Cryptanalysis of the Goldreich–Goldwasser–Halevi Cryptosystem from Crypto'97

Phong Nguyen

École Normale Supérieure, Laboratoire d'Informatique
45 rue d'Ulm, 75230 Paris Cedex 05, France
Phong.Nguyen@ens.fr
http://www.dmi.ens.fr/~pnguyen/

Abstract. Recent results of Ajtai on the hardness of lattice problems have inspired several cryptographic protocols. At Crypto '97, Goldreich, Goldwasser and Halevi proposed a public-key cryptosystem based on the closest vector problem in a lattice, which is known to be NP-hard. We show that there is a major flaw in the design of the scheme which has two implications: any ciphertext leaks information on the plaintext, and the problem of decrypting ciphertexts can be reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

1 Introduction

Historically, public-key cryptosystems have almost without exception been built on the assumed hardness of one of the three following problems: knapsacks, discrete logarithms in some groups, and integer factorization. Despite the NP-completeness of the knapsack problem, all knapsack-based systems have been broken (see the survey [24]), mainly due to the connection between lattice problems and knapsacks arising from cryptography. The narrowness of the remaining options has often been cited as a potential fragility of public-key cryptography. Recently, Ajtai [1] found a surprising worst-case/average-case connection for certain lattice problems, which caused a revival of knapsack-based cryptography [3,16,15,19,5]. In particular, two lattice-based public-key cryptosystems have received wide attention: the Ajtai-Dwork cryptosystem (AD) [3] and the Goldreich-Goldwasser-Halevi cryptosystem (GGH) [16].

The AD scheme has a fascinating property: it is provably secure unless some worst-case lattice problem can be solved in probabilistic polynomial time. The problem is a variant of the famous shortest vector problem (SVP), which refers to the question of computing a lattice vector with minimum non-zero Euclidean

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 288–304, 1999.

length. The GGH scheme relies on the non-homogeneous analog of SVP, the so-called closest vector problem (CVP) in which one has to find a lattice vector minimizing the distance to a given vector. GGH has no proven worst-case/average-case property, but it is much more practical than AD. Specifically, for security parameter n, key-size and encryption time are $O(n^2)$ for GGH, *vs.* $O(n^4)$ for AD. For RSA and El-Gamal systems, key size is $O(n)$ and computation time is $O(n^3)$. The authors of GGH argued that the increase in size of the keys was more than compensated by the decrease in computation time.

Ajtai's work [1] initiated a substantial amount of research [7,2,22,13,9,6,17] on the hardness of SVP, CVP, and related problems. We now know that SVP is NP-hard for polynomial random reductions [2], even up to some constant [22]. CVP is NP-hard [11], and approximating CVP to within almost-polynomial factors is also NP-hard [9]. In fact, SVP cannot be harder than CVP: it was recently shown [17] that one can approximate SVP to any factor in polynomial time given an approximation CVP-oracle for the same factor and dimension. On the other hand, approximating SVP or CVP to $\sqrt{n/\log(n)}$ are unlikely to be NP-hard [13] (n denotes the lattice dimension). Furthermore, there exist polynomial-time approximation algorithms [20,4,25,27], such as the celebrated LLL algorithm, that can achieve exponential bounds. And it is well-known that these reduction algorithms behave much better in practice than theoretically expected. Therefore, the practical security of AD and GGH had to be assessed.

In the case of AD, such an assessment proved to be deadly: Nguyen and Stern [23] showed that any realistic implementation of the AD scheme was insecure. However, the attack was specific to AD, and had no implications on the security of GGH. Moreover, since GGH is much more efficient than AD, it has larger security parameters: breaking GGH apparently meant solving hard lattice problems in dimensions much higher than what had previously been done, such as in cryptanalyses of knapsack systems. Based on numerous experiments, the authors of GGH conjectured that the closest vector problem arising from their scheme was intractable in practice for dimension 300 or so. To bring confidence in their scheme, they published on the Internet a series of five numerical challenges [14], in dimensions 200, 250, 300, 350 and 400. In each of these challenges, a public key and a ciphertext were given, and the challenge was to recover the plaintext.

To our knowledge, all previous attempts to solve these challenges have failed, except in dimension 200. The most successful attempts used the so-called embedding technique, which (heuristically) reduces CVP to a shortest vector problem in a lattice of similar dimension. By applying high-quality reduction algorithms, one hopes to recover the closest vector. Using this technique, Schnorr *et al.* [26] were able to decrypt ciphertexts up to dimension 150. And Nguyen used the NTL package in Oct. 97 to solve the 200-challenge in a few days (see the documentation of NTL [28]) with the improved algorithm of [27]. But higher dimensions seemed to be out of reach, confirming the predictions of the authors of GGH.

In this paper, we show that there is dangerous flaw in the GGH cryptosystem. More precisely, we observe that each encryption leaks information on the

cleartext, and this information leakage allows an attacker to reduce the problem of decrypting ciphertexts to solving particular CVP-instances which are much easier than the general problem. Namely, for these instances, the given vector is very close to the lattice, which makes it possible in practice to find the closest vector by standard techniques. As an application, we solved four out of the five Internet GGH challenges from dimension 200 to 350, in a reasonable time. In dimension 400, we obtained 1/8th of the plaintext. This proves that GGH is insecure for the parameters suggested by Goldreich, Goldwasser and Halevi. Learning the result of our experiments, one of the authors of GGH declared the scheme as "dead" [12]. We suggest modifications to fix the encryption process, but estimate that, even modified, the scheme cannot provide security without being impractical, compared to existing schemes.

The rest of the paper is organized as follows. We first review necessary material about lattices in section 2. Then we briefly describe the GGH cryptosystem. In section 4, we explain how the encryption process leaks information, using a basic observation. In section 5, we show how to exploit the information leakage to simplify the problem of decrypting ciphertexts. Section 6 presents the experiments done on the Internet challenges. And in section 7, we discuss ways to repair the scheme.

2 Background on Lattices

In the sequel, we denote vectors by bold-face lowercase letters (*e.g.* $\mathbf{b}$, $\mathbf{c}$, $\mathbf{r}$); we use capital letters (*e.g.* B, C, L) to denote matrices or sets of vectors. If $\mathbf{b}$ is a vector in $\mathbb{R}^n$, then $\lceil \mathbf{b} \rfloor$ denotes the vector in $\mathbb{Z}^n$ which is obtained by rounding each entry in $\mathbf{b}$ to the nearest integer.

2.1 Definitions

In this paper, we only care about integral lattices of full rank, so the definitions below only deal with those. Let M be a non-singular $n \times n$ integral matrix. Denote by $\mathbf{b}_1, \ldots, \mathbf{b}_n$ the row vectors of M. The *lattice* L spanned by $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ (or M) is the set $L(M)$ of all integral linear combinations of the $\mathbf{b}_i$'s. The set of $\mathbf{b}_i$'s is called a *basis* of L. We identify a basis with the square matrix whose rows are the basis vectors (note that the convention of [16] used columns instead of rows). Naturally, a lattice is a set $L \subset \mathbb{Z}^n$ for which there exists a non-singular matrix M such that $L = L(M)$. For a given lattice L, there exist many bases, which all differ by multiplication with some unimodular matrix. Thus, all bases have the same determinant in absolute value, which is called the lattice determinant $\det(L)$. In every lattice L, there is a non-zero vector whose Euclidean length is $O(\sqrt{n}\det(L)^{1/n})$. In a "random" lattice, one generally assumes that there are no non-zero vectors with substantially shorter length. The goal of lattice reduction is to find a reduced basis, that is, a basis consisting of reasonably short vectors. Define the lattice *gap* as the ratio between the second successive minimum (the smallest real number r such that there are two linearly independent lattice points

of length at most r) and the length of a shortest non-zero vector. Experiments suggest that the larger the lattice gap is, the easier reduction becomes.

2.2 Algorithmic Problems

We recall that the shortest vector problem (SVP) is: given a lattice basis, find a non-zero lattice vector with minimal Euclidean length. The closest vector problem (CVP) is: given a lattice basis and a vector $\mathbf{c} \in \mathbb{Z}^n$, find a lattice vector which minimizes the distance to $\mathbf{c}$. A related problem is the smallest basis problem (SBP): given a lattice basis, find a basis which minimizes the product of the lengths of its elements. All these problems are NP-hard. And no polynomial-time algorithm is known for approximating either SVP, CVP or SBP in $\mathbb{Z}^n$ to within a polynomial factor in n. In fact, the existence of such algorithms is an important open problem. The best polynomial time algorithms achieve exponential factors, and are based on the LLL algorithm [20]. The LLL algorithm can be viewed as an approximation algorithm for both SVP and SBP. Variants [25,27] of LLL achieve better approximation factors for SVP, but with slower running time. More precisely, Schnorr defined a family of algorithms (BKZ [25]) whose performances depend on a parameter called the blocksize. These algorithms use some kind of exhaustive search which is exponential in the blocksize. So far, the best reduction algorithms in practice are variants [27] of those BKZ-algorithms, which apply a heuristic to reduce exhaustive search.

Babai [4] showed how to use a reduced basis to approximate CVP. The more reduced the basis is, the better the approximation is. For an LLL-reduced basis, this yields an exponential factor. But in practice, the best method to solve CVP is the so-called embedding technique (see [16]), which reduces the problem to a shortest vector problem. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a lattice L, and $\mathbf{c}$ the given vector corresponding to a CVP-instance. The embedding technique builds the lattice in $\mathbb{Z}^{n+1}$ spanned by the rows of the following matrix:

$$L' = \begin{pmatrix} - & \mathbf{b}_1 & - & 0 \\ & \vdots & & \\ - & \mathbf{b}_n & - & 0 \\ - & \mathbf{c} & - & 1 \end{pmatrix}$$

L' and L have the same determinant and almost the same dimension, therefore one would expect that the shortest lattice vector of L' has about the same length than the shortest lattice vector of L. Now, assume that the vector $\mathbf{v} \in L$ minimizes the distance to $\mathbf{c}$. One can see that the vector $(\mathbf{c} - \mathbf{v}, 1) \in \mathbb{Z}^{n+1}$ is short and belongs to L'. In fact, one hopes that it is the shortest vector of L', so that one solves the CVP-instance from the SVP-instance defined by L'. Clearly, this embedding method is heuristic. Still, if $\mathbf{c}$ is very close to the lattice L, then $\mathbf{c} - \mathbf{v}$ will be much shorter than the shortest vector of L, so that L' has a large gap, and hopefully, is easy to reduce.

3 The Goldreich-Goldwasser-Halevi Cryptosystem

We make a brief description of the GGH Cryptosystem. More details can be found in [16]. In particular, we do not specify the key generation because it is unnecessary for the understanding of our attack. Roughly speaking, GGH is the lattice-analog of the McEliece [21] cryptosystem based on algebraic coding theory. In both schemes, a ciphertext is the addition of a random noise vector to a vector corresponding to the plaintext. The public key and the private key are two representations of the same object (a lattice for GGH, a linear code for McEliece). The private key has a particular structure which allows to suppress noise vectors up to a certain bound. However, the domains in which all these operations take place are vastly different.

The security parameter is $(n, \sigma) \in \mathbb{N}^2$. The integer n is much larger than σ: a typical value is $(n, \sigma) = (300, 3)$. A lattice L in $\mathbb{Z}^n$ is generated together with a reduced basis R of L. One actually generates a non-singular matrix R with short row vectors, and defines the lattice spanned by this matrix. The basis R, which is kept private, is transformed to a non-reduced basis B, which will be public. Several transformation methods were proposed in [16]. Roughly speaking, these methods multiply R by sufficiently many small unimodular matrices. Computing a basis as "good" as the private basis, given only the non-reduced basis, means approximating SBP.

The message space is a "large enough" parallelepiped in $\mathbb{Z}^n$. For instance, in the numerical challenges, cleartexts were randomly chosen in $[-128 \cdots + 127]^n$ so that they could be stored within $8n$ bits. A message $\mathbf{m} \in \mathbb{Z}^n$ is encrypted into $\mathbf{c} = \mathbf{m}B + \mathbf{e}$ where $\mathbf{e}$ is an error vector uniformly chosen from $\{-\sigma, \sigma\}^n$. In [16], other methods to embed messages into lattice points are discussed. Note, that as it stands, the cryptosystem is not semantically secure, because one can check if a ciphertext $\mathbf{c}$ corresponds to a plaintext $\mathbf{m}$ by computing $\mathbf{c} - \mathbf{m}B$. A ciphertext $\mathbf{c}$ is decrypted as $\lfloor \mathbf{c}R^{-1} \rceil RB^{-1}$ (note: this is Babai's round method [4] to solve CVP). The private basis R is generated in such a way that the decryption process succeeds with high probability. More precisely, the following result is proved in [16]:

Theorem 1. *Let R be the private basis, and denote the maximum L_∞ norm of the columns in R^{-1} by $\gamma/\sqrt{n}$. Then the probability of decryption errors is bounded by $2ne^{-1/(8\sigma^2\gamma^2)}$.*

The larger σ is, the harder the CVP-instances are expected to be. But the previous theorem shows that σ must be small for the decryption process to succeed. In practice, one chooses a private basis R, looks at the norm of the columns in R^{-1}, and then takes the maximal possible value for σ. The authors of GGH considered that the value $\sigma = 3$ was a good compromise. Based on extensive experiments (see [16]) with all known methods to solve CVP, they conjectured that the cryptosystem was secure in practice for dimension 300 or so. The most successful attack (embedding technique using improved reduction algorithm) against GGH could decrypt ciphertexts up to dimension 200, in about a few days.

We now present new results on the security of GGH, which show that GGH cannot provide sufficient security even for dimensions as high as 400. In fact, there is an intrinsic weakness in the encryption process which will prove very dangerous.

4 Leaking Remainders

Let (n, σ) be the security parameter, and B be a public basis. Assume that a message $\mathbf{m} \in \mathbb{Z}^n$ is encrypted into a ciphertext $\mathbf{c} \in \mathbb{Z}^n$ with B. There is a vector $\mathbf{e} \in \{\pm\sigma\}^n$ such that:

$$\mathbf{c} = \mathbf{m}B + \mathbf{e} \tag{1}$$

We will see that this equation defining the encryption process has a major flaw. The key to our results is to look at (1) modulo some well-chosen integer. By an appropriate choice of the modulus, the error vector $\mathbf{e}$ will disappear, from which information on $\mathbf{m}$ can be derived. Since each entry of $\mathbf{e}$ is $\pm\sigma$, a natural candidate is σ, which gives: $\mathbf{c} \equiv \mathbf{m}B \pmod{\sigma}$. But we notice that 2σ is a better modulus. Indeed, if we let $\mathbf{s} = (\sigma, \ldots, \sigma) \in \mathbb{Z}^n$, then we have $\mathbf{e}+\mathbf{s} \equiv 0 \pmod{2\sigma}$, so that:

$$\mathbf{c} + \mathbf{s} \equiv \mathbf{m}B \pmod{2\sigma} \tag{2}$$

This reads as a modular system in the unknown $\mathbf{m}$. If we can solve this system, we obtain $\mathbf{m}$ modulo 2σ, which we denote by $\mathbf{m}_{2\sigma}$. Two questions arise: how many solutions are there ? And how can we compute all of them ? We will see that with high probability, there are very few solutions, which are easy to compute. With non-negligible probability, there is even a single solution. The probability is with respect to B. Since B is obtained by randomly mixing a private basis R, and since 2σ is a small number, we assume that the entries of B modulo 2σ are uniformly and independently distributed in $\mathbb{Z}_{2\sigma}$. Recall that, in practice, the standard choice is $\sigma = 3$, so that $2\sigma = 6$.

We now discuss the general problem of solving a linear system $\mathbf{y} = \mathbf{x}B \pmod{N}$, where the vector $\mathbf{y}$, the (random) matrix B and the (small) modulus N are known and we know that there is at least one solution. Clearly, two solutions differ by an element of the kernel of B, defined as the set of $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{x}B \equiv 0 \pmod{N}$. It follows that all solutions can be found from the kernel of B and a particular solution to the system, and their number is equal to the cardinal of the kernel. If the matrix B is invertible modulo N, that is, if $\det(B)$ is coprime to N, then there is only one solution, which can be found by matrix inversion: $\mathbf{x} = \mathbf{y}B^{-1} \pmod{N}$. This is of course the simplest case. Let us see how often such a case occurs.

4.1 Invertible Matrices

The material covered in sections 4.1 and 4.2 is quite intuitive and is probably already known. Since we have not been able to locate appropriate references,

we provide proofs in the appendix. The following result gives the proportion of invertible matrices modulo N among all matrices:

Theorem 2. *Let N be some positive integer. Let $p_1, \ldots, p_\ell$ be the distinct prime factors of N. Consider the ring of $n \times n$ matrices with entries in $\mathbb{Z}_N$. Then the proportion of invertible matrices (*i.e.*, with determinant coprime to N) is equal to:*

$$\prod_{i=1}^{\ell} \prod_{k=1}^{n} (1 - p_i^{-k}).$$

Note that the above proportion converges rapidly to its limit. It follows that the proportion can be considered as constant for dimensions of interest, since those dimensions are high (higher than 200). Table 1 gives numerical results. It shows that with non-negligible probability, the public matrix B is invertible

Table 1. The proportion of invertible matrices modulo N.

Modulus N	2	3	4	5	6	7	8	9	10
%	28.9	56.0	28.9	76.0	16.2	83.7	28.9	56.0	22.0

modulo 2σ, which discloses any plaintext modulo 2σ. We now show that when the matrix is not invertible, its kernel is most of the time very small.

4.2 Matrices with Small Kernel

We first treat the case of a prime modulus p. The kernel is then a $\mathbb{Z}_p$-vector space. If d is the kernel dimension, the number of solutions is p^d. If both p and d are small, this number is small. The following theorem shows that the vast majority of non-invertible matrices modulo p have a kernel of dimension less or equal to two:

Theorem 3. *Let $\mathbb{F}_q$ be the finite field with q elements, where q is a prime power. Consider the set of $n \times n$ matrices with entries in $\mathbb{F}_q$. We have:*

1. *The proportion of matrices with one-dimensional kernel is equal to:*

$$\frac{q}{(q-1)^2}(1 - q^{-n}) \prod_{k=1}^{n} (1 - q^{-k}).$$

2. *The proportion of matrices with two-dimensional kernel is equal to:*

$$\frac{q^2}{(q-1)^2(q^2-1)^2}(1 - q^{1-n} - q^{-n} + q^{1-2n}) \prod_{k=1}^{n} (1 - q^{-k}).$$

Table 2. The proportion of square modular matrices of low-dimensional kernel.

Prime modulus p	2			3			5		
Kernel dimension	1	2	≥ 3	1	2	≥ 3	1	2	≥ 3
%	57.8	12.8	0.5	42.0	2.0	< 0.1	23.8	0.2	$\ll 0.1$

Again, we note that the above proportions converge quickly to their limit, so that the numerical results of table 2 hold for any sufficiently high dimension. It shows that with high probability, the kernel has dimension less than 2, which means that there are most p^2 solutions to the modular system. Furthermore, it is very simple to compute these solutions. One can compute a kernel basis of any matrix modulo any prime in polynomial time (see [8]). To find a particular solution of the system, one can build a $(n+1) \times (n+1)$ matrix from the image vector $\mathbf{y}$ and the matrix B, as in the embedding technique. It can be shown that a particular solution can be derived from any kernel basis of the new matrix. We refer to [8] for more details.

Now, if the modulus N is not prime but square-free (for instance, 6), the previous results allow us to conclude. Indeed, if $N = p_1 \times \cdots \times p_\ell$ where the p_i's are distinct primes, then all the solutions can be recovered by Chinese remainders from the solutions modulo each prime p_i. And the total number of solutions is obtained by multiplying the number of solutions for each prime. Furthermore, one can compute the proportion of matrices with respect to the number of elements of their kernel, from the previous proportions. Table 3 gives the results for a modulus equal to 6, which is the case of interest. It shows that only a very small minority of matrices modulo 6 have a kernel with more than 12 elements. The most probable cases are 2, 6, 1 and 3 elements. Otherwise,

Table 3. The proportion of square matrices modulo 6 with small kernel

Kernel cardinal	1	2	3	4	6	9	12	18	36	Other cases
%	16.2	32.4	12.1	7.2	24.3	0.6	5.4	1.1	0.3	0.6

if the modulus N is not square-free, the previous methods do not apply. Still, there exist methods to solve such modular systems, as a method is implemented in the Lidia package [18]. This implementation can compute all the solutions to a linear system modulo any small composite number, in about the same time required for a prime modulus. We have not been able yet to locate further references, but we believe such a classical problem has already been studied. We also believe that the number of solutions is still small most of the time, although we do not know whether this number can easily be estimated. The problem is that we have $\mathbb{Z}_N$-modules instead of vector spaces. The ring $\mathbb{Z}_N$ has zero divisors,

and all its ideals are principal ideals. The modules will not usually be free modules, so counting the number of elements is harder. We stress that the case of non-square-free numbers does not occur for the suggested choice of parameters in GGH. And if one selects different parameters for which the modulus has a square factor, then one can apply the mentioned algorithm and still hope that the number of solutions is small.

4.3 Security Implications

We saw that for the suggested choice of parameters, the public matrix B had very small kernel with high probability. Then, for any ciphertext $\mathbf{c}$, the underlying linear system has very few solutions. It follows that, even though the encryption method is probabilistic, one can check whether a given ciphertext corresponds to a given plaintext (knowing only a small fraction of the plaintext), or whether two given ciphertexts correspond to the same plaintext, with overwhelming probability. Thus, the GGH cryptosystem is far from being semantically secure. One might argue that other methods of embedding plaintexts into lattice points were proposed in [16] to make sure the scheme was semantically secure. But the only other practical method suggested to embed the message in the least-significant bits of the coordinates. Namely, instead of picking $\mathbf{m}B$ as the lattice point to be perturbed, it was suggested to pick $\mathbf{v}B$ where the message would form the least-significant bits of $\mathbf{v}$'s entries, and the remaining bits would be chosen at random. The method is even worse, since our techniques recover the remainders of $\mathbf{v}$, that is, the least-significant bits. We will discuss ways to fix the flaw in the encryption process in section 7.

5 Simplifying the Closest Vector Problem

Assume now that one knows the plaintext $\mathbf{m}$ modulo 2σ which we denote by $\mathbf{m}_{2\sigma}$. We explain how this partial information simplifies the decryption problem. Recall that we have:

$$\mathbf{c} = \mathbf{m}B + \mathbf{e}.$$

Therefore:

$$\mathbf{c} - \mathbf{m}_{2\sigma}B = (\mathbf{m} - \mathbf{m}_{2\sigma})B + \mathbf{e}.$$

The vector $\mathbf{m} - \mathbf{m}_{2\sigma}$ is of the form $2\sigma\mathbf{m}'$ where $\mathbf{m}' \in \mathbb{Z}^n$. It follows that:

$$\frac{\mathbf{c} - \mathbf{m}_{2\sigma}B}{2\sigma} = \mathbf{m}'B + \frac{\mathbf{e}}{2\sigma}.$$

The rational point $\frac{\mathbf{c}-\mathbf{m}_{2\sigma}B}{2\sigma}$ is known, so that the previous equation reads as a closest vector problem for which the error vector $\mathbf{e}/(2\sigma) \in \{\pm\frac{1}{2}\}^n$ is much smaller. The error vector length is now $\sqrt{n/4}$, compared to $\sigma\sqrt{n}$ previously. And if one can solve the new CVP-instance, one can easily solve the former CVP-instance, due to the relationship between the two error vectors. In other

words, we have reduced the problem of decrypting ciphertexts (that is, CVP-instances for which the error vector has entries $\pm\sigma$), to a simpler CVP-problem for which the error vector has entries $\pm\frac{1}{2}$. At this point, we note that there might exist specialized CVP-algorithms to solve such CVP-instances. But we can also apply traditional methods such as the embedding technique, which are more likely to work now that the error vector is smaller.

The previous section showed that one was not guaranteed to obtain $\mathbf{m}_{2\sigma}$ (the message modulo 2σ). However, with high probability, one can confine $\mathbf{m}_{2\sigma}$ to a very small set which can be computed. It follows that the general problem of decrypting ciphertexts can be reduced to solving a very small number of CVP-instances, among which one is easier than the former CVP-instance. Note that it is not necessary to solve all the CVP-instances: it suffices to solve the right one. In practice, this means reducing several lattices in parallel until one of them provides the solution.

The fact that the new CVP-instance involves rational points is not a problem. Rationals can be avoided by multiplying by two the equation defining the CVP-instance. One thus obtains an integer CVP-instance for which the error vector has entries ± 1 and the lattice is the original lattice with doubled entries.

Another way to avoid rational arithmetic is to make the error vector a multiple of 2σ by translation. For instance, letting again $\mathbf{s} = (\sigma, \ldots, \sigma) \in \mathbb{Z}^n$, we have: $\mathbf{c} + \mathbf{s} - \mathbf{m}_{2\sigma}B = (\mathbf{m} - \mathbf{m}_{2\sigma})B + \mathbf{e} + \mathbf{s}$. The vector $\mathbf{e} + \mathbf{s}$ is of the form $2\sigma\mathbf{e}'$, where $\mathbf{e}' \in \{0, 1\}^n$. It follows that the vector $\mathbf{c} + \mathbf{s} - \mathbf{m}_{2\sigma}B$ is of the form $2\sigma\mathbf{c}'$, where $\mathbf{c}' \in \mathbb{Z}^n$. Therefore, dividing by 2σ, one gets: $\mathbf{c}' = \mathbf{m}'B + \mathbf{e}'$. This is a closest vector problem for which the error vector has expected length $\sqrt{n/2}$, which is slightly worse than $\sqrt{n/4}$.

6 Experiments on the Internet Challenges

To validate our results, we tested the method on the Internet challenges [14] provided by the authors of the cryptosystem, because the hardness of the CVP-instances partly depends on the way the private basis is transformed to the public basis. For each challenge, the cleartext was a vector of $\mathbb{Z}^n$ with entries in $[-128 \cdots + 127]$, and σ was equal to 3. The private basis was generated by multiplying the $n \times n$ identity matrix by $4\lceil\sqrt{n} + 1\rfloor$, and then adding to each entry a random integer chosen from $[-4 \cdots + 3]$. Although we do not know the private basis, we can estimate the length of the shortest vector by $\sqrt{(4\lceil\sqrt{n} + 1\rfloor)^2 + (n-1) \times 11/2}$. This allows to estimate the gap of the embedded lattice, so that we get a feeling on the hardness of the SVP-instance defined by the embedding attack. The larger the gap is, the easier the reduction is expected to be.

We implemented the attack using the NTL library [28] developed by Victor Shoup. Timings are given for a 500-MHz 64-bit DEC Alpha running on Linux. For all experiments, we had to use the so-called Givens floating-point variants of reduction algorithms, which are slower than the standard floating-point variants but less prone to stability problems. The results are summarized in table 4 and

5. Table 4 refers to an error vector in $\{\pm\frac{1}{2}\}^n$, whereas table 5 refers to an error vector in $\{0,1\}^n$. In each dimension, we first solved the linear system modulo 6 described in section 4. The number of solutions is indicated in the corresponding row. It is worth noting that these numbers correspond to what was theoretically predicted by table 3 of section 4. For each solution, we built the corresponding CVP-instance described in section 5. We tried to solve these CVP-instances using the embedding technique and improved reduction algorithms. The *Expected gap* row gives an approximation of the embedded lattice gap. Of course, all the reductions were performed in parallel: each workstation took care of a single CVP-instance, and the running time is with respect to the workstation who found the solution. In all our experiments, at most six workstations were required at the same time. The reduction process was the following: apply a LLL reduction; then a BKZ reduction [25] with blocksize 20; and if necessary, a pruned-BKZ reduction [27] with blocksize 60 and pruning factor 14. From dimension 200 to 300, the embedding used the public basis: it is likely that the running times could have been decreased if we had started after reduction of the public basis. Therefore, the running times should not be considered as optimal. We stress that it is not necessary to perform a complete BKZ-reduction: one just keeps reducing until the correct solution has been found (that is the time indicated in the *Time* row). In our experiments, the ratio between the total reduction time and the actual time for which the basis was sufficiently reduced to provide the solution is around 3. Thus, stopping the reduction as soon as the solution is found significantly reduces the running time, which is not very surprising. In dimension 350, the embedding used the public basis after BKZ-reduction with blocksize 20, and the corresponding running time does not include the reduction of the public basis. This is because we were not able to recover the plaintext after 4 days of computation, when the embedding started with the public basis (and not the public basis reduced). We are unable to explain this phenomenon. In fact, the behavior of lattice reduction algorithms is still not very well understood.

Table 4. Experiments on the Internet Challenges, with a $\{\pm\frac{1}{2}\}$-error.

Dimension	200	250	300	350	400
Number of solutions mod 6	2	2	6	6	1
Expected gap	9.7	9.4	9.5	9.4	9.6
Block size	20	20	20	60	60
Type of reduction	BKZ	BKZ	BKZ	Pruning	Pruning
Time in minutes	30	60	165	270	Unsolved

The results demonstrate the power of the new attack. We are able to solve all challenges except in dimension 400, in a reasonable time, although challenges in dimension 300 and 350 were assumed to be intractable. We even obtained information on the plaintext in dimension 400, since we recovered all the remainders

Table 5. Experiments on the Internet Challenges, with a $\{0, 1\}$-error.

Dimension	200	250	300	350	400
Expected gap	6.9	6.6	6.7	6.6	6.8
Block size	20	20	20	60	60
Type of reduction	BKZ	BKZ	BKZ	Pruning	Pruning
Time in minutes	30	60	240	1290	Unsolved

modulo 6. The 200-challenge took only 30 minutes, whereas previous methods required a few days. Challenges in dimension 250 and 300 take respectively 1 and 3 hours, and these timings are not even optimal. The running time in dimension 350 is much larger than in dimension 300 (if we take into account the reduction of the public basis) because a stronger reduction algorithm was required. It is hard to guess what the required time would be for higher dimensions.

Our results suggest that in order to be secure, GGH would require working in dimensions at least higher than 400. But already in dimension 400, the scheme is not really practical: for the 400-challenge, the public key takes 1.8 Mbytes, and the ciphertext takes 6.4 Kbytes, which represents a message expansion of 16.6. And each encryption requires $400^2 \approx 2^{17}$ multiplications with numbers of bit-length up to 129.

7 Repairing the Scheme

Our attack used two "qualitatively different" weaknesses of the scheme. The first one is inherent to the GGH construction: the error vectors are always quite shorter than the vectors in the lattice. This results in a gap in the embedded lattice, which was exploited in previous embedding attacks (success up to dimension 200). There seems to be no easy way to fix this problem, making CVP-instances arising from GGH easier than general CVP-instances. The second weakness is the particular form of the error vectors in the encryption process. This can be fixed by choosing the error vector $\mathbf{e}$ in such a manner that an attacker no longer knows the value of $\mathbf{e}$ modulo some well-chosen integer, and theorem 1 is valid. A careful analysis of the proof of theorem 1 suggests that the theorem remains correct when the entries of the error vector are less than σ in absolute value, and have zero mean. Therefore, the most natural way to prevent the flaw is to choose the entries of the error vector $\mathbf{e}$ at random in $[-\sigma \cdots + \sigma]$ instead of $\{\pm\sigma\}$. The drawback is that the error vector is smaller now. Indeed, if the entries of $\mathbf{e}$ are uniformly chosen from $[-\sigma \cdots + \sigma]$, then the expected length of $\|\mathbf{e}\|$ is approximately $\sigma\sqrt{n/3}$ instead of $\sigma\sqrt{n}$. One can obtain a larger error by choosing the entries at random in $\{\pm\sigma, \pm(\sigma - 1)\}$ instead, but the special form of the vector might be dangerous.

In any case, the error vector is smaller than in the original scheme, which makes the scheme more vulnerable to the first weakness. If we choose the entries in $[-\sigma \cdots + \sigma]$, the gap of the embedded lattice is around 3. Our experiments

showed that similar CVP-instances could be solved by a single computer when the gap was slightly less than 7, up to dimension 350. So it might be reasonable to assume that using larger resources, CVP-instances with a gap of 3 can be solved up to about the same dimension. And if one believes that the problem is much harder, one has to keep in mind that such problems can be solved in dimension 200 with small resources. Unfortunately, GGH would need such small dimensions to hope to be competitive with existing public-key cryptosystems. In fact, in early drafts of GGH [16], the proposed dimension was 150-200. Furthermore, one should not forget about possible improvements over current lattice reduction algorithms. Hence, we feel that it would be dangerous to use a dimension less than 400 even if we fix the flaw, which clearly limits interests in the scheme, even compared to the McEliece cryptosystem. Actually, we do not suggest any precise dimension, because we feel that the lattice reduction area is not understood enough yet.

Finally, we note that the hardness of the GGH CVP-instances is somehow related to the hardness of the problem arising in Ajtai's worst-case/average-case connection [1], and in the AD security proof [3]. The problem is a shortest vector problem in a lattice for which the gap is polynomial in the dimension. Our experiments showed that SVP could effectively be solved for a class of lattices with small gap (around 7), up to dimension 350. Although our class of lattices is particular, it seems to suggest that Ajtai's problem might be tractable up to moderate dimensions. It would be nice to assess the hardness of such problems, both from a theoretical and a practical point of view.

8 Conclusion

We showed that the GGH cryptosystem had a major flaw. The special form of the error vector in GGH CVP-instances is dangerous: partial information on plaintexts can be recovered, and the problem of decrypting ciphertexts can be reduced to solving CVP-instances much easier than the general problem. We demonstrated the effectiveness of these results by solving all the numerical challenges proposed by the authors of the GGH scheme, except in the highest dimension 400. Two of those challenges were conjectured to be intractable. There exist simple ways to prevent the flaw, but even modified, we estimate that the scheme cannot achieve sufficient security without large parameters. Our experiments seem to indicate that, in practice, hard lattice problems can be solved up to high dimensions. We feel that, for the moment, it is risky to speculate on the practical performances of the best lattice-reduction algorithms, because their behavior is still not well understood. Our results suggest that, unless major improvements are found, lattice-based cryptography cannot provide a serious alternative to existing public-key encryption algorithms and digital signatures such as RSA and DSS.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments. We are very grateful to Victor Shoup for developing the Num-

ber Theory Library, and in particular, its efficient and easy-to-use lattice reduction module.

References

1. M. Ajtai. Generating hard instances of lattice problems. In *Proc. 28th ACM STOC*, pages 99–108, 1996. Available at [10] as TR96-007.
2. M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th ACM STOC*, 1998. Available at [10] as TR97-047.
3. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM STOC*, pages 284–293, 1997. Available at [10] as TR96-065.
4. L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.
5. M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *Proc. of Eurocrypt '97*, volume 1233 of *LNCS*. Springer-Verlag, 1997.
6. J. Blömer and J.-P. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proc. 31th ACM STOC*, 1999. To appear.
7. J.-Y. Cai and A. P. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proc. 38th IEEE FOCS*, pages 468–477, 1997.
8. H. Cohen. *A course in computational algebraic number theory*. Springer, 1995. 2nd Edition.
9. I. Dinur, G. Kindler, and S. Safra. Approximating-CVP to within almost-polynomial factors is NP-hard. In *Proc. 39th IEEE FOCS*, pages 99–109, 1998. Available at [10] as TR98-048.
10. ECCC. `http://www.eccc.uni-trier.de/eccc/`. The Electronic Colloquium on Computational Complexity.
11. P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981. Report 81-04.
12. O. Goldreich. Private communication, Jan. 99.
13. O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proc. 30th ACM STOC*, 1998. Available at [10] as TR97-031.
14. O. Goldreich, S. Goldwasser, and S. Halevi. Challenges for the GGH-Cryptosystem. Available at `http://theory.lcs.mit.edu/~shaih/challenge.html`.
15. O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Available at [10] as TR96-056., 1996.
16. O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto'97*, volume 1294 of *LNCS*, pages 112–131. Springer-Verlag, 1997. Available at [10] as TR96-056.
17. O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. Available at [10] as TR99-002.
18. The LIDIA Group. LiDIA - a library for computational number theory. Can be obtained at `http://www-jb.cs.uni-sb.de/LiDIA/linkhtml/lidia/lidia.html`.
19. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

20. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
21. R.J. McEliece. A public-key cryptosystem based on algebraic number theory. Technical report, Jet Propulsion Laboratory, 1978. DSN Progress Report 42-44.
22. D. Micciancio. The shortest vector problem is NP-hard to approximate within some constant. In *Proc. 39th IEEE FOCS*, 1998. Available at [10] as TR98-016.
23. P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork Cryptosystem. In *Proc. of Crypto '98*, volume 1462 of *LNCS*, pages 223–242. Springer-Verlag, 1998.
24. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*, pages 75–88. A.M.S., 1990.
25. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
26. C-P. Schnorr, M. Fischlin, H. Koy, and A. May. Lattice attacks on GGH-Cryptosystem. Rump session of Crypto '97.
27. C.P. Schnorr and H.H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt'95*, volume 921 of *LNCS*, pages 1–12. Springer-Verlag, 1995.
28. V. Shoup. Number Theory C++ Library (NTL) version 3.6. Can be obtained at `http://www.shoup.net/ntl/`.

A The Solutions to the Internet Challenges

For completeness, we provide the solutions to the numerical challenges [14] published on the Internet by Goldreich, Goldwasser and Halevi. The message entries are given from left to right, and top to bottom.

A.1 Dimension 200

The message is:

```
-37  -46   73   73  -55  -65   74  100   38   71   87  110 -113  -10  109   70   36  116 -114  -94  -38   32   59    6   54
 86  -81   48  108  -24   79  -57   58   25 -112 -124   88   90  104   -1   33   64  -19   49  -73  -38   -9   92  -49    6
126   41  -90   57  -80   92   16  -33   12   -3   72  -36   68    3  117   85    7   77   48  -16  -52  -47  -80   58 -125
-25 -108   -5  -61  -28 -127  -62 -115  -89  124  -67 -124   13   29   17 -118  -26  109   79  105   98   36  -16   48  -44
 96  125   38 -112   55   41   -8   76  -91   60  -80  -90  126  -66  -50  122   -4  -45    7 -102  100   18    0   81  -31
-23   52 -123   90  -28  -38   58  -30 -128   75 -103   42  -60  102   79 -128 -105  118  127  -43  -59  122   82   24    1
108   -4 -108  -20  -50  -11   86 -125   -6   48  -24   84  -21   74   85  -74  100   -1   -5   74  -49   -5   98  -58   -6
-73   11  -11 -118  -93  118  118   32 -117  -29  111    0  -70  114 122   106  -38   79  -42  -92   37   12 -120  -91 -120
```

A.2 Dimension 250

The message is:

```
 77  -81 -118 -123  -22  -46  120   85   70   27 -123   78   18  -21   46 -122  -64   91  115   -7    7   55   92  -71  -77
 90  122   72  119   62  108   68  110  119   73  -39  -54   65  -82 -112  -35  -76   95  112   31   13  118   96  -23  106
-38  112   33  -74  -86   85 -111  -92   29 -120   99   10   76   82    1   22  -85   75  -40  -39   92   53   13   59   38
-83  -56   29 -114   50    7  -23   34  -88   31   77  125   48  114   27   57   85  -91    6   39   39 -100   83  -14  -11
-84  -50  -85   58 -118   81  104  -45  -18  119 -123  117  -32   40  -98 -128  -11 -100   49  -25   55  -22  -67   93  112
101    4 -115   56  119    2  101  -58   45   32  -48   -1    8   35  110    0   41  100   96  -47 -126   96   71  -98   18
-82   86  124  -20   51  109  -46  -73  123 -117  -81   -3  112  -11  -85   17   69   43 -103  -23   25   26 -110   -3   -5
-28    0  -37  -85   31  110  -38  -11  106   70  -88   87 -103  -32   82  -92   15  -48 -108 -124   -4   38   74  -88   64
-76   65  -38  -58  -65   85   42   64  -79   85  -33  -97  -81   84 -118  118  124   98 -113  -35   52  -77  108 -123  -56
113    2  110   59   42   46  111  -20 -119  -74   44   94   97  -20 -112   55   75  -81  103   31  -71   93   28 -101  109
```

A.3 Dimension 300

The message is:

```
  3   31  -96   68  -80  122  -73  -19  -74   94 -115  -83  -46   59  -39   84  117  -62 -119  -83   90 -108 -112  -56  -33
-25  114  -46  107   90  -33  -18   -7   -1  -77   42   -6  -22 -104   48  -55  -90   94   28   98  -72  -15   87   -6   -5
-124  -44   15 -108   28  110   -4   14  -64  -24  -23   32   86   99  -97 -119   14   26  -12   38  -53   62   77  -87  -38
 47  -31   76    6   91  -57   10  -80   86   31   76   69 -100  -37 -122    4   69   38  -37   40   70  -27  -73  -32   89
 93   43   24   42   84  115  -39  -74   63   96 -110 -122  -21  -62   92 -118 -114   33  -89  -22   39   43  -81  -50 -122
 88 -108  -21 -113  -12  -59  109  -97   93   24  116   81  114  -86   16   82  -68   22  -67   -2  -13  -56   13 -107  -17
120   61 -102  -89 -117  -95 -128  -96   12 -112   20   82  125   52   48   21   40    1    7   83 -111  -39 -112   40   23
 14 -100   95 -100  -79  -49 -108  111  -22  -68  123   11  -67 -101 -104  -51   47  106  -54  -29   26   96 -116 -100  -24
 95  -83   66  112   86  -39   -1  114  -71   27  -92    8   48 -109  114  109 -114   -2  -86  -87   22   -9   88    1   66
-68 -100   34  -55  -72 -117  -87  -26   77 -103  -68  -90   24  -81  -32  -76  -45  105  101  -26   91   82  -12  -38  124
 30  113  -12   -9  -14  -74   52   15   89  125   72  100  -90  -82   49  -64  -21   88   89   26  -72 -115  109  -95  114
-44  125   68   73  -40   65  -25  -55  -75   94   60  -21   18  -53  -59   16 -109  -87   55  -62   91  119   46   51   80
```

A.4 Dimension 350

The message is:

```
-23  -12  114  123   49  -17   17    3  116  -93   54   27  121   83  123  126  125  -70  -25  -67   80   -8   75 -112  -18
-46  -99   51  -69   70   -1   37  -70  -14   32  108   98  -79  -16   87  -43  -90  -14  -50  122  -19  -51  119   40   52
-76   -7  -83 -128    9   28  -46   39  -48   14  -19   80  -76   40  -62   85   21  -91 -121 -123  124  -35  -84  -17   43
-90   92  121  -98    5   46  -45  126   91  -45    8   -8   38  -81   72  -76   29   24  -23  -58   91  -66   91 -127  -58
 97  126   35 -114  -19  -49  -76  -54  -56  -45   79  118   38   78   82  121  -42  -53   31    6 -109   84  -93   44  -66
-23    8   -3  -59    9   67  -90    7  -26  -76  116   53  -23  -66  126   60 -115  117  -29   92   71  -36  -78 -110  124
 56  -90   81   92   82 -112  -59  -38   13   10  100  -48   49  -21   55  101  -32  -19  -49   30  -21 -116   44  -32  -17
  8  -88   76   59  -69   73  -13  -30   26  -49  -75   42   21   16  -72  -96  -12    8   81   96  -65   55  -64   44 -121
 95 -104 -109   11   -7    3 -108   33  -49   80   93 -104  -60  -65   51   20  -11  -34  -87 -123   22  -54   -7  -98 -101
-39  -34   83   26 -117   90   -7  -93  110 -124   28  -15 -103  -66   65  105 -100   89  -83  -37   13   66   80  107  107
 86    1  -75  -49   32  -47   41  126  -91  -61 -119 -128   60  -83  110   65   73  -32   91 -120  -95   69  -92  123  115
  0    9   53  -47  -12   32  -89  118   86   -9   22  -88  -96   20  -51  -28   30  -50  -95   75  -67   98   21   29   62
 29   63    3   66   59  -10  -62  -60  -84   19   57   77   59   47   35  -78  -59   76  -46   90 -102   55   -7  -24   88
-59   37   59   90  -61   -7   -8 -126  124  -70   62  -13   -3 -125   31 -112   60  -20   75  -21 -112   126 48   92  -47
```

A.5 Dimension 400

The message modulo 6 is:

```
5 1 1 3 2 0 3 1 1 3 2 1 5 3 0 4 2 4 4 2 4 1 1 4 0 0 5 2 3 1 1 0 5 5 0 5 3 0 3 3 2 4 0 3 4 2 0 4 3 0 5 5 3 2 5 1 1 1 2 2 4 5 5
1 3 4 3 4 0 3 3 1 3 0 3 5 5 5 2 0 2 3 1 4 2 5 2 1 2 0 0 5 4 3 3 4 4 5 5 3 4 4 0 0 3 5 4 0 1 2 2 1 2 2 2 2 3 2 0 4 5 4 0 2 3 1
2 4 4 5 3 5 2 1 3 1 2 5 4 1 4 4 1 5 2 2 4 4 0 2 5 4 4 3 3 4 2 4 0 3 0 1 0 1 4 0 1 5 4 1 5 4 1 3 2 2 1 2 3 0 2 5 1 4 1 0 1 0 3
3 5 1 3 2 4 0 0 1 3 0 1 5 3 1 0 1 0 5 1 1 2 1 5 1 2 2 5 5 4 5 1 2 2 2 0 5 4 4 3 4 2 0 1 4 5 0 1 1 2 0 1 2 4 2 1 4 1 3 0 4 4 0
2 4 5 1 0 0 3 1 3 2 5 0 2 0 5 2 0 3 4 3 3 0 2 0 1 2 1 0 2 4 3 3 5 0 2 3 3 2 0 4 2 2 3 3 1 4 1 3 5 4 3 1 1 4 4 0 2 3 3 2 4 5 4
5 1 2 0 3 0 3 3 1 3 4 0 0 5 0 2 2 0 3 0 5 3 2 2 3 2 4 4 2 5 4 5 5 3 2 4 1 4 5 4 3 0 1 2 3 3 0 2 2 5 4 0 0 5 4 0 3 1 1 4 2 3 0
5 2 4 0 0 4 4 3 0 2 0 0 1 2 2 2 3 5 2 5 4 5
```

B Proofs for Section 4

B.1 Proof of Theorem 2

We notice that it suffices to prove the statement for prime powers, by Chinese remainders. Thus, let N be a prime power q^α. Since numbers not coprime to N are exactly multiples of q, singular matrices are the matrices for which the determinant is divisible by q. It follows that the proportion of invertible matrices in $\mathcal{M}_n(\mathbb{Z}_{q^\alpha})$ is exactly the proportion of invertible matrices in $\mathcal{M}_n(\mathbb{Z}_q)$. Denote by $\mathbb{F}_q$ the finite field with q elements. The number of invertible matrices in $\mathcal{M}_n(\mathbb{F}_q)$ is equal to the number of families $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of linearly independent vectors. A simple counting argument shows that this number is equal to:

$$\prod_{k=0}^{n-1}(q^n - q^k) = q^{n^2}\prod_{k=1}^{n}(1 - q^{-k})$$

B.2 Proof of Theorem 3

We count the number of matrices with one-dimensional kernel by the number of families $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank $n-1$. Denote by B_k the subspace spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_k$, with the convention B_0 being the nullspace. Recall that a k-dimensional subspace has cardinality q^k. For each family $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank $n-1$, there exists a unique i such that $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ are linearly independent, $\mathbf{b}_i \in B_{i-1}$, and for all $j > i$, $\mathbf{b}_j \notin B_{j-1}$. There are $\prod_{k=0}^{i-2}(q^n - q^k)$ possibilities for $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. There are q^{i-1} choices for $\mathbf{b}_i$. And there are $\prod_{k=i-1}^{n-2}(q^n - q^k)$ possibilities for $(\mathbf{b}_{i+1}, \ldots, \mathbf{b}_n)$. It follows that the total number of families is:

$$\sum_{i=1}^{n} q^{i-1} \prod_{k=0}^{n-2}(q^n - q^k) = \frac{(q^n-1)q^{n^2}}{(q-1)(q^n - q^{n-1})} \prod_{k=1}^{n}(1 - q^{-k}).$$

Now, consider a family $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank $n-2$. There exists a unique (i_1, i_2) with $i_1 < i_2$ such that: $\mathbf{b}_1, \ldots, \mathbf{b}_{i_1-1}$ are linearly independent, $\mathbf{b}_{i_1} \in B_{i_1-1}$, for all $i_1 < j < i_2$, $\mathbf{b}_j \notin B_{j-1}$, $\mathbf{b}_{i_2} \in B_{i_2-1}$, and for all $j > i_2$, $\mathbf{b}_j \notin B_{j-1}$. That way, we know the dimension of B_j for all j, and therefore, the number of $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ corresponding to a given (i_1, i_2) is:

$$\prod_{k=0}^{i_1-2}(q^n - q^k) \times q^{i_1-1} \times \prod_{k=i_1-1}^{i_2-3}(q^n - q^k) \times q^{i_2-2} \times \prod_{k=i_2-2}^{n-3}(q^n - q^k).$$

It follows that the total number of families of rank $n-2$ is:

$$\prod_{k=0}^{n-3}(q^n - q^k) \times \sum_{i_1=1}^{n-1} \sum_{i_2=i_1+1}^{n} q^{i_1+i_2-3}.$$

The double sum is equal to:

$$\sum_{i_1=1}^{n-1} \frac{q^{i_1+n-2} - q^{2i_1-2}}{q-1} = \frac{q^{2n-1} - q^n - q^{n-1} + 1}{(q-1)(q^2-1)}.$$

Therefore, the number of families is:

$$\frac{q^{2n-1} - q^n - q^{n-1} + 1}{(q^n - q^{n-1})(q^n - q^{n-2})(q-1)(q^2-1)} \prod_{k=0}^{n-1}(q^n - q^k).$$

The result follows after a few simplifications.