Topics in Post-Quantum Cryptography

Lattice-Based Methods

Jesko Hüttenhain Lars Wallenborn

January 18th, 2011

Contents

Algorithms

Acknowledgements

This short summary of lattice-based encryption methods is based on the book chapter [MR09]. It was presented in the seminar on modern cryptographic methods at the Mathematical Institute of the University Bonn from January 18th to January 28th in 2011. The authors would like to express their heartfelt thanks to Professor Nitin Saxena for organizing the seminar and being available for helpful advice, even on national holidays.

1 Basic Definitions

1.1 General Lattices

Definition 1.1. *Let $n \in \mathbb{N}$. A **lattice of dimension** n is a set of the form*

$$\mathcal{L} = \mathcal{L}(B) := \{\, Bx \mid x \in \mathbb{Z}^n \,\} \subseteq \mathbb{R}^n$$

*for some invertible matrix $B \in \mathrm{Gl}_n(\mathbb{R})$. We then write $\dim(\mathcal{L}) = n$. In this case, B is also called the **lattice basis** of $\mathcal{L}$. If $b_1, \ldots, b_n$ are the columns of B, we write $\|B\| := \max_i \|b_i\|$.*

Proposition 1.2. *Let $B, C \in \mathrm{Gl}_n(\mathbb{R})$. Then, $\mathcal{L}(B) = \mathcal{L}(C)$ if and only if there exists a matrix $U \in \mathrm{Gl}_n(\mathbb{Z})$ such that $C = BU$.*

Proof. For "$\Rightarrow$", we let $c_1, \ldots, c_n$ denote the columns of C. By assumption, $c_i = Bu_i$ for certain $u_i \in \mathbb{Z}^n$. Let U denote the matrix whose columns are the u_i, then $C = BU$ is immediate. Since $U = CB^{-1}$, it has to be invertible over $\mathbb{R}$. We let $v_1, \ldots, v_n$ be the columns of $V := U^{-1} \in \mathrm{Gl}_n(\mathbb{R})$. Since $B = CV$, it follows that the Cv_i are the columns of B, hence $Cv_i \in \mathcal{L}(C)$. This means that $v_i \in \mathbb{Z}^n$ and therefore $V, U \in \mathrm{Gl}_n(\mathbb{Z})$.

For "$\Leftarrow$", simply note that $Ux \in \mathbb{Z}^n \Leftrightarrow x \in \mathbb{Z}^n$ and $Cx = BUx$. □

Corollary/Definition 1.3. *If $\mathcal{L} = \mathcal{L}(B)$ is a lattice, we define*

*(i). the **determinant** $\det(\mathcal{L}) := |\det(B)|$.*

*(ii). the **dual** $\mathcal{L}^* := \mathcal{L}\left(B^{-T}\right)$.*

By the above proposition, this does not depend on the choice of B. □

Fact 1.4. *Let $\mathcal{L} = \mathcal{L}(B)$ be a lattice, then $\det(\mathcal{L}^*) = 1/\det(\mathcal{L})$.* □

Fact 1.5. *Let $\mathcal{L} = \mathcal{L}(B) \subseteq \mathbb{R}^n$ be a lattice. Then,*

$$\mathcal{L}^* = \{\, y \in \mathbb{R}^n \mid \forall x \in \mathcal{L} : \langle x, y \rangle \in \mathbb{Z} \,\}.$$

Proof. First, let us assume that $\langle x, y\rangle \in \mathbb{Z}$ for all $x \in \mathcal{L}$, so $\langle Bx', y\rangle \in \mathbb{Z}$ for all $x' \in \mathbb{R}^n$. This means $\langle x', B^T y\rangle \in \mathbb{Z}$ for all $x' \in \mathbb{R}^n$. Choosing x' as the i-th standard vector implies that the i-th component of $B^T y$ is an integer number. In other words, $y' := B^T y \in \mathbb{Z}^n$. Thus, $B^{-T} y' = y$ as claimed.

Now, let us assume that $y = B^{-T} y'$ for some $y' \in \mathbb{Z}^n$. Then, for any $x \in \mathcal{L}$ we can write $x = Bx'$ with $x' \in \mathbb{Z}^n$, and thus,

$$\langle x, y\rangle = \langle x, B^{-T} y'\rangle = \langle B^{-1} x, y'\rangle = \langle x', y'\rangle \in \mathbb{Z}. \qquad \square$$

1.2 q-ary Lattices

Definition 1.6. *Let $q \in \mathbb{Z}_+$. A lattice $\mathcal{L}$ of dimension n is said to be q-**ary** if $q\mathbb{Z}^n \subseteq \mathcal{L}$.*

Fact 1.7. *Let $\mathcal{L} = \mathcal{L}(B)$ be an integer lattice, i.e. $B \in \mathbb{Z}^{n\times n}$, and assume that $q \in \mathbb{Z}_+$ is an integer multiple of $\det(\mathcal{L})$. Then, $\mathcal{L}$ is a q-ary lattice.*

Proof. Let $x \in \mathbb{Z}^n$ be an integer vector. Let $B^{\#} \in \mathbb{Z}^{n\times n}$ be the adjoint of B, so we have $B^{-1} = \det(B)^{-1} B^{\#}$ by the Cramer rule[1]. Thus,

$$y := B^{-1} \cdot qx = \frac{q}{\det(B)} B^{\sharp} x \in \mathbb{Z}^n$$

and therefore, $qx = By \in \mathcal{L}(B)$. $\square$

Corollary 1.8. *Every rational lattice is q-ary for some $q \in \mathbb{N}$.*

Proof. Let $\mathcal{L} = \mathcal{L}(B)$ for $B \in \mathbb{Q}^{n\times n}$. We can write $p \cdot B = B'$ with $B' \in \mathbb{Z}^{n\times n}$ and $p \in \mathbb{N}$. Let $q := |\det(B')|$, then $\mathcal{L}(B')$ is (pq)-ary by the above result. Consequently, $\mathcal{L}(B) = p^{-1}\mathcal{L}(B')$ is a q-ary lattice. $\square$

Lemma 1.9. *Let $v \in \mathbb{Z}^n$ and $B = (b_1 \cdots b_n)$ a lattice basis. The sets*

$$Z_v := \left\{ \sum\nolimits_{i=1}^{n} \alpha_i b_i \;\middle|\; \alpha_i \in \mathbb{R} \text{ with } v_i \le \alpha_i < v_i + 1 \right\} \tag{1.1}$$

satisfy $\mathrm{vol}(Z_v) = \det(B)$ and $\mathbb{R}^n = \dot{\bigcup}_v Z_v$.

Proof. Note that $\mathrm{vol}(Z_v) = \mathrm{vol}(Z_0)$ for all v and $B \cdot [0,1]^n = Z_0$. It is a standard result of multivariate calculus[2] that

$$\det(B) = \det(B) \cdot \mathrm{vol}([0,1]^n) = \mathrm{vol}(Z_0). \qquad \square$$

Remark 1.10. The above 1.9 can be phrased as saying that the inverse of the determinant of a lattice is equal to its density.

[1] See also [Lor03, Chapter 4, Satz 4, Page 148]
[2] See, for instance, [Koe04, Chapter 9].

Definition 1.11. *Let $n \leq m$ and $A \in \mathbb{Z}^{n\times m}$ with $\operatorname{rank}(A) = n$. If $A = (a_{ij})$, we write $\max A := \max_{ij} a_{ij}$. For $q \in \mathbb{Z}_+$, we define*

$$\begin{aligned} \Lambda_q(A) &:= \left\{ y \in \mathbb{Z}^m \;\middle|\; \exists s \in \mathbb{Z}^n : y \equiv A^T s \pmod q \right\} \\ \Lambda_q^\perp(A) &:= \left\{ y \in \mathbb{Z}^m \mid Ay \equiv 0 \pmod q \right\} \end{aligned}$$

To give terms like $\det(\Lambda_q(A))$ a meaning, we want to assign a square matrix B to the matrix A. For this we will use the following algorithm:

Algorithm 1 BASE-REDUCTION-ALGORITHM

Input: $A \in \mathbb{Z}^{n\times m}$ for $n \leq m$ with $\operatorname{rank}(A) = n$
Output: $B \in \mathbb{Z}^{n\times n}$ with $B \cdot \mathbb{Z}^n = A \cdot \mathbb{Z}^m$ and $\operatorname{rank}(B) = n$.
/* We denote the i-th column of A by a_i and its j-th entry by a_{ij} */

1: **for** $j = 1$ **to** n **do**
2: **set** $g := \gcd(a_{jj}, \ldots, a_{mj})$
3: **choose** $g_j, \ldots, g_m \in \mathbb{Z}$ with $\sum_{i=1}^m g_i a_{ij} = g$
4: **set** $a_j := \sum_{i=1}^m g_i a_i$
5: **for** $k = j + 1$ **to** m **do**
6: **set** $a_k := a_k - \frac{a_{kj}}{g} a_j$
7: **end for**
8: **end for**
9: **return** $(a_1, \ldots, a_n)$

Proposition 1.12. *The* BASE-REDUCTION-ALGORITHM *works correctly and outputs a lower triangular matrix.*

Proof. After step 4 the j-th entry of a_j is

$$a_{jj} = \sum_{i=1}^m g_i a_{ij} = g$$

and therefore, after step 6, the j-th entry of a_k is

$$a_{kj} = a_{kj} - \frac{a_{kj}}{g} a_{jj} = 0$$

Since in the j-th iteration of the outer loop, we only change columns a_i of A with $i \geq j$ the output is of the desired form. We only replaced columns of A by integral linear combinations of other columns of A so during the whole algorithm A remains integral and $\mathcal{L}(A)$ never changes. □

Proposition 1.13. *We claim that $\Lambda_q(A)$ is a q-ary lattice $\mathcal{L}$ of dimension m. Furthermore, $\Lambda_q^\perp(A) = q \cdot \mathcal{L}^*$.*

Proof. We first note that

$$\Lambda_q(A) = \begin{pmatrix} A \\ qI_m \end{pmatrix}^T \cdot \mathbb{Z}^{m+n}$$

Thus, the output of the Base-Reduction-Algorithm is a matrix $B \in \mathbb{Z}^{m \times m}$ of full rank such that $\Lambda_q(A) = B \cdot \mathbb{Z}^m = \mathcal{L}(B) =: \mathcal{L}$. This is a q-ary lattice by definition of $\Lambda_q(A)$. We are left to show that $\Lambda_q^\perp(A) = q\mathcal{L}(B)^*$. To see this, we will make use of 1.5.

First, let $\tilde{y} \in q\mathcal{L}^*$ meaning that $\tilde{y} = qy$ for some $y \in \mathbb{R}^m$, with the property that $\forall x \in \mathcal{L} : \langle x, y \rangle \in \mathbb{Z}$. If both y and Ay were integral, we would be done since $A\tilde{y} = qAy \equiv 0 \pmod q$. By either choosing $x = qe_i$ or $x = A^T e_i$, we can see that both $\langle A^T e_i, y \rangle = \langle e_i, Ay \rangle$ and $\langle qe_i, y \rangle = \langle e_i, \tilde{y} \rangle$ are integer numbers.

Next, let $\tilde{y} \in \mathbb{Z}^m$ such that $A\tilde{y} \equiv 0 \pmod q$. In other words, there exists a $z \in \mathbb{Z}^n$ with $A\tilde{y} = qz$. Define $y := q^{-1}\tilde{y}$. It's left to show that $\langle x, y \rangle \in \mathbb{Z}$ for all $x \in \Lambda_q(A)$. So let $x \in \Lambda_q(A)$ and write $x = A^T s + qr$ for some $s \in \mathbb{Z}^n$ and $r \in \mathbb{Z}^m$. Then,

$$\begin{aligned} \langle x, y \rangle &= \left\langle A^T s + qr, q^{-1}\tilde{y} \right\rangle = \left\langle A^T s, q^{-1}\tilde{y} \right\rangle + \langle r, \tilde{y} \rangle \\ &= q^{-1} \langle s, A\tilde{y} \rangle + \langle r, \tilde{y} \rangle = \langle s, z \rangle + \langle r, \tilde{y} \rangle \in \mathbb{Z}. \end{aligned}$$

□

Corollary 1.14. *If $\Lambda_q(A)$ is a lattice, then $\Lambda_q(A) = q \cdot \Lambda_q^\perp(A)^*$.*

Proof. Let $\Lambda_q(A) = \mathcal{L}(B)$. Then, the statement directly follows from 1.13:

$$q\Lambda_q^\perp(A)^* = q\left(q\mathcal{L}\left(B^{-T}\right)\right)^* = q\mathcal{L}\left(qB^{-T}\right)^* = q\mathcal{L}\left(q^{-1}B\right) = \mathcal{L}(B) = \Lambda_q(A).$$

□

1.3 Lattice Problems

Let $B \in \mathrm{Gl}_n(\mathbb{R})$ be a lattice basis and $\gamma \geq 1$ be an approximation parameter. The most well-known approximation problems on this lattice are the following:

Shortest Vector Problem ($\mathbf{SVP}_\gamma$). Find a vector $v \in \mathcal{L}(B) \setminus \{0\}$ such that

$$\|v\| \leq \gamma \cdot \min_{\substack{w \in \mathcal{L}(B) \\ w \neq 0}} \|w\|$$

Closest Vector Problem ($\mathbf{CVP}_\gamma$). For $t \in \mathbb{R}^n$, find a lattice point $v \in \mathcal{L}(B)$ such that

$$\|v - t\| \leq \gamma \cdot \min_{w \in \mathcal{L}(B)} \|w - t\|$$

Shortest Independent Vectors Problem ($\mathbf{SIVP}_\gamma$). Find $U \in \mathrm{Gl}_n(\mathbb{Z})$ with

$$\|BU\| \leq \gamma \cdot \min_{V \in \mathrm{Gl}_n(\mathbb{Z})} \|BV\|$$

We then write $\mathrm{SVP} := \mathrm{SVP}_1$, $\mathrm{CVP} := \mathrm{CVP}_1$ and $\mathrm{SIVP} := \mathrm{SIVP}_1$ for the non-approximative problems.

We now give a brief historical analysis of the hardness of the SVP_γ. From the algorithms known so far, it seems that we can either achieve a polynomial runtime or a polynomial approximation factor, but not both.

Approximation	**Runtime**	**Space**	**Reference**
1	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(n)}$	see below
1	$2^{\mathcal{O}(n \log n)}$	$p(n)$	[Kan83]
$p(n)$	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(n)}$	
$2^{\mathcal{O}(n)}$	$p(n)$		[LLL82]

This has led to the following conjecture:

Conjecture 1.15. *There is no polynomial time algorithm that approximates lattice problems to within polynomial factors.*

As far as exponential-time exact solvers are concerned, they have become practical even for small instances just in the recent years:

Year	**Authors**	**Time**	**Space**
2001	Ajtai, Kumar, Sivakumar	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(n)}$
2004	Regev	2^{16n}	2^{8n}
2008	Nguyen, Vidick	$2^{5.9n}$	$2^{2.95n}$
2010	Pujol, Stelhé	$2^{2.46n}$	$2^{1.233n}$

One should note, however, that lattice reduction methods such as [LLL82] seem to perform better in practice than their theoretic worst-case guarantees suggest. This is not fully explained yet, but has experimental evidence: In [GN08], different algorithms and several distrubutions on lattices where compared with the result that they provide an approximation ratio of roughly δ^n where δ is close to 1.012. Moreover, it seems that approximation rations of $(1.01)^n$ are outside the reach of known lattic reduction alogrithms.

We should note that for $\gamma > \sqrt{n/\log(n)}$, the SVP_γ is not NP-hard unless the polynomial time hierarchy collapses. For $\gamma = 1$ however, it is – see [Ajt98]. Furthermore, there are no quantum algorithms known that perform better than the classical ones. Because of this, lattice-based cryptography is often labelled "post-quantum" cryptography. In summary, we may very well assume that the SVP is a hard problem.

For certain cryptosystems based on lattices, we will be able to prove that breaking the cryptographic construction would imply an efficient algorithm for solving *any* instance of some underlying lattice problem. Because of the above facts, this provides a very strong security guarantee, as opposed to other cryptographic schemes.

2 Finding Short Vectors

Definition 2.1. *Let $\mathcal{L}$ be a lattice. We denote by $\lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L}} \|v\|$ the length of a shortest vector in $\mathcal{L}$.*

Definition 2.2. *Let $\mathbb{Z}_q := \mathbb{Z}/(q)$. We denote by $\pi_q : \mathbb{Z} \to \mathbb{Z}_q$ the canonical projection. For matrices $A = (a_{ij}) \in \mathbb{Z}^{n \times m}$, we write $\pi_q(A)$ for the matrix $(\pi_q(a_{ij})) \in \mathbb{Z}_q^{n \times m}$.*

Fact 2.3. *If $\Lambda_q(A)$ is a lattice such that $\pi_q(A)$ has full rank. Then,*

$$\det(\Lambda_q^\perp(A)) = q^n, \qquad \det(\Lambda_q(A)) = q^{m-n}.$$

Proof. We note that $\#\ker(\pi_q(A)) = q^{m-n}$ by assumption. If we denote by $Q_k := [0, kq[^m$ the cube of length kq, this means that $V_k := Q_k \cap \Lambda_q^\perp(A)$ contains precisely $k^m q^{m-n}$ elements. Recall 1.9. If $\Lambda_q^\perp(A) = \mathcal{L}(B)$, we set

$$Z_k := \bigcup_{Bv \in V_k} Z_v.$$

Intuitively speaking, we cover each lattice point in Q_k by one of the parallelepipeds Z_v. Now, we know that Z_k can differ by a volume of at most $C(qk)^{m-1}$ from that of Q_k, where C is some constant. In other words,

$$\mathrm{vol}(Q_k) - C(qk)^{m-1} \leq \mathrm{vol}(Z_k) \leq \mathrm{vol}(Q_k) + C(qk)^{m-1}$$

Since $\mathrm{vol}(Q_k) = (qk)^m$, the quotient $\mathrm{vol}(Z_k)/\mathrm{vol}(Q_k)$ tends to 1 as k tends to infinity. On the other hand, the left side of

$$\frac{\det(\Lambda_q^\perp(A))}{q^n} = \frac{\det(\Lambda_q^\perp(A)) \cdot k^m q^{m-n}}{k^m q^m} = \frac{\mathrm{vol}(Z_k)}{\mathrm{vol}(Q_k)}$$

does not depend on k and must therefore be equal to 1.

We now write $\Lambda_q^\perp(A) = \mathcal{L}(B)$ and thus,

$$\Lambda_q(A) \overset{1.14}{=} qt\mathcal{L}(B)^* \overset{1.3}{=} q\mathcal{L}\left(B^{-T}\right) = \mathcal{L}\left(qI_m \cdot B^{-T}\right)$$

and taking the determinant yields the desired result. □

In the following subsections, we assume that $n < m$ and $A \in \mathbb{Z}^{n \times m}$ is an integer matrix, chosen at random with $\max A < q$. We aim to solve the SVP_γ for $\Lambda_q^\perp(A)$. With high probability, $\pi_q(A)$ has full rank. Even if it does not, we can eliminate redundant columns in polynomial time and obtain a smaller instance.

2.1 A Length Estimate

We have seen that the determinant of a lattice can be understood as the inverse of its density. Hence, the density of $\Lambda_q^\perp(A)$ is q^{-n}. This means that in a ball of volume $q^n + \varepsilon$ around 0, we expect to find one nonzero lattice point, namely the shortest such vector.

The volume of the ball[3] $\mathcal{B}_r := \mathcal{B}_r(0) \subseteq \mathbb{R}^m$ with radius r is equal to

$$\mathrm{vol}(\mathcal{B}_r) = \frac{\sqrt{\pi^m} \cdot r^m}{\Gamma\left(\frac{m}{2}+1\right)}.$$

Thus, the radius of the ball with volume q^n can be calculated as

$$r = \sqrt[m]{q^n \cdot \frac{\Gamma\left(\frac{m}{2}+1\right)}{\sqrt{\pi^m}}}.$$

We want to use this radius as an estimate for $\lambda_1(\Lambda_q^\perp(A))$. Note that for even m, we have $\Gamma\left(\frac{m}{2}+1\right) = (m/2)!$ by [For04, §20, Satz 2, Page 220]. The faculty can be approximated by

$$n! \approx \sqrt{2\pi n}\,(n/e)^n\,,$$

see also [For04, §20, Satz 6, Page 225]. Thus, $\sqrt[m]{(m/2)!} \approx \sqrt{m/2e}$ where we use that $(\pi m)^{1/2m} \approx 1$. In conclusion, we estimate the length of a shortest vector as

$$\lambda_1(\Lambda_q^\perp(A)) \approx q^{n/m} \cdot \sqrt{\frac{m}{2\pi e}}.$$

This estimate has been shown to be very good for values of m that are neither too close to n nor too large.

2.2 Lattice Reduction Methods

Lattice reduction is the process of calculating a nearly orthogonal lattice basis from an arbitrary one – neither shall we make the term *nearly orthogonal* precise here, nor will we describe the various algorithms. We should however mention to the interested reader that the LLL-algorithm was the first such algorithm and is probably suited best for study – see [LLL82].

It has been established empirically that the shortest vector one can find in $\Lambda_q^\perp(A)$ is approximately of length

$$\min\left\{ q,\ 2^{\sqrt{4n\log(q)\log(\delta)}} \right\} \tag{2.1}$$

where δ depends on the exact lattice reduction algorithm used. Faster algorithms provide $\delta \approx 1.013$ whereas more precise algorithms provide $\delta \approx 1.012$ or even $\delta \approx 1.011$.

It is noteworthy that m has no impact on the quality of the vector that is found. This might indicate that n and q alone determine the difficulty of the SVP_γ.

[3] See [Koe04, Chapter 8.4, Page 288].

2.3 Combinatorial Methods

The best known combinatorial method to solve the SVP_γ is given below. It results from the best known algorithm to solve the k-list birthday problem, see [Wag02].

Algorithm 2 Combinatorial SVP_γ-Solver

Input: An integer $q \in \mathbb{Z}_+$, a matrix $A \in \mathbb{Z}^{n\times m}$ with $\max A < q$, $\operatorname{rank}(A) = n$, and a length bound $b \in \mathbb{Z}_+$.

Output: A vector $w = (w_1, \ldots, w_m) \in \Lambda_q^\perp(A)$ such that $|w_i| \le b$ for all i.

1: **choose** $k \in \mathbb{Z}_+$ such that $\left|\frac{2^k}{k+1} - \frac{m\log(2b+1)}{n\log(q)}\right|$ is minimal
2: **set** $M := \lceil m/2^k \rceil$
3: **set** $V_i := \{ A_{\bullet,i} \ldots A_{\bullet,i+M} \}$ for $1 \le i \le 2^k$. Here, $A_{\bullet,i}$ denotes the i-th column of $\pi_q(A)$ for $i \le m$ and the zero vector otherwise
4: **set** $L_i := \left\{ \sum_{j=1}^M b_j v_j \;\middle|\; v_j \in V_i \text{ and } b_j \in [-b, b] \cap \mathbb{Z} \right\}$
/* Note: We store vectors as formal linear combinations of columns */
/* of A and the L_i are multisets, i.e. implemented as lists. */
5: **set** $L := \#L_1 = (2b+1)^M$
6: **for** $j = k-1$ **to** 0 **do**
7: **for** $1 \le i \le 2^j$ **do**
8: **set** $L_i := \left\{ x + y \;\middle|\; \begin{array}{l} x \in L_i,\ y \in L_{i+2^j} \text{ and} \\ \forall l \le (k-j)\log_q(L) : (x+y)_l = 0 \end{array} \right\}$
9: **end for**
10: **end for**
11: **if** $0 \in L_1$ **then**
12: **return** $(w_1, \ldots, w_m)$ for $0 = \sum_{i=1}^m w_i A_{\bullet,i} \in L_1$
13: **else**
14: **return** 0
15: **end if**

Proposition 2.4. *The* Combinatorial SVP_γ-Solver *works correctly. With a probability of at least* $1 - e^{-1} \approx 0.63$*, the returned vector is nonzero.*

Proof. Correctness is obvious, but we have to verify that the returned vector is nonzero with high probability. First, we claim that after step 8, the size of L_i has not increased. The list $\{ x + y \mid x \in L_i, y \in L_{i+2^j} \}$ contains L^2 many elements. For any integer $t < m$:

$$\Pr_{v\in\mathbb{Z}_q^n}[\forall 1 \le i \le t : v_i = 0] = \frac{q^{n-t}}{q^n} = q^{-t}.$$

Performing this experiment for $t = \log_q(L)$ exactly L^2 times yields an expectancy value of $L^2 \cdot q^{-\log_q(L)} = L$.

Consequently, in step 11, the list L_1 contains L vectors which are zero in all but the last $n - k\log_q(L)$ components. The probability that a random vector with this property

is the zero vector is precisely $1/q^{n-k\log_q L}$. Since in step 1, we chose

$$\frac{n}{k+1} \approx \frac{m\log(2b+1)}{2^k\log(q)} = \frac{M\log(2b+1)}{\log(q)} = \log_q(L),$$

we know that $n \approx (k+1)\log_q(L)$. The probability that the zero vector is among the elements of L_1 is therefore equal to

$$1-\left(1-q^{k\log_q(L)-n}\right)^L \approx 1-\left(1-\frac{1}{L}\right)^L = 1-\left(\frac{L-1}{L}\right)^L$$

which tends to $1-e^{-1}$ from above, for $L\to\infty$. □

The running time of the algorithm is mainly dominated by the size of the lists L_i, namely $L=(2b+1)^{\frac{m}{2^k}}$ for k chosen as in step 1.

3 Encryption Schemes

3.1 GGH/HNF

This cryptosystem, proposed by Goldreich, Goldwasser and Halevi has already been broken in practice but is presented here for educational purposes.

Fact/Definition 3.1. *Given a matrix $A\in \mathrm{Gl}_n(\mathbb{R})$ with integer entries, there exists a unique $U\in\mathrm{Gl}_n(\mathbb{Z})$ such that $(h_{ij})=H=AU$ has the following properties:*

(i). For all i, $h_{ii}>0$.

(ii). For all $j>i$, $h_{ij}=0$.

(iii). For all $j<i$, $|h_{ij}|<h_{ii}$.

Then, H is called the ***hermite normal form of*** *A.*

Proof. See [Coh93, Theorem 2.4.3, page 67]. □

Remark 3.2. The hermite normal form of an invertible integer matrix can be efficiently computed, see [SL96].

Algorithm 3 GGH/HNF-Key-Generation

Input: A security parameter $n\in\mathbb{N}$ and a small value $b\in\mathbb{N}$.
Output: Two lattice bases B and H with $\mathcal{L}(H)=\mathcal{L}(B)$.
/* B will be the private and H the public key. */

1: **choose** nearly orthogonal vectors $b_1,\ldots,b_n\in_R\mathbb{Z}^n$ with $b_{ij}<b$
2: calculate the Hermite Normal Form H of B
3: **return** (B,H)

Algorithm 4 GGH/HNF-Encryption

Input: A lattice basis H and a message $m \in \mathbb{Z}^n$.
Output: A ciphertext $c \in \mathbb{Q}^n$

1: **set** $v := Hm$
2: **choose** $r \in_R \mathbb{Q}^n$ such that the lattice vector closest to $r + v$ is v
3: **return** $v + r$

Definition 3.3. *We denote by* $\lfloor - \rceil$ *the babai rounding method; see [Bab86].*

Algorithm 5 GGH/HNF-Decryption

Input: A ciphertext $c \in \mathbb{Q}^n$ and a nearly orthogonal lattice basis B.
Output: The decrypted message $m \in \mathbb{Z}^n$.

1: let $H = BU$ be the hermite normal form of B
2: **return** $U^{-1} \cdot \lfloor B^{-1}c \rceil$

Proposition 3.4. *The* GGH/HNF *Cryptosystem works correctly.*

Proof. We show that GGH/HNF-Decryption calculates the plaintext from a ciphertext that is the output of a call to GGH/HNF-Encryption. Assume that $c = Hm + r$ for some $m \in \mathbb{Z}^n$ and a noise vector $r \in \mathbb{Q}^n$ where $H = BU$ is the hermite normal form of B. Then, the Babai rounding technique yields

$$\begin{aligned} U^{-1} \cdot \lfloor B^{-1}c \rceil &= U^{-1} \cdot \lfloor B^{-1}Hm + B^{-1}r \rceil = U^{-1} \cdot \lfloor B^{-1}BUm + B^{-1}r \rceil \\ &= U^{-1} \cdot \lfloor Um + B^{-1}r \rceil = U^{-1}Um = m. \end{aligned}$$ □

Remark 3.5. It is a general understanding that the hermite normal form of a lattice basis serves as the best public key since it can be computed from any lattice basis efficiently.

3.2 NTRU

We now proceed to present the NTRU cryptosystem, the most practical known lattice-based cryptosystem.

Definition 3.6. *Let* $v \in \mathbb{R}^n$ *be a vector and* $A \in \mathbb{R}^{n \times n}$ *a matrix. Then, we define* $A_v^* := (v, Av, \ldots, A^{n-1}v)$. *Furthermore, we define the matrix*

$$T := \left(\begin{array}{ccc|c} 0 & \cdots & 0 & 1 \\ \hline \ddots & & & 0 \\ & I & & \vdots \\ & & \ddots & 0 \end{array}\right)$$

and consequently, T_v^* *is the matrix whose* i*-th column is equal to* v *rotated by* i.

Lemma 3.7. *For any two vectors $f, g \in \mathbb{R}^n$, we have*

(i). $T_f^* g = T_g^* f$.

(ii). $T \cdot T_f^* = T_f^* \cdot T$.

(iii). $T_f^* \cdot T_g^* = T_{T_f^* g}^*$.

Proof. Consider the matrices

$$I_k^* := \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & \iddots & 0 \\ 0 & \iddots & & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix} \in \mathrm{Gl}_k(\mathbb{R})$$

and for $1 \leq k \leq n-1$, define the symmetric matrices

$$S_i := \begin{pmatrix} I_i^* & \mathbf{0} \\ \mathbf{0} & I_{n-i}^* \end{pmatrix} \in \mathrm{Gl}_n(\mathbb{R}).$$

Then, we have

$$T_g^* f = \begin{pmatrix} g_1 & g_n & \cdots & g_2 \\ g_2 & g_1 & \cdots & g_3 \\ \vdots & \vdots & & \vdots \\ g_n & g_{n-1} & \cdots & g_1 \end{pmatrix} \cdot f = \begin{pmatrix} \langle f,\, S_1 g \rangle \\ \vdots \\ \langle f,\, S_n g \rangle \end{pmatrix} = \begin{pmatrix} \langle S_1 f,\, g \rangle \\ \vdots \\ \langle S_n f,\, g \rangle \end{pmatrix} = T_f^* g =: h$$

For the second statement, we calculate

$$\begin{aligned} \left\langle S_{i-1} f,\, T^j g \right\rangle &= \sum_{k=1}^{n} (S_{i-1} f)_k \cdot (T^j g)_k = \sum_{k=1}^{i-1} f_{i-k} \cdot g_{k-j} + \sum_{k=i}^{n} f_{n+i-k} \cdot g_{k-j} \\ &= \sum_{k=2}^{i} f_{i-k-1} \cdot g_{k-j-1} + \sum_{k=i+1}^{n+1} f_{n+i-k-1} \cdot g_{k-j-1} \\ &= \sum_{k=1}^{i} f_{i-k-1} \cdot g_{k-j-1} + \sum_{k=i+1}^{n} f_{n+i-k-1} \cdot g_{k-j-1} = \left\langle S_i f,\, T^{j+1} g \right\rangle, \end{aligned}$$

where all index operations are considered modulo n. Thus,

$$T \cdot \begin{pmatrix} \langle S_1 f,\, T^{j-1} g \rangle \\ \vdots \\ \langle S_n f,\, T^{j-1} g \rangle \end{pmatrix} = \begin{pmatrix} \langle S_n f,\, T^{j-1} g \rangle \\ \vdots \\ \langle S_{n-1} f,\, T^{j-1} g \rangle \end{pmatrix} = \begin{pmatrix} \langle S_1 f,\, T^{j} g \rangle \\ \vdots \\ \langle S_n f,\, T^{j} g \rangle \end{pmatrix}$$

and therefore,

$$T^{j-1} h = (T_h^*)_j = \begin{pmatrix} \langle S_1 f,\, T^{j-1} g \rangle \\ \vdots \\ \langle S_n f,\, T^{j-1} g \rangle \end{pmatrix}.$$

This yields the desired equality

$$T_f^* \cdot T_g^* = \begin{pmatrix} f_1 & f_n & \cdots & f_2 \\ f_2 & f_1 & \cdots & f_3 \\ \vdots & \vdots & & \vdots \\ f_n & f_{n-1} & \cdots & f_1 \end{pmatrix} \cdot \begin{pmatrix} g_1 & g_n & \cdots & g_2 \\ g_2 & g_1 & \cdots & g_3 \\ \vdots & \vdots & & \vdots \\ g_n & g_{n-1} & \cdots & g_1 \end{pmatrix}$$
$$= \left(\langle S_i f, T^{j-1} g \rangle\right)_{ij} = T_h^*. \qquad \square$$

Definition 3.8. *We say that a q-ary lattice $\mathcal{L} = \Lambda_q(A)$ is a **convolutional modular lattice** if $A \in \mathbb{Z}^{n \times 2n}$ and $\binom{Tx}{Ty} \in \mathcal{L}$ for all $\binom{x}{y} \in \mathcal{L}$.*

Definition 3.9. *Let $n, d \in \mathbb{N}$ and $d < n$. A vector $f \in \mathbb{Z}^n$ is called a d-**vector** if f has exactly d negative and $d+1$ positive nonzero entries.*

Algorithm 6 NTRU-Key-Generation

Input: prime $n \in \mathbb{N}$, modulus $q \in \mathbb{N}$, $p \in \mathbb{N}$ with $p < q$, weight bound $d \in \mathbb{N}$.
Output: A private key $\binom{f}{g} \in \mathbb{Z}^{2n}$ and a public key $h \in \mathbb{Z}_q^n$.

1: **choose** two d-vectors $f', g \in_R \{p, 0, -p\}^n$
2: **set** $f := f' + e_1$
3: **if** $\pi_q(T_f^*) \notin \mathrm{Gl}_n(\mathbb{Z}_q)$ **then**
4: **goto** step 1
5: **end if**
6: **set** $h := (T_f^*)^{-1} g \bmod q$
7: **return** $(\binom{f}{g}, h)$

Proposition 3.10. *Let $(\binom{f}{g}, h)$ be a key pair generated by* NTRU-Key-Generation *and $A := (T_f^* \; T_g^*)$. Then,*

(i). $\Lambda_q(A)$ is the smallest convolutional modular lattice containing $\binom{f}{g}$.

(ii). We have $T_f^* \equiv I \pmod{p}$ and $T_g^* \equiv \mathbf{0} \pmod{p}$.

(iii). If $\Lambda_q(A) = \mathcal{L}(A')$, the hermite normal form of A' is given by

$$H = \begin{pmatrix} I & \mathbf{0} \\ T_h^* & qI \end{pmatrix}.$$

Proof. Note that all entries of g and f' (chosen in step 1) are divisible by p. Thus, the two congruencies in part (ii) are obvious.

For part (i), note that $x \in \Lambda_q(A)$ if and only if there exist $y \in \mathbb{Z}^n$ and $z \in \mathbb{Z}^m$ with $x = A^T y + qz$. Since

$$A^T y = \begin{pmatrix} T_f^* \\ T_g^* \end{pmatrix} y = \sum_{i=0}^{n-1} \begin{pmatrix} T^i f \\ T^i g \end{pmatrix} \cdot y_i \tag{3.1}$$

and

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in \mathbb{Z}^m \Leftrightarrow \begin{pmatrix} Tz_1 \\ Tz_2 \end{pmatrix} \in \mathbb{Z}^m,$$

we have verified that $\Lambda_q(A)$ is convolutional. Also, if any convolutional modular lattice contains $\binom{f}{g}$, it must contain all $\mathbb{Z}$-linear combinations of the form (3.1). For part (iii), we have to show three things:

- The vector $\binom{f}{g}$ is contained in $\mathcal{L}(H)$.
- The lattice $\mathcal{L}(H)$ is convolutional.
- The inclusion $\mathcal{L}(H) \subseteq \Lambda_q(A)$ holds.

First, we are looking for $\binom{x}{y} \in \mathbb{Z}^{2n}$ such that $H \cdot \binom{x}{y} = \binom{f}{g}$. Since application of H does not change x, we must choose $x = f$. For $y := q^{-1}(g - T_h^* f)$,

$$H \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ T_h^* f + qIy \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}.$$

To see that y is integral, one has to show that $g - T_h^* f$ is an integer multiple of q. By 3.7 and the definition of h,

$$T_h^* f = T_f^* h = T_f^* \cdot (T_f^*)^{-1} g \equiv g \pmod{q}.$$

By part (i), we have shown that $\Lambda_q(A) \subseteq \mathcal{L}(H)$. Next, we show that $\mathcal{L}(H)$ is convolutional. Observe that

$$\begin{pmatrix} x \\ y \end{pmatrix} \in \mathcal{L}(H) \Leftrightarrow \exists \tilde{y} \in \mathbb{Z}^n : H \cdot \begin{pmatrix} x \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \Leftrightarrow T_h^* x + qI\tilde{y} = y. \tag{3.2}$$

We now choose

$$\begin{aligned} x' &:= Tx \\ y' &:= q^{-1}(Ty - T_h^* x') \end{aligned}$$

and calculate that $H \cdot \binom{x'}{y'} = \binom{Tx}{Ty}$. Finally, we calculate

$$Ty \overset{(3.2)}{=} T\left(T_h^* x + qI\tilde{y}\right) \equiv TT_h^* x \overset{3.7}{=} T_h^* Tx = T_h^* Tx = T_h^* x' \pmod{q}$$

Therefore, y' is integral.

Finally, the determinant of $\mathcal{L}(H)$ is obviously equal to q^n, which is the same as $\det \Lambda_q(A)$ by 2.3. On the other hand, we already know that $\mathcal{L}(H) \supseteq \Lambda_q(A)$, and so $\mathcal{L}(H)$ can not contain any further points since both lattices have the same density – see 1.9 and the following remark. □

Algorithm 7 NTRU-Encryption

Input: prime $n \in \mathbb{N}$, modulus $q \in \mathbb{N}$, weight bound $d \in \mathbb{N}$, public key $h \in \mathbb{Z}_q^n$, d-vector message $m \in \{1, 0, -1\}^n$.

Output: ciphertext $c \in \mathbb{Z}_q^n$.

1: **choose** a d-vector $r \in_R \{1, 0, -1\}^n$
2: **set** $h := T_f^* g$
3: **return** $m + T_h^* r \bmod q$

Definition 3.11. *Let $a, b \in \mathbb{Z}^n$ be two integral vectors and $A \in \mathbb{Z}^{n \times n}$ a matrix of rank n. We say that a **is congruent** b **modulo** A if $A^{-1}(a - b) \in \mathbb{Z}^n$. We then write $a \equiv b \pmod{A}$.*

The following algorithm reduces a vector modulo a lower triangular matrix A:

Algorithm 8 Matrix-Reduction

Input: A lower triangular matrix $A = (a_{ij}) \in \mathbb{Z}^{n \times n}$ of rank n and $b \in \mathbb{Z}^n$.

Output: A vector $\tilde{b} \in \mathbb{Z}^n$ such that $0 \leq \tilde{b}_i < a_{ii}$ for all i and $b \equiv \tilde{b} \pmod{A}$.

1: **for** $i = 1$ **to** n **do**
2: **set** $b := b - \lfloor a_{ii}/b_i \rfloor \cdot (a_{1,i} \cdots a_{n,i})^T$
3: **end for**
4: **return** b

Definition 3.12. *We denote the output of* Matrix-Reduction *by $a \bmod A$.*

Fact 3.13. *For two integral d-vectors $r, m \in \{1, 0, -1\}^n$,*

$$\begin{pmatrix} -r \\ m \end{pmatrix} \bmod \begin{pmatrix} I & \mathbf{0} \\ T_h^* & qI \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ (m + T_h^* r) \bmod q \end{pmatrix}.$$

Proof. For $i \leq n$, the i-th iteration of the loop in line 1 of the Matrix-Reduction algorithm will add r_i times the i-th column of T_h^* to m. The following n steps simply perform a reduction modulo q. □

Remark 3.14. The above, together with 3.10.(iii) means that the NTRU encryption step basically reduces $\binom{-r}{m}$ modulo the hermite normal form of the lattice basis. As mentioned in 3.5, this seems to be a sane choice.

Algorithm 9 NTRU-Decryption

Input: prime $n \in \mathbb{N}$, modulus $q \in \mathbb{N}$, $p \in \mathbb{N}$ with $p < q$, weight bound $d \in \mathbb{N}$, private key $\binom{f}{g} \in \mathbb{Z}_q^{2n}$ and a ciphertext $c \in \mathbb{Z}_q^n$.

Output: plaintext $m \in \{1, 0, -1\}^n$.

1: **set** $\tilde{v} := T_f^* c$
2: **for** $i = 1$ **to** n **do**
3: **set** $v_i := \underset{\pi_q(v)=\tilde{v}_i}{\arg\min} |v|$
4: **end for**
5: **return** $(v_1 \cdots v_n) \bmod p$

Proposition 3.15. *With a parameter choice satisfying* $8dp + 4p + 2 < q$*, the* NTRU *Cryptosystem works correctly.*

Proof. Assume that c is a ciphertext generated by NTRU-Encryption. Then,

$$T_f^* \cdot c \equiv T_f^* m + T_f^* T_h^* r \overset{3.7}{=} T_f^* m + T_{T_f^* h}^* r \equiv T_f^* m + T_g^* r \pmod{q}.$$

If the absolute values of all entries of the vector $v := T_f^* m + T_g^* r$ are bounded by $q/2$, the loop in steps 2 to 4 of NTRU-Decryption would reconstruct the value of $v \in \mathbb{Z}^n$ correctly. By 3.10.(ii), this would mean

$$v \bmod p = T_f^* m + T_g^* r \bmod p = m.$$

Hence, let us inspect the vector v more closely. Its i-th entry is given by the formula

$$\begin{aligned} v_i = \sum_{j=1}^{n} \left((T_f^*)_{ij} m_j + (T_g^*)_{ij} r_j \right) &= \sum_{j=1}^{n} \left((T^{j-1} f)_i m_j + (T^{j-1} g)_i r_j \right). \\ &= \sum_{j=1}^{n} \left(f_{i-j+1} m_j + g_{i-j+1} r_j \right). \end{aligned}$$

We write $f' = f - e_1$ as in the NTRU-Key-Generation step 1. Estimating the absolute value of v_i, the worst case would certainly be

$$f'_{i-j+1} = \begin{cases} -p & ; \quad m_j = -1 \\ p & ; \quad m_j = 1 \end{cases} \quad \text{and} \quad g_{i-j+1} = \begin{cases} -p & ; \quad r_j = -1 \\ p & ; \quad r_j = 1 \end{cases}$$

Since $f = f' + e_1$, we obtain the condition

$$|v_i| \leq (dp + (p+1) + dp) + ((d+1)p + dp) = 4dp + 2p + 1,$$

which yields the condition $8dp + 4p + 2 < q$ if we want the absolute values to be bounded by $q/2$. □

3.3 LWE-Based

We present what is perhaps the most efficient lattice-based cryptosystem that admits a theoretical proof of security. We first define the problem on whose worst-case difficulty the system is based, the **Learning With Errors** problem.

Learning-With-Errors (LWE)	
Parameters:	Integers $n, m, q \in \mathbb{Z}_+$ and a probability distribution $\chi : \mathbb{Z}_q \to [0,1]$.
Instance:	A pair (A, v) with $A \in \mathbb{Z}_q^{m \times n}$ and $v \in \mathbb{Z}_q^m$.
Problem Task:	Decide if $v \in_R \mathbb{Z}_q^m$ was chosen uniformly at random or $v = As + e$ was chosen with $s \in_R \mathbb{Z}_q^n$ and $e \in_\chi \mathbb{Z}_q^m$.

This problem can be equivalently described as a bounded distance decoding problem in q-ary lattices: Given $A \in_R \mathbb{Z}_q^{m \times n}$ and a vector $v \in \mathbb{Z}_q^m$, we need to distringuish between the case that v is chosen uniformly from $\mathbb{Z}_q^m$ and the case in which v is chosen by mangling each coordinate of a random point in $\Lambda_q(A^T)$ using χ^m.

Remark 3.16. Throughout this section, we assume q to be odd. We also make frequent use of the embedding of sets

$$\begin{array}{rcl} \mathbb{Z}_q & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \displaystyle\arg\min_{\pi_q(\alpha)=x} |\alpha| \end{array}$$

when considering distributions on $\mathbb{Z}_q$. In other words, we choose $[-\frac{q-1}{2}, \frac{q-1}{2}] \cap \mathbb{Z}$ as a system of representatives for $\mathbb{Z}_q$.

Definition 3.17. *Recall the normal distribution $f_{\mu,\sigma} : \mathbb{R} \to [0,1]$ defined by*

$$f_{\mu,\sigma}(x) := \tfrac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

and the cummulative distribution function

$$\phi(t) := \int_{-\infty}^{t} f_{0,1}(x)\, dx\,.$$

We now define distributions

$$\begin{array}{ll} g_{\mu,\sigma} : \mathbb{Z} \to [0,1] & g_{\mu,\sigma}(n) := \displaystyle\int_{n-\frac{1}{2}}^{n+\frac{1}{2}} f_{\mu,\sigma}(x)\, dx \\ h^q_{\mu,\sigma} : \mathbb{Z}_q \to [0,1] & h^q_{\mu,\sigma}(n) := \displaystyle\sum_{k \in \mathbb{Z}} g_{\mu,\sigma}(n + kq) \end{array}$$

We call $h^q_{\mu,\sigma}$ the ***rounded normal distribution on*** $\mathbb{Z}_q$*. We also define*

$$\Psi_\alpha := h^q_{0,(\alpha q/\sqrt{2\pi})}.$$

For the LWE problem, one usually chooses $\chi = \Psi_\alpha$ as a parameter. The LWE is assumed to be quite hard: there are no subexponential algorithms known that solve it to date. Furthermore, the following Theorem was proven in [Reg05]:

Theorem 3.18. *Assume access to an oracle that solves the* LWE *problem with a parameter choice* (n, m, q, χ) *such that* $\chi = \Psi_\alpha$, $\alpha q > \sqrt{n}$, *a prime* $q \leq \text{poly}(n)$ *and* $m \leq \text{poly}(n)$. *Then, there exists a quantum algorithm running in time* $\text{poly}(n)$ *for solving the* SIVP_γ *and the decision variant of* SVP_γ *for* $\gamma = \widetilde{\mathcal{O}}(n/\alpha)$ *in any lattice of dimension* n. □

In other words, assuming the security of lattice-based problems against quantum computers, any cryptosystem based on LWE is also secure against quantum computers. Moreover, it is very much possible that the proof for Theorem 3.18 may some day be dequantized, i.e. ported to a classical computational model.

Algorithm 10 LWE-Key-Generation

Input: $n, m, \ell, t, r, q \in \mathbb{Z}_+$ and $\alpha \in \mathbb{R}_+$.
Output: private key $S \in \mathbb{Z}_q^{n\times\ell}$ and public key $(A, P) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times\ell}$

1: **choose** $S \in_R \mathbb{Z}_q^{n\times\ell}$, $A \in_R \mathbb{Z}_q^{m\times n}$, $E \in_{\Psi_\alpha} \mathbb{Z}_q^{m\times\ell}$
2: **set** $P := AS + E$
3: **return** $(S, (A, P))$

Remark 3.19. Note now that an oracle to construct the private key only from (A, P) would imply an efficient algorithm to solve LWE: Given an instance (A, v), we understand $v \in \mathbb{Z}_q^{m\times 1}$ as a matrix and apply our oracle to construct a "private key" $s \in \mathbb{Z}_q^{n\times 1}$. If this succeeds, we know that $e \in_{\Psi_\alpha} \mathbb{Z}_q^{m\times 1}$ such that $v = As + e$. Otherwise, v was chosen uniformly at random. We will discuss this in further detail in 3.3.3.

Definition 3.20. *For integers* $q, t \in \mathbb{Z}_+$, *we define a function* $\rho_t^q : \mathbb{Z}_t \to \mathbb{Z}_q$ *by*

$$\rho_t^q(n) := \left\lceil \frac{nq}{t} \right\rfloor .$$

We also write ρ_t^q *instead of* $(\rho_t^q)^\ell$ *when we apply this function to vectors in* $\mathbb{Z}_t^\ell$.

Algorithm 11 LWE-Encryption

Input: $n, m, \ell, t, r, q \in \mathbb{Z}_+$, $\alpha \in \mathbb{R}_+$, public key $(A, P) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times\ell}$, message $v \in \mathbb{Z}_t^\ell$.
Output: ciphertext $(u, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

1: **choose** $a \in_R [-r, r]^m \cap \mathbb{Z}^m$
2: **set** $u := A^T a$
3: **set** $c := P^T a + \rho_t^q(v)$
4: **return** (u, c)

Algorithm 12 LWE-Decryption

Input: ciphertext $(u, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, private key $S \in \mathbb{Z}_q^{n\times\ell}$.
Output: $v \in \mathbb{Z}_t^\ell$.

1: **return** $\rho_q^t(c - S^T u)$

We now proceed to discuss the choice of parameters for the LWE cryptosystem under the aspects of efficiency, possibility of decryption errors and security.

3.3.1 Efficiency

It is obvious that the algorithms 10, 11 and 12 can be implemented efficiently since they only involve matrix operations modulo certain integers. Due to this, the algorithms are also heavily parallelizable. It seems natural to choose $t = 2^k$ for k equaling the size of a computer register.

3.3.2 Decryption Errors

Let $b := E^T a$ and assume $|b_i| < \frac{q-t}{2t}$. We also set $w := \rho_t^q(v)$, so we know

$$\left|\frac{qv_i}{t} - w_i\right| \leq \frac{1}{2} \Leftrightarrow \left|v_i - \frac{tw_i}{q}\right| \leq \frac{t}{2q}$$

We now get

$$\rho_q^t(b_i + \rho_t^q(v_i)) = \left\lceil \frac{t \cdot (b_i + w_i)}{q} \right\rfloor.$$

Therefore, we can calculate the estimate

$$\left|v_i - \frac{t \cdot (b_i + w_i)}{q}\right| = \left|v_i - \frac{tw}{q}\right| + \left|\frac{tb_i}{q}\right| < \frac{t}{2q} + \left(\frac{1}{2} - \frac{t}{2q}\right) = \frac{1}{2}. \tag{3.3}$$

Thus, in this case,

$$\begin{aligned}\rho_q^t(c - S^T u) &= \rho_q^t(P^T a + \rho_t^q(v) - S^T A^T a) \\ &= \rho_q^t((AS + E)^T \cdot a + \rho_t^q(v) - S^T A^T a) \\ &= \rho_q^t(E^T a + \rho_t^q(v)) \overset{(3.3)}{=} v.\end{aligned}$$

In other words, under this assumption,

$$\Pr\left[\rho_q^t(c - S^T u) = v\right] \geq \Pr\left[\forall i : |b_i| < \frac{q-t}{2t}\right]. \tag{3.4}$$

Now we analyze the behaviour of $b = E^T a$. Since each coordinate of a is uniformly chosen from $[-r, r] \cap \mathbb{Z}$, we know that the variance of each coordinate is equal to

$$\mathrm{Var}(a_i) = \frac{1}{2r+1} \cdot \sum_{k=-r} k^2 = \frac{1}{2r+1} \cdot \frac{r \cdot (r+1) \cdot (2r+1)}{6} = \frac{r(r+1)}{3}.$$

We denote by X the random variable measuring b_i and by $Z := \frac{X-\mu(X)}{\sigma(X)}$ its normalization, such that $\Pr[z_1 \leq Z \leq z_2] = \phi(z_2) - \phi(z_1)$. Thus, we can calculate

$$\begin{aligned}\sigma(X)^2 = \mathrm{Var}(b_i) &= \mathrm{Var}\left(\sum\nolimits_{j=1}^{m} E_{ji} a_j\right) \\ &= \sum\nolimits_{j=1}^{m} E_{ji}^2 \cdot \mathrm{Var}(a_j) = \|E_{\bullet,i}\|^2 \cdot \frac{r(r+1)}{3}\end{aligned}$$

and deduce an upper bound for the decryption error probability per letter:

$$\begin{aligned}\varepsilon_i &:= \Pr\left[|X| \geq \frac{q-t}{2t}\right] = \Pr\left[Z \geq \frac{q-t}{2t\sigma(X)}\right] + \Pr\left[Z \leq \frac{-q}{2t\sigma(X)}\right] \\ &= 1 - \phi\left(\frac{q-t}{2t\sigma(X)}\right) + \phi\left(\frac{-q}{2t\sigma(X)}\right) = 2 - 2\cdot\phi\left(\frac{q-t}{2t\cdot\sigma(X)}\right) \\ &= 2\cdot\left(1 - \phi\left(\frac{q-t}{2t\cdot\|E_{\bullet,i}\|}\cdot\sqrt{\frac{3}{r(r+1)}}\right)\right). \end{aligned} \tag{3.5}$$

Using $\sigma(\|E_{\bullet,i}\|) = \frac{\alpha q\cdot\sqrt{m}}{\sqrt{2\pi}}$, we get an expected decryption error probability per letter

$$\varepsilon = 2\cdot\left(1 - \phi\left(\frac{q-t}{2t\alpha q}\cdot\sqrt{\frac{6\pi}{mr(r+1)}}\right)\right). \tag{3.6}$$

It is now a matter of mere arithmetics to adjust parameters accordingly in order to obtain low error margins. We mention that one can always deploy error-correcting codes to the plaintext in order to reduce the probability of decoding errors to arbitrary small values.

3.3.3 Security

To meet the requirements of Theorem 3.18, we will choose q to be prime and $\alpha q > \sqrt{n}$. This leaves us with a choice for m and α where we will attempt to choose α as large as possible since it leads to harder lattice instances. Under these conditions, we may assume public keys to be completely indistinguishable from pairs (A, P) chosen uniformly at random.

Althought these theoretical results stand, we would also like to choose parameters in such a way that the best algorithms to attack LWE fail to achieve acceptable running times. Hence, we first describe such an attack. Assume that (A, v) is an LWE-instance.

- Choose a short vector $w \in \Lambda_q(A^T)^*$.
- Calculate $\lambda := \langle w, v\rangle$.
- If λ is close to an integer, we guess that $v = As + e$ for some $e \in_{\Psi_\alpha} \mathbb{Z}_q^m$.

This routine makes use of the fact that

$$\langle As + e,\, w\rangle = \underbrace{\langle As,\, w\rangle}_{\in\mathbb{Z}} + \langle e,\, w\rangle$$

and relies on $\langle e,\, w\rangle$ being very small. Fixing any vector w and for $e \in_{\Psi_\alpha} \mathbb{Z}_q^m$,

$$\mathrm{Var}\left(\langle e,\, w\rangle\right) = \mathrm{Var}\left(\sum\nolimits_{i=1}^{m} e_i w_i\right) = \sum_{i=1}^{m} w_i^2 \mathrm{Var}(e_i) = \|a\|^2 \cdot \frac{\alpha^2 q^2}{2\pi}$$

yields a standard deviation of $\|w\| \cdot \frac{\alpha q}{\sqrt{2\pi}}$ for this value. Since we want the above algorithm to fail, we choose

$$\frac{\alpha q}{\sqrt{2\pi}} \gg \frac{1}{\|w\|}. \tag{3.7}$$

The estimate (2.1) applied to $\Lambda_q(A^T)^* = q^{-1}\Lambda_q^\perp(A^T)$ yields

$$\|w\| \approx q^{-1} \cdot \min\left(q,\ 2^{\sqrt{4n\log(q)\log(\delta)}}\right).$$

If right and left hand side of (3.7) differ by a factor of 1.5, this brings the distribution of $\lambda \bmod \mathbb{Z}$ into negligible statistical distance from the uniform distribution. Hence, we pick

$$\alpha \geq 1.5 \cdot \sqrt{2\pi}\max\left(q^{-1},\ 2^{-2\sqrt{n\log(q)\log(\delta)}}\right). \tag{3.8}$$

For a pair $(A, P) \in_R \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times \ell}$ chosen uniformly at random and used as a public key for LWE-Encryption, we would like the ciphertext to be virtually indistinguishable from a uniformly chosen vector. This would establish the systems invulnerability against chosen plaintext attacks.

Theorem 3.21. *Let $r < q$ be integers, $M \in_R \mathbb{Z}_q^{n\times m}$ and $a \in_R [-r, r]^m \cap \mathbb{Z}_q^m$. Then, the statistical distance between the distribution of Ma and the uniform distribution on $\mathbb{Z}_q^n$ is bounded by*

$$\beta_{n,m}(q, r) := \sqrt{\frac{(2r+1)^m}{q^n}}.$$

Proof. This can be proven by an argument similar to the one in [Reg05, PVW08]. □

Choosing $M = \binom{A^T}{P^T}$, the above tells us that we should choose parameters in such a way that $\beta_{n+\ell,m}(q, r)$ is negligible, say 2^{-100}.

3.3.4 Choice of Parameters

In this summary, we present some concrete choices of parameters, satisfying our aforementioned requirements. Thus, we expect them to deliver high levels of both security and efficiency. Equation (3.7) motivates the following choice:

$$m := \frac{(n+\ell)\log(q) \overbrace{+200}^{\text{for “}\gg\text{”}}}{\log(2r+1)}$$

Using an approximation parameter of $\delta = 1.01$ corresponds to the most exact algorithm known, so the estimate (3.8) justifies a choice of

$$\alpha := 4 \cdot \max\left\{q^{-1},\ 2^{-2\sqrt{n\log(q)\log(1.01)}}\right\}.$$

Thus, we are left with the parameters n, ℓ, q, r and t. To obtain a balance between encryption blowup and public key size, we choose $n = l$. We now try to minimize q while maximizing r and t and maintaining a tolerable probability of decryption errors. We give such parameter configurations below.

Private key size: $s_{\text{prv}} := n\ell\log(q)$

Public key size: $s_{\text{pub}} := m(n+\ell)\log(q)$

Message size: $s_{\text{m}} := \ell\log(t)$

Ciphertext size: $s_{\text{c}} := (n+\ell)\log(q)$

Encryption blowup factor: $s_{\text{c}}/s_{\text{m}} = \left(1+\frac{n}{\ell}\right)\log_t(q)$

Error probability per letter: Equation (3.6) yields

$$\varepsilon := 2 \cdot \left(1 - \phi\left(\frac{q-t}{2t\alpha q} \cdot \sqrt{\frac{6\pi}{mr(r+1)}}\right)\right)$$

Lattice of dimension in attack: The value (2.1) was experimentally obtained in instances where the lattice had dimension

$$d := \sqrt{n\log(q)/\log(\delta)}.$$

n	ℓ	m	q	r	t	α	s_{pub}	$s_{\text{c}}/s_{\text{m}}$	ε	d
136	136	2008	2003	1	2	0.0065000	$6 \cdot 10^6$	21.9	0.84%	322
166	166	1319	4093	4	2	0.0024000	$5.25 \cdot 10^6$	24.0	0.54%	372
192	192	1500	8191	5	4	0.0009959	$7.5 \cdot 10^6$	13.0	1.02%	417
214	214	1333	16381	12	4	0.0004500	$8 \cdot 10^6$	14.0	0.82%	457
233	233	1042	32749	59	2	0.0002170	$7.3 \cdot 10^6$	30.0	0.92%	493
233	233	4536	32749	1	40	0.0002170	$31.7 \cdot 10^6$	5.6	0.87%	493

References

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. 261 (1982), 515-534.

[Kan83] Ravi Kannan, *improved algorithms for integer programming and related lattice problems*, In Proc. 15th ACM Symp. on Theory of Computing (STOC) (1983), 193-206.

[Bab86] László Babai, *On Lovász lattice reduction and the nearest lattice point problem*, Combinatorica 6 (1986), 1-13.

[Coh93] H. Cohen, *A Course in Computational Algebraic Number Theory*, edition 3, Springer 1993.

[SL96] Arne Storjohann and George Labahn, *Asymptotically Fast Computation of Hermite Normal Forms of Integer Matrices*, Proc. Internat. Symp. on Symbolic and Algebraic Computation: ISSAC (1996), ACM Press, 259-266.

[Ajt98] M. Ajtai, *The shortest vector problem in L2 is NP-hard for randomized reductions*, Proc. of 30th STOC. ACM (1998), 10-19.

[Wag02] David Wagner, *A generalized birthday problem*, Advances in cryptology (CRYPTO), Volume 2442 of LNCS, 288-303, Springer 2002.

[Lor03] Falko Lorenz, *Lineaere Algebra 1*, edition 3, Spektrum.

[For04] Otto Forster, *Analysis 1*, edition 7, Friedr. Vieweg & Sohn.

[Koe04] Konrad Königsberger, *Analysis 2*, edition 5, Springer.

[Sto04] J.Stoer *Numerische Mathematik 1*, edition 9, Springer 2004

[Reg05] O.Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Proc. 37th ACM Symp. on Theory of Computing (STOC), 84-93, 2005.

[GN08] N.Gama and P.Q.Nguyen, *Predicting lattic reduction*, Advances in Cryptology, Proc. Eurocrypt '08, Lecture Notes in Computer Science, Springer 2008

[PVW08] C. Peikert and V. Vaikuntanathan and B. Waters, *A framework for efficient and composable oblivious transfer*, Advances in Cryptology, LNCS, Springer 2008

[MR09] D.J. Bernstein, J. Buchmann and E. Dahmen, *Post Quantum Cryptography*, chapter *Lattice-based Cryptography* by Daniele Micciancio and Oded Regev, 147-191, Springer 2009.