Dimension Reduction Methods for Convolution Modular Lattices

Alexander May and Joseph H. Silverman

[1] Department of Mathematics and Computer Science, University of Paderborn, 33095 Paderborn, Germany, alexx@uni-paderborn.de

[2] NTRU Cryptosystems, Inc., 5 Burlington Woods, Burlington, MA 01803 USA, and Mathematics Department, Brown University, Providence, RI 02912 USA
jhs@ntru.com, jhs@math.brown.edu

Abstract. We describe a dimension reduction method for convolution modular lattices. Its effectiveness and implications for parallel and distributed computing are analyzed.

Keywords: lattice reduction, cryptography, convolution modular lattice

Introduction

The theory of lattices provides a tempting source of hard problems for cryptography, because the linearity of lattice operations offers speed advantages over cryptosystems based on other known hard problems. Two proposed cryptosystems based on (approximate) shortest and closest vector problems are GGH [4] and NTRU [6]. Unfortunately (for cryptographers), current lattice reduction methods such as LLL and its variants [11–13, 18, 19] are quite effective at finding short vectors in lattices of moderate dimension. This renders the GGH cryptosystem impractical [17], since its public keys consist of a full basis for the lattice, so the key size is proportional to the square of the lattice dimension.

The NTRU public key cryptosystem [6] and the related NSS signature scheme [7, 8] overcome this difficulty by using a class of lattices in which an entire basis can be specified by a single vector. These *convolution modular lattices* are defined using convolution products and reduction modulo q, and the resulting cryptosystems have key size that is proportional to $n \log q$ for a lattice of dimension n.

Convolution modular lattices have a cyclic structure that makes them nice to use in cryptography, but that same structure is a potential liability, since it offers an additional avenue for solving the underlying hard lattice problem. In this note we explore a method, originally proposed by the first author [14, 15], that exploits the cyclic structure in order to decrease the dimension of the underlying lattice. Since the solution time is roughly proportional to the dimension of the lattice, this offers the possibility of significant savings; and indeed in low dimensions, the method is fairly effective. However, in higher dimensions such as those used in commercial implementations of NTRU [9], the speedup does not significantly affect basic security estimates. We also find that although the dimension reduction method can be parallelized, the effect is not linear in the number of processors.

1 Convolution Products and Convolution Modular Lattices

The *convolution product* of two vectors

$$\mathbf{u} = [u_0, u_1, \ldots, u_{N-1}] \in \mathbb{Z}^N \qquad \text{and} \qquad \mathbf{v} = [v_0, v_1, \ldots, v_{N-1}] \in \mathbb{Z}^N$$

is the vector $\mathbf{w} = [w_0, w_1, \ldots, w_{N-1}] = \mathbf{u} * \mathbf{v}$ whose k^{th} coordinate is given by the formula

$$\begin{aligned} w_k &= u_0 v_k + u_1 v_{k-1} + \cdots + u_k v_0 + u_{k+1} v_{N-1} + \cdots + u_{N-1} v_{k+1} \\ &= \sum_{i+j \equiv k \pmod{N}} u_i v_j. \end{aligned}$$

Convolution product makes the lattice $\mathbb{Z}^N$ into a ring. In a similar manner, if we start with vectors whose coordinates are in the finite field $\mathbb{F}_q$, then convolution product makes the vector space $\mathbb{F}_q^N$ into a ring, and reduction modulo q gives a natural ring homomorphism $\mathbb{Z}^N \to \mathbb{F}_q^N$. In this note we will study certain sublattices of $\mathbb{Z}^N$ that are defined by convolution congruences.

Remark 1. An alternative description of the convolution ring $\mathbb{Z}^N$ is the polynomial quotient ring $\mathbb{Z}[X]/(X^N - 1)$. Under this isomorphism, the vector $\mathbf{v}$ is identified with the polynomial $v_0 + v_1 X + v_2 X^2 + \cdots + v_{N-1} X^{N-1}$. We will mainly use the vector description, but it is sometimes useful to deal with polynomials, especially when discussing the mod q multiplicative inverse of a vector.

Let $q \geq 1$ be an integer, which we will call the *modulus*, and let $\mathbf{c} \in \mathbb{Z}^N$ be a fixed vector. The collection of pairs of vectors $[\mathbf{u}, \mathbf{v}] \in \mathbb{Z}^{2N}$ satisfying the congruence

$$\mathbf{c} * \mathbf{u} \equiv \mathbf{v} \pmod{q} \tag{1}$$

forms a lattice in $\mathbb{Z}^{2N}$. We denote this lattice by $L(\mathbf{c}, q)$ and call it the *Convolution Modular Lattice* (CML) associated to $\mathbf{c}$ and q. It is clear that $L(\mathbf{c}, q)$ is a lattice of dimension $2N$, since directly from (1) we see that $L(\mathbf{c}, q)$ is a subgroup of $\mathbb{Z}^{2N}$ and satisfies

$$q\mathbb{Z}^{2N} \subset L(\mathbf{c}, q) \subset \mathbb{Z}^{2N}.$$

Using the formula for the convolution product, the lattice $L(\mathbf{c}, q)$ may be described very explicitly as the lattice spanned by the rows of the following matrix:

$$L = L(\mathbf{c}, q) = \mathrm{RowSpan}\left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{N-1} \\ 0 & 1 & \cdots & 0 & c_{N-1} & c_0 & \cdots & c_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & c_1 & c_2 & \cdots & c_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

Remark 2. The N-by-N upper righthand block of the matrix for $L(\mathbf{c}, q)$ is the circulant matrix of the sequence $[c_0, c_1, \ldots, c_{N-1}]$. For this reason an alternative name for $L(\mathbf{c}, q)$ might be *Circulant Modular Lattice.*

Remark 3. In typical cryptographic applications such as NTRU and NSS [6–8], convolution modular lattices are constructed to contain a particular short vector $[\mathbf{a}, \mathbf{b}]$. For example, $\mathbf{a}$ and $\mathbf{b}$ might be chosen from the set of binary (or trinary) vectors with a specified number of ones (and negative ones). The corresponding CML is given by the solutions $[\mathbf{u}, \mathbf{v}]$ to the congruence

$$\mathbf{b} * \mathbf{u} \equiv \mathbf{a} * \mathbf{v} \pmod{q}.$$

Equivalently, it is the lattice $L(\mathbf{c}, q)$ with

$$\mathbf{c} \equiv \mathbf{a}^{-1} * \mathbf{b} \pmod{q}.$$

The public information is the generating vector $\mathbf{c}$, and the secret information used as a trapdoor is the short vector $[\mathbf{a}, \mathbf{b}]$.

2 The Shortest Vector Problem and the Closest Vector Problem in Convolution Modular Lattices

Two classical and much-studied problems in the theory of lattices are the shortest vector problem (SVP) and the closest vector problem (CVP). These are, respectively, the problem of finding a shortest nonzero vector in L and the problem of finding a vector in L that is closest to a given target vector $\mathbf{t}$ not lying in L. An overview of the theoretical study of these problems may be found in [3]. The practical problem of solving SVP and CVP has also attracted considerable attention. The LLL algorithm [13] and its improvements and variants (e.g., [11, 12, 18, 19]) provide a practical method for finding moderately short and moderately close vectors, but the computation of shortest and closest vectors remains very difficult from both a theoretical [1, 16] and a practical perspective [2].

The standard notation for the length of a shortest nonzero vector in a lattice L is

$$\lambda_1 = \lambda_1(L) = \min\left\{\|\mathbf{v}\| : \mathbf{v} \in L,\ \mathbf{v} \neq \mathbf{0}\right\}.$$

If $\mathbf{t}$ is any vector, we use a similar notation

$$\lambda_1(\mathbf{t}) = \lambda_1(\mathbf{t}, L) = \min\left\{\|\mathbf{t} - \mathbf{v}\| : \mathbf{v} \in L\right\}$$

to denote the distance from $\mathbf{t}$ to a closest vector in L.

In terms of theoretical complexity theory, CVP may be a little harder to solve than SVP (but see [5]). However, for most problems there appears to be very little difference in practice. More precisely, if $(L, \mathbf{t})$ is an n-dimensional CVP to be solved, then one forms the $(n+1)$-dimensional lattice

$$L' = \{(\mathbf{v}, 0) : \mathbf{v} \in L\} + \{k(\mathbf{t}, c) : k \in \mathbb{Z}\} \subset \mathbb{R}^{n+1}.$$

For an appropriately chosen value of c, a shortest nonzero vector in L' will often have the form $(\mathbf{u}, c)$, and then it is likely that the vector $\mathbf{t} - \mathbf{u}$ (which is in L) will be a closest vector to $\mathbf{t}$. See, e.g., [4, 17] for further details.

Example 1. Typical convolution modular lattices used in cryptography will contain short (probably shortest) vectors $\mathbf{w} = [\mathbf{u}, \mathbf{v}]$ that are binary or trinary of known form. For ease of exposition, we will consider the case of binary vectors, the trinary case being similar. For example, if we let

$$\mathcal{B}_N(d) = \{\text{binary vectors with } d \text{ ones and } N - d \text{ zeros}\},$$

then an adversary may know that $L(\mathbf{c}, q)$ contains a vector $\mathbf{w} = [\mathbf{u}, \mathbf{v}]$ satisfying $\mathbf{u}, \mathbf{v} \in \mathcal{B}_N(d)$. His objective is to find this vector (or one of its rotations, see Section 3).

The most straightforward approach is to solve SVP. The vector $\mathbf{w}$ has length $\|\mathbf{w}\| = \sqrt{2d}$, so if $\mathbf{w}$ is a smallest vector than solving SVP will find $\mathbf{w}$. However, it will generally be easier to solve CVP with target vector

$$\mathbf{t} = \left(\frac{d}{N}, \frac{d}{N}, \frac{d}{N}, \ldots, \frac{d}{N}\right),$$

since the distance from $\mathbf{t}$ to $\mathbf{w}$ satisfies

$$\|\mathbf{t} - \mathbf{w}\| = \sqrt{\frac{2d(N-d)}{N}} < \sqrt{2d} = \|\mathbf{w}\|.$$

More generally, the two halves of the shortest vector may have unequal sizes, say $\mathbf{u} \in \mathcal{B}_N(d_1)$ and $\mathbf{v} \in \mathcal{B}_N(d_2)$. In this case the lattice should be balanced before formulating the appropriate CVP. For ease of exposition, we will restrict attention in this note to the case that $\|\mathbf{u}\| = \|\mathbf{v}\|$, but see Section 5 for a brief discussion of the balancing process.

Extensive experiments with convolution modular lattices using NTL's implementation [10] of current lattice reduction methods have yielded the following result.

Heuristic 1 *Let $\mathcal{L} = \{(L, \mathbf{t})\}$ be a collection of pairs consisting of a convolution modular lattice L and a CVP target vector $\mathbf{t}$, and suppose that the collection has the following properties:*

1. *The ratio N/q is (approximately) constant for the lattices in $\mathcal{L}$.*
2. *The ratio $\lambda_1(\mathbf{t})/\sqrt{q}$ is (approximately) constant for the lattices in $\mathcal{L}$.*
3. *$\lambda_1(\mathbf{t})$ is significantly smaller than $\sqrt{Nq/\pi e}$, say by at least a factor of 2.*

Then there are positive constants α and β so that the time required to solve the CVP for every $(L, \mathbf{t})$ in $\mathcal{L}$ is given by

$$\log_{10}(\text{Time}) \approx \alpha N + \beta.$$

In other words, the time to solve CVP is exponential in the dimension of the lattice.

Remark 4. The key conditions in Heuristic 1 are conditions (1) and (2) requiring that N/q and $\lambda_1(\mathbf{t})/\sqrt{q}$ be held approximately constant. Condition (3) is less important, it simply says that the lattice contains a vector that is significantly closer to $\mathbf{t}$ than would have been predicted by the Gaussian heuristic. More precisely, the Gaussian heuristic suggests that a (random) lattice L of dimension n and discriminant D will generally have few vectors that are significantly closer than the Gauss constant

$$\gamma = \gamma(L) = D^{1/n}\sqrt{n/2\pi e}$$

to a given vector $\mathbf{t}$. See [6] or [9, Annex A.3.5] for further details of this observation.

Remark 5. We can rephrase Heuristic 1 more intrinsically without direct reference to convolution modular lattices, although we do not know whether it holds for more general lattices. The Gaussian heuristic says that in a "random" lattice L, SVP and CVP have solutions of size approximately $\gamma(L)$. Then the heuristic says that in a collection of (CML) lattices and target vectors satisfying

$$\frac{\gamma(L)}{\dim(L)} \approx c_1 \qquad \text{and} \qquad \lambda_1(\mathbf{t}) \approx c_2 \frac{\gamma(L)}{\sqrt{\dim(L)}},$$

the solution time is approximately exponential in the dimension of L.

Example 2. A series of experiments on convolution modular lattice CVP's (with $67 \le N \le 82$) satisfying

$$\frac{N}{q} \approx 1.96 \qquad \text{and} \qquad \frac{\lambda_1(\mathbf{t})}{\sqrt{q}} \approx 0.896$$

yielded a solution time

$$\log_{10}(\text{Time}) \approx 0.083N - 7.5,$$

where the time is measured in MIPS-years. (A MIPS-year is an approximate amount of computation that a machine capable of performing one million arithmetic instructions per second would perform in one year, about $3 \cdot 10^{13}$ arithmetic instructions.) These values correspond to taking target binary vectors (as described in Example 1) in which each half of the target vector has approximately $0.287N$ ones and $0.713N$ zeros. Extrapolating to the higher value $N = 251$ gives a search time of approximately $2.1 \cdot 10^{13}$ MIPS-years for target vector halves having 72 ones and 179 zeros.

3 Rotation Invariance of Convolution Modular Lattices

For any vector $\mathbf{u} = [u_0, u_1, \ldots, u_{N-1}]$, define the *(right) rotation of u* by

$$\rho(\mathbf{u}) = [u_{N-1}, u_0, u_1, \ldots, u_{N-2}].$$

(In terms of the polynomial ring $\mathbb{Z}[X]/(X^N - 1)$ described in Remark 1, right rotation is simply multiplication by X.) Applying ρ several times gives a k-fold rotation, which we denote by

$$\mathbf{u}^{(k)} = \rho^k(\mathbf{u}) = [u_{N-k}, u_{N-k+1}, \ldots, u_{N-k-1}].$$

An easy computation using the definition of the convolution product shows that $\rho(\mathbf{u} * \mathbf{v}) = \mathbf{u} * \rho(\mathbf{v})$, and repeated application of this rule gives

$$\rho^k(\mathbf{u} * \mathbf{v}) = \mathbf{u} * \rho^k(\mathbf{v}) \quad \text{for all } k \in \mathbb{Z}.$$

(Negative powers of ρ are defined as left rotations.)

Let $L(\mathbf{c}, q)$ be a CML. Applying ρ^k to the defining congruence (1) for $L(\mathbf{c}, q)$ yields

$$\mathbf{c} * \rho^k(\mathbf{u}) \equiv \rho^k(\mathbf{v}) \pmod{q},$$

so we see that if $[\mathbf{u}, \mathbf{v}]$ is in $L(\mathbf{c}, q)$, then every rotation $[\mathbf{u}^{(k)}, \mathbf{v}^{(k)}]$ is also in $L(\mathbf{c}, q)$.

Note that the rotations of a vector have the same length as the original vector. In particular, the SVP for a CML will have up to N different solutions corresponding to the N rotations of any one shortest vector. Similarly, if the target vector $\mathbf{t} \in \mathbb{R}^n$ for a CML CVP is rotation invariant (i.e., has all of its coordinates the same), then the CVP will have up to N different solutions, since if $\mathbf{w} \in L$ solves the CVP for $\mathbf{t}$, then all of the rotations $\rho^k(\mathbf{w}) \in L$ will also solve the CVP for $\mathbf{t}$. See Example 1 for a typical situation where this occurs.

4 The Pattern Method for Convolution Modular Lattices

The convolution lattices typically used in cryptography [6–8] are constructed to contain short binary (or trinary) vectors, which means in particular they tend to contain short vectors in which many of the coordinates are equal to zero. More precisely, they are constructed by choosing two short vectors $\mathbf{a}$ and $\mathbf{b}$ and taking the lattice L defined by

$$\mathbf{b} * \mathbf{u} \equiv \mathbf{a} * \mathbf{v} \pmod{q}.$$

(See Remark 3. The lattice L is the convolution modular lattice $L(\mathbf{c}, q)$ with $\mathbf{c} = \mathbf{a}^{-1} * \mathbf{b} \pmod{q}$.)

In this situation, the first author [14, 15] suggested looking for vectors $[\mathbf{u}, \mathbf{v}]$ in L such that the first r coordinates of $\mathbf{u}$ (or $\mathbf{v}$) are all zero. The hope, of course, is that the generating vector $[\mathbf{a}, \mathbf{b}] \in L$ has this property. If it does, then this method helps to efficiently find it. More precisely, one multiplies the first r columns of the matrix for L by a large constant θ, which has the effect of biasing lattice reduction against nonzero coordinates in those columns.

It is further noted in [14, 15] that the method is more effective for convolution modular lattices than for general lattices because of the rotation invariance inherent in a convolution modular lattice. Thus for a CML it is not necessary that $\mathbf{a}$ (or $\mathbf{b}$) have its first r coordinates zero, it suffices that the vector contains a string of r consecutive zeros somewhere in its list of coordinates. This is because $L(\mathbf{c}, q)$ also contains all of the rotations $[\mathbf{a}^{(k)}, \mathbf{b}^{(k)}]$ of each of its vectors, so if $\mathbf{a}$ (or $\mathbf{b}$) contains a run of r consecutive zeros, then one of the rotations of $[\mathbf{a}, \mathbf{b}]$ will have the zero run in the correct position.

More generally, as noted by the authors and others, rather than choosing a run of r consecutive zeros, one can choose a random pattern of r coordinates and hope that L contains a short vector with the chosen pattern of zeros. And just as before, since the CML is rotation invariant, it suffices that the generating vectors have some rotation containing the preselected pattern of zeros. Even more generally, one might choose a pattern consisting of both zero and nonzero entries. We will call this the *CML Pattern Method.* For simplicity, we will concentrate in this note on patterns of zeros, but see Remark 8 for some brief comments on the more general situation.

From a cryptographic viewpoint, it is clearly better to choose essentially random patterns. Thus if $L(\mathbf{c}, q) = L(\mathbf{a}^{-1} * \mathbf{b}, q)$ is being used for cryptography, then the person who creates the lattice by choosing $[\mathbf{a}, \mathbf{b}]$ can easily thwart any particular pattern (e.g., long runs of zeros) by discarding $\mathbf{a}$ and $\mathbf{b}$ if they contain the chosen pattern.

There is a second, less obvious, reason why it is disadvantageous for an attacker to choose a pattern consisting of consecutive zeros. The attacker "wins" if the pattern of coordinates that he chooses corresponds to zero coordinates in any one of the shifts $\mathbf{a}^{(k)}$ of the unknown vector $\mathbf{a}$. Now it may happen that there are actually two different shifts $\mathbf{a}^{(k_1)}$ and $\mathbf{a}^{(k_2)}$ that win. This means that $L(\mathbf{c}, q)$ contains two different target vectors. In practice having two target vectors does not seem to help lattice reduction very much. (Indeed, underlying May's original idea is taking the lattice $L(\mathbf{c}, q)$, which has N shortest vectors, and breaking the symmetry until there is only one shortest vector.) It turns out that multiple winners are more likely to occur if the attacker chooses consecutive coordinates than if he chooses random coordinates. Since the attacker's goal is simply to choose a single winning set of coordinates, he does best if most winners are only single winners. Thus by choosing random coordinates, he will spread the winning entries more widely.

We illustrate this last point with a small example. We take vectors of length $N = 13$ containing $d = 4$ non-zero entries. There are 715 such vectors. We consider various patterns and for each vector $\mathbf{a}$ we count how many times that pattern appears in some cyclic rotation of the coordinates of $\mathbf{a}$. For example, if we take the pattern $\langle 1, 2, 3, 4 \rangle$, then we are counting how many times there are 4 consecutive zeros in $\mathbf{a}$, while if we take $\langle 1, 3, 5, 8 \rangle$, we are counting how many times the vector $\mathbf{a}$ contains consecutive coordinates of the form 0#0#0##0. (Note that the coordinates of $\mathbf{a}$ wrap, so for example the vector $\mathbf{a} = (0,1,0,1,1,1,0)$ contains the pattern $\langle 1,2,4\rangle$.) We write m_k for the number of vectors in which the given pattern appears k times. In particular, m_0 is the number of vectors which contain no copies of the pattern. Thus in terms of guessing patterns, it is best to have m_0 as small as possible. Table 1 gives the results of this experiment.

Pattern	m_0	m_1	m_2	m_3	m_4	m_5
$\langle 1,2,3,4,5\rangle$	260	195	130	78	39	13
$\langle 1,2,6,10,12\rangle$	130	299	247	39	0	0
$\langle 1,2,4,7,11\rangle$	117	338	208	52	0	0
$\langle 1,2,3,6,10\rangle$	117	338	208	52	0	0
$\langle 1,2,5,8,10\rangle$	78	377	247	13	0	0
$\langle 1,3,5,8,9\rangle$	78	377	247	13	0	0

Table 1. Multiplicity of Patterns — $(N,d) = (13,4)$

Thus for vectors of length 13 containing 4 nonzero entries, the probability of containing the pattern $\langle 1,2,3,4,5\rangle$, that is, the probability of containing five consecutive zeros, is $(715-260)/715 = 64\%$. This is reasonably high, but notice that the probability of containing the pattern $\langle 1,2,4,7,11\rangle$ is 84%, while the probability of containing the pattern $\langle 1,2,5,8,10\rangle$ is 89%. It seems likely that the pattern $\langle 1,2,\dots,r\rangle$ always gives the lowest probability. It would be interesting to try to prove this.

In conclusion, it appears to be in the attacker's best interest either to choose random coordinates or to do a further analysis and determine patterns which have particularly high probability of being successful. However, the disadvantage of choosing a particular pattern is that if the lattice creator knows which pattern(s) the attacker will use, he can always ensure that the shortest vectors in his lattices do not contain that pattern.

Remark 6. It appears that there are some patterns that discourage multiple winners, just as a pattern of consecutive zeros appears to encourage multiple winner. More precisely, it seems that there are some patterns J with the property that if $\mathbf{a}^{(k)}$ does not contain J for some given k, then it is more likely that other rotations will contain J. It seems to be a difficult combinatorial problem to even quantify this precisely, and from a practical perspective, using randomly selected patterns appears to give reasonable performance. We formulate some natural questions and conjectures which we believe are of interest.

- *Conjecture.* Among patterns of length r, the pattern $J = [1,2,\dots,r]$ of consecutive zeros gives the fewest distinct winners.
- *Question.* What is the average number of distinct winners as J runs over all patterns of length r?

– *Question.* What pattern (or patterns) J gives the maximum number of distinct winners?

5 Balancing the Target in a Convolution Modular Lattice

Let $L(\mathbf{c}, q)$ be a CML that is constructed by setting $\mathbf{c} \equiv \mathbf{a}^{-1} * \mathbf{b} \pmod{q}$ with short vectors $\mathbf{a}$ and $\mathbf{b}$ as described in Remark 3. If $\mathbf{a}$ and $\mathbf{b}$ have different lengths, it may be easier to find them by using a balanced CML.

Given $\mathbf{c}$, q and a real number λ, the associated *λ-balanced CML* is the lattice generated by the rows of the following matrix:

$$L(\mathbf{c}, q, \lambda) = \mathrm{RowSpan} \left(\begin{array}{cccc|cccc} \lambda & 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{N-1} \\ 0 & \lambda & \cdots & 0 & c_{N-1} & c_0 & \cdots & c_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda & c_1 & c_2 & \cdots & c_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array} \right)$$

The real number λ is called the *balancing constant*. It is selected to make lattice reduction algorithms work more efficiently.

More precisely, lattice reduction works best if we minimize the ratio between the length of the shortest vector and the length of the next shortest (independent) vector. [For a CML, $[\mathbf{u}', \mathbf{v}']$ is independent of $[\mathbf{u}, \mathbf{v}]$ if it does not lie in the subspace generated by $[\mathbf{u}, \mathbf{v}]$ and all of its rotations.] The shortest vector in $L(\mathbf{c}, q, \lambda)$ is probably the vector $[\lambda\mathbf{a}, \mathbf{b}]$, while the Gaussian heuristic predicts that the next shortest independent vector has length approximately

$$\sqrt{\frac{\dim L(\mathbf{c}, q, \lambda)}{2\pi e}} \cdot \mathrm{Disc}(L(\mathbf{c}, q, \lambda))^{1/\dim L(\mathbf{c}, q, \lambda)} = \sqrt{\frac{Nq\lambda}{\pi e}}.$$

Thus we want to choose λ to minimize the ratio

$$\sqrt{\frac{\lambda^2 \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2}{Nq\lambda/\pi e}}.$$

The optimal value is $\lambda = \|\mathbf{b}\|/\|\mathbf{a}\|$. Note that this equalizes the lengths of the two halves of the vector $[\lambda\mathbf{a}, \mathbf{b}]$. Further, the fundamental ratio that influences the difficulty of lattice reduction is equal to

$$\frac{\text{Length of shortest vector}}{\text{Length of next independent vector}} \approx \sqrt{\frac{\lambda^2 \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2}{Nq\lambda/\pi e}} = \sqrt{\frac{2\pi e \|\mathbf{a}\| \|\mathbf{b}\|}{Nq}}. \quad (2)$$

Remark 7. In practice, one generally approximates the optimal value of λ with a rational number $\alpha/\beta \approx \lambda$ satisfying $\gcd(\alpha, q) = \gcd(\beta, q) = 1$. The balanced CML for the balancing constant α/β is the set of solutions $[\mathbf{u}, \mathbf{v}] \in \mathbb{Z}^N$ to the congruence

$$\mathbf{c} * \beta\mathbf{u} \equiv \alpha\mathbf{v} \pmod{q}.$$

6 Probability of CML Pattern Success

The CML Pattern Method will not work unless the lattice contains a vector with the selected pattern. It is thus important to calculate the probability that a random vector will contain a given pattern. For example, among all binary vectors of dimension 251 containing 72 ones, what is the probability that a randomly selected vector will contain a particular pattern of 17 zeros? As we have seen in Section 4, the exact probability will depend on the chosen pattern, but by making a certain independence assumption we can obtain a reasonable approximate formula. This estimate is given in the next result.

Theorem 1. *Fix positive integers N, d, r and a set of indices $J = \{j_1, j_2, \dots, j_r\}$ with $0 \le j_1 < j_2 < \cdots < j_r < N$. Let*

$$\mathcal{B}_N(d) = \{\textit{binary vectors of dimension } N \textit{ with exactly } d \textit{ ones}\}$$

(a)

$$\operatorname*{Prob}_{\mathbf{a}\in\mathcal{B}_N(d)}(a_{j_1} = a_{j_2} = \cdots = a_{j_r} = 0) = \frac{\binom{N-r}{d}}{\binom{N}{d}} = \prod_{i=0}^{d-1}\left(1 - \frac{r}{N-i}\right).$$

(b)

$$\operatorname*{Prob}_{\mathbf{a}\in\mathcal{B}_N(d)}\left(\begin{array}{c} a_{j_1+k} = a_{j_2+k} = \cdots = a_{j_r+k} = 0 \\ \textit{for some } 0 \le k < N \end{array}\right) \approx 1 - \left(1 - \prod_{i=0}^{d-1}\left(1 - \frac{r}{N-i}\right)\right)^N$$

(The indices on $\mathbf{a} = [a_0, a_1, \dots, a_{N-1}]$ are taken modulo N.)

Proof. (a) In order to create a vector $\mathbf{a}$, we choose d coordinates of $\mathbf{a}$ equal to 1, and the others will be 0. There are $\binom{N}{d}$ ways to do this. If we specify a particular set of r coordinates that are required to be 0, then there are only $\binom{N-r}{d}$ possible choices for $\mathbf{a}$. This proves the first formula for the probability of success, and the second formula follows from a little bit of algebra.
(b) For any particular k, the probability of success is given by the formula in (a). If we assume that the success probabilities for different values of k are independent, then we can use the formula

$$\begin{aligned}
\operatorname{Prob}(\text{success for some } k) &= 1 - \operatorname{Prob}(\text{failure for every } k) \\
&= 1 - \prod_{0\le k<N} \operatorname{Prob}(\text{failure for } k) \\
&= 1 - \prod_{0\le k<N} \bigl(1 - \operatorname{Prob}(\text{success for } k)\bigr).
\end{aligned}$$

Substituting in the formulas from (a) completes the proof of (b), assuming that the different k events are independent. (See Section 4 for a discussion on the extent to which the different values of k give independent events. The precise amount of independence depends on the particular set of indices J.) $\square$

In order to test the accuracy, on average, of Theorem 1, we performed experiments to compute the probability that an r-pattern wins in $\mathcal{B}_N(d)$. For each r we fixed a random vector $\mathbf{a} \in \mathcal{B}_N(d)$ and choose 10,000 random patterns of length r. The results, given in Table 2, show an excellent match between theory and experiment.

$(N,d) = (167,30)$	r	15	20	25	30	35	40	45
	Prob-Theory	0.999	0.913	0.533	0.204	0.063	0.017	0.004
	Prob-Exp	1.000	0.918	0.534	0.209	0.059	0.017	0.004

$(N,d) = (251,72)$	r	5	10	15	20	25	30	35
	Prob-Theory	1.000	1.000	0.734	0.189	0.031	0.005	0.001
	Prob-Exp	1.000	1.000	0.731	0.188	0.032	0.004	0.001

$(N,d) = (347,64)$	r	20	25	30	35	40	45	50
	Prob-Theory	0.995	0.823	0.432	0.166	0.056	0.017	0.005
	Prob-Exp	0.995	0.834	0.426	0.167	0.059	0.017	0.005

Table 2. Probability that an r-pattern wins in $\mathcal{B}_N(d)$

Remark 8. As noted in Section 4, it may be advantageous to use patterns that contain nonzero entries, as well as zero entries. For example, suppose that $\mathbf{a}$ is a binary vector of dimension N with d ones and $N-d$ zeros, and suppose that an attacker guesses a pattern of length r. The numbers N, d, and r are fixed, but we are free to specify r_1 ones and r_0 zeros, where $r_1 + r_0 = r$. There are two issues to consider. First, it may be easier to guess a mixed pattern of ones and zeros. Second, the length of the associated CVP may be smaller using a mixed pattern. We briefly consider these two issues.

First consider the probability of guessing a correct pattern. If we take indices $J = \{j_1, \ldots, j_{r_0}\}$ for the zeros and other indices $L = \{\ell_1, \ldots, \ell_{r_1}\}$ for the ones, then the probability in Theorem 1(a) becomes

$$\operatorname*{Prob}_{\mathbf{a}\in\mathcal{B}_N(d)} \begin{pmatrix} a_{j_1} = a_{j_2} = \cdots = a_{j_{r_0}} = 0 \text{ and} \\ a_{\ell_1} = a_{\ell_2} = \cdots = a_{\ell_{r_1}} = 1 \end{pmatrix} = \frac{\binom{N-r}{d-r_1}}{\binom{N}{d}}.$$

The attacker wants to maximize this probability. In order to maximize the probability of guessing correctly, he chooses a pattern of length r such that the difference of the number of zeros and the number of ones in the remaining $N-r$ vector entries is minimal. Thus, the optimal choice is

$$r_1 = \frac{r-(N-2d)}{2} \qquad \text{and} \qquad r_0 = \frac{r+(N-2d)}{2}, \tag{3}$$

but there is also the obvious restriction that r_0 and r_1 must be nonnegative. In particular, if $d \approx N/2$, then he should take $r_1 \approx r_2 \approx r/2$. On the other hand, if d and r are small, more precisely if $r+2d \leq N$, then the best that the attacker can do is choose $r_1 = 0$ and $r_0 = r$, i.e., he should choose a pattern consisting entirely of zeros.

Next consider the CVP that arises after a pattern has been correctly guessed. The fundamental length that appears in the associated CVP (see Example 1) is proportional to

$$\sqrt{\frac{(d-r_1)(N-d-r_0)}{N-r}},$$

so the attackers wants to choose r_0 and r_1 (subject to $r_0 + r_1 = r$) to minimize this quantity. But the choice made in (3) maximizes this function. If the attacker wants to make the CVP easiest, then he chooses a pattern of length r such that the difference between the number of zeros and the number of ones in the remaining $N-r$ vector entries is maximal. For example, if he can guess all zeros or all ones in his pattern, he obtains length zero in the associated CVP.

We will concentrate in this note on maximizing the probability of guessing correct patterns. It is an open question how to choose an optimal pattern that takes also the vector length in the associated CVP into account.

7 The Net Advantage of the CML Pattern Method

The effect of the CML Pattern Method is to reduce the dimension of the search lattice. More precisely, a correct r-fold pattern has the effect of reducing the effective dimension from $2N$ to $2N-2r$. This can be quite significant, since as explained in Section 2, the search time is roughly exponential in N.

However, one does not know, a priori, that the given lattice has a vector with the given pattern. So balanced against the gain from the reduction in dimension is the loss from choosing a pattern that has no chance of working. Thus the effective gain in efficiency is the probability of guessing a correct pattern multiplied by the reduction in search time.

Proposition 1. *Let $\mathcal{L} = \{(L, \mathbf{t})\}$ be a collection of pairs consisting of a convolution modular lattice L and a CVP target vector $\mathbf{t}$. Suppose that there are constants α and β so that the time required to solve the CVP for each $(L, \mathbf{t}) \in \mathcal{L}$ is given by*

$$\log_{10}(\text{Time}) \approx \alpha N + \beta.$$

Then the gain in applying the CML Pattern Method with a random set of indices $J = \langle j_1, j_2, \ldots, j_r \rangle$ is approximately

$$\text{Gain} = \begin{pmatrix} \textit{Probability that the lattice} \\ \textit{contains the pattern } J \textit{ of zeros} \end{pmatrix} \cdot \frac{\begin{pmatrix} \textit{Time to solve CVP} \\ \textit{in the original lattice} \end{pmatrix}}{\begin{pmatrix} \textit{Time to solve CVP in} \\ \textit{the } J\textit{-eliminated lattice} \end{pmatrix}}$$

$$\approx \left(1 - \left(1 - \prod_{i=0}^{d-1} \left(1 - \frac{r}{N-i}\right)\right)^N\right) 10^{\alpha r} \tag{4}$$

The optimal value of r and approximate corresponding gain may be determined from the formula (4) once an experimental value of α is determined.

Example 3. We use formula (4) to estimate the approximate gain in speed when the CML Pattern Method is applied to a lattice of dimension $N = 251$ with short vectors having $d = 72$. We use the value $\alpha = 0.083$ from Example 2. The results are given in Table 3. We see that the optimal choice of $r = 15$ gives a speed gain of approximately 13.

r	1	2	3	4	5	6	7	8	9	10
Gain	1.21	1.47	1.77	2.15	2.60	3.15	3.81	4.61	5.58	6.76

r	11	12	13	14	15	16	17	18	19	20
Gain	8.16	9.71	11.22	12.36	12.90	12.79	12.14	11.12	9.92	8.66

Table 3. Speed Gain for $(N, d, \alpha) = (251, 72, 0.083)$

8 Parallelization of the CML Pattern Method

The CML Pattern Method allows a partial parallelization of the CVP in convolution modular lattices. The basic idea is to have a large number of separate processors each choose a possible pattern and attempt to solve CVP using that pattern. The method is successful if any one machine guesses a pattern that appears in the solution vector.

In a typical situation, one has a certain number of machines available and one wants to choose a value for r (the pattern length) in order to minimize the expected running time. An important observation is that for a fixed value of r, the gain from adding more and more machines becomes less and less. This is due to the fact that it does not help the running time if more than one machine guesses a valid pattern.

We now quantify these remarks. Let

$$T(r) = \text{time to run if we guess a valid pattern of length r},$$
$$P(r) = \text{probability of guessing a valid pattern of length r}.$$

Using one machine, the running time is more-or-less $T(r)/P(r)$. Suppose we instead use two machines. If $P(r)$ is small, the running time is approximately $T(r)/2P(r)$, so we gain a factor of 2. But if $P(r)$ is large, say 50%, then we save much less (on average), because there is a good chance that both machines will guess right the first time, which doesn't help.

So suppose that we have K machines, each of which tries to guess a pattern of length r. Then the formula for the expected running time is given by the following computation.

$$\begin{aligned}
\mathrm{TotalTime}(K,r) &= \begin{pmatrix}\text{total expected running time using } K \text{ machines}\\ \text{and guessing patterns of length } r\end{pmatrix}\\
&= \frac{T(r)}{\text{probability that at least one machine guesses correctly}}\\
&= \frac{T(r)}{1-\text{probability that all machines guess wrong}}\\
&= \frac{T(r)}{1-(1-P(r))^K}
\end{aligned}$$

Notice that if $P(r)$ is small (compared to $1/K$), then we obtain

$$\mathrm{TotalTime}(K,r) \approx \frac{T(r)}{K\cdot P(r)},$$

which agrees with our intuition that using K machines gives a K-fold speedup. And as K gets very large with r fixed, we find that

$$\lim_{K\to\infty} \mathrm{TotalTime}(K,r) = T(r),$$

which also makes sense, since if there are a huge number of machines, then at least one of them will almost certainly guess a valid pattern the first time.

Hence given access to K machines, the optimal strategy is to choose r so as the minimize the running time function

$$\mathrm{TotalTime}(K,r) = \frac{T(r)}{1-(1-P(r))^K}. \tag{5}$$

Letting r_K denote this optimal value of r, we see that as a function of the number of machines K, the expected running time is given by the function $\mathrm{TotalTime}(K,r_K)$, a highly nonlinear function of K.

In order to analyze parallelization more deeply, we thus need to know the functions $T(r)$ and $P(r)$. The formula for $P(r)$ is given in Theorem 1(b). Further, as noted in Section 2, for certain classes of lattices it appears that the running time is exponential in the dimension of the lattice. Since use of a pattern of length r has the effect of reducing the dimension by $2r$ (equivalently, reducing the value of N by r), it is reasonable to take $T(r)$ to have the form

$$T(r) = 10^{\alpha(N-r)+\beta}$$

for certain constants α and β that are independent of r. However, we note that this neglects the fact that the length of the associated CVP problem is also reduced (see Remark 8), so the following computation is only a rough estimate.

With this caveat, we consider the rough approximation

$$\mathrm{TotalTime}(K,r) = \frac{T(r)}{1-(1-P(r))^K} \approx \frac{10^{\alpha(N-r)+\beta}}{1-\left(1-\prod_{i=0}^{d-1}\left(1-\frac{r}{N-i}\right)\right)^{KN}}$$

and ask for the value r_K of r that minimizes this function of K, treating α, β, d, and N as constants. Then the gain obtained by using K processors is approximately

$$\mathrm{Gain}(K) = \frac{\mathrm{TotalTime}(1,r_1)}{\mathrm{TotalTime}(K,r_K)}.$$

It seems quite difficult to obtain a closed form for this expression. We have estimated it numerically for the parameters $(N,d,\alpha,\beta) = (251, 72, 0.083, -7.5)$ and list the results in Table 4. (The value of β does not affect the gain.)

K	r_K	$P(r_K)$	$K \cdot P(r_K)$	$\mathrm{TotalTime}(r_K)$	$\mathrm{Gain}(K)$
1	15	0.734	0.73	$1.6698 \cdot 10^{12}$	1.000
2	17	0.471	0.94	$1.1607 \cdot 10^{12}$	1.439
3	18	0.357	1.07	$9.4145 \cdot 10^{11}$	1.774
5	20	0.189	0.95	$7.2489 \cdot 10^{11}$	2.303
10	21	0.135	1.35	$5.0929 \cdot 10^{11}$	3.279
20	23	0.066	1.32	$3.5717 \cdot 10^{11}$	4.675
50	26	0.021	1.07	$2.2631 \cdot 10^{11}$	7.378
75	27	0.015	1.10	$1.8491 \cdot 10^{11}$	9.030
100	27	0.015	1.46	$1.6044 \cdot 10^{11}$	10.408
150	28	0.010	1.49	$1.3151 \cdot 10^{11}$	12.697
200	29	0.007	1.35	$1.1376 \cdot 10^{11}$	14.678
300	30	0.005	1.37	$9.3382 \cdot 10^{10}$	17.881

Table 4. Gain Using K Processors—$(N,d,\alpha,\beta) = (251, 72, 0.083, -7.5)$

The numbers in Table 4 are puzzling at first, since as noted above, if $P(r)$ is small, then we expect the gain to be approximately K. However, this will only be true if

$$1-\bigl(1-P(r)\bigr)^K \approx KP(r), \tag{6}$$

so in particular $P(r)$ must be considerably smaller than $1/K$. But in Table 4, the optimal choice of $P(r)$ is approximately equal to $1/K$, so the approximation (6) is not at all accurate.

References

1. M. Ajtai, *The shortest vector problem in ℓ_2 is NP-hard for randomized reductions*, Proc. 30th ACM Symposium on the Theory of Computing, pages 10-19, 1998
2. M. Ajtai, R. Kumar, D. Sivakumar, *A sieve algorithm for the shortest lattice vector problem*, Proc. 33rd ACM Symposium on Theory of Computing, 2001 (to appear)
3. J.W.S. Cassels, *An Introduction to the Geometry of Numbers*, Die Grundlehren Der Mathematischen Wissenschaften, Springer-Verlag, 1959.
4. O. Goldreich, S. Goldwasser, S. Halevi, *Public-key cryptography from lattice reduction problems*, CRYPTO'97, Lect. Notes in Computer Science 1294, Springer-Verlag, 1997, 112-131.
5. O. Goldreich, D. Micciancio, S. Safra and J.P. Seifert, *Approximating shortest lattice vectors is not harder than approximating closest vectors*, Information Processing Letters, vol. 71, pp. 55-61, 1999.
6. J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J.P. Buhler, ed.), Springer-Verlag, Berlin, 1998, 267–288.
7. J. Hoffstein, J. Pipher, J.H. Silverman, *NSS: An NTRU Lattice-Based Signature Scheme*, Advances in Cryptology—Eurocrypt 2001, Lecture Notes in Computer Science, Springer-Verlag, 2001.
8. J. Hoffstein, J. Pipher, J.H. Silverman, *The NTRU Signature Scheme: Theory and Practice*, preprint, June 2001.
9. IEEE P1363.1, Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, Draft 2, 2001.
10. Number Theory Library, Victor Shoup, `http://www.cs.wisc.edu/~shoup/ntl`
11. H. Koy, C.-P. Schnorr, *Segment LLL-Reduction of Lattice Bases*, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science, Springer-Verlag, this volume.
12. H. Koy, C.-P. Schnorr, *Segment LLL-Reduction with Floating Point Orthogonalization*, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science, Springer-Verlag, this volume.
13. A.K. Lenstra, H.W. Lenstra Jr., L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Ann. 261 (1982), 513-534.
14. A. May, *Auf Polynomgleichungen basierende Public-Key-Kryptosysteme*, Johann Wolfgange Goethe-Universitat, Frankfurt am Main, Fachbereich Informatik. (Masters Thesis in Computer Science, 4 June, 1999; Thesis advisor C.P. Schnorr).
15. A. May, *Cryptanalysis of NTRU-107*, preprint, April 1999 (unpublished).
16. D. Micciancio, *The Shortest Vector in a Lattice is Hard to Approximate within Some Constant*, Proc. 39th IEEE Symposium on Foundations of Computer Science, pages 92-98, 1998
17. P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem*, Advances in Cryptology - Proceedings of CRYPTO '99, M. Wiener (ed.), Lecture Notes in Computer Science, Springer-Verlag, 1999.

18. C.P. Schnorr, M. Euchner, *Lattice basis reduction: improved practical algorithms and solving subset sum problems*, Math. Programming 66 (1994), no. 2, Ser. A, 181-199.

19. C.P. Schnorr, *A hierarchy of polynomial time lattice basis reduction algorithms*, Theoretical Computer Science 53, pages 201-224, 1987