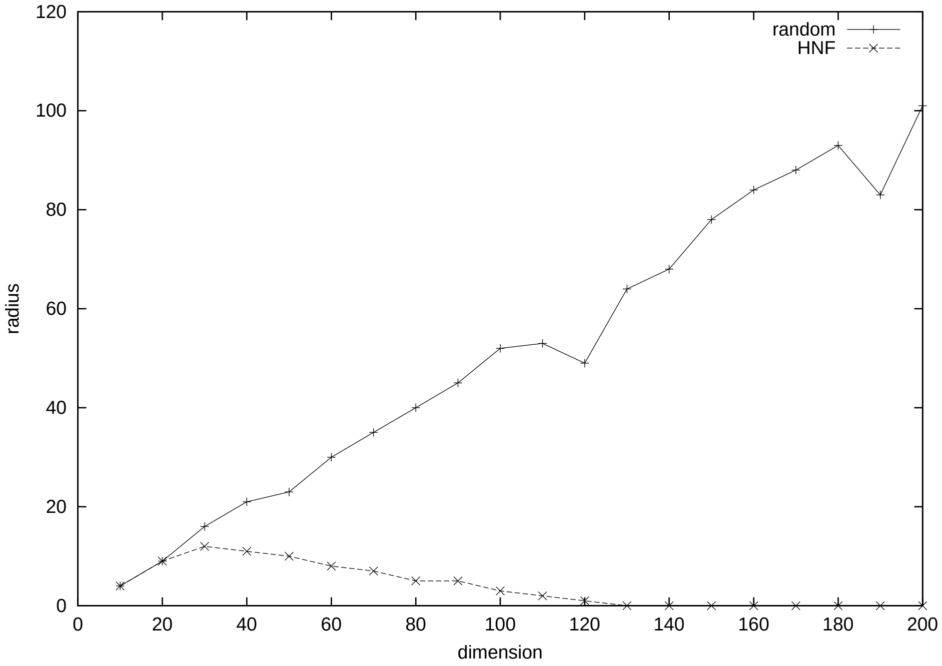

Fig. 4. Correction radius obtained applying the LLL algorithm to random matrices and their Hermite Normal Forms.

These preliminary data clearly show that applying the HNF algorithm reduces the effectiveness of lattice reduction. Still, the plot in Fig. 4 is not clear evidence of the increased security of the scheme for various reasons: first of all better result can be achieved using more sophisticated basis reduction algorithms (e.g., [29, 30, 32]) than LLL. Moreover, the correction radius is only a worst case measure of the quality of a basis. It is still possible that the HNF basis allows to recover from most small errors.

To support our claim that the modified scheme is secure against the strongest lattice attacks we performed the following experiment. We generated a private basis $\boldsymbol{R}$ in dimension 400×400 by choosing each entry at random in the interval $\{-400, \ldots, +400\}$. After running the fast LLL reduction algorithm, the correction radius of the random matrix was $\rho = 340$. The public basis $\boldsymbol{B} = \mathrm{HNF}(R)$ had size 260 KB. Notice that although the size of $\boldsymbol{B}$ is even smaller than the first GGH challenge (which was about 330 KB), our theoretical analysis suggests that the new function should still be secure because the underlying lattice has much higher dimension.

We then generated random error messages $\boldsymbol{r}$ choosing each entry at random in the interval $\{-28, \ldots, +28\}$ and computed the ciphertexts $\boldsymbol{c} = \boldsymbol{r} \bmod \boldsymbol{B}$. The size of $\boldsymbol{c}$ was around 800 B. The error vector had length 320 (i.e. less than the correction radius) and could be correctly recovered from $\boldsymbol{c}$ using $\boldsymbol{R}$. We then tried to recover $\boldsymbol{r}$ using the public basis $\boldsymbol{B}$.

Following [24], we first applied a strong basis reduction algorithm (Block Korkine-Zolotarev reduction [29] with block size 20, as implemented in Victor Shoup's Number Theory Library [33]) to the public basis $\boldsymbol{B}$ to obtain a reduced basis $\boldsymbol{G}$. The computation took over 10 days on a 700 MHz Pentium III workstation. Still the correction radius was only 6.77. Finally, we applied the "embedding technique" (see [24]) to recover $\boldsymbol{r}$ from $\boldsymbol{c}$ and $\boldsymbol{G}$ using the Block KZ reduction algorithm with block size 60 and pruning factor 14 [32]. The clear-text could not be recovered, and even after several days of computation the best lattice vector found by the attack was 500, 000 away from the target, over three orders of magnitude worse than the optimal solution $\|\boldsymbol{r}\| = 320$.

Previous cryptanalytic results [24] warn to be cautious about the security of the cryptosystem, and dimension $n = 400$ should be considered only borderline secure. In fact we suggest the dimension should be at least $n \geq 500$, but only after careful cryptanalysis the minimum dimension for which the system is secure in practice can be determined.

7 Other cryptosystems

In this section we discuss the McEliece and NTRU cryptosystems, and compare them to the general scheme presented in this paper.

7.1 Comparison with the McEliece Cryptosystem

In 1978, McEliece [21] suggested a "cryptosystem" based on the hardness of coding problems that in retrospect is very similar to the GGH cryptosystem. The proposal is to use as a secret key the generating matrix $\boldsymbol{G}$ of a Goppa code, together with a random permutation matrix $\boldsymbol{P}$ and a random invertible matrix $\boldsymbol{S}$ (over GF(2)). The public key is given by the product $\boldsymbol{G}' = \boldsymbol{PGS}$. Then the trapdoor function is defined by $f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{G}'\boldsymbol{x} + \boldsymbol{r}$ (all arithmetic performed over GF(2)), where $\boldsymbol{x}$ is a random binary vector, and $\boldsymbol{r}$ is chosen at random among the binary vectors with small Hamming weight. Let $\boldsymbol{c} = \boldsymbol{G}'\boldsymbol{x} + \boldsymbol{r}$ be the output of the trapdoor function. Using the secret key, we can first compute the permuted target $\boldsymbol{P}^{-1}\boldsymbol{c} = \boldsymbol{GS} + \boldsymbol{P}^{-1}\boldsymbol{r}$. Then we can correct from the small error $\boldsymbol{P}^{-1}\boldsymbol{r}$ using the decoding algorithm for Goppa codes, retrieving codeword $\boldsymbol{G}(\boldsymbol{Sx})$. Finally, we can compute $\boldsymbol{x}$ from $\boldsymbol{Sx}$ solving a system of linear equations over GF(2). Notice that this is essentially the same of the GGH cryptosystem with the message m encoded in the coefficients of the lattice vector. A variant of McEliece cryptosystem roughly corresponding to encoding the message in the error vector has also been proposed [25] and the two are known to be equivalent [20]. Interestingly, essentially the same techniques presented in this paper for lattices, can also be used for codes (e.g., they can be applied to [21, 25] and many of their variants). Here instead of the Hermite normal form, we use systematic form for the public code $\boldsymbol{G'}$. If $[\boldsymbol{I}|\boldsymbol{H}]^T$ is the systematic generating matrix, then the trapdoor function is given by $f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{H}^T\boldsymbol{x}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have total weight less than the correction radius of the code. However, the advantage of this transformation for codes is not as good as in the lattice case, giving only a constant (typically a factor 2) improvement over the original McEliece scheme.

A detailed comparison of lattice and coding based cryptosystems is beyond the scope of this paper. However, an apparent advantage of coding based schemes is a potentially smaller public key: since the matrix defining the code has $\{0,1\}$ entries, the size of the public key is only $O(n^2)$, as opposed to $O(n^2 \log n)$. Of course, this kind of comparisons is not necessarily meaningful if we first do not achieve a better understanding of the relation between the hardness of lattice and coding problems. For example, if decoding binary linear codes in dimension n is easier than breaking lattices in the same dimension, then for a fair comparison of the two schemes one should use different values of the security parameter. In fact, the original McEliece scheme [21] was already proposing codes with block-length $n = 1024$, and recent cryptanalytic work [7] shows that even this large dimension might be insufficient. Interestingly, the nearest codeword problem for binary (or ternary) codes can be efficiently reduced to the closest vector problem over the integers: in order to find the codeword $\boldsymbol{Cx}$ closest to $\boldsymbol{y}$, look for a lattice vector in $[\boldsymbol{C}|2\boldsymbol{I}]$ closest to $\boldsymbol{y}$. This suggests that lattice problems might be harder than coding problems, at least for the binary case. For larger alphabets, the relation between codes and lattices is less clear. However, if large alphabets are used, then the public key size for code based cryptosystems would increase. For example, if Reed-Solomon codes were used, then the alphabet size would be n, giving keys of total size $O(n^2 \log n)$, matching the asymptotic key size of lattice based cryptosystems. (Here Reed-Solomon codes are just an hypothetical example, as these codes are known not to be secure [34].) We hope that our work will stimulate further investigations on the relation between lattice and coding problems.

7.2 The NTRU cryptosystem

NTRU is a cryptosystem based on polynomial ring arithmetics proposed by Hoffstein, Pipher and Silverman in [17]. The system works essentially as follows. Let n, p, q be system parameters where n is a security parameter (say $n = 200$), p is a small prime (say, $p = 3$) and q is a relatively large prime (tipically $q = \Omega(n)$). The secret key is a pair of degree $n-1$ polynomials $a(X), b(X) \in \mathbb{Z}[X]$ with small coefficients, such that $a(X)$ is invertible modulo $(X^n - 1, pq)$. The public key is given by $c(X) = p \cdot g(X)/f(X) \bmod (X^n - 1, q)$. The encryption function takes as input two polynomials $m(X), r(X) \in \mathbb{Z}[X]/(X^N - 1)$ with small coefficients (the first interpreted as a message, and the second as a randomizer), and outputs $c(X) \cdot r(X) + m(X) \bmod (X^n - 1, q)$. (For the decription procedure, as well as a more detailed description of the system, see the original article [17].) As for the GGH and the McEliece cryptosystem, NTRU does not provide semantic security, so it is better described as a deterministic trapdoor function, instead of a full fledged probabilistic cryptosystem. Interestingly, this function can be formulated in terms of lattices as follows. Consider the lattice generated by the $2n \times 2n$ matrix

$$\left[\begin{array}{c|c} c \cdot \boldsymbol{I} & \boldsymbol{C} \\ \hline \boldsymbol{0} & \boldsymbol{I} \end{array}\right]$$

where $\boldsymbol{C}$ is the $n \times n$ matrix whose rows are given by all cyclic permutations of the coefficients $[c_0, c_1, \ldots, c_{n-1}]$ of polynomial $c(X) = \sum_{i=0}^{n} c_i X^i$, i.e., $\boldsymbol{C}_{i,j} = c_{(j-i \bmod n)}$. Notice that this public basis is in Hermite normal form. Moreover, it is easy to see that the output $c(X)r(X) + m(X)$ of the trapdoor function is exactly the result of reducing vector $[m_{n-1}, \ldots, m_0, -r_{n-1}, \ldots, -r_0]^T$ modulo the public lattice basis. So, when viewed as a lattice based trapdoor function, NTRU has the same high level structure of the functions described in this paper: the public key is an HNF lattice basis, and the function is computed reducing small "error" vector modulo the public basis. What sets NTRU apart from all other lattice based functions is the use of a class of lattices with special structure: convolutional modular lattices. While the security offered by this class of lattices is still largely to be investigated, the performance advantage is clear: as the HNF basis can be represented by only n $(\log n)$-bit numbers, the public key is much smaller than those obtained using general lattices which require $(n^2 \log n)$ bits.

8 Conclusion

We presented a new trapdoor function based on the hardness of lattice problems. The trapdoor function can be transformed into a full fledged encryption algorithm using standard techniques [16, 5]. The new function can be formally proved to be at least as secure as any other function from a general scheme to design lattice based trapdoor functions that includes the GGH trapdoor function [14] and the tensor based trapdoor function [10] as special cases. Moreover, the new function substantially improves previous proposals from the efficiency point of view: for the same level of security the new function reduces both the time and space requirements by a factor $O(n)$. The improved efficiency allows to use bigger values of the security parameter while maintaining the scheme reasonably practical. One last advantage of the new scheme is simplicity. While previous schemes computed the public key and function values using a substantial amount of randomness, in the new function these operations are substituted by simple deterministic procedures. This is important both from the theoretical and practical point of view, because it makes the algorithms easier to implement and also easier to analyze.

At this point the main question about lattice based cryptography is how to choose the private key, i.e., finding families of easily decodable lattices for which decoding becomes infeasible when the lattice is presented in Hermite normal form. We believe that in order to be really competitive with RSA, key sizes even smaller than those achieved in this paper would be desirable. However, the public key size cannot be further reduced unless one considers classes of lattices with special structure. (A simple counting argument shows that the number of lattices in a certain dimension is exponential in the bit-size representation of their Hermite normal forms, so the HNF representation is essentially optimal if one considers arbitrary lattices.) The search for easily decodable lattices for which decoding is hard when the lattice is presented in Hermite normal form becomes particularly interesting if one could find special classes of hard lattices that have small HNF representation. We observed that one such family of lattices is given by the NTRU trapdoor function. Again, the main question is security. Are the convolutional modular lattices used by NTRU really hard to decode? We hope that our work will stimulate further research on the computational complexity of decoding this and other classes of lattices.

9 Acknowledgements

The author wishes to thank Shai Halevi, Shafi Goldwasser and Salil Vadhan for many stimulating conversations on lattice based cryptography. Thanks also to the anonymous referees for their useful comments.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108, Philadelphia, Pennsylvania, 22–24 May 1996.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 284–293, El Paso, Texas, 4–6 May 1997.

[3] S. Arora, L. Babai, J. Stern, and E. Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, Apr. 1997. Preliminary version in FOCS'93.

[4] L. Babai. On Lovasz' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the first ACM Conference on Computer and Communications Security*. ACM, Nov. 1993.

[6] J.-Y. Cai and T. W. Cusick. A lattice-based public-key cryptosystem. *Information and Computation*, 151(1–2):17–31, May–June 1999.

[7] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. In K. Ohta and D. Pei, editors, *Advances in Cryptology — Proceedings of Asiacrypt 98*, volume 1514 of *Lecture Notes in Computer Science*, pages 187–199, Beijing, China, 1998.

[8] H. Daude and B. Vallée. An upper bound on the average number of iterations of the LLL algorithm. *Theoretical Computer Science*, 123(1):95–115, Jan. 1994.

[9] I. Dinur, G. Kindler, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. In *39th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, 7–10 Nov. 1998. IEEE.

[10] R. Fischlin and J.-P. Seifert. Tensor-based trapdoors for CVP and their application to public key cryptography. In *7th IMA International Conference "Cryptography and Coding"*, volume 1746 of *Lecture Notes in Computer Science*, pages 244–257. Springer-Verlag, 1999.

[11] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, University of California, Santa Barbara, Aug. 1999. IACR, Springer-Verlag.

[12] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Transaction of Fundamentals of electronic Communications and Computer Science*, E38-A(1):24–32, Jan. 2000.

[13] O. Goldreich, S. Goldwasser, and S. Halevi. The GGH cryptosystem, challenge page. `http://theory.lcs.mit.edu/~cis/lattice/challenge.html`.

[14] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In B. S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer-Verlag, 17–21 Aug. 1997.

[15] O. Goldreich and L. Levin. A hard predicate for all one-way functions. In *Proceedings of the 21st Annual Symposium on Theory of Computing (STOC)*. ACM, 1989.

[16] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sience*, 28(2):270–299, 1984. Preliminary version in STOC'82.

[17] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring based public key cryptosystem. In J. Buhler, editor, *Algorithmic Number Theory (ANTS III)*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288, Portland, OR, 1998. Springer.

[18] P. Klein. Finding the closest lattice vector when it's unusually close. In *Proceedings of the 11th Symposium on Discrete Algorithms*, San Francisco, California, Jan. 2000. SIAM.

[19] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[20] Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, Jan. 1994.

[21] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, Jet Propulsion Laboratory, Pasadena, 1978.

[22] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 2001. To Appear.

[23] D. Micciancio and B. Warinschi. A linear space algorithm for computing the Hermite Normal Form. In B. Mourrain, editor, *International Symposium on Symbolic and Algebraic Computation*. ACM, ACM, 2001. To Appear.

[24] P. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In M. Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer-Verlag, Aug. 1999.

[25] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[26] T. Okamoto and D. Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In D. Naccache, editor, *Proceedings of the Cryptographers' Track of the RSA Conference '2001 (RSA '2001)*, Lecture Notes in Computer Science, San Francisco, California, USA, 8–12Apr. 2001. Springer-Verlag.

[27] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology: Proceedings of Crypto 91*, volume 576 of *Lecture Notes in Computer Science*, University of California, Santa Barbara, Aug. 1991. IACR, Springer-Verlag.

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

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

[30] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In L. Budach, editor, *Proceedings of Fundamentals of Computation Theory*, volume 529 of *lncs*, pages 68–85. Springer-Verlag, 1991.

[31] C.-P. Schnorr, M. Fischlin, H. Koy, and A. May. Lattice attacks on GGH cryptosystem. Rump session of Crypto'97, 1997.

[32] C.-P. Schnorr and H. H. Hörner. Attacking the Chor–Rivest cryptosystem by improved lattice reduction. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology—EUROCRYPT 95*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 21–25 May 1995.

[33] V. Shoup. NTL: A library for doing number theory. URL http://www.shoup.net/ntl/index.html.

[34] V. Sidelnikov and S. Shestakov. On cryptosystems based on generalized Reed-Solomon codes. *Diskretnaya Math*, 4(3):57–63, 1992. In Russian.

[35] N. J. A. Sloane. Encryption by random rotations. In *Workshop on Cryptography Burg Feuerstein 1982*, volume 149 of *Lecture notes in computer science*, pages 71–129, 1983.

[36] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Mathematische Instituut, Universiry of Amsterdam, 1981. Available on-line at URL `http://turing.wins.uva.nl/~peter/`.

[37] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, Chicago, IL, 1982. IEEE.