Kryptographie

Studienbrief 9

universität wien

Gitterbasierte Kryptographie

Gerald und Susanne Teschl

Druckfehler/Feedback bitte an:
gerald.teschl@univie.ac.at

SS 23

Version:
2023-07-05

Studienbrief 9

Gitterbasierte Kryptographie

Inhalt

9.1 Gitter 386
9.2 Gitterreduktion 394
9.2.1 Abschätzungen an den kürzesten Vektor 404
9.2.2 Anwendung: Coppersmith-Algorithmus 405
9.3 Gitterbasierte Kryptographie 409
9.3.1 GGH 409
9.3.2 NTRU 412
9.3.3 LWE 421
9.4 Kontrollfragen 433
9.5 Übungen 436

Wie bereits wissen, bedeutet der Shor-Algortihmus, dass alle derzeit gängigen PK Verfahren (RSA, DH, ECDH) mithilfe eines Quantencomputers gebrochen werden können. Der Algorithmus von Grover bedeutet ebenfalls einen signifikanten Vorteil beim Brechen klassischer Verfahren (AES). Da diese Verbesserung bei der Suche aber nur quadratisch ist, kann man hier leicht durch Verdoppelung der Schlüssellänge gegensteuern. Vom NIST wurde daher 2016 der Standardisierungsprozess für **Post-Quanten-Kryptographie**, also PK Verfahren die vor Quantencomputern sicher sind, gestartet:

`https://csrc.nist.gov/projects/post-quantum-cryptography`

Die Einreichungen stammen aus folgenden mathematischen Bereichen:

- Gitter
- multivariaten Polynome

- kryptographische Hashfunktionen (Lamport-Einmal-Signaturen und das darauf basierende **Merkle-Signaturverfahren**)
- fehlerkorrigierenden Codes (**Classic McEliece**)
- Isogenien (Homomorphismen) zwischen supersingulären elliptische Kurven

Viele dieser Verfahren haben signifikante Nachteile in Bezug auf Größe der Schlüssel und Ausführungsgeschwindigkeit (das Verfahren aus dem Isogenie-Bereich wurde inzwischen gebrochen, was das Vertrauen in diesem Bereich erschüttert hat) und die derzeit vielversprechensden Algorithmen sind aus dem Bereich der **gitterbasierten Kryptographie**. Wir werden daher hier nur auf diesen Bereich näher eingehen.

Praktische Implementierungen aktueller Verfahren findet man auf:

https://openquantumsafe.org/liboqs/algorithms/

9.1 Gitter

Als Vorbereitung besprechen wir in diesem Abschnitt den namensgebenden Begriff eines Gitters.

Erinnern Sie sich daran, dass m linear unabhängige Vektoren $\mathbf{u}_1, \dots \mathbf{u}_m \in \mathbb{R}^n$ einen m-dimensionalen Teilraum des $\mathbb{R}^n$ aufspannen. Wenn wir in der Linearkombination dieser Vektoren nur ganzzahlige Koeffizienten zulassen, dann erhalten wir ein Gitter.

Definition 9.1 Sind $\mathbf{u}_1, \dots \mathbf{u}_m \in \mathbb{R}^n$ gegebene linear unabhängige Vektoren, so heißt die Menge

$$\mathcal{L}(\mathbf{u}_1, \dots, \mathbf{u}_m) = \{k_1\mathbf{u}_1 + \cdots k_m\mathbf{u}_m | k_1, \dots, k_m \in \mathbb{Z}\}$$

das von $\mathbf{u}_1, \dots \mathbf{u}_m \in \mathbb{R}^n$ aufgespannte **Gitter** (*lattice*). Die Zahl m wird auch als **Dimension** oder **Rang** des Gitters bezeichnet.

Wir werden die Vektoren $\mathbf{u}_1, \dots \mathbf{u}_m$ als Spalten[1] einer Matrix $U = (\mathbf{u}_1, \dots \mathbf{u}_m)$ auffassen und können dann kompakt $\mathcal{L}(U)$ schreiben.

Beispiel 9.2 Der einfachste Fall ist das von den Standardbasisvektoren $\mathbf{e}_1, \dots \mathbf{e}_m$ aufgespannte Gitter

$$\mathcal{L}(\mathbf{e}_1, \dots, \mathbf{e}_m) = \mathbb{Z}^m \subseteq \mathbb{R}^n$$

[1] Achtung, insbesondere bei Softwareprogrammen wird meist mit Zeilenvektoren gearbeitet.

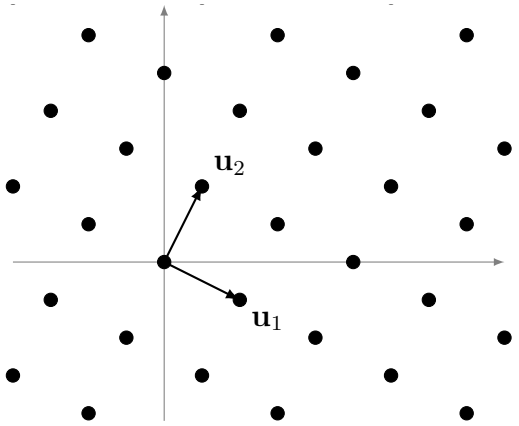

Abbildung 9.1: Das von $\mathbf{u}_1$ und $\mathbf{u}_2$ aufgespannte Gitter in $\mathbb{R}^2$.

Mithilfe von U können wir

$$\mathcal{L}(U) = \{U\mathbf{k} | \mathbf{k} \in \mathbb{Z}^m\} = U\mathcal{L}(\mathbf{e}_1, \ldots, \mathbf{e}_m) = U\mathbb{Z}^m$$

auf das Standardgitter $\mathbb{Z}^m$ zurückführen.

Sind die Koordinaten der Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_m$ alle rational, so kann man durch skalieren der Achsen immer voraussetzten, dass die Koordinaten sogar ganzzahlig sind und wir werden deshalb nur diesen Fall betrachten.

Die Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_m$ werden als **Basis** des Gitters bezeichnet und es ist wichtig zu beachten, dass es für ein Gitter verschiedene Basen geben kann. Es kann also

$$\mathcal{L}(\mathbf{u}_1, \ldots, \mathbf{u}_m) = \mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_r)$$

gelten. In diesem Fall kann man die zweite Basis $\mathbf{v}_1, \ldots, \mathbf{v}_r$ durch die erste $\mathbf{u}_1, \ldots, \mathbf{u}_m$ ausdrücken: Da $\mathbf{v}_1 \in \mathcal{L}(\mathbf{u}_1, \ldots, \mathbf{u}_m)$ gibt es $w_{11}, \ldots w_{1m} \in \mathbb{Z}$ mit

$$\mathbf{v}_1 = w_{11}\mathbf{u}_1 + \cdots + w_{m1}\mathbf{u}_m,$$

etc. Wenn wir die Entwicklungskoeffizienten w_{jk} zu einer Matrix W zusammenfassen, können wir das kompakt als

$$V = UW$$

schreiben. Durch Vertauschen der Rollen von U und V sehen wir, dass auch

$$U = V\tilde{W}$$

gelten muss. Da Basen linear unabhängig sind, muss $\tilde{W} = W^{-1}$ gelten (und damit auch $r = m$, da eine nicht-quadratische Matrix nicht invertierbar sein kann).

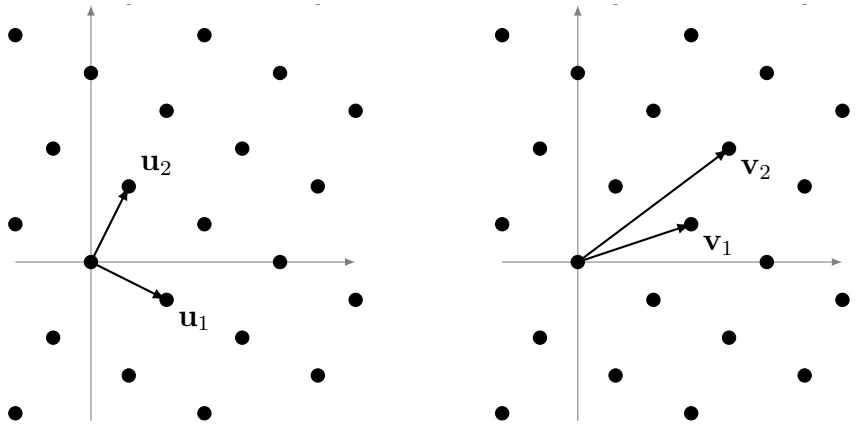

Abbildung 9.2: Verschiedene Basen können das gleiche Gitter aufspannen.

Ist für eine invertierbare ganzzahlige Matrix W auch die inverse W^{-1} ganzzahlig, dann muss $\det(W) \in \{\pm 1\}$ gelten, denn in diesem Fall sind auch die Determinanten ganzzahlig und wegen $\det(W^{-1}) = \det(W)^{-1}$ bleibt nur noch diese Möglichkeit. Matrizen W mit $\det(W) \in \{\pm 1\}$ werden als **unimodular** bezeichnet.

Satz 9.3 Zwei Mengen von linear unabhängigen Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_m$ und $\mathbf{v}_1, \ldots, \mathbf{v}_r$ erzeugen genau dann das gleiche Gitter, wenn $r = m$ gilt und es eine ganzzahlige unimodulare Matrix W gibt, mit $V = UW$.

Beispiel 9.4 Spannen $\mathbf{u}_1$, $\mathbf{u}_2$ und $\mathbf{v}_1$, $\mathbf{v}_2$ das gleiche Gitter auf?
a) $\mathbf{u}_1 = (2, -1)$, $\mathbf{u}_2 = (1, 2)$ und $\mathbf{v}_1 = (4, 3)$, $\mathbf{v}_2 = (3, 1)$.
a) $\mathbf{u}_1 = (2, -1)$, $\mathbf{u}_2 = (1, 2)$ und $\mathbf{v}_1 = (4, 3)$, $\mathbf{v}_2 = (5, 0)$

Lösung zu 9.4 a) Es gilt $\mathbf{v}_1 = \mathbf{u}_1 + 2\mathbf{u}_2$ und $\mathbf{v}_2 = \mathbf{u}_1 + \mathbf{u}_2$, also ist

$$W = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}.$$

Es gilt $\det(W) = -1$, also spannen beide Basen das gleiche Gitter auf. Die inverse Matrix ist

$$W^{-1} = \begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix}.$$

Siehe Abbildung 9.2.
b) Es gilt $\mathbf{v}_1 = \mathbf{u}_1 + 2\mathbf{u}_2$ und $\mathbf{v}_2 = 2\mathbf{u}_1 + \mathbf{u}_2$, also ist

$$W = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

Es gilt $\det(W) = -3 \notin \{\pm 1\}$, also spannen die Basen verschiedene Gitter auf! Die Inverse von W ist nicht ganzzahlig. ■

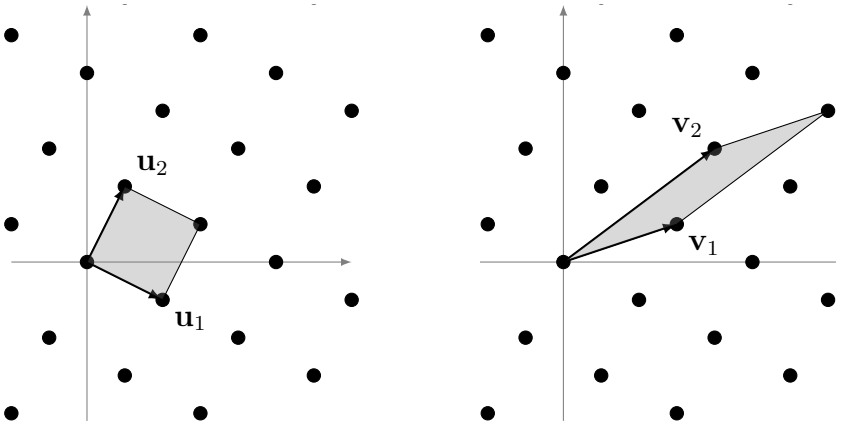

Abbildung 9.3: Fundamentabereich in Abhängigkeit von der Basis. Die Fläche ist immer gleich.

Es ist leicht zu sehen, dass folgende Spaltenoperationen auf der Matrix U der Basisvektoren das Gitter invariant lassen:

- Vertauschen zweier Spalten.
- Eine Spalte mit -1 multiplizieren.
- Ein ganzzahliges Vielfache einer Spalte zu einer anderen addieren.

Ein Gitter kann auch als Untergruppe der additiven Gruppe des $\mathbb{R}^n$ betrachtet werden, da es abgeschlossen unter der Addition

$$\mathbf{a}, \mathbf{b} \in \mathcal{L} \quad \Rightarrow \mathbf{a} + \mathbf{b} \in \mathcal{L}$$

ist. Natürlich gilt damit auch Abgeschlossenheit bezüglich Multiplikation mit ganzen Zahlen, $\mathbf{a} \in \mathcal{L}$ und $k \in \mathbb{Z}$ impliziert $k\mathbf{a} \in \mathcal{L}$. Solche Untergruppen werden als **diskret** bezeichnet, weil es um jeden Punkt $\mathbf{a} \in \mathcal{L}$ eine kleine Kugel $B_\varepsilon(\mathbf{a}) = \{\mathbf{x} | \|\mathbf{x} - \mathbf{a}\| < \varepsilon\}$ gibt, in der keine weiteren Gitterpunkte sind.

Die Menge

$$\mathcal{F}(U) = \{U\mathbf{s} | 0 \leq s_j < 1\}$$

wird als **Fundamentalbereich** (auch **Grundmasche**) des Gitters $\mathcal{L}(U)$ bezeichnet. Der Fundamentalbereich hängt offensichtlich von der Basis ab (Abbildung 9.3), aber sein Volumen (das auch als **Gitterdeterminante** oder **Gitterdiskriminante** bezeichnet wird) ist durch die **Gram-Determinante**

$$\det(\mathcal{L}) = \sqrt{\det(U^T U)}$$

gegeben und ist unabhängig von der Basis (da $\det(W)^2 = 1$).

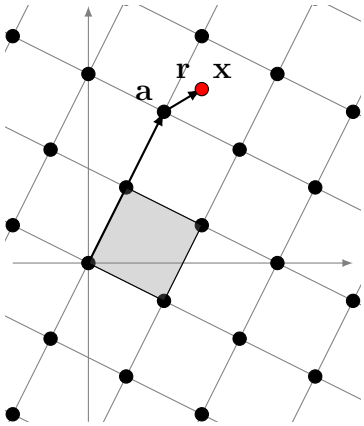

Abbildung 9.4: Zerlegung eines Vektors $\mathbf{x}$ in einen Gittervektor $\mathbf{a}$ und einen Vektor $\mathbf{r}$ im Fundamentalbereich.

Ist $m = n$, so kann jeder Punkt $\mathbf{x} \in \mathbb{R}^n$ in der Form

$$\mathbf{x} = \mathbf{a} + \mathbf{r}, \qquad \mathbf{a} \in \mathcal{L}(U), \mathbf{r} \in \mathcal{F}(U)$$

geschrieben werden (Abbildung 9.4).

Da die Basis des Gitters auch eine Basis des $\mathbb{R}^n$ ist, können wir $\mathbf{x} = s_1\mathbf{u}_1 + \cdots + s_n\mathbf{u}_n = U\mathbf{s}$ mit $\mathbf{s} = U^{-1}\mathbf{x}$ schreiben. Weiters können wir $s_j = k_j + t_j$ mit $k_j = \lfloor s_j \rfloor \in \mathbb{Z}$ und $t_j = s_j - k_j \in [0, 1)$ schreiben. Damit gilt $\mathbf{a} = U\mathbf{k}$ und $\mathbf{r} = U\mathbf{t}$.

Die beiden zentralen Probleme der Gittertheorie sind nun

CVP Das Nähester-Vektor-Problem (*closest vector problem*), zu einem gegebenen Vektor $\mathbf{x} \in \mathbb{R}^n$ einen Gittervektor $\mathbf{a} \in \mathcal{L}$ zu finden, der den kürzesten Abstand $\|\mathbf{x} - \mathbf{a}\|$ hat.

SVP Das Kürzester-Vektor-Problem (*shortest vector problem*), einen kürzesten von Null verscheidenen Gittervektor zu finden.

Die Länge eines kürzesten Vektors wird mit

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{a} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{a}\| = \min_{\mathbf{a} \in \mathcal{L} \setminus \{\mathbf{b}\}} \|\mathbf{a} - \mathbf{b}\|$$

bezeichnet.

Auf den ersten Blick sieht es so aus, als ob man bei der Approximation von $\mathbf{x}$ einfach nur die vorherige Zerlegung etwas modifizieren muss indem man die Koordinaten von $\mathbf{a}$ nicht abrundet, sondern rundet:

$$\mathbf{a} = U\mathbf{k}, \qquad k_j = \lfloor (U^{-1}\mathbf{x})_j \rceil.$$

Wenn die Vektoren der Basis orthogonal sind, dann löst dieser Rundungsalgorithmus (**Babai-Rundungsalgorithmus**, nach dem ungarischen Mathematiker László Babai (*1950)) in der Tat das Problem des nähesten Vektors.

Sind die Vektoren $\mathbf{u}_1,\ldots,\mathbf{u}_n$ orthogonal und gilt $\mathbf{x}=U\mathbf{s}$, so folgt $\|\mathbf{x}-U\mathbf{k}\|^2=\sum_j(s_j-k_j)^2\|\mathbf{u}\|_j^2$ und diese Summe wird genau dann minimal, wenn $k_j=\lfloor s_j\rceil$ gewählt wird.

In einer orthogonalen Basis ist auch das Problem des kürzesten Vektors nicht schwer: Es ist einfach der kürzeste Basisvektor.

Wiederum unmittelbar aus $\|U\mathbf{k}\|^2=\sum_j k_j^2\|\mathbf{u}_j\|^2$ klar.

Sobald die Basisvektoren aber nicht mehr orthogonal sind, bricht das alles zusammen. Mithilfe der Zerlegung $\mathbf{x}=\mathbf{a}+\mathbf{r}$ kann man das Problem zwar immer auf den Fall $\mathbf{r}\in\mathcal{F}$ reduzieren, aber Runden gibt im Allgemeinen nicht den nähesten Gitterpunkt, wie unser nächstes Beispiel zeigt.

Beispiel 9.5 Gegeben sind die zwei Basen

$$U=\begin{pmatrix}2&1\\-1&2\end{pmatrix},\qquad V=\begin{pmatrix}5&8\\0&1\end{pmatrix}.$$

Weisen Sie nach, das beide Basen das gleiche Gitter aufspannen und versuchen Sie das CVP für $\mathbf{x}=(5.2,0.4)$ mit dem Rundungsalgorithmus bezüglich beider Basen zu lösen.

Lösung zu 9.5 Die Transformationsmatrix

$$W=U^{-1}V=\begin{pmatrix}2&3\\1&2\end{pmatrix}$$

ist ganzzahlig und unimodular, $\det(W)=1$. Also spannen beide Basen das gleiche Gitter auf.

Die Koordinaten von $\mathbf{x}$ bezüglich U sind

$$\mathbf{x}=2\mathbf{u}_1+1.2\mathbf{u}_2$$

und durch Runden ergibt sich der näheste Gittervektor zu

$$\mathbf{a}=\lfloor 2\rceil\mathbf{u}_1+\lfloor 1.2\rceil\mathbf{u}_2=2\mathbf{u}_1+\mathbf{u}_2=(0,5).$$

Die Koordinaten von $\mathbf{x}$ bezüglich V sind

$$\mathbf{x}=0.4\mathbf{v}_1+0.4\mathbf{v}_2$$

und durch Runden ergibt sich der vermeintlich näheste Gittervektor $\tilde{\mathbf{a}}=0$. Dieser hat aber einen größeren Abstand $\|\mathbf{x}-\tilde{\mathbf{a}}\|=5.22$ als unser vorheriger Vektor $\|\mathbf{x}-\mathbf{a}\|=0.45$, der optimal ist, da $\langle\mathbf{u}_1,\mathbf{u}_2\rangle=0$ gilt. Die Situation ist in Abbildung 9.5 veranschaulicht. ■

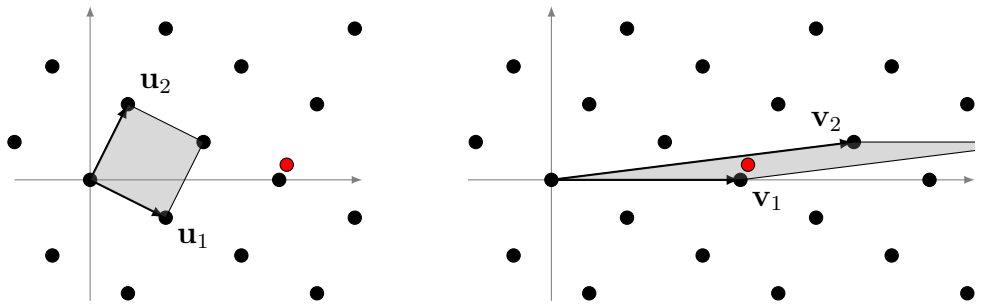

Abbildung 9.5: Gute vs. schlechte Basis für CVP

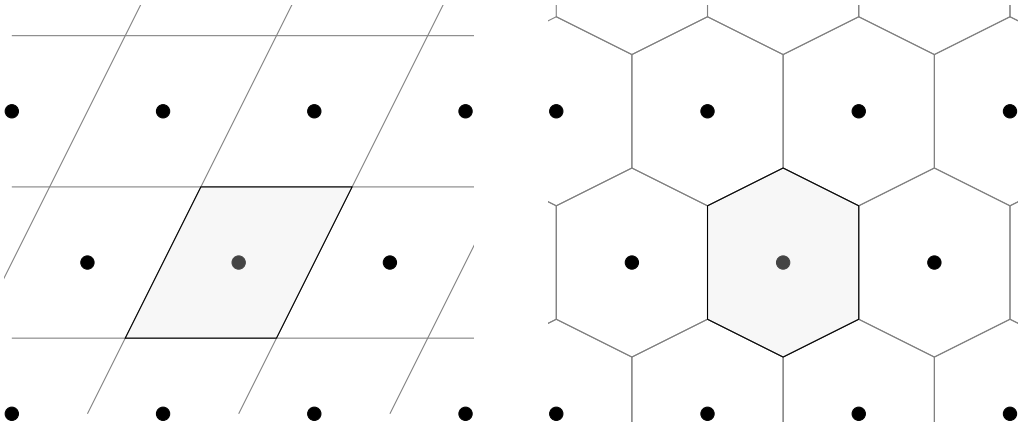

Abbildung 9.6: Runden (links) vs. nähester Nachbar (rechts)

Um das Problem besser zu verstehen, betrachten wir die Bereiche, die einem bestimmten Gitterpunkt zugeordnet werden.

Die Menge der Punkte die zum Gittervektor $\mathbf{a}$ gerundet werden ist

$$\{\mathbf{a} + U\mathbf{s} | -\tfrac{1}{2} \leq s_j < \tfrac{1}{2}\} = \mathbf{a} - \tfrac{1}{2}(\mathbf{u}_1 + \cdots + \mathbf{u}_m) + \mathcal{F}(U),$$

also der Fundamentalbereich so verschoben, dass der Punkt zu dem gerundet wird in der Mitte liegt.

Bei Zuordnung zum nähesten Gittervektor (also der Lösung des CVP) erhält man die Grenzen indem man die Normale durch die Mitte der Verbindungsgerade zwischen den benachbarten Gitterpunkten zieht. In einem orthogonalen Gitter fallen beide Verfahren zusammen, anderenfalls ergeben sich (auch für eine gute Basis) verschiedene Bereiche wie in Abbildung 9.6 veranschaulicht ist. Die Menge der Punkte für die 0 der näheste Gitterpunkt ist, wird übrigens als **Voronoi-Zelle** des Gitters bezeichnet.

Der Rundungsalgorithmus löst also auch für eine gute Basis das CVP nicht immer. Der Fehler der dabei gemacht wird ist allerdings klein. Bei einer schlechten

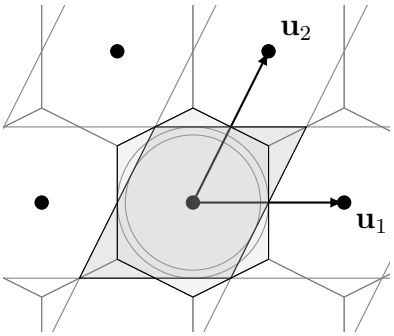

Abbildung 9.7: Durchschnitt von Rundungsbereich und Voronoi-Zelle

Basis ist der Rundungsbereich langgezogen und Punkte nahe den entfernten Ecken werden auf den Mittelpunkt gerundet, obwohl es in der Umgebung nähere Punkte gibt. Genau das ist in Beispiel 9.5 passiert.

Insbesondere findet der Rundungsalgorithmus nicht einmal die näheste Ecke des Fundamentalbereichs. Es mag auf den ersten Blick zwar trivial erscheinen, zumindest die näheste Ecke des Fundamentalbereichs durch Probieren zu finden, aber alleine schon dafür gibt es 2^n Möglichkeiten! Eine Brute-Force-Suche um das SVP oder CVP zu lösen ist also hoffnungslos.

Wir halten jedoch fest, dass der Rundungsalgorithmus das CVP genau dann löst, wenn $\mathbf{x}$ in der Schnittmenge der Voronoi-Zelle und des Rundungsbereichs des nähesten Gitterpunkts liegt (vgl. Abbildung 9.7). Insbesondere also, wenn $\mathbf{x}$ nahe genug an einem Gittervektor liegt, z.B. in der größten Kugel um den Gitterpunkt, die gerade noch in den Rundungsbereich passt (kleiner Kreis in Abbildung 9.7). Nennen wir diesen Radius $\rho(U)$, dann löst der Rundungsalgorithmus das CVP für $\mathbf{x}$ falls

$$\min_{\mathbf{b}\in\mathcal{L}} \|\mathbf{x}-\mathbf{b}\| < \rho(U).$$

Der Radius $\rho(U)$ ist das Minimum der Normalabstände der Begrenzungsflächen des Fundamentalbereichs, also $\rho(u) = \min_{1\le j\le m} \|\mathbf{u}_j - P_j\mathbf{u}_j\|$, wobei P_j die orthogonale Projektion auf $\mathrm{LH}\{\mathbf{u}_1,\dots,\mathbf{u}_{j-1},\mathbf{u}_{j+1},\dots,\mathbf{u}_m\}$ ist. Der Radius der größten Kugel, die in die Voronoi-Zelle passt, ist genau der halbe Gitterabstand $\lambda_1(\mathcal{L})/2$ (großer Kreis in Abbildung 9.7) und das liefert eine Abschätzung $\rho(U) \le \lambda_1(\mathcal{L})$, die wir aber noch verbessern werden.

Zusammenfassend gilt, das CVP ist im Allgemeinen nur dann mit dem Rundungsalgorithmus (näherungsweise) lösbar, wenn man eine gute Basis hat, die zumindestens näherungsweise orthogonal ist. In der Tat stellt sich heraus, dass das CVP ein algorithmisch sehr schwieriges Problem ist. Diese Tatsache versucht man sich in der gitterbasierten Kryptographie zu Nutze zu machen. Die gute Basis für ein Gitter, von dem öffentlich nur eine schlechte Basis bekannt ist, fungiert als Falltüre für das CVP (siehe Abschnitt 9.3.1).

9.2 Gitterreduktion

In diesem Abschnitt wollen wir uns überlegen, wie man eine gegebene Basis verbessern kann. Solche Verfahren sind als **Gitterreduktion** bekannt. Zunächst erinnern wir an das **Gram–Schmidt-Verfahren** zu Orthogonalisierung der Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_m$ (wir verzichten auf das Normieren): Der erste Vektor ist $\mathbf{u}_1$:

$$\mathbf{u}_1^\star = \mathbf{u}_1.$$

Den zweiten Vektor erhalten wir, indem wir die orthogonale Komponente von $\mathbf{u}_2$ bezüglich $\mathbf{u}_1$ bilden,

$$\mathbf{u}_2^\star = \mathbf{u}_2 - \frac{\langle \mathbf{u}_1^\star, \mathbf{u}_2 \rangle}{\|\mathbf{u}_1^\star\|^2} \mathbf{u}_1^\star.$$

Das setzen wir so lange fort, bis alle Vektoren aufgebraucht sind.

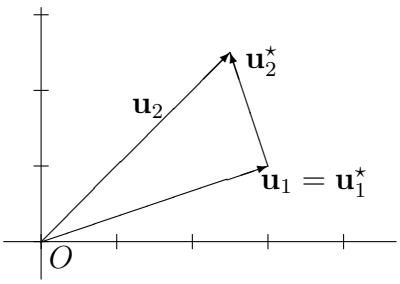

Abbildung 9.8: Gram–Schmidt-Verfahren

Dieses Verfahren ist nach dem dänischen Mathematiker Jorgen Pedersen Gram (1850–1916) und dem deutschen Mathematiker Erhard Schmidt (1876–1959) benannt.

Die Vektoren $\mathbf{u}_j^\star$ werden also rekursiv gemäß

$$\mathbf{u}_j^\star = \mathbf{u}_j - \sum_{i=1}^{j-1} \mu_{j,i} \mathbf{u}_i^\star$$

mit

$$\mu_{i,j} = \frac{\langle \mathbf{u}_j, \mathbf{u}_i^\star \rangle}{\|\mathbf{u}_i^\star\|^2}, \qquad 1 \leq i < j,$$

berechnet. Für $i = j$ setzten wir $\mu_{j,j} = 1$ und für $i > j$ setzten wir $\mu_{i,j} = 0$. Schreiben wir die Gewichte $\mu_{i,j}$ in eine Matrix

$$M = \begin{pmatrix} 1 & \mu_{1,2} & \ldots & \ldots & \mu_{1,m} \\ 0 & 1 & \mu_{2,3} & \ldots & \mu_{2,m} \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & 1 & \mu_{m-1,m} \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix},$$

so können wir das kompakt als

$$U = U^\star M$$

schreiben.

Beispiel 9.6 Berechnen Sie die Gram–Schmidt-Basis $V^\star$ für V aus Beispiel 9.5.

Lösung zu 9.6 Wir beginnen mit

$$\mathbf{v}_1^\star = \mathbf{v}_1 = (5,0), \qquad \|\mathbf{v}_1^\star\|^2 = 25.$$

Wir berechnen

$$\mu_{1,2} = \frac{\langle \mathbf{u}_1^\star, \mathbf{u}_2 \rangle}{\|\mathbf{u}_1^\star\|^2} = \frac{40}{25} = \frac{8}{5}$$

Der zweite Vektor ist

$$\mathbf{u}_2^\star = \mathbf{u}_2 - \mu_{1,2}\mathbf{u}_1^\star = (8,1) + \frac{8}{5}(5,0) = (0,1).$$

Insgesamt erhalten wir

$$V^\star = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}, \qquad M = \begin{pmatrix} 1 & \frac{8}{5} \\ 0 & 1 \end{pmatrix}.$$

■

Am Ende haben wir orthogonale Vektoren $\mathbf{u}_1^\star, \ldots, \mathbf{u}_m^\star$ erhalten, die denselben Teilraum aufspannen wie die ursprünglichen Vektoren. Allerdings ist (im Allgemeinen) weder $V^\star$ noch M ganzzahlig und wir erhalten daher keine Basis für unser Gitter. Eine interessante Beobachtung ist jedoch, dass sich wegen $\det(M) = 1$ die Gram-Determinante nicht ändert. Daraus folgt

$$\det(\mathcal{L}) = \sqrt{\det(U^T U)} = \sqrt{\det((U^\star)^T U^\star)} = \|\mathbf{u}_1^\star\| \cdot \|\mathbf{u}_2^\star\| \cdots \|\mathbf{u}_m^\star\|,$$

da $(U^\star)^T U^\star$ eine Diagonalmatrix mit Diagonalelementen $\|\mathbf{u}_j^\star\|^2$ ist.

Das spiegelt die rekursive Definition des Volumens eines Parallelepipeds als Grundfläche mal Höhe wieder: $\|\mathbf{u}_1^\star\|$ ist die Seitenlänge und $\|\mathbf{u}_2^\star\|$ ist die Höhe des von $\mathbf{u}_1$ und $\mathbf{u}_2$ aufgespannten Parallelograms. Die Fläche also $\|\mathbf{u}_1^\star\| \cdot \|\mathbf{u}_2^\star\|$. Dieses Parallelogram ist die Grundfläche des von $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ aufgespannten Parallelepipeds und die Höhe ist $\|\mathbf{u}_3^\star\|$. Das Volumen ist also $\|\mathbf{u}_1^\star\| \cdot \|\mathbf{u}_2^\star\| \cdot \|\mathbf{u}_3^\star\|$. Etc.

Daraus erhalten wir folgende Ungleichung:

Satz 9.7 (Hadamard Ungleichung) Für jede Gitterbasis gilt

$$\det(\mathcal{L}) \leq \|\mathbf{u}_1\| \cdots \|\mathbf{u}_m\|$$

mit Gleichheit genau dann, wenn die Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_m$ orthogonal sind.

Da die Gram-Schmidt-Basis $\mathbf{u}_j^\star$ orthogonal ist folgt aus dem Satz von Pythagoras

$$\|\mathbf{u}_j\|^2 = \|\mathbf{u}_j^\star\|^2 + \sum_{i=1}^{j-1} \mu_{j,i}^2 \|\mathbf{u}_i^\star\|^2 \geq \|\mathbf{u}_j^\star\|^2.$$

Der Ausdruck

$$\mathcal{H}(U) = \sqrt[n]{\frac{\det(\mathcal{L})}{\|\mathbf{u}_1\| \cdots \|\mathbf{u}_m\|}}$$

wird als **Hadamard-Verhältnis** der Basis bezeichnet und dient als Maß für die Güte einer Basis. Umso näher es bei 1 ist, um so weniger weicht die Basis von einer Orthogonalbasis ab.

Zusätzlich bekommen wir auch noch eine untere Schranke an $\lambda_1(\mathcal{L})$:

Satz 9.8 Sei U eine Gitterbasis mit zugehöriger Gram–Schmidt-Basis $U^\star$. Dann gilt für die Länge eines kürzesten Vektors

$$\lambda_1(\mathcal{L}) \geq \min_{1 \leq j \leq m} \|\mathbf{u}_j^\star\|.$$

Es sei $\mathbf{a} = U\mathbf{k}$ ein beliebiger Gitterpunkt und j der größte Index mit $k_j \neq 0$. Dann gilt $\langle \mathbf{a}, \mathbf{u}_j^\star \rangle = k_j \|\mathbf{u}_j^\star\|^2$ da $\mathbf{u}_j^\star$ auf alle $\mathbf{u}_i$ mit $i < j$ orthogonal steht. Zusammen mit der Cauchy–Schwarz-Ungleichung folgt daraus $\|\mathbf{a}\| \geq \frac{|\langle \mathbf{a}, \mathbf{u}_j^\star \rangle|}{\|\mathbf{u}_j^\star\|} = |k_j| \|\mathbf{u}_j^\star\| \geq \|\mathbf{u}_j^\star\|$.

Wie findet man nun eine *gute* Basis? Betrachten wir dazu den einfachsten Fall mit zwei Dimensionen. Wir wollen also eine Basis finden, bei der der Winkel zwischen den Vektoren $\pm\pi/2$ ist, bzw. möglichst nahe bei $\pm\pi/2$ liegt. Da die Fläche konstant bleiben muss, müssen die Längen der Vektoren kürzer werden, wenn der Winkel größer wird.

Wir nehmen daher $\|\mathbf{u}_1\| \leq \|\mathbf{u}_2\|$ an und versuchen Gram–Schmidt anzuwenden. Da das Ergebnis ein Gittervektor sein muss, bleibt uns nichts anderes übrig als zu runden. Die Lagrange-Reduktion von $\mathbf{u}_2$ bezüglich $\mathbf{u}_1$ lautet somit:

$$\tilde{\mathbf{u}}_2 = \mathbf{u}_2 - \lfloor \frac{\langle \mathbf{u}_1, \mathbf{u}_2 \rangle}{\|\mathbf{u}_1\|^2} \rceil \mathbf{u}_1.$$

Dass die Lagrange-Reduktion in der Tat $\mathbf{u}_2$ verkürzt, sieht man wie folgt: Wir schreiben $\mathbf{u}_2 = \mathbf{u}_2^\star + m\mathbf{u}_1$ mit $m = \|\mathbf{u}_1\|^{-2} \langle \mathbf{u}_1, \mathbf{u}_2 \rangle = k + t$, wobei $k = \lfloor m \rceil \in \mathbb{Z}$ und $|t| \leq \frac{1}{2}$. Damit folgt aus dem Satz von Pythagoras $\|\mathbf{u}_2 - k\mathbf{u}_1\|^2 = \|\mathbf{u}_2^\star\|^2 + t^2 \|\mathbf{u}_1\|^2 \leq \|\mathbf{u}_2^\star\|^2 + m^2 \|\mathbf{u}_1\|^2 = \|\mathbf{u}_2\|^2$ (mit Gleichheit nur falls $k = 0$ oder $k = \pm 1$, $t = \mp\frac{1}{2}$). Übrigens kann man die Rechnung leicht erweitern um zu sehen, dass unsere Wahl für k optimal ist und dass die Verkürzung zumindest $(k^2 - k)\|\mathbf{u}_1\|$ beträgt.

Die beiden Vektoren $\mathbf{u}_1$, $\tilde{\mathbf{u}}_2$ bilden wieder eine Basis, denn die Transformationsmatrix W ist von der Form $\left(\begin{smallmatrix} 1 & * \\ 0 & 1 \end{smallmatrix}\right)$, also unimodular.

Ist der neue Vektor $\tilde{\mathbf{u}}_2$ kürzer als $\mathbf{u}_1$, so können wir beide Vektoren vertauschen und das Verfahren wiederholen. Ansonsten sind wir fertig. Das ist der **Lagrange-Algorithmus** (auch Gauß-Algorithmus) zur Gitterreduktion.

Da das Quadrat der Länge eines der beiden Vektoren in jedem Schritt um mindestens 1 abnimmt, terminiert der Algorithmus.

Beispiel 9.9 (Lagrange'sche Gitterreduktion) Wenden Sie den Lagrange-Algorithmus auf die Basis

$$V = \begin{pmatrix} 5 & 8 \\ 0 & 1 \end{pmatrix}$$

an. Geben Sie das Hadamard-Verhältnis von V und von der reduzierten Basis an.

Lösung zu 9.9 Es gilt

$$\mathcal{H}(V) = 0.73$$

und wir berechnen

$$\mathbf{v}_2 - \lfloor \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{\|\mathbf{v}_1\|^2} \rceil \mathbf{v}_1 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}.$$

Da die Norm kleiner als die Norm von $\mathbf{v}_1$ ist, ist unsere neue Basis

$$V = \begin{pmatrix} -2 & 5 \\ 1 & 0 \end{pmatrix}.$$

Das Hadamard-Verhältnis hat sich auf $\mathcal{H}(V) = 0.96$ verbessert. Wir berechnen wieder

$$\mathbf{v}_2 - \lfloor \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{\|\mathbf{v}_1\|^2} \rceil \mathbf{v}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Da die Norm gleich der Norm von $\mathbf{v}_1$ ist, sind wir fertig. Unser Endresultat ist somit

$$V = \begin{pmatrix} -2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Es gilt $\mathcal{H}(V) = 1$ und wir haben eine Orthogonalbasis erhalten. ■

Die Lagrange'sche Gittereduktion löst insbesondere das SVP:

Satz 9.10 Ist V eine Gitterbasis und ist U die mit dem Lagrange-Algorithmus reduzierte Basis, dann ist $\mathbf{u}_1$ ein kürzerster Vektor und $\mathbf{u}_2$ ist ein kürzester Vektor, der kein Vielfaches von $\mathbf{u}_1$ ist. Zusätzlich gilt

$$\|\mathbf{u}_1\| \|\mathbf{u}_2\| \leq \frac{2}{\sqrt{3}} \det(\mathcal{L}).$$

Am Ende des Lagrange-Algorithmus gilt

$$\|\mathbf{u}_1\| \leq \|\mathbf{u}_2\| \quad \text{und} \quad |\mu_{1,2}| = \frac{|\langle \mathbf{u}_1, \mathbf{u}_2 \rangle|}{\|\mathbf{u}_1\|^2} \leq \frac{1}{2}$$

und damit folgt für einen beliebigen Gittervektor $\mathbf{a} = k_1\mathbf{u}_1 + k_2\mathbf{u}_2$

$$\begin{aligned} \|\mathbf{a}\|^2 &= k_1^2\|\mathbf{u}_1\|^2 + 2k_1k_2\langle \mathbf{u}_1, \mathbf{u}_2 \rangle + k_2^2\|\mathbf{u}_2\|^2 \\ &\geq (k_1^2 - |k_1k_2| + k_2^2)\|\mathbf{u}_1\|^2 \\ &= \left(\frac{3}{4}k_2^2 + (|k_1| - \frac{|k_2|}{2})^2\right)\|\mathbf{u}_1\|^2. \end{aligned}$$

Ist $|k_2| \geq 2$ so ist $\|\mathbf{a}\|^2 > \|\mathbf{u}_1\|^2$. Ist $|k_2| = 1$ so ist ebenfalls $\|\mathbf{a}\|^2 \geq \|\mathbf{u}_1\|^2$. Und auch im Fall $k_2 = 0$ gilt $\|\mathbf{a}\|^2 \geq \|\mathbf{u}_1\|^2$.

Gäbe es einen Vektor $\mathbf{v}_2 = k_2\mathbf{u}_2 + k_1\mathbf{u}_1$ der kürzer als $\mathbf{u}_2$ und kein Vielfaches von $\mathbf{u}_1$ ist, so ist $\mathbf{u}_1$, $\mathbf{v}_2 - k_1\mathbf{u}_1 = k_2\mathbf{u}_2$ ebenfalls eine Basis und daher muss $k_2 = \pm 1$ sein. Ohne Beschränkung der Allgemeinheit, können wir $k_2 = 1$ voraussetzten. Daraus folgt $\lfloor \|\mathbf{u}_1\|^{-2}\langle \mathbf{u}_1, \mathbf{v}_2 \rangle \rceil = k_1 + \lfloor \|\mathbf{u}_1\|^{-2}\langle \mathbf{u}_1, \mathbf{u}_2 \rangle \rceil = k_1$ und somit $\|\mathbf{u}_2\| = \|\mathbf{v}_2 - k_1\mathbf{u}_2\| \leq \|\mathbf{v}_2\|$.

Um die Ungleichung zu sehen verwenden wir

$$\|\mathbf{u}_2\|^2 = \|\mathbf{u}_2^\star\|^2 + \mu_{1,2}^2\|\mathbf{u}_1^\star\|^2 \leq \|\mathbf{u}_2^\star\|^2 + \frac{1}{4}\|\mathbf{u}_1\|^2 \leq \|\mathbf{u}_2^\star\|^2 + \frac{1}{4}\|\mathbf{u}_2\|^2$$

also

$$\|\mathbf{u}_2^\star\|^2 \geq \frac{3}{4}\|\mathbf{u}_2\|^2$$

und indem wir die Ungleichung noch mit $\|\mathbf{u}_1\|^2$ multiplizieren folgt die Behauptung.

Insbesondere erhalten wir in zwei Dimensionen die Ungleichung

$$\lambda_1(\mathcal{L}) \leq \left(\frac{4}{3}\right)^{1/4} \det(\mathcal{L})^{1/2}.$$

Die Strategie zur näherungsweisen Lösung des CVP ist nun auch klar: Zuerst mit dem Lagrange-Algorithmus die Basis reduzieren und danach das CVP mit dem Rundungsalgorithmus lösen.

Nun versuchen wir diese Strategie auf mehr als zwei Dimensionen zu übertragen. Die erweiterte Strategie von Hermite ist es, die Basisvektoren so lange paarweise zu reduzieren, bis keine Verbesserung mehr möglich ist.

Definition 9.11 Es sei

$$m_{i,j} = \frac{\langle \mathbf{u}_j, \mathbf{u}_i \rangle}{\|\mathbf{u}_i\|^2}.$$

Eine Gitterbasis U heißt **längenreduziert** falls

- $\|\mathbf{u}_1\| \leq \|\mathbf{u}_2\| \leq \cdots \leq \|\mathbf{u}_m\|$ und
- $|m_{i,j}| \leq \frac{1}{2}$ für alle $1 \leq i < j$.

In zwei Dimensionen liefert uns der Lagrange-Algorithmus eine längenreduziert Basis und von Hermite stammen verschiedene Strategien diesen zu erweitern. Eine einfache Variante ist (**Hermite-Algorithmus**):

- Wiederhole bis keine Verbesserung mehr möglich ist
  - Sortiere die Basisvektoren anhand ihrer Länge
  - Für $i = 1, \ldots, j-1$

    Falls $|m_{i,j}| \leq \frac{1}{2}$ gehe zum nächsten i.

    Reduziere: $\mathbf{u}_j \leftarrow \mathbf{u}_j - \lfloor m_{i,j} \rceil \mathbf{u}_i$ und springe zum Sortieren.

Da das Quadrat der Länge eines der Vektoren in jedem Schritt um mindestens 1 abnimmt, terminiert der Algorithmus.

Dieser Algorithmus liefert eine längenreduzierte Basis. Allerdings ist er vergleichsweise langsam, da bei jeder erfolgreichen Reduktion von vorne begonnen werden muss.

Bei der Reduktion kann man auch immer nur einen Vektor, und nicht wie bei Gram–Schmidt alle vorhergehenden Vektoren auf einmal, nehmen, da ansonsten die Länge des reduzierten Vektors größer werden könnte.

Bezeichnen wir mit $N = (\|\mathbf{u}_1\|^2 + \cdots + \|\mathbf{u}_m\|^2)^{1/2}$ die Hilbert–Schmidt-Norm der Basis, dann brauchen wir im schlimmsten Fall N^2 Schritte und in jedem Schritt müssen die Koeffizienten $m_{i,j}$ in einer Zeile und in einer Spalte neu berechnet werden. Also brauchen wir ganz grob $O(B^2 n^3)$ Operationen mit $B = \max\{\|\mathbf{u}_1\|, \ldots, \|\mathbf{u}_m\|\}$.

Eine Alternative ist es, eine möglichst orthogonale Basis zu erreichen indem man die Gewichte bezüglich der Gram–Schmidt-Basis reduzieren.

Definition 9.12 Es sei $U^\star$ die zu U gehörige Gram–Schmidt-Basis und

$$\mu_{i,j} = \frac{\langle \mathbf{u}_j, \mathbf{u}_i^\star \rangle}{\|\mathbf{u}_i^\star\|^2}.$$

Eine Gitterbasis U heißt **gewichtsreduziert**, falls

- $|\mu_{i,j}| \leq \frac{1}{2}$ für alle $1 \leq i < j$.

Eine gewichtsreduzierte Basis erhalten wir mit folgendem Algorithmus:

- Für $j = 2, \ldots, m$
  - Für $i = j-1, \ldots, 1$

    Falls $r = \lfloor \mu_{i,j} \rceil \neq 0$:

    $\mathbf{u}_j \leftarrow \mathbf{u}_j - r\mathbf{u}_i$ und $\mu_{k,j} \leftarrow \mu_{k,j} - r\mu_{k,i}$ für $1 \leq k \leq i$

Zunächst ist zu beachten, dass zu $\mathbf{u}_j$ nur Vielfache von $\mathbf{u}_i$ mit $i < j$ addiert werden, und dabei ändert sich $\mathbf{u}_j^\star$ nicht. Die Vektoren $\mathbf{u}_1^\star, \ldots, \mathbf{u}_m^\star$ bleiben also während der Gewichtsreduktion gleich. Die einzelnen Reduktionsschritte

$$\mathbf{u}_j \leftarrow \mathbf{u}_j - \lfloor \mu_{i,j} \rceil \mathbf{u}_i$$

ändern

$$\mu_{i,j} \leftarrow \mu_{i,j} - \lfloor \mu_{i,j} \rceil$$

und stellen die Gewichts-Bedingung für das aktuelle i sicher. Dass die zweite Schleife in umgekehrter Reihenfolge durchlaufen wird stellt sicher, dass alle nachfolgenden Änderungen von $\mathbf{u}_j$ nur noch Vielfache von $\mathbf{u}_k$ mit $k < i$ addieren und sich die Gewichte $\mu_{j,i}$ für $i > k$ nicht mehr ändern (da ja $\langle \mathbf{u}_k, \mathbf{u}_i^\star \rangle = 0$ für $i > k$ gilt).

Der Vorteil ist, dass nun die Reduktion in einem Durchlauf erfolgt und auch das Sortieren entfällt. Mit einer weiteren Operation versucht man noch (wie beim Lagrange-Algorithmus) die Basisvektoren nach den Längen von $\mathbf{u}_j^\star$ umzusortieren und die Reduktion zu wiederholen. Dazu vertauscht man $\mathbf{u}_j$ mit $\mathbf{u}_{j+1}$ falls sich dadurch die Länge von $\mathbf{u}_j^\star$ verkleinert. Das ist auch schon fast der berühmten **LLL-Algorithmus** von Arjen Lenstra, Hendrik Lenstra and László Lovász [LLL82]. Allerdings vertauscht man dabei nur wenn eine vorgegebene Toleranz überschritten wird, da nur dann garantiert werden kann, dass der Algorithmus die angestrebte Laufzeit erreicht.

Um diese Vertauschungsbedingung konkret zu formulieren, erinnern wir uns daran, dass wir $\mathbf{u}_j^\star$ berechnen indem wir die orthogonale Projektion von $\mathbf{u}_j$ auf $\mathrm{LH}\{\mathbf{u}_1, \ldots, \mathbf{u}_{j-1}\}$ bilden. Wenn wir anstelle von $\mathbf{u}_j$ den Vektor $\mathbf{u}_{j+1}$ nehmen, dann ist die Projektion von $\mathbf{u}_{j+1}$ auf $\mathrm{LH}\{\mathbf{u}_1, \ldots, \mathbf{u}_{j-1}\}$ gegeben durch $\mathbf{u}_{j+1}^\star + \mu_{j+1,j}\mathbf{u}_j^\star$. Es wird daher getauscht, wenn die **Lovász-Bedingung**

$$\delta \|\mathbf{u}_j^\star\|^2 \leq \|\mathbf{u}_{j+1}^\star + \mu_{j+1,j}\mathbf{u}_j^\star\|^2 = \|\mathbf{u}_{j+1}^\star\|^2 + \mu_{j+1,j}^2 \|\mathbf{u}_j^\star\|^2$$

verletzt ist, wobei $\delta \in (0, 1]$ der zuvor erwähnte Toleranzfaktor ist. Je näher δ bei 1 liegt, desto öfter wird getauscht und man erwartet sich daher auch eine bessere Basis. Demensprechend konvergiert der Algorithmus langsamer um so größer δ ist und für $\delta = 1$ kann nicht mehr garantiert werden, dass die Laufzeit polynomial ist. Üblicherweise wird $\delta = \frac{3}{4}$ gewählt.

Definition 9.13 Es sei $U^\star$ die zu U gehörige Gram–Schmidt-Basis und $\delta \in (0, 1]$. Eine Gitterbasis U heißt **LLL-reduziert**, falls

- $|\mu_{i,j}| \leq \frac{1}{2}$ für alle $1 \leq i < j$ (Gewichtsbedingung)
- $(\delta - \mu_{j+1,j}^2)\|\mathbf{u}_j^\star\|^2 \leq \|\mathbf{u}_{j+1}^\star\|^2$ (Lovász-Bedingung).

Der Ablauf des LLL-Algorithmus ist nun wie folgt:

- Führe die Gewichtsreduktion durch.

- Teste ob alle Vektoren die Lovász-Bedingung erfüllen. Erfüllt $\mathbf{u}_j$ die Lovász-Bedingung nicht, so tausche $\mathbf{u}_j$ mit $\mathbf{u}_{j+1}$ und springe zur Gewichtsreduktion.

  Nun ist die Lovász-Bedingung bis zur Stelle j erfüllt, allerdings müssen $\mathbf{u}_j^\star$, $\mathbf{u}_{j+1}^\star$ aktualisiert werden und die Vektoren sind ab der Stelle j ev. nicht mehr reduziert und wir müssen den Reduktionsschritt wiederholen. Die Gewichte können dabei rekursiv aus den alten Gewichten berechnet werden.

Satz 9.14 Der LLL-Algorithmus liefert eine LLL-reduzierte Basis in polynomialer Laufzeit $O(n^2(\log(B)+\log(n)))$, wobei $B = \max\{\|\mathbf{u}_1\|, \dots, \|\mathbf{u}_m\|\}$.

Die Strategie des LLL-Algorithmus ist es die Teilgitterdeterminanten $d_l = \det(\mathcal{L}(\mathbf{u}_1, \dots, \mathbf{u}_l))$ zu minimieren. Setzten wir $D = d_1 \cdots d_m$, dann ändert sich D genau dann, wenn wir $\mathbf{u}_j$ und $\mathbf{u}_{j+1}$ tauschen. In dem Fall gilt für den neuen Wert $\tilde{D} \leq \delta D$, denn die einzige Teilgitterdeterminante die sich ändert ist $\tilde{d}_j = \|\tilde{\mathbf{u}}_1^\star\| \cdots \|\tilde{\mathbf{u}}_1^\star\| = \|\mathbf{u}_j^\star\| \cdots \|\mathbf{u}_{j-1}^\star\| \|\mathbf{u}_{j+1}^\star + \mu_{j+1,j}\mathbf{u}_j^\star\| \leq \delta d_j$. Der Wert von D sinkt also exponentiell und da $D \geq 1$ ist, terminiert der Algorithmus nach maximal $\log_\delta(D)$ Schritten. Für $\delta = 1$ wissen wir nur $\tilde{D} < D$ und wir haben keine explizite Rate mehr. Für weitere Informationen sei auf auf [NV10] verwiesen. Eine gut lesbare Einführung findet sich in [Bre12].

Es ist zu beachten, dass das Ergebnis von der Reihenfolge der Vektoren zu Beginn abhängt und die Vektoren am Ende nicht der Länge nach sortiert sind. Ist man also auf der Suche nach einem kürzesten Vektor (SVP), so muss das Ergebnis vorher noch sortiert werden. In kleinen Dimensionen ist der kürzeste Vektor der reduzierten Basis oft der kürzeste Gittervektor und man kann Gleichheit in Satz 9.8 als hinreichendes Kriterium verwenden um nachzuweisen, dass man in der Tat einen kürzesten Vektor gefunden hat. Im Allgemeinen kann man zumindest zeigen, dass die Länge des gefundene Vektor maximal um einen festen Faktor daneben liegen kann:

Satz 9.15 Es sei $\delta > \frac{1}{4}$. Für eine LLL-reduzierte Basis U gilt

$$\|\mathbf{u}_1\| \leq \gamma^{(n-1)/2}\lambda_1(\mathcal{L}), \qquad \gamma = (\delta - \frac{1}{4})^{-1}.$$

Aus der Gewichts-Bedingung zusammen mit der Lovász-Bedingung folgt $\|\mathbf{u}_j^\star\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{u}_{j-1}^\star\|^2 \geq \cdots \geq (\delta - \frac{1}{4})^{j-1}\|\mathbf{u}_1\|^2$. Somit folgt die Behauptung aus Satz 9.8.

Das klingt auf den ersten Blick vielleicht nicht beeindruckend, ist in der Praxis aber sehr gut und der Algorithmus ermöglicht eine Vielzahl von interessanten Anwendungen, wie wir noch sehen werden.

Es gibt alternative Varianten, wie z.B. den **Block-Korkine–Zolotarev-Algorithmus** (**BKZ**). Dabei werden nicht schrittweise einzelne Vektoren reduziert, sondern mehrere Vektoren zu Blöcken zusammengefaßt und gemeinsam reduziert. Das liefert eine bessere Basis auf Kosten von längerer Laufzeit.

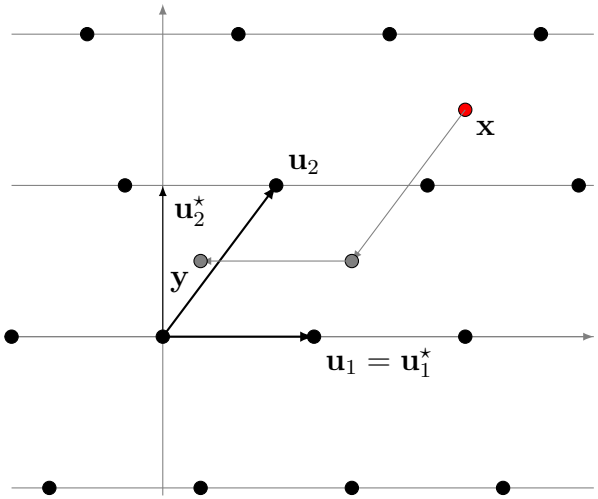

Abbildung 9.9: Babai-Näheste-Ebene-Algorithmus

Sucht man nach einer näherungsweisen Lösung des CVP, so gibt es eine Reihe von Möglichkeiten. Abgesehen vom Babai-Rundungsalgorithmus kann man den **Babai-Näheste-Ebene-Algorithmus** verwenden:

- $\mathbf{y} \leftarrow \mathbf{x}$
- Für $j = m, \ldots, 1$

  $$\mathbf{y} \leftarrow \mathbf{y} - \left\lfloor \frac{\langle \mathbf{u}_j^\star, \mathbf{y} \rangle}{\|\mathbf{u}_j^\star\|^2} \right\rceil \mathbf{u}_j$$

- Ausgabe: $\mathbf{x} - \mathbf{y}$

Der Algorithmus reduziert sukzessive den Normalabstand bezüglich der Gram–Schmidt-Basis. Er funktioniert insbesondere gut, wenn die Basis LLL reduziert ist (und somit die Längen der Gram–Schmidt-Vektoren geordnet sind).

Satz 9.16 (Babai) Der Gittervektor $\mathbf{a}$ den der Babai-Näheste-Ebene-Algorithmus mit Eingabe $\mathbf{x}$ liefert erfüllt

$$\|\mathbf{x} - \mathbf{a}\| \leq \frac{1}{2} \sqrt{\|\mathbf{u}_1^\star\|^2 + \cdots + \|\mathbf{u}_m^\star\|^2}.$$

Im Fall

$$\inf_{\mathbf{b} \in \mathcal{L}} \|\mathbf{x} - \mathbf{b}\| \leq \frac{1}{2} \min_{1 \leq j \leq m} \|\mathbf{u}_j^\star\|$$

löst der Algorithmus das CVP.

Die Idee ist in Abbildung 9.9 veranschaulicht: Wir zerlegen das Gitter in Geraden die normal zu $\mathbf{u}_2^\star$ stehen und in die Gitterpunkte auf diesen Geraden. Im ersten Schritt reduzieren wir $\mathbf{x}$ indem wir den Punkt möglichst nahe an die Gerade die durch den Ursprung geht bringen. Im zweiten Schritt suchen wir den Gitterpunkt auf der Geraden der dem Ursprung am nächsten liegt.

Allgemein passiert Folgendes: Wir schreiben $\mathbf{x} = \mathbf{x}_{m-1} + \mu_m \mathbf{u}_m^\star$ mit $\mathbf{x}_{m-1} \in \mathrm{LH}\{\mathbf{u}_1^\star, \ldots, \mathbf{u}_{m-1}^\star\}$ und $k_m = \frac{\langle \mathbf{u}_m^\star, \mathbf{x}\rangle}{\|\mathbf{u}_m\|^2}$. Indem wir $\mu_m = k_m + t_m$ mit $k_m = \lfloor \mu_m \rceil$ schreiben, erhalten wir einen Vektor $\mathbf{y}_{m-1} = \mathbf{x} - k_m \mathbf{u}_m = \tilde{\mathbf{x}}_{m-1} + t_m \mathbf{u}_m^\star$ der sich von $\mathbf{x}$ nur um einen Gittervektor unterscheidet und die Komponente in Richtung $\mathbf{u}_m^\star$ minimiert. Im nächsten Schritt zerlegen wir $\tilde{\mathbf{x}}_{m-1} = \mathbf{x}_{m-2} + \mu_{m-1}\mathbf{u}_{m-1}^\star$ und verfahren wie vorher. Am Ende erhalten wir einen Vektor $\mathbf{y}_1 = t_1\mathbf{u}_1^\star + \ldots + t_m\mathbf{u}_m^\star$ der in der Nähe des Ursprungs liegt und sich von $\mathbf{x}$ um einen Gittervektor unterscheidet. Insbesondere gilt $\|\mathbf{y}_1\|^2 \le \frac{1}{4}(\|\mathbf{u}_1^\star\|^2 + \cdots + \|\mathbf{u}_m^\star\|^2)$.

Die Zuordnung von Punkten zu Gitterpunkten partitioniert den Raum (vgl. auch Abbildung 9.10) in Quader mit den Seitenlängen $\|\mathbf{u}_1^\star\|, \ldots, \|\mathbf{u}_m^\star\|$ und Mittelpunkten an den Gitterpunkten. Setzten wir an jeden Gitterpunkt eine Kugel vom Radius $\frac{1}{2}\min_{1\le j\le m}\|\mathbf{u}_j^\star\|$, so passt diese vollständig in den zugehörigen Quader und der Algorithmus trifft für jeden Punkt in der Kugel die optimale Entscheidung, löst also am Ende das CVP.

Bemerkung: Indem wir statt mit Kugeln, mit Würfeln arbeiten, sehen wir, dass wir die Euklid'sche Norm auch durch $\|\mathbf{x}\|_\infty = \max_{1\le j\le n}|x_j|$ ersetzen können.

Das CVP ist also in jedem Fall dann leicht, wenn $\mathbf{x}$ nahe genug an einem Gitterpunkt liegt.

Beispiel 9.17 Lösen Sie das CVP aus dem Beispiel 9.5 mit der Basis V näherungsweise mit dem Babai-Näheste-Ebene-Algorithmus.

Lösung zu 9.17 Die Gram–Schmidt-Basis zu V kennen wir bereits aus Beispiel 9.6. Also ist im ersten Schritt

$$\mathbf{y} = \mathbf{x} - \lfloor \frac{\langle \mathbf{v}_2^\star, \mathbf{x}\rangle}{\|\mathbf{u}_2^\star\|^2} \rceil \mathbf{v}_2 = (5.2, 0.4) - \lfloor 0.4 \rceil (8,1) = (5.2, 0.4)$$

und im zweiten Schritt

$$\mathbf{y} = \mathbf{y} - \lfloor \frac{\langle \mathbf{v}_1^\star, \mathbf{x}\rangle}{\|\mathbf{u}_1^\star\|^2} \rceil \mathbf{v}_1 = (5.2, 0.4) - \lfloor 1.04 \rceil (5,0) = (0.2, 0.4).$$

Somit ist der zugehörige Gitterpunkt

$$\mathbf{a} = \mathbf{x} - \mathbf{y} = (5, 0).$$

Die beiden Verfahren sind in Abbildung 9.10 veranschaulicht. ■

Alternativ kann man sich auch eines Tricks bedienen: Man erweitert das Gitter um eine weitere Dimension und fügt $\mathbf{x}$ zur Basis hinzu, wobei die neue Komponente bei den Basisvektoren mit 0 und bei $\mathbf{x}$ mit 1 ergänzt wird. Nun wendet man LLL an und erhält als Ergebnis einen Vektor mit kleiner Differenz zwischen $\mathbf{x}$ und dem Gitter (der kleinste Vektor in der LLL-reduzierten Basis mit letzter Komponente gleich 1). Siehe auch [MV13].

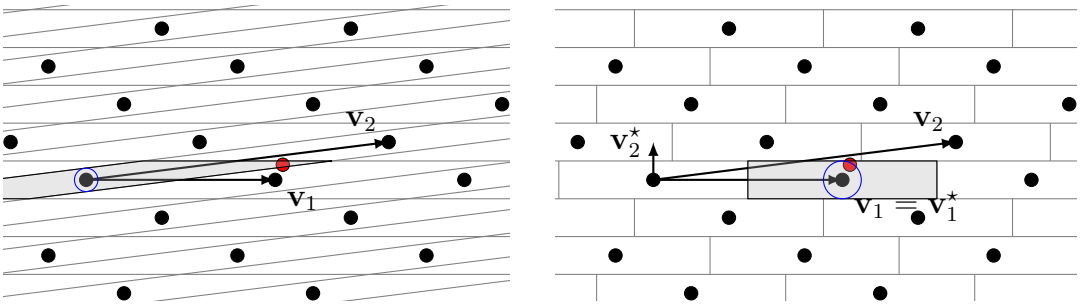

Abbildung 9.10: Rundungsbereiche für den Babai-Rundungsalgorithmus (links) und den Babai-Näheste-Ebene-Algorithmus (rechts). Innerhalb der Kreise trifft der Algorithmus sicher die richtige Entscheidung.

9.2.1 Abschätzungen an den kürzesten Vektor

Da eine exakte Lösung des SVP meist nicht möglich ist, möchte man zumindest Schranken an $\lambda_1(\mathcal{L})$ haben. Eine untere Schranke haben wir bereits in Satz 9.8 gefunden. Nun wollen wir auch eine obere Schranke herleiten. Als Vorbereitung brauchen wir ein Kriterium, wann eine Menge zumindest einen Gittervektor enthält.

Satz 9.18 (Blichfeldt) Es sei $\mathcal{L}$ ein Gitter von vollem Rang und S eine beschränkte Menge vom Volumen $|S|$. Gilt $|S| > \det(\mathcal{L})$, dann gibt es zwei verschiedene Punkte $\mathbf{x}_1, \mathbf{x}_2 \in S$ mit $\mathbf{x}_1 - \mathbf{x}_2 \in \mathcal{L}$.

Die Translationen um Gittervektoren des Fundamentalbereichs partitionieren den $\mathbb{R}^n$. Insbesondere wird dadurch S in eine endliche Anzahl von Teilen zerlegt. Wenn wir diese Teile um die entsprechenden Gittervektoren in den Fundamentalbereich verschieben, dann muss es aufgrund der Annahme $\mathrm{Vol}(S) > \det(\mathcal{L})$ zu einer Überschneidung kommen.

Satz 9.19 (Hermite) Es sei $\mathcal{L}$ ein n-dimensionales Gitter. Dann gilt

$$\lambda_1(\mathcal{L}) \leq 2V_n^{-1/n} \det(\mathcal{L})^{1/n},$$

wobei

$$V_n = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$$

das Volumen der Einheitskugel $B_1(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^n | \|x\| \leq 1\}$ in $\mathbb{R}^n$ ist.

Es sei $|B_r(\mathbf{0})| > \det(\mathcal{L})$. Dann gibt es zwei verschiedene Punkte $\mathbf{x}_1, \mathbf{x}_2 \in B_r(\mathbf{0})$ mit $\mathbf{a} = \mathbf{x}_1 - \mathbf{x}_2 \in \mathcal{L}$ und aus der Dreiecksungleichung folgt $\|\mathbf{a}\| \leq \|\mathbf{x}_1\| + \|\mathbf{x}_2\| < 2r$, also $\mathbf{a} \in B_{2r}(\mathbf{0}) \setminus \{\mathbf{0}\}$. Da

$B_{\lambda_1(\mathcal{L})}(\mathbf{0})$ keinen Gitterpunkt ausser $\mathbf{0}$ enthält, muss $|B_{\lambda_1(\mathcal{L})/2}(\mathbf{0})| = V_n(\lambda_1(\mathcal{L})/2)^n < \det(\mathcal{L})$ gelten.

Aus der Stirling'schen Näherung folgt

$$V_n^{-1/n} \approx \sqrt{\frac{n}{2\pi \mathrm{e}}}$$

und indem man verwendet, dass der Würfel mit Seitenlänge $\frac{2}{\sqrt{n}}$ in die Einheitskugel passt, erhält man $V_n \geq (\frac{2}{\sqrt{n}})^n$ und

$$V_n^{-1/n} \leq \frac{\sqrt{n}}{2}.$$

9.2.2 Anwendung: Coppersmith-Algorithmus

Als eine erste Anwendung können wir den **Coppersmith-Algorithmus** zum Auffinden von *kleinen* Nullstellen eines Polynoms besprechen, der einen möglichen Angriff auf RSA eröffnet.

Wir suchen also die Nullstellen eines Polynoms

$$P(x) = x^d + P_{d-1}x^{d-1} + \cdots + P_1 x + P_0$$

vom Grad d in $\mathbb{Z}_n$. Interessant ist nur der Fall in dem n zusammengesetzt ist, denn im Fall einer Primzahl haben wir einen Körper und da gibt es effektive Algorithmen zur Faktorisierung (**Berlekamp-Algorithmus** bzw. **Cantor-Zassenhaus-Algorithmus**).

Die Idee ist es, das Polynom P durch ein Polynom

$$Q(x) = x^d + Q_{d-1}x^{d-1} + \cdots + Q_1 x + Q_0$$

zu ersetzten, das die gleichen Nullstellen in $\mathbb{Z}_n$ besitzt, aber *kleinere* Koeffizienten besitzt. Und zwar hätten wir gerne

$$C^d + |Q_{d-1}|C^{d-1} + \cdots + |Q_1|C + |Q_0| < n,$$

denn dann gilt für $|x| \leq C$, dass $|Q(x)| < n$. Also haben wir

$$|x| \leq C: \qquad P(x) = 0 \bmod n \;\Leftrightarrow\; Q(x) = 0.$$

Die Nullstellen von Q können leicht numerisch gefunden werden und um die Nullstellen modulo n nicht zu ändern, addieren wir zu P nur (ganzzahlige) Vielfache der Monome

$$Q^{(j)}(x) = nx^j, \qquad 0 \leq j < d.$$

Indem wir die Koeffizienten unserer Polynome als Vektoren auffassen, können wir die Suche nach einem optimalen Q als SVP im Gitter mit der Basis

$$U = \begin{pmatrix} n & & & & & P_0 \\ & Cn & & & & P_1C \\ & & nC^2 & & & P_2C^2 \\ & & & \ddots & & \vdots \\ & & & & nC^{d-1} & P_{d-1}C^{d-1} \\ & & & & & C^d \end{pmatrix}$$

auffassen. Aber eigentlich ist das gar nicht notwendig, den die best mögliche Reduktion ergibt sich wenn wir P_j durch $P_j - n$ ersetzten falls $|P_j - n| < |P_j|$ (unabhängig von C).

Beispiel 9.20 Gesucht sind kleine Nullstellen des Polynoms

$$P(x) = x^2 + 33x + 32$$

in $\mathbb{Z}_{35}$.

Lösung zu 9.20 Indem wir P wie beschrieben reduzieren erhalten wir

$$Q(x) = x^2 - 2x - 3 = (x+1)(x-3)$$

und da

$$3^2 + 2 \cdot 3 + 3 = 18 < 35$$

sind beide Nullstellen $x = -1, 3$ von $Q(x)$ in $\mathbb{Z}$ auch Nullstellen von $P(x)$ in $\mathbb{Z}_{35}$. Achtung: $P(x)$ hat in $\mathbb{Z}_{35}$ noch weitere Nullstellen $x = -11, 13$, die mit diesem Verfahren nicht gefunden wurden. ■

Das Problem ist, dass unsere Bedingung an die Koeffizienten von Q nur selten erfüllt ist. Wir müssen also noch nachbessern. Dazu erweitern wir unsere Funktionen und betrachten

$$Q^{(j,k)}(x) = n^{m-k} x^j P(x)^k, \qquad 0 \le j < d,\ 0 \le k \le m$$

für ein noch zu wählendes $m \in \mathbb{N}$. Die entscheidende Tatsache ist, dass jede Nullstelle x_0 von P in $\mathbb{Z}_n$

$$Q^{(j,k)}(x_0) = 0 \pmod{n^m}$$

erfüllt. Das gilt natürlich auch für jede (ganzzahlige) Linearkombination

$$Q(x) = Q_0 + Q_1 x + \cdots + Q_{d(m+1)-1} x^{d(m+1)-1}$$

aus diesen Funktionen. Und falls

$$|Q_0| + |Q_1| C + \cdots + |Q_{d(m+1)-1}| C^{d(m+1)-1} < n^m$$

erfüllt ist, dann gilt

$$|x| \le C: \qquad P(x) = 0 \bmod n \;\Rightarrow\; Q(x) = 0.$$

Für $d = 2$ und $m = 2$ sieht die Gitterbasis z.B. wie folgt aus:

$$\begin{pmatrix} n^2 & 0 & nP_0 & 0 & P_0^2 & 0 \\ 0 & n^2C & nCP_1 & nCP_0 & 2CP_0P_1 & CP_0^2 \\ 0 & 0 & nC^2 & nC^2P_1 & C^2(P_1^2+2P_0) & 2C^2P_0P_1 \\ 0 & 0 & 0 & nC^3 & 2C^3P_1 & C^3(P_1^2+2P_0) \\ 0 & 0 & 0 & 0 & C^4 & 2C^4P_1 \\ 0 & 0 & 0 & 0 & 0 & C^5 \end{pmatrix}$$

Erfüllt der kürzeste Vektor $\mathbf{u}$ der LLL-reduzierten Basis $\sum_j |u_j| < n^m$, so können wir $Q_j = u_{j-1}C^{-j}$ wählen. Bleibt noch die Frage wie C zu wählen ist.

Die Gitterdimension ist $l = d(m+1)$ und die Gitterdeterminante ist

$$\det(\mathcal{L}) = C^{0+1+\cdots l-1} n^{(m+(m-1)+\cdots+1+0)d} = C^{l(l-1)/2} n^{d(m+1)m/2}.$$

Der kürzeste Vektor einer LLL-reduzierten Basis erfüllt (Satz 9.15 mit $\delta = \frac{3}{4}$, $\gamma = 2$ und Satz 9.19)

$$\|\mathbf{u}\| \le 2^{(l-1)/2}\lambda_1(\mathcal{L}) \le 2^{(l+1)/2} V_l^{-1/l} \det(\mathcal{L})^{1/l} = 2^{(l+1)/2} V_l^{-1/l} C^{(l-1)/2} n^{m/2}.$$

Wir brauchen

$$|u_1| + \cdots + |u_l| \le \sqrt{l}\|\mathbf{u}\| < n^m,$$

also müssen wir

$$C < \frac{C_l}{2} n^{m/(l-1)}, \qquad C_l = (V_l/(2\sqrt{l}))^{2/(l-1)},$$

wählen. Für $m \to \infty$ gilt $C_l \to 1$ und $\frac{m}{l-1} \to 1/d$. In der Praxis sind dreistellige Gitterreduktionen aber schon eine Herausforderung.

Beispiel 9.21 Gesucht sind kleine Nullstellen des Polynoms

$$P(x) = x^2 + 1304x + 455$$

in $\mathbb{Z}_{1400}$.

Lösung zu 9.21 Wir versuchen $m = 1$, also $l = 2(1+1) = 4$. Für C wählen wir

$$C = \lfloor \frac{C_l}{2} n^{m/(l-1)} \rfloor = 2.$$

Das ist nicht berauschend und liegt daran, dass $n = 1400$ zu klein ist um in Bereiche zu kommen bei denen der Algorithmus praktisch relevante Ergebnisse

liefert. Ein kleines Computerprogramm könnte in diesem Beispiel sowieso leicht alle 1400 Möglichkeiten durchprobieren. Im Prinzip kann man auch einfach versuchen C größer zu wählen, da der LLL-Algorithmus in der Regel bessere Ergebnisse liefert als durch die Schranke garantiert. Da wir am Ende die erhaltenen Nullstellen sowieso testen müssen, kann dabei nicht viel schief gehen.

Wir bleiben bei $C = 2$ und die Gitterbasis lautet dann

$$\begin{pmatrix} 1400 & 0 & 455 & 0 \\ 0 & 2800 & 2608 & 910 \\ 0 & 0 & 4 & 5216 \\ 0 & 0 & 0 & 8 \end{pmatrix}.$$

Gitterreduktion (und Sortieren der Vektoren nach ihrer Länge) liefert

$$\begin{pmatrix} 35 & 35 & -70 & -35 \\ -54 & 36 & 28 & -126 \\ -44 & 84 & 24 & 76 \\ 24 & 48 & 80 & -8 \end{pmatrix}$$

und das zum kürzesten Vektor gehörige Polynom ist

$$Q(x) = 3x^3 - 11x^2 - 27x + 35.$$

Die ganzzahligen Nullstellen sind $x = 1$ und $x = 5$, wovon aber nur $x = 5$ eine Nullstelle von $P(x)$ in $\mathbb{Z}_n$ ist.

Wenn man obige Bedingung $3x^3 + 11x^2 + 27x + 35 < n^m$ an die Koeffizienten von Q überprüft, dann findet man, dass sie bis $|x| \leq 8$ erfüllt ist, also haben wir in der Tat alle Nullstellen mit $|x| \leq 8$ gefunden. Möchte man weiter kommen, so muss man m erhöhen. Mit $m = 2$ erhält man $C = 5$ und die Gitterbasis lautet

$$\begin{pmatrix} 1960000 & 0 & 637000 & 0 & 207025 & 0 \\ 0 & 9800000 & 9128000 & 3185000 & 5933200 & 1035125 \\ 0 & 0 & 35000 & 45640000 & 42533150 & 29666000 \\ 0 & 0 & 0 & 175000 & 326000 & 212665750 \\ 0 & 0 & 0 & 0 & 625 & 1630000 \\ 0 & 0 & 0 & 0 & 0 & 3125 \end{pmatrix}.$$

Wie zuvor liefert Gitterreduktion

$$\begin{pmatrix} -22050 & -11025 & -22050 & -22050 & -19600 & 335650 \\ -23275 & -23450 & 350 & 31850 & -7175 & 302575 \\ 24700 & -7150 & -17300 & -4300 & 4400 & 334900 \\ 13750 & -6500 & 31500 & -20500 & 16750 & 326250 \\ 3750 & 29375 & 13750 & 8750 & -60000 & 326250 \\ 3125 & 18750 & -6250 & 6250 & 65625 & 334375 \end{pmatrix}$$

und das zum kürzesten Vektor gehörige Polynom ist

$$Q(x) = x^5 + 6x^4 + 110x^3 + 988x^2 - 4655x - 22050,$$

dessen ganzzahlige Nullstellen $x = -9$ und $x = 5$ sind. Einsetzten ergibt, dass -9 ebenfalls eine Nullstelle von $P(x)$ ist. Überprüfen der Bedingung $x^5 + 6x^4 + 110x^3 + 988x^2 + 4655x + 22050 < n^m$ zeigt, dass sie bis $|x| \leq 15$ erfüllt ist. Wir haben also nun alle Nullstellen mit $|x| \leq 15$ gefunden.

Achtung: Auch wenn unser Polynom nur Grad 2 hat, sind damit noch lange nicht alle Nullstellen gefunden, da $\mathbb{Z}_n$ kein Körper ist! Eine vollständige Suche liefert: $-609, -595, -359, -345, -259, -245, -9, 5, 91, 105, 341, 355, 441, 455, 691$. ■

Weitere Verbesserungen findet man in [Bre12].

9.3 Gitterbasierte Kryptographie

Wir besprechen nun einige ausgewählte Algorithmen deren Sicherheit auf der Schwierigkeit von Gitterproblemen basiert. Einen Überblick findet man auch in [MR09]. Da es mit den elliptischen Kurven aktuell extrem effektive Verfahren gibt, die eigentlich keinerlei Wünsche offen lassen, liegt die Latte für Alternativen extrem hoch, zumindest solange leistungsfähige Quantencomputer noch ausreichend weit entfernt scheinen. Wir wollen uns hier in erster Linie mit den zugrundeliegenden mathematischen Ideen vertraut machen. Diese Ideen wurden in den letzten Jahren sukzessive verbessert, wobei es dabei meist nicht um die Sicherheit ging, sondern darum, die Effektivität zu erhöhen um mit aktuellen Verfahren mithalten zu können. Wir werden daher in der Regel vereinfachte Varianten beschreiben und danach Hinweise geben wie diese in der Praxis angewandt werden um maximale Effektivität zu erreichen.

9.3.1 GGH

Wir beginnen mit dem **GGH** Verfahren von O. Goldreich, S. Goldwasser und S. Halevi [GGH97], dessen Grundidee auf der Tatsache basiert, dass eine gute Gitterbasis als Falltüre für eine schlechte Basis fungiert.

- Alice wählt eine gute Basis $U \in \mathbb{Z}^{n\times n}$ für ein Gitter und erzeuge eine schlechte Basis $V = UW$. Die gute Basis U ist ihr geheimer Schlüssel und die schlechte Basis V ist ihr öffentlicher Schlüssel.
- Möchte Bob eine Nachricht $\mathbf{x} \in \mathbb{Z}_2^n$ an Alice schicken, so wählt er eine kleine Nonce $\mathbf{r} \in \mathbb{Z}^n$ und sendet $\mathbf{y} = V\mathbf{x} + \mathbf{r}$ an Alice.
- Alice löst CVP mit U um $\mathbf{x}$ zu erhalten.

Als erstes stellt sich die Frage, wie die Basen zu generieren sind. Die gute Basis könnte man in der Form

$$U = a_0 \mathbb{I}_n + A$$

mit $|A_{ij}| \leq M$ und $a_0 = 2Mn$ wählen. Dann gilt mit

$$\|A\| = \max_i \sum_j |A_{ij}|,$$

dass

$$\|U^{-1}\| \leq \frac{1}{a_0 - \|A\|} \leq \frac{2}{Mn}.$$

Gilt

$$|r_j| < \frac{Mn}{4},$$

dann wird sicher richtig entschlüsselt, denn es folgt

$$U^{-1}\mathbf{y} = W\mathbf{x} + U^{-1}\mathbf{r}$$

mit

$$|(U^{-1}\mathbf{r})_i| \leq \|U^{-1}\| \max |r_j| < \frac{1}{2}.$$

Die schlechte Basis kann man wählen indem man elementare Spaltenoperationen auf U anwendet, also ganzzahlige Vielfache der anderen Spalten zu einer Spalte addiert und Spalten vertauscht.

Die Sicherheit des Verfahrens beruht darauf, dass Eve mit V das CVP nicht lösen kann. Dazu muss einerseits die Gitterdimension n groß sein, damit eine Gitterreduktion praktisch nicht möglich ist (in der Praxis ca. $n \geq 400$) und anderseits darf auch $\mathbf{r}$ nicht zu klein sein, da wir ja gesehen haben, dass die Rundungsalgorithmen das CVP immer lösen, wenn $\mathbf{r}$ (abhängig von der Basis) genügend klein ist. Der ursprüngliche Vorschlag in [GGH97] war es, die Komponenten von $\mathbf{r}$ zufällig als $\pm\sigma$ mit einem festen kleinen Wert σ (z.B. $\sigma = 3$) zu wählen. Das hat sich aber als schlechte Entscheidung herausgestellt, da dann der Klartext modulo σ bekannt ist, wodurch sich durch Lösen des Gleichungssystems in $\mathbb{Z}_\sigma$ die Möglichkeiten für den Klartext so weit einschränken lassen, dass das CVP lösbar wird [Ngu99].

Das kann man natürlich beheben, aber das Verfahren hat noch einige weitere Nachteile: Aus praktischer Sicht sind das die zu großen Schlüssel und aus theoretischer Sicht das Problem, dass es nicht ganz klar ist, in wie weit durch die spezielle Wahl der Basen bzw. der Forderungen an $\mathbf{r}$ das CVP erleichtert wird.

Beispiel 9.22 Wählen Sie für $n = 4$, $M = 100$ eine private und eine öffentliche Basis. Was ist für Ihre Basis eine passende Schranke an $\mathbf{r}$? Verschlüsseln Sie die Nachricht „`FAD!`“ indem Sie sie die ASCII-Codes der Buchstaben als Vektor auffassen.

Lösung zu 9.22 Wir wählen $a_0 = 2Mn = 800$ und addieren zu $a_0\mathbb{I}_4$ eine Zufallsmatrix mit Einträgen zwischen -100 und 100:

$$U = \begin{pmatrix} 867 & -92 & -67 & -12 \\ 22 & 733 & -86 & -3 \\ 94 & 70 & 900 & 74 \\ 65 & -5 & 32 & 789 \end{pmatrix}.$$

Unsere Abschätzung garantiert uns eine korrekte Entschlüsselung falls $|r_j| < \frac{Mn}{4} = 100$. Wenn wir nicht die Abschätzung verwenden, sondern U^{-1} berechnen, dann sehen wir das der Rundungsalgorithmus sogar für

$$|r_j| < \frac{1}{2\|U^{-1}\|} = 327.6$$

garantiert korrekt entschlüsselt. Mit dem Babai-Näheste-Ebene-Algorithmus kann die Korrektheit sogar für $|r_j| < 369.6$ garantiert werden.

Für die öffentliche Basis V mischen wir U mit elementaren Spaltenoperationen durch:

$$V = \begin{pmatrix} 186547 & 331514 & -210901 & 93221 \\ -89908 & 18530 & 10576 & 26499 \\ -11326 & 62238 & -29314 & 27290 \\ -36693 & -122021 & 70449 & -41141 \end{pmatrix}.$$

Das Hadamardverhältnis ist $\mathcal{H}(V) = 0.0056$ und mit dieser Basis kann eine korrekte Entschlüsselung nur für $|r_j| < \frac{1}{2\|V^{-1}\|} = 0.041$ garantiert werden (mit dem Babai-Näheste-Ebene-Algorithmus verbessert sich der Wert nur unwesentlich auf $|r_j| < 0.08$). Es ist also extrem unwahrscheinlich, dass für ein genügend großes $\mathbf{r}$ mit V korrekt entschlüsselt werden kann.

Um die Nachricht zu verschlüsseln, codieren wir also $\mathbf{x} = (70, 65, 68, 33)$, und wählen einen Zufallsvektor $\mathbf{r}$ mit $|r_j| < 327$:

$$\mathbf{r} = (-121, -177, 82, -134).$$

Damit erhalten wir

$$\mathbf{y} = V\mathbf{x} + \mathbf{r} = (23341775, -3495533, 2159879, -7067034).$$

Mit dem Rundungsalgorithms kann die Nachricht wieder entschlüsselt werden

$$V^{-1}U\lfloor U^{-1}\mathbf{y}\rceil = \mathbf{x}.$$

Versucht Eve mit V zu entschlüsseln, $\tilde{\mathbf{x}} = \lfloor V^{-1}\mathbf{y}\rceil = \mathbf{x} + \lfloor V^{-1}\mathbf{r}\rceil$, so wird das in der Regel nicht klappen.

Die Matrizen $W = V^{-1}U$ und U^{-1} sollten natürlich vorberechnet werden und als privater Schlüssel (anstelle von U) gespeichert werden.

Da in unserem Fall die Matrixdimension n zu klein ist, kann man leicht V mittels Gitterreduktion reduzieren um

$$V_L = \begin{pmatrix} 92 & -867 & -12 & 67 \\ -733 & -22 & -3 & 86 \\ -70 & -94 & 74 & -900 \\ 5 & -65 & 789 & -32 \end{pmatrix}$$

zu erhalten. Damit klappt die Entschlüsselung mittels Rundungsalgorithmus ebenfalls für $|r_j| < 327$. ■

Es ist übrigens auch möglich ein Signaturverfahren zu bekommen. Die Idee ist es, den Hashwerten Vektoren $\mathbf{x} \in \mathbb{R}^n$ zuzuordnen. Alice signiert dann indem Sie das CVP löst und den Gittervektor $\mathbf{a}$ als Signatur bekannt gibt. Bob überprüft die Signatur indem er prüft ob $\mathbf{a} \in \mathcal{L}(V)$ ein Gittervektor ist und ob der Abstand $\|\mathbf{a} - \mathbf{x}\|$ unter einer vorgegeben Toleranz ist. Der Wert für die Toleranz muss heuristisch geschätzt bzw. experimentell bestimmt werden. Allerdings ist die Signatur auch für alle Vektoren innerhalb der Toleranz gültig und das wiederum erleichtert den Geburtstagsangriff. Hat man viele Dokumente signiert, so liegen die Differenzen zwischen Signatur und zugehörigen Hashwert im Fundamentalbereich von U und dieser kann darüber erhalten werden [NR06].

9.3.2 NTRU

In diesem Abschnitt wollen wir das **NTRU** Verfahren aus dem Jahr 1996 von J. Hoffstein, J. Pipher und J. H. Silverman [HPS08] besprechen, das im Standard ANSI X9.98-2010 (R2017) für die Finanzindustrie akkreditiert ist und (in der Variante NTRU-Prime) aktuell bereits bei OpenSSH eingesetzt wird.

Die prinzipielle Idee ist es, die Nachricht in einer Kongruenz zu verstecken:

- Gegeben sei p und wir nennen eine Zahl $< \sqrt{p/2}$ klein.
- Alice wählt $\sqrt{p/4} < g$ klein und $f \in \mathbb{Z}_p^* \cap \mathbb{Z}_g^*$ klein und berechnet

  $$F_p = f^{-1} \pmod{p}, \qquad F_g = f^{-1} \pmod{g}$$

  und

  $$h = F_p g \pmod{p}.$$

  Öffentlicher Schlüssel: (h, p)
  Privater Schlüssel: (f, F_g)
- Bob verschlüsselt die Nachricht m klein mit einer Nonce r klein via

  $$c = rh + m \pmod{p}.$$

- Alice entschlüsselt mit

$$m = F_g(fc \bmod p) \pmod g.$$

Da f, g, r und m klein sind gilt $fc \bmod p = rg + fm < p$. Damit folgt $F_g(rg + fm) \bmod g = m$ und somit funktioniert das Verfahren.

Beispiel 9.23 Alice wählt

$$p = 20011, \qquad f = 83, \quad g = 87.$$

a) Wie lautet der öffentliche bzw. private Schlüssel?
b) Wie verschlüsselt Bob die Nachricht $m = 21$?
c) Wie entschlüsselt Alice die Nachricht von Bob?

Lösung zu 9.23 a) Alice berechnet $F_p = f^{-1} = 7474 \pmod p$, $F_g = f^{-1} = 65 \pmod g$ und $h = gF_p = 9886 \pmod p$. Ihr öffentlicher Schlüssel ist $(h, p) = (9886, 0011)$ und ihr privater $(f, F_g) = (83, 65)$.
b) Bob wählt (z.B.) $r = 65$ und verschlüsselt $c = rh + m = 2259 \pmod p$.
c) Alice berechnet $fc = 187497$ und entschlüsselt $F_g(fc) \pmod g = 21$. ■

Die (vermeintliche) Sicherheit des Verfahrens basiert auf der Tatsache, dass die beiden Operationen, modulo p und modulo g zu rechnen, nicht miteinander kompatibel sind. Insbesondere können diese Operationen im allgemeinen nicht vertauscht werden $((5 \bmod 2) \bmod 3 = 1 \neq 0 = (5 \bmod 3) \bmod 2)$. Deshalb mussten wir die Zahlen f, g, m, r klein wählen, damit bei Entschlüsselung an einer bestimmten Stelle in $\mathbb{Z}$ kein Überlauf auftritt und somit Alice korrekt entschlüssel kann. Haben p und g gemeinsame Teiler, so wird es für Eve leichter, denn sie kann dann alle (kleinen) Teiler t von p durchprobieren und für $t = \mathrm{ggT}(p, g)$ erhält sie $m = c \pmod t$. Im Fall $\mathrm{ggT}(p, g) = g$ also sogar die ganze Nachricht. Wir sollten also $\mathrm{ggT}(p, g) = 1$ verlangen (was z.B. durch p prim sicher gestellt ist). Leider hat das Verfahren aber noch eine weitere Schwachstelle, da der private Schlüssel aus dem öffentlichen berechnet werden kann.

Eve möchte also aus dem öffentlichen Schlüssel (h, p) den privaten (f, g) berechnen. Auf den ersten Blick scheint das schwer, da h das Produkt aus f^{-1} und g ist. Allerdings wissen wir, dass f und g klein sind. Wir suchen also einen kurzen Vektor (f, g). Von f wissen wir nichts, aber g erfüllt die Gleichung $g = fh + tp$, also

$$\begin{pmatrix} f \\ g \end{pmatrix} = f \begin{pmatrix} 1 \\ h \end{pmatrix} + t \begin{pmatrix} 0 \\ p \end{pmatrix},$$

wobei wir die (für Eve) unbekannten Größen vor die Vektoren gezogen haben. Wir suchen also einen kurzen Vektor in einem zweidimensionalen Gitter.

Beispiel 9.24 Berechnen Sie den geheimen Schlüssel von Alice aus dem letzten Beispiel indem Sie das zugehörige Gitterproblem lösen.

Lösung zu 9.24 Das zugehörige Gitter lautet

$$V = \begin{pmatrix} 1 & 0 \\ h & p \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 9886 & 20011 \end{pmatrix}$$

und der Hermite-Algorithmus liefert die kürzeste Basis

$$U = \begin{pmatrix} 83 & -85 \\ 87 & 152 \end{pmatrix}$$

Also $(f, g) = (83, 87)$. ■

Das Verfahren ist also so unbrauchbar und muss verbessert werden! Idee: Ersetzt man $\mathbb{Z}_p$ durch Polynomringe, so erhöht sich bei einem Angriff die Gitterdimension und das SVP ist (praktisch) nicht mehr lösbar.

Dazu betrachtet man die drei Ringe

$$\mathcal{R} = \mathbb{Z}[x]_{x^n-1}, \qquad \mathcal{R}_p = \mathbb{Z}_p[x]_{x^n-1}, \qquad \mathcal{R}_q = \mathbb{Z}_q[x]_{x^n-1}$$

mit $\mathrm{ggT}(q, p) = 1$. Es handelt sich also jedesmal um Polynome vom Grad (maximal) $n-1$ wobei modulo $m(x) = x^n - 1$ gerechnet wird.

Beispiel 9.25 Beschreiben Sie:
b) Die Multiplikation mit x bzw. x^j in $\mathcal{R}$. Was ist das multiplikative Inverse von x^j?
a) Die Multiplikation in $\mathcal{R}$.
c) Ist $m(x) = x^n - 1$ irreduzibel?

Lösung zu 9.25 a) Es gilt

$$\begin{aligned} x \cdot a(x) &= x(a_{n-1}x^{n-1} + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0) \\ &= a_{n-2}x^{n-1} + a_{n-3}x^{n-1} + \cdots + a_0 x + a_{n-1} \end{aligned}$$

und die Multiplikation entspricht einer zyklischen Rotation der Koeffizienten um eine Stelle nach links. Also entspricht die Multiplikation mit x^j einer zyklischen Rotation der Koeffizienten um j Stellen nach links. Die Koeffizienten von $x^j \cdot a(x)$ sind also

$$a_{k-j \bmod n}, \qquad 0 \le k \le n-1.$$

Das multiplikative Inverse von x^j ist x^{n-j}. b) Die Koeffizienten von $a(x)b(x) = c(x)$ sind

$$c_k = \sum_{j=0}^{n-1} a_j b_{k-j \bmod n}, \qquad 0 \le k \le n-1$$

und das ist als **Faltung** der Koeffizientenvektoren bekannt.
c) Nein, da $m(1) = 0$, also $m(x) = (x-1)(x^n + x^{n-1} + \cdots + x + 1)$. ■

Das letzte Beispiel erklärt die Wahl des Polynomes x^n-1, da sich in diesem Fall die Multiplikation in $\mathcal{R}$ besonders einfach und effektiv durchführen lässt. Ist ausserdem eines der beiden Polynome ternär, hat also nur Koeffezienten aus $\{-1,0,1\}$, dann sind für die Berechnung nur Addition und Subtraktion (also keine Multiplikation) notwendig. Dadurch ist die Berechnung am Computer extrem effizient und schneller als ECC (allerdings sind die Schlüssel größer). Der Ring $\mathcal{R}$ wird auch als zyklotomischer Ring bzw. Faltungsring bezeichnet.

Achtung: Manchmal wir x^n-1 in diesem Zusammenhang auch als zyklotomisches Polynom bezeichnet, aber dieser Begriff hat in der Mathematik eine andere Bedeutung.

Wie zuvor basiert die Sicherheit darauf, dass Operationen modulo q und modulo p inkompatibel sind (um das sicher zu stellen wird $\text{ggT}(q,p)=1$ gefordert) und wir müssen zwischen den Ringen $\mathcal{R}$, $\mathcal{R}_q$ und $\mathcal{R}_p$ wechseln. Da wir die Kleinheit der Polynome über den Betrag der Koeffizienten definieren, die Repräsentanten der Koeffizienten in $\mathbb{Z}_q$ nicht wie üblich durch $0 \le a_j < q$ sondern über $-\frac{q}{2} < a_j \le \frac{q}{2}$ wählen. Wenn wir also aus einem Polynom $a \in \mathcal{R}_q$ ein Polynom $a^\sharp \in R$ machen wollen, dann wählen wir die Repräsentanten so dass $-\frac{q}{2} < a_j^\sharp \le \frac{q}{2}$ gilt.

Beispiel 9.26 Für $q=11$ und $a(x)=5x^3+8x^2+9x+1$ gilt $a^\sharp(x)=5x^3-3x^2-2x+1$.

Ausserdem benötigen wir die folgende Menge der ternären Polynome

$$\mathcal{T} = \{a_{n-1}x^{n-1}+\cdots xa_1+a_0 \in \mathcal{R} | a_j \in \{0,\pm 1\}\}$$

und

$$\begin{aligned}\mathcal{T}(d_+,d_-) &= \{a(x)\in\mathcal{T} \,|\, |\{j|a_j=\pm 1\}| = d_\pm\},\\ \mathcal{T}(d) &= \{a(x)\in\mathcal{T} \,|\, |\{j\,||a_j|=1\}| = d\}.\end{aligned}$$

Ein Polynom in $\mathcal{T}(d_+,d_-)$ hat also d_+ Koeffizienten gleich $+1$, d_- Koeffizienten gleich -1, und alle anderen $n-d_--d+$ Koeffizienten gleich 0. Ein Polynom in $\mathcal{T}(d)$ hat d Koeffizienten gleich ± 1 und alle anderen $n-d$ Koeffizienten gleich 0.

Beispiel 9.27 $a(x)=x^3-x^2+1 \in \mathcal{T}(2,1) \subset \mathcal{T}(3)$.

Nun sind wir in der Lage die Verschlüsselung mit NTRU zu beschreiben:

- **Parameter**: (n,q,p,d) mit $\text{ggT}(q,p)=1$ und $q>(6d+1)p$.
- **Schlüsselerzeugung**: Alice wählt

$$f(x) \in \mathcal{T}(d+1,d), \qquad g(x) \in \mathcal{T}(d,d)$$

und berechnet

$$F_q(x) = f(x)^{-1} \text{ in } \mathcal{R}_q, \qquad F_p(x) = f(x)^{-1} \text{ in } \mathcal{R}_p$$

Falls eine der beiden Inversen nicht existiert, wählt sie eine neues f. Weiters berechnet sie

$$h(x) = F_q(x)g(x) \text{ in } \mathcal{R}_q.$$

Öffentlicher Schlüssel: $h(x)$
Privater Schlüssel: $(f(x), F_p(x))$

- **Verschlüsselung**: Bob verschlüsselt die Nachricht $m(x) \in \mathcal{R}_p$ mit $-\frac{p}{2} < m_j \leq \frac{p}{2}$ zusammen mit einer Nonce $r \in \mathcal{T}$ via

$$c(x) = p\,h(x)r(x) + m(x) \text{ in } \mathcal{R}_q.$$

- **Entschlüsselung**: Alice entschlüsselt mit

$$\tilde{c}(x) = f(x)c(x) \text{ in } \mathcal{R}_q$$

und

$$m(x) = F_p(x)\tilde{c}^\sharp(x) \text{ in } \mathcal{R}_p.$$

Zunächst gilt

$$\tilde{c}(x) = p\,g(x)r(x) + f(x)m(x) \text{ in } \mathcal{R}_q$$

Betrachten wir $p\,g(x)r(x) + f(x)m(x)$ in $\mathcal{R}$ und wählen wir die Koeffizienten aller Polynome so dass $-\frac{q}{2} < m_j \leq \frac{q}{2}$ gilt. Dann gilt (Aufgabe 10) für die Koeffizienten von $g(x)r(x)$, dass $|(gr)_j| \leq 2d$ und für die Koeffizienten von $f(x)m(x)$, dass $|(fm)_j| \leq (2d+1)\frac{p}{2}$ (da $|m_j| \leq \frac{p}{2}$ nach Voraussetzung an m). Insgesamt also $|\tilde{c}^\sharp_j| \leq (6d+1)\frac{p}{2} < \frac{q}{2}$ und damit gilt

$$\tilde{c}^\sharp(x) = p\,g(x)r(x) + f(x)m(x) \text{ in } \mathcal{R}.$$

Also gilt $F_p(x)\tilde{c}^\sharp(x) = m(x)$ in $\mathcal{R}_p$, wie gewünscht.

Beispiel 9.28 Wir wählen

$$n = 5, \quad q = 41, \quad p = 3, \quad d = 2,$$

so dass $q > (6d+1)p$ erfüllt ist. Ausserdem sei

$$f(x) = -x^4 + x^3 - x^2 + x + 1, \qquad g(x) = -x^3 + x^2 - x + 1.$$

a) Wie lautet der öffentliche bzw. private Schlüssel?
b) Wie verschlüsselt Bob die Nachricht $m = (0, 1, 0, 1)$?
c) Wie entschlüsselt Alice die Nachricht von Bob?

Lösung zu 9.28 a) Wir berechnen

$$F_q(x) = 21x^4 + 21, \qquad F_p(x) = 2x^4 + 2, \qquad h(x) = 21x^4 + 20x^3.$$

Der öffentliche Schlüssel ist $(h(x), q)$ und der private Schlüssel $(f(x), F_p(x))$. b) Die Nachricht als Polynom lautet $m(x) = x^3 + x$

$$r(x) = x^4 + x^2 - x - 1$$

und die verschlüsselte Nachricht ist

$$c(x) = 39x^3 + 23x + 22.$$

c) Zum Entschlüsseln berechnet Alice

$$\tilde{c}(x) = 40x^4 + 38x^3 + 3x^2 + 2x + 1$$

und $\tilde{c}^\sharp(x) = -x^4 - 3x^3 + 3x^2 + 2x + 1$. Die Nachricht lautet

$$F_p(x)\tilde{c}^\sharp(x) = x^3 + x.$$

■

Um die Sicherheit des Verfahrens zu gewährleisten müssen bei der Wahl der Parameter einige Bedingungen beachtet werden.

- Wie zuvor muss $\mathrm{ggT}(q, p) = 1$ sein, damit Eve durch Probieren der Teiler von p keinerlei Information über die Nachricht erhält.
- Kann man $m(x) = x^n - 1$ in kleinere Polynome faktorisieren, so kann man auch die Ringe entsprechend zerteilen und das erleichtert verschiedene Angriffe. Ist z.B. $n = 2m$ gerade, so gilt $x^n - 1 = (x^m - 1)(x^m + 1)$. Die Faktorisierung $x^n - 1 = (x-1)(x^{n-1} + \cdots x + 1)$ ist natürlich immer möglich, aber die Reduktion ist nur gering. Ist n eine Primzahl so ist $x^{n-1} + \cdots x + 1$ irreduzibel und daher wird n in der Regel als Primzahl gewählt.
- Aus Effektivitätsgründen sollte q nicht zu groß werden; üblicherweise wird eine Potenz von 2 gewählt. Auf der anderen Seite muss aber d groß sein, damit eine Brute-Force Suche in $\mathcal{T}(d+1, d)$ und $\mathcal{T}(d, d)$ unmöglich ist (Aufgabe 11). Dabei ist zu berücksichtigen, dass auch ein Meet-in-the-Middle Angriff möglich ist, der den Aufwand entsprechend reduziert. Deshalb wird p klein gewählt, typischerweise ist $p = 3$.
- Der Zweck von d ist es, eine genügend große Anzahl von Nullstellen in den ternären Polynomen sicherzustellen, damit die Faltungsprodukte nicht zu groß werden, damit die Entschlüsselung klappt. Dass man f in $\mathcal{T}(d+1, d)$ wählt, liegt daran, dass Polynome in $\mathcal{T}(d, d)$ kein multiplikatives Inverses

haben können (Aufgabe 9). In der Tat unterscheiden sich verschiedene NTRU Varianten in der Wahl der Mengen der ternären Polynome. Das Ziel ist es durch eine spezielle Struktur die Schranke and $q > (6d+1)p$ zu verbessern und damit die Effektivität des Verfahrens zu erhöhen.

Sind die Parameter entsprechend gewählt, so basiert die Sicherheit des Verfahrens darauf, dass der einzige bekannte Angriff über ein Gitterproblem der Dimension $2n$ läuft. Es muss also n genügend groß sein damit ein Angriff mittels Gitterreduktion nicht möglich ist. Der Aufwand es Angriffs muss dabei anhand der aktuelle besten verfügbaren Algorithmen zu Gitterreduktion geschätzt werden. In der NIST-Einreichung von NTRU[2] findet man folgende Angaben:

Variante	n	q	p	Geschätztes Sicherheitsniveau (Bit)
NTRU-HPS	509	2^{11}	3	128
NTRU-HPS	677	2^{11}	3	192
NTRU-HPS	821	2^{12}	3	256
NTRU-HRSS	701	2^{13}	3	256

Von Experten wurde allerdings kritisiert, dass das zugrundeligende Gitterproblem eine sehr einfache Struktur besitzt, was in der Zukunft für potentielle Angriffe genutzt werden kann. Deshalb wurde vorsorglich eine Variante NTRU-Prime [Ber+18] vorgeschlagen, bei der anstelle von $x^n - 1$ das Polynom $x^n - x - 1$ (mit n prim) verwendet wird. Für die konkrete Wahl $n = 761$ und $q = 4591$ ergeben sich nur geringfügig längere Berechnungszeiten.

Kommen wir nun zu möglichen Angriffen auf NTRU. Wir beginnen mit der Beobachtung, dass (f, q) nicht die einzigen Polynome sind, mit denen die Entschlüsselung klappt. Es reicht wenn $(\tilde{f}, \tilde{g})$ folgende Eigenschaften erfüllen:

- $\tilde{f} \in \mathcal{S}(2d+1)$, $\tilde{g} \in \mathcal{S}(2d)$ mit der Eigenschaft, dass $\tilde{f}^{-1}(x)$ in $\mathcal{R}_q$ existiert und $\tilde{f}^{-1}(x)\tilde{g}(x) = h(x)$ in $\mathcal{R}_q$.

Hier ist

$$\mathcal{S}(d) = \{a_{n-1}x^{n-1} + \cdots xa_1 + a_0 \in \mathcal{R} | \, |a_{n-1}| + \cdots + |a_0| = d\}$$

und wir haben $\mathcal{T}(d_+, d_-) \subset \mathcal{S}(d_+ + d_-)$.

Insbesondere funktionieren also auch die zyklischen Rotationen $(x^j f(x), x^j g(x))$.

Wird die selbe Nachricht mit verschiedenen r verschlüsselt, so bekommt Eve Informationen über die verwendeten r. Ebenso ist es für Eve hilfreich, wenn verschiedene Nachrichten mit dem gleichen r verschlüsselt werden. Eine Möglichkeit dem vorzubeugen ist es, r als Hash von m zu wählen.

Sehen wir uns nun den Angriff via Gitter an: Wie zuvor schreiben wir

$$\begin{pmatrix} f \\ g \end{pmatrix} = f \begin{pmatrix} 1 \\ h \end{pmatrix} + t \begin{pmatrix} 0 \\ q \end{pmatrix},$$

[2] https://ntru.org

allerdings haben wir hier noch kein Gitter, da die Komponenten der Vektoren Polynome sind und die Multiplikation in $\mathcal{R}$ auszuführen ist. Das können wir aber leicht beheben indem wir die Polynome mit den Vektoren ihrer Koeffizienten identifizieren:

$$f(x) \mathrel{\widehat{=}} \mathbf{f} = (f_0, \ldots, f_{n-1}), \quad g(x) \mathrel{\widehat{=}} \mathbf{g} = (g_0, \ldots, g_{n-1})$$

und für $h(x)$ verwenden wir

$$H = \begin{pmatrix} h_0 & h_1 & \ldots & h_{n-1} \\ h_{n-1} & h_0 & \ldots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \ldots & h_0 \end{pmatrix}^T$$

damit der Koeffeizientenvektor von $f(x)g(x)$ durch

$$f(x)h(x) = H\mathbf{f}$$

gegeben ist. Dann können wir obige Gleichung als Vektorgleichung

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbb{I}_n & 0_n \\ H & q\mathbb{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{t} \end{pmatrix}$$

schreiben. Das ist das gewünschte Gitterproblem.

Beispiel 9.29 Berechnen sie den geheimen Schlüssel aus dem letzten Beispiel mittels Gitterreduktion.

Lösung zu 9.29 Die Gitterbasis ist

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ h_0 & h_4 & h_3 & h_2 & h_1 & q & 0 & 0 & 0 & 0 \\ h_1 & h_0 & h_4 & h_3 & h_2 & 0 & q & 0 & 0 & 0 \\ h_2 & h_1 & h_0 & h_4 & h_3 & 0 & 0 & q & 0 & 0 \\ h_3 & h_2 & h_1 & h_1 & h_4 & 0 & 0 & 0 & q & 0 \\ h_4 & h_3 & h_2 & h_0 & h_0 & 0 & 0 & 0 & 0 & q \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 21 & 20 & 0 & 0 & 41 & 0 & 0 & 0 & 0 \\ 0 & 0 & 21 & 20 & 0 & 0 & 41 & 0 & 0 & 0 \\ 0 & 0 & 0 & 21 & 20 & 0 & 0 & 41 & 0 & 0 \\ 20 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 41 & 0 \\ 21 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 41 \end{pmatrix}.$$

Nach Gitterreduktion

$$U = \begin{pmatrix} 0 & 0 & 1 & -1 & -1 & 6 & 6 & -5 & -1 & -12 \\ 0 & 0 & -1 & 1 & -1 & -7 & 0 & -1 & -5 & 5 \\ -2 & 0 & -1 & -1 & -1 & -7 & -6 & -1 & 12 & 1 \\ 0 & -2 & -1 & -1 & -1 & 6 & 5 & -5 & -5 & 1 \\ 0 & 0 & -1 & -1 & -1 & 2 & -5 & 12 & -1 & 5 \\ 1 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 12 & 2 \\ -1 & 1 & 0 & 0 & 0 & 14 & 15 & 2 & -12 & 0 \\ 0 & -1 & 0 & 0 & 0 & 2 & 5 & 12 & -2 & -2 \\ 0 & 0 & -1 & 0 & 0 & -2 & 15 & -12 & 0 & -12 \\ 0 & 0 & 1 & -1 & 0 & -14 & 3 & -2 & 2 & 12 \end{pmatrix}.$$

Die ersten beiden Vektoren $\mathbf{u}_1$, $\mathbf{u}_2$ sind zwar nicht tärner, aber wir erhalten (z.B.) aus dem ersten Vektor

$$\tilde{f}(x) = -2x^2, \quad \tilde{g}(x) = 1 - x, \quad F_p(x) = x^3$$

und damit klappt die Entschlüsselung.

Die Vektoren $\mathbf{u}_3, \ldots, \mathbf{u}_5$ sind zwar ternär, allerdings sind die zughörigen Polynome $f(x)$ nicht teilerfremd zu $x^n - 1$ modulo p und damit unbrauchbar. ■

Es ist möglich mit der bei GGH beschriebenen Idee auch einen Signaturalgorithmus zu erhalten. Der Vorteil ist, dass das zugrundeliegende Gitter durch $h(x)$ gegeben ist und somit die Schlüssel signifikant kleiner sind (ein Vektor anstatt einer Matrix). Das Problem ist, dass man nun keine gute Basis kennt. Es ist allerdings möglich aus den n bekannten kurzen Vektoren $(x^j f(x), x^j g(x))$, $1 \leq j \leq n$ eine gute Basis zu erhalten. Für Details siehe [HPS08].

9.3.3 LWE

Wir betrachten ein lineares Gleichungssystem

$$A\mathbf{x} = \mathbf{b}$$

in $\mathbb{Z}$ mit A einer $n \times m$ Matrix. Wir interessieren uns für den Fall $n > m$ in dem das Gleichungssystem überbestimmt ist, also im Allgemeinen keine Lösung besitzt. Geben wir umgekehrt $\mathbf{x}$ vor und berechnen wir $\mathbf{b} = A\mathbf{x}$, dann kann man leicht $\mathbf{x}$ aus $\mathbf{b}$ bestimmen indem man das Gleichungssystem löst (Gauß-Algorithmus). Wenn man nun $\mathbf{b}$ mit einem Vektor $\mathbf{e}$ stört, $\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{e}$, dann ist es algorithmisch ein sehr schwieriges Problem den Vektor $\mathbf{x}$ zu finden, der das Gleichungssystem bis auf den Fehler $\mathbf{e}$ löst. Das Problem dabei ist, dass bei jedem Schritt des Gauß-Algorithmus der anfangs kleine Fehler vergrößert wird und man somit schnell die Kontrolle über den Fehler verliert. Alternativ können wir A auch als Basis eines Gitters auffassen, und dann suchen wir einen Gittervektor $\mathbf{b}$ (mit Entwicklungskoeffizienten $\mathbf{x}$) der möglichst nahe an $\tilde{\mathbf{b}}$ liegt. Das ist das **Learning with Errors**[3] (**LWE**) Problem.

Um daraus einen Verschlüsselungsalgorithmus zu erhalten, ist die prinzipielle Idee ähnlich wie bei Elgamal: Die Nachricht wird mit einer Zufallszahl maskiert, so dass die Maskierung nur mithilfe des geheimen Schlüssels entfernt werden kann. Dabei muss der Geheimtext für Eve wie ein Zufallstext aussehen, so dass durch statistische Analyse keinerlei Informationen über die Nachricht erhalten werden können. Um diese Eigenschaft sicherzustellen spielt die Wahrscheinlichkeitsverteilung zur Erzeugung von zufälligen Elementen eine zentrale Rolle und wird daher bei modernen Verfahren vorgeschrieben. Da die Sicherheit unseres LWE-Problems auf dem Hinzufügen von Fehlern basiert, ist man nicht in der Lage die Maskierung vollständig zu entfernen, sondern es bleibt ein gewisser Fehler. Dashalb muss der Fehler klein sein. Dann kann man die Nachricht in den höheren Stellen verpacken und beim Entschlüsseln die niedrigen (fehlerbehafteten) Stellen entfernen. Nun muss der Fehler auf der einen Seite genügend groß sein damit die Maskierung ihre Aufgabe erfüllen kann, und auf der anderen Seite klein, damit beim Runden möglichst wenige Stellen verworfen werden müssen (und somit viele für die Nachricht bleiben). Die Sicherheit geht hier also auf Kosten der Effizienz. Nun basiert das Verfahren auf linearer Algebra und der Fehler beim Entschlüsseln setzt sich aus einer Summe von Einzelfehlern zusammen, die sich statistisch wegheben. Der

[3] wird meist wörtlich als „Lernen mit Fehlern“ übersetzt; sinngemäß wäre „Lösen unter Fehlern“

typische Fehler wird also viel kleiner sein, als der Fehler im schlimmsten Fall, wo sich nichts weghebt. Man kann dadurch die Fehlerschranken erhöhen (bzw. überhaupt fallen lassen und die Fehlergröße über die Verteilung kontrollieren), wenn man mit gelegentliche Dekodierfehlern lebt (die mit klassischer Fehlerkorrektur effizient behoben werden können).

Konkret gehen wir nach Oded Regev [Reg05] wie folgt vor:

- **Parameter**: p und $m \leq n$ natürliche Zahlen. Alle Berechnungen erfolgen in $\mathbb{Z}_p$ und für $a \in \mathbb{Z}_p$ bezeichne $a^\sharp$ den Repräsentanten mit $-\frac{p}{2} < a^\sharp \leq \frac{p}{2}$.
- **Schlüsselerzeugung**: Wir wählen eine zufällige Matrix $A \in \mathbb{Z}_p^{n\times m}$, zufällige Vektoren $\mathbf{s} \in \mathbb{Z}_p^m$, $\mathbf{e} \in \mathbb{Z}_p^n$ mit $|e_1^\sharp| + \cdots + |e_n^\sharp| < \lfloor \frac{p}{4} - \frac{1}{2} \rfloor$ und berechnen

  $$\mathbf{b} = A\mathbf{s} + \mathbf{e} \in \mathbb{Z}_p^n.$$

  öffentlicher Schlüssel: $(A, \mathbf{b})$
  privater Schlüssel: $\mathbf{s}$
- **Verschlüsselung**: Die Nachricht sei ein einzelnes Bit $x \in \mathbb{Z}_2$. Bob wählt zufällig $\mathbf{r} \in \mathbb{Z}_2^n$ und berechnet

  $$\mathbf{u} = A^T\mathbf{r} \in \mathbb{Z}_p^m, \qquad v = \langle \mathbf{b}, \mathbf{r}\rangle + \lfloor p/2 \rfloor x \in \mathbb{Z}_p.$$

  Die verschlüsselte Nachricht ist

  $$y = (\mathbf{u}, v) \in \mathbb{Z}_p^{m+1}.$$

- **Entschlüsselung**: Alice entschlüsselt mit:

  $$x = \begin{cases} 0, & -\lfloor \frac{p}{4} \rfloor < (v - \langle \mathbf{s}, \mathbf{u}\rangle)^\sharp \leq \lfloor \frac{p}{4} \rfloor, \\ 1, & \text{sonst.} \end{cases}$$

Beim Entschlüsseln gilt

$$\begin{aligned} v - \langle \mathbf{s}, \mathbf{u}\rangle &= \langle \mathbf{b}, \mathbf{r}\rangle + \lfloor p/2 \rfloor x - \langle \mathbf{s}, A^T\mathbf{r}\rangle \\ &= \lfloor p/2 \rfloor x + \langle A\mathbf{s} + \mathbf{e}, \mathbf{r}\rangle - \langle A\mathbf{s}, \mathbf{r}\rangle = \lfloor p/2 \rfloor x + \langle \mathbf{e}, \mathbf{r}\rangle \end{aligned}$$

und x wird durch Runden korrekt bestimmt wenn $-\lfloor \frac{p}{4} \rfloor < \langle \mathbf{e}, \mathbf{r}\rangle^\sharp \leq \lfloor \frac{p}{4} \rfloor$ ist. Da $r_j \in \{0, 1\}$ und damit $|\langle \mathbf{e}, \mathbf{r}\rangle| = |e_1 r_1 + \cdots + e_n r_n| \leq |e_1| + \cdots + |e_n|$ wird das durch unsere Forderung an $\mathbf{e}$ sichergestellt.

Beispiel 9.30 Wir wählen

$$p = 29, \qquad n = 4, \quad m = 2.$$

Ausserdem sei

$$A = \begin{pmatrix} 20 & 28 \\ 4 & 2 \\ 2 & 22 \\ 7 & 16 \end{pmatrix}, \qquad \mathbf{s} = \begin{pmatrix} 25 \\ 2 \end{pmatrix}, \qquad \mathbf{e} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -3 \end{pmatrix}.$$

a) Wie lautet der öffentliche bzw. private Schlüssel?
b) Wie verschlüsselt Bob die Nachricht $x = 1$?
c) Wie entschlüsselt Alice die Nachricht von Bob?

Lösung zu 9.30 a) Es ist (wir rechnen in $\mathbb{Z}_{29}$!)

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} = \begin{pmatrix} 5 \\ 17 \\ 7 \\ 1 \end{pmatrix}$$

und somit lautet der öffentliche Schlüssel $(A, \mathbf{b})$ und der private ist $\mathbf{s}$.
b) Wir wählen (zufällig)

$$\mathbf{r} = (1, 1, 0, 0)$$

und berechnen

$$\mathbf{u} = A^T\mathbf{r} = (24, 1), \qquad v = \langle \mathbf{b}, \mathbf{r} \rangle + \lfloor p/2 \rfloor x = 7.$$

c) Wegen $\lfloor p/4 \rfloor = 7$ und

$$(v - \langle \mathbf{s}, \mathbf{u} \rangle)^\sharp = 14 \notin (-7, 7]$$

ist die Nachricht gleich 1. ■

Zur Sicherheit halten wir Folgendes fest:

- Damit $\mathbf{r}$ nicht aus $\mathbf{u}$ bestimmt werden kann, muss n wesentlich größer als m sein. Die Dimension des Kerns von A^T ist mindestens $n - m$ und Lösen des Gleichungssystems $A^T\mathbf{r} = \mathbf{u}$ liefert dann zu viele Möglichkeiten und damit keine verwertbaren Informationen über $\mathbf{r}$. Alternativ kann man auch zu $\mathbf{u}$ einen weiteren zufälligen Fehler hinzufügen

- Ausserdem ist, wenn n groß genug ist, $\langle \mathbf{b}, \mathbf{r} \rangle$ für typisches $\mathbf{b}$ gleichmäßig über $0, \ldots, p-1$ verteilt und v verrät keinerlei Informationen über x.

- Man kann die Wahl von $\mathbf{r}$ auch als eine zufällige Auswahl von Zeilen aus dem öffentlichen Schlüssle $(A, \mathbf{b})$ auffassen. Insbesondere wird das Verfahren auch manchmal so formuliert, dass eine vorgegebene Anzahl von Zeilen aus $(A, \mathbf{b})$ zufällig auszuwählen ist. In unsere Formulierung wäre dann $\mathbf{r} \in \mathbb{Z}_2^n$ mit einer

vorgegebene Anzahl von Einsen zu wählen. Dadurch kann man sicherstellen, dass $\mathbf{r}$ nicht klein sein kann (denn dann würde die Entschlüsselung auch mit einer schlechten Lösung $\tilde{\mathbf{s}}$ des LWE Problems klappen).

- Damit $\mathbf{s}$ nicht über Brute-Force bestimmt werden kann, muss m groß sein. Ebenso muss p groß genug sein, damit $\mathbf{e}$ nicht über Brute-Force bestimmt werden kann.

  Dann bleibt nur noch der Angriff über das Gitter. Da A und $\mathbf{e}$ zufällig gewählt werden und keinen Einschränkungen (abgesehen von der Größe von $\mathbf{e}$) unterliegen, ist sicher gestellt, dass das CVP keine spezielle Form hat die eine Lösung erleichtern könnte.

  Unsere obige Kleinheitsbedingung an $\mathbf{e}$ stellt sicher, dass die Entschlüsselung in jedem Fall korrekt ist. Wenn man bereit ist, gelegentliche Fehler bei der Entschlüsselung in Kauf zu nehmen, kann man die Schranke an $\mathbf{e}$ erhöhen und dadurch die Sicherheit erhöhen. In der Praxis arbeitet man nicht mit einer Schranke sondern wählt die Koeffizienten von $\mathbf{e}$ anhand einer vorgegebenen Wahrscheinlichkeitsverteilung (z.B. eine diskreten Normalverteilung) mit Mittelwert 0 und kleiner Standardabweichung σ (über die die Fehlerwahrscheinlichkeit vorgegeben wird). Dann ist $\langle \mathbf{r}, \mathbf{e} \rangle$ eine Summe von unabhängigen und identisch verteilten Zufallsvariablen mit Standardabweichung $\frac{\sqrt{n}}{2}\sigma$ (der Vektor $\mathbf{r}$ entspricht einer zufälligen Auswahl von ca. 50% der Koeffizienten von $\mathbf{e}$). Die Wahrscheinlichkeit für einen Fehler kann damit leicht über die Verteilungsfunktion abgeschätzt werden.

- A könnte im Prinzip ein öffentlicher Parameter sein (wodurch der öffentliche Schlüssel kleiner wird), aber dann stellt sich wieder die Frage nach absichtlich eingebauten Schwachstellen.

Für eine detaillierte Sicherheitsanalyse (inklusive mathematische Beweise) müssen wir auf die umfangreiche Literatur verweisen. Ein Startpunkt ist der Übersichtsartikel [Reg10]. Wir sehen uns aber noch an, wie der private Schlüssel aus dem öffentlichen bestimmt werden kann.

Beispiel 9.31 Bestimmen Sie den privaten Schlüssel aus dem letzten Beispiel via Gitterreduktion aus dem öffentlichen Schlüssel.

Lösung zu 9.31 Wir suchen den zu $\mathbf{b}$ nähesten Vektor $\mathbf{b}_c$ im von A aufgespannten Gitter. Allerdings nicht in $\mathbb{Z}$, sondern in $\mathbb{Z}_p$. Deshalb müssen wir einen kleinen Trick anwenden und p mal der Einheitsmatrix zur Basis hinzunehmen, damit alle Komponenten modulo p reduziert werden, wenn das die Länge reduziert. Unsere Gitterbasis lautet also

$$\begin{pmatrix} 20 & 28 & 29 & 0 & 0 & 0 \\ 4 & 2 & 0 & 29 & 0 & 0 \\ 2 & 22 & 0 & 0 & 29 & 0 \\ 7 & 16 & 0 & 0 & 0 & 29 \end{pmatrix}.$$

Hier sind wir allerdings schon außerhalb unserer bisherigen Theorie, da die Basisvektoren nicht linear unabhängig sind. Wir können aber linear abhängige Vektoren entfernen indem wir die Matrix auf **Hermit'sche Normalform** (**HNF**) bringen (siehe z.B. [Coh93]; gängige Implementationen von LLL machen das automatisch) und der LLL-Algorithmus liefert uns

$$\begin{pmatrix} 0 & 4 & 3 & -5 \\ -2 & -4 & -1 & -4 \\ 1 & -3 & 4 & 1 \\ -3 & 0 & 3 & 1 \end{pmatrix}.$$

Wenn wir nun das CVP mit dem Babai-Näheste-Ebene-Algorithmus lösen erhalten wir

$$\tilde{\mathbf{b}}_c = (5, 15, 8, 1)$$

und lösen des Gleichungssystems $A\tilde{\mathbf{s}} = \tilde{\mathbf{b}}_c$ in $\mathbb{Z}_{29}$ liefert $\tilde{\mathbf{s}} = (21, 9)$. Das ist zwar ungleich dem ursprünglichen Schlüssel, aber da der zugehörige Fehlervektor $\tilde{\mathbf{e}} = \mathbf{b} - \tilde{\mathbf{b}}_c = (0, 2, -1, 0)$ sogar kürzer ist (und insbesondere die Bedingung zum Entschlüsseln erfüllt), kann damit korrekt entschlüsselt werden. ■

Wir haben also ein praktikables Verfahren das eine hohe Sicherheit bietet, die auch mathematisch gut verstanden ist (siehe wiederum [Reg10]). Allerdings ist der Aufwand um ein einzelnes Bit zu übertragen zu hoch. Man kann die Effektivität aber leicht steigern indem man mehrere Bits mit dem gleichen A und dem gleichen $\mathbf{r}$ verschlüsselt. Um die Sicherheit weiter zu gewährleisten, benötigen wir für jedes Bit verschiedene Vektoren $\mathbf{s}$ und $\mathbf{e}$.

Wollen wir l Bits verschlüsseln, so modifizieren wir das Verfahren wie folgt:

- **Parameter**: p und $m < n$, l natürliche Zahlen. Alle Berechnungen erfolgen in $\mathbb{Z}_p$ und für $a \in \mathbb{Z}_p$ bezeichne $a^\sharp$ den Repräsentanten mit $-\frac{p}{2} < a^\sharp \leq \frac{p}{2}$.
- **Schlüsselerzeugung**: Wir wählen eine zufällige Matrix $A \in \mathbb{Z}_p^{n\times m}$, zufällige Matrizen $S \in \mathbb{Z}_2^{m\times l}$, $E \in \mathbb{Z}_p^{n\times l}$ mit $|E_{1j}^\sharp| + \cdots + |E_{nj}^\sharp| < \frac{1}{2}\lfloor\frac{p}{4} - \frac{1}{2}\rfloor$ für $1 \leq j \leq l$ und berechnen
$$B = AS + E.$$
öffentlicher Schlüssel: (A, B)
privater Schlüssel: S.
- **Verschlüsselung**: Die Nachricht sei $x \in \mathbb{Z}_2^l$. Bob wählt $\mathbf{r} \in \mathbb{Z}_2^n$, $\mathbf{e}^1 \in \mathbb{Z}_2^m$, $\mathbf{e}^2 \in \mathbb{Z}_p^l$ mit $|e_1^1| + \ldots e_m^1 + e_1^2 + \cdots + e_l^2| < \frac{1}{2}\lfloor\frac{p}{4} - \frac{1}{2}\rfloor$ und berechnet
$$\mathbf{u} = A^T\mathbf{r} + \mathbf{e}^1, \qquad \mathbf{v} = B^T\mathbf{r} + \mathbf{e}^2 + \lfloor p/2\rfloor\mathbf{x}.$$
Die verschlüsselte Nachricht ist
$$y = (\mathbf{u}, \mathbf{v}).$$

- **Entschlüsselung**: Alice entschlüsselt mit:

$$x_j = \begin{cases} 0, & -\lfloor \frac{p}{4} \rfloor < (\mathbf{v} - S^T\mathbf{u})^\sharp_j \leq \lfloor \frac{p}{4} \rfloor, \\ 1, & \text{sonst.} \end{cases}$$

Wie zuvor erhalten wir

$$\begin{aligned} \mathbf{v} - S^T\mathbf{u} &= B^T\mathbf{r} + \mathbf{e}^2 + \lfloor p/2 \rfloor \mathbf{x} - S^T(A^T\mathbf{r} + \mathbf{e}^1) \\ &= \lfloor p/2 \rfloor \mathbf{x} + \mathbf{e}^2 + (AS + E)^T\mathbf{r} - (AS)^T\mathbf{r} - S^T\mathbf{e}^1 = \lfloor p/2 \rfloor \mathbf{x} + E^T\mathbf{r} + \mathbf{e}^2 - S^T\mathbf{e}^1. \end{aligned}$$

und gilt für alle Spaltenvektoren $\mathbf{e}_j$ von E, dass $-\lfloor \frac{p}{4} \rfloor < \langle \mathbf{e}_j, \mathbf{r} \rangle \leq \lfloor \frac{p}{4} \rfloor$, so wird korrekt entschlüsselt.

Mit einigen weiteren Modifikationen kann die Effizienz noch weiter erhöht werden. Bei **Frodo** werden dazu für $\mathbf{r}$, $\mathbf{u}$ und $\mathbf{v}$ (und die zugehörigen Fehler) ebenfalls Matrizen verwendet und es wird auch mehr als ein Bit in die Koeffizienten verpackt. Details dazu sind in [Alk+21] zu finden. Frodo wird aktuell vom BSI (zusammen mit **Classic McEliece**) empfohlen [BSI].

Eine weitere Variante, die mehr Möglichkeiten bei der Auswahl konkreter Parameter ermöglicht, ist das **Ring-LWE** Problem bei dem $\mathbb{Z}_p$ durch einen Polynomring $\mathcal{R}$ ersetzt wird (wie wir das schon bei NTRU gesehen haben). Arbeitet man zusätzlich mit Vektoren, so hat man nun keinen Vektorraum mehr, da der Skalarkörper ja durch einen Ring ersetzt wurde. Man spricht in diesem Fall von einem **Modul** und man nennt diese Variante dementsprechend **Module-LWE**.

Wir wollen uns das konkret am Beispiel von **Kyber** ansehen, der 2022 vom NIST standardisiert wurde. Dabei wird als Ring konkret $\mathbb{Z}_p[x]_{x^n+1}$ verwendet und wir sehen uns zuerst wieder die Multiplikation in diesem Ring an.

Beispiel 9.32 Beschreiben Sie im Ring $\mathcal{R} = \mathbb{Z}[x]_{x^n+1}$:
b) Die Multiplikation in $\mathcal{R}$ mit x bzw. x^j. Was ist das multiplikative Inverse von x^j?
a) Die Multiplikation in $\mathcal{R}$.

Lösung zu 9.32 a) Es gilt

$$\begin{aligned} x \cdot a(x) &= x(a_{n-1}x^{n-1} + a_{n-1}x^{n-1} + \cdots + a_1x + a_0) \\ &= a_{n-2}x^{n-1} + a_{n-3}x^{n-1} + \cdots + a_0x - a_{n-1} \end{aligned}$$

und die Multiplikation entspricht einer zyklischen Rotation der Koeffizienten um eine Stelle nach links und einem Vorzeichenwechsel beim konstanten Koeffizienten. Also entspricht die Multiplikation mit x^j einer zyklischen Rotation der Koeffizienten um j Stellen nach links und j Vorzeichenwechsel bei den ersten j Koeffizienten. Die Koeffizienten von $x^j \cdot a(x)$ sind also

$$\begin{cases} -a_{k+n-j}, & 0 \leq k \leq j-1, \\ a_{k-j}, & j \leq k \leq n-1. \end{cases}$$

Das multiplikative Inverse von x^j ist $-x^{n-j}$.
b) Die Koeffizienten von $a(x)b(x) = c(x)$ sind

$$c_k = \sum_{j=0}^{k-1} a_j b_{k-j} - \sum_{j=k}^{n-1} a_j b_{k+n-j}, \qquad 0 \le k \le n-1.$$

■

Bis auf das Vorzeichen erhalten wir also wieder die Faltung. Insbesondere bleiben die Abschätzungen aus Aufgabe 10 gültig. Wir schreiben abkürzend $\|a(x)\|_1 = |a^\sharp_{n-1}| + \cdots + |a^\sharp_0|$ und $\|a(x)\|_\infty = \max_{0\le j\le n-1} |a^\sharp_j|$.

Damit sieht Kyber wie folgt aus. Zum Unterschied zu vorher werden auch noch zusätzliche kleine Fehlerterme bei $\mathbf{u}$ und $\mathbf{v}$ ergänzt.

- **Parameter**: p prim und n, k natürliche Zahlen. Alle Berechnungen erfolgen in $\mathcal{R}_p = \mathbb{Z}_p[x]_{x^n+1}$.
- **Schlüsselerzeugung**: Wir wählen eine zufällige Matrix $A \in \mathcal{R}_p^{k\times k}$, zufällige Vektoren $\mathbf{s} \in \mathcal{R}_p^k$ mit $\|s_j\|_\infty \le 1$, $\mathbf{e} \in \mathcal{R}_p^k$ mit $\|e_1\|_1 + \cdots + \|e_k\|_1 < \frac{1}{2}\lfloor \frac{p}{4} - \frac{1}{2}\rfloor$ und berechnen
$$\mathbf{b}(x) = A(x)\mathbf{s}(x) + \mathbf{e}(x)$$
öffentlicher Schlüssel: $(A(x), \mathbf{b}(x))$
privater Schlüssel: $\mathbf{s}(x)$.
- **Verschlüsselung**: Die Nachricht sei $m \in \mathbb{Z}_2^n$ bzw. als Polynom
$$m(x) = \big(m_{n-1}x^{n-1} + \cdots + m_1 x + m_0\big).$$
Bob wählt $\mathbf{r}(x) \in \mathcal{R}_2^k$ und die Fehler $\mathbf{e}^1(x) \in \mathcal{R}_p^k$, $e^2(x) \in \mathcal{R}_p$ mit $\|e_1^1\|_1 + \cdots + \|e_k^1\|_1 + \|e^2\|_1 < \frac{1}{2}\lfloor \frac{p}{4} - \frac{1}{2}\rfloor$. Nun berechnet er
$$\mathbf{u}(x) = A(x)^T\mathbf{r}(x) + \mathbf{e}^1(x), \qquad v(x) = \mathbf{b}(x)^T\mathbf{r}(x) + e^2(x) + \lfloor p/2\rfloor m(x).$$
Die verschlüsselte Nachricht ist
$$c(x) = (\mathbf{u}(x), v(x)).$$
- **Entschlüsselung**: Alice entschlüsselt mit:
$$x_j = \begin{cases} 0, & -\lfloor \frac{p}{4}\rfloor < [v(x) - \mathbf{s}(x)^T\mathbf{u}(x)]^\sharp_j \le \lfloor \frac{p}{4}\rfloor, \\ 1, & \text{else}, \end{cases}$$
wobei $[p(x)]_j$ den Koeffizient von x^j des Polynoms $p(x)$ bezeichnet.

Es gilt
$$v(x) - \mathbf{s}(x)^T\mathbf{u}(x) = \mathbf{e}(x)^T\mathbf{r}(x) + e^2(x) - \mathbf{s}(x)^T\mathbf{e}^1(x) + \lfloor p/2\rfloor m(x).$$
Korrekt falls die Koeffizienten von $\mathbf{e}(x)^T\mathbf{r}(x) + e^2(x) - \mathbf{s}(x)^T\mathbf{e}^1(x)$ alle im Intervall $(-\lfloor \frac{p}{4}\rfloor, \lfloor \frac{p}{4}\rfloor]$ sind.

Beispiel 9.33 Wir wählen

$$p = 97, \qquad n = 4, \quad k = 2.$$

Ausserdem sei

$$\begin{aligned} A(x) &= \begin{pmatrix} 79x^3 + 61x^2 + 88x + 66 & 67x^3 + 57x^2 + 8x + 34 \\ 21x^3 + 50x^2 + 85x + 66 & 66x^3 + 95x^2 + 74x + 90 \end{pmatrix} \in \mathbb{Z}_{97}[x]^{2\times 2}_{x^4+1}, \\ \mathbf{s}(x) &= \big(-x^3 + x + 1, -x^3 - x^2 + x - 1\big) \in \mathbb{Z}_p[x]^2_{x^4+1} \end{aligned}$$

und

$$\mathbf{e}(x) = \big(-x^3 + 2x^2 - x - 1, -3x^3 - 2x\big) \in \mathbb{Z}_{97}[x]^2_{x^4+1}.$$

a) Wie lautet der öffentliche bzw. private Schlüssel?
b) Wie verschlüsselt Bob die Nachricht $m = (0, 0, 1, 1)$?
c) Wie entschlüsselt Alice die Nachricht von Bob?

Lösung zu 9.33 a) Es ist (wir rechnen in $\mathbb{Z}_{97}[x]_{x^4-1}$!)

$$\mathbf{b}(x) = A(x)\mathbf{s}(x) + \mathbf{e}(x) = (21x^3 + 20x^2 + 73x + 38, 61x^3 + 14x^2 + 85x + 46)$$

und somit lautet der öffentliche Schlüssel $(A(x), \mathbf{b}(x))$ und der private ist $\mathbf{s}(x)$.
b) Zum Verschlüsseln wandeln wir zunächst die Nachricht $(m_3, m_2, m_1, m_0) \in \mathbb{Z}_2^n$ in ein Polynom um

$$m(x) = x^3 + x^2$$

und wählen

$$\mathbf{r}(x) = \big(x + 1, x^3 + x^2\big) \in \mathbb{Z}_2[x]^2_{x^n+1}$$

und die Fehler

$$\mathbf{e}^1(x) = \big(-x^3 - x^2 - x, 3x\big) \in \mathbb{Z}_{97}[x]^2_{x^n+1}, \quad e^2(x) = -x^3 - x^2 - x - 1 \in \mathbb{Z}_{97}[x]_{x^n+1}.$$

Nun berechnen wir

$$\mathbf{u}(x) = A(x)^T\mathbf{r}(x) + \mathbf{e}^1(x) = \big(x^3 + x^2 + 84x + 46, 94x^3 + 89x^2 + 72x + 89\big)$$

und

$$v(x) = \mathbf{b}(x)^T\mathbf{r}(x) + e^2(x) + \lfloor p/2 \rfloor m(x) = 25x^3 + 28x^2 + 35x + 14.$$

c) Wegen $\lfloor p/4 \rfloor = 24$ und

$$(v(x) - \mathbf{s}(x)^T\mathbf{u}(x))^\sharp = (41x^3 + 51x^2 + 92x + 4)^\sharp = 41x^3 - 46x^2 - 5x + 4$$

ist die Nachricht gleich $(0, 0, 1, 1)$. ■

Unsere Darstellung ist wieder vereinfacht.

- Wie bereits bei LWE erwähnt, nimmt man eine kleine Fehlerwahrscheinlichkeit beim Entschlüssel in Kauf und kann dafür die Schranken an die Fehler durch die Wahl einer zentrierten Wahrscheinlichkeitsverteilungen ersetzten. Bei Kyber wird eine zentrierte Binomialverteilung B_η verwendet.

- Ausserdem gibt es auch noch eine Kompressionsfunktion mit der die Koeffizenten der verschlüsselten Nachricht leicht gerundet werden. Dadurch wird ein weiterer kleiner Fehler hinzugefügt, der allerdings keine signifikaten Auswirkung hat. Dabei geht es in erster Linie darum die Effizienz leicht zu verbessern.

- Bei der Schlüsselerzeugung werden die Koeffizienten nicht zufällig gewählt, sondern werden ausgehend von einem Zufallswert d deterministisch mithilfe von Hashfunktionen erzeugt. Dazu werden von d mittels Hashfunktion (SHA3-512) zwei Werte erzeugt: $(\rho, \sigma) = H(d)$.

  Aus ρ wird A erzeugt ($A_{j,k} = \mathrm{XOF}(\rho, j, k)$ mit XOF basierend auf SHAKE-128) und ρ wird als Teil des öffentlichen Schlüssels anstelle von A abgespeichert. Dadurch wird der öffentliche Schlüssel kleiner, A muss vor dem Verschlüssel allerdings jedesmal neu berechnet werden.

  Aus σ werden $\mathbf{b}$ und $\mathbf{e}$ unter Beachtung der geforderten Wahrscheinlichkeitsverteilung generiert. Dazu werden mit einem Pseudozufallsgenerator (SHAKE-256) 2η Bits $(a_1, \ldots, a_\eta, b_1, \ldots, b_\eta)$ erzeugt, dann hat $a_1 + \ldots + a_\eta - b_1 - \ldots - b_\eta$ die gewünschte zentrierte Binomialverteilung $\mathcal{B}_\eta$. Für η wird je nach Variante 2 bzw. 3 gewählt.

- Bei der Verschlüsselung wird ebenfalls ein Zufallswert erzeugt und dann die Werte von $\mathbf{r}$, $\mathbf{e}^1$ und e^2 mittels Pseudozufallsgenerator (SHAKE-256) wie schon beschrieben generiert.

Für weitere Details siehe [Ava+21].

Folgende Parameter werden bei Kyber verwendet:

- $n = 256$, $p = 3329$ für den Polynomring $\mathbb{Z}_p[x]_{x^n+1}$ über dem wir rechnen.

- $k = 2, 3, 4$ für die Modul-Dimension, also die Anzahl der Polynome pro Vektor, je nach gewünschtem Sicherheitsniveau.

Die Wahl der Primzahl $p = 2^8 \cdot 13 + 1$ hat den Grund, dass $x^{256} + 1$ über $\mathbb{Z}_p$ in kleiner Faktoren zerfällt,

$$x^{256} + 1 = \prod_{j=0}^{127} (x^2 - 17^{2j+1}),$$

wodurch die Multiplikation in $\mathbb{Z}_p[x]_{x^{256}+1}$ besonders effizient implementiert werden kann. Dazu wird ein Polynom in die Reste modulo der Faktoren zerlegt (Chinesischer Restsatz im Polynomring) und das ist als **NTT** (*number-theoretic transform*) bekannt.

Zum Schluß kommen wir noch dazu wie mit LWE eine Signatur erzeugt werden kann. Der zu Kyber gehörige Signaturalgorithmus ist **Dilithium** und bevor wir zur groben Struktur kommen vorweg ein paar Ideen. Wie zuvor wird mit dem privaten Schlüssel eine Maskierung erzeugt, die mit dem öffentlichen bis auf Fehler entfernt werden kann. Der Fehler ist klein und wirkt sich nur auf die *niedrigen* Stellen aus; die eigentlichen Informationen stecken in den *höheren* Stellen. Bei Verfahren mit dieser Struktur können in gewissen Fällen Informationen über den Schlüssel verraten werden. Deshalb müssen diese Fälle verworfen werden. Ausserdem sind Dekodierfehler nun auch keine Option und diese Fälle müssen ebenfalls verworfen werden.

Die Zerlegung in niedrige und hohe Stellen bezüglich eines festen Parameters $\gamma > 0$ erfolgt gemäß

$$u = u_l + 2\gamma u_h, \qquad -\gamma < u_l \leq \gamma.$$

Wiederum wählen wir also bei der Division mit 2γ den Rest u_l symmetrisch bezüglich 0.

- **Parameter**: p prim und n, k, l, η, γ_1, γ_2, τ natürliche Zahlen. $\beta = \tau\eta$. Alle Berechnungen erfolgen in $\mathcal{R}_p = \mathbb{Z}_p[x]_{x^n+1}$. Eine Hashfunktion $H : \mathbb{Z}_2^* \to \mathcal{T}(\tau)$.

- **Schlüsselerzeugung**: Wir wählen eine zufällige Matrix $A(x) \in \mathcal{R}_p^{k\times l}$, zufällige Vektoren $\mathbf{s}(x) \in \mathcal{R}_p^l$, $\mathbf{e}(x) \in \mathcal{R}_p^k$ mit $\|\mathbf{s}(x)\|_\infty \leq \eta$, $\|\mathbf{e}\|_\infty \leq \eta$ und berechnen
$$\mathbf{b}(x) = A(x)\mathbf{s}(x) + \mathbf{e}(x) \in \mathcal{R}_p^k$$
öffentlicher Schlüssel: $(A(x), \mathbf{b}(x))$
privater Schlüssel: $\mathbf{s}(x)$.

- **Signatur**: Nachricht m. Wähle $\mathbf{r}(x) \in \mathcal{R}_p^l$ mit $\|\mathbf{r}(x)\|_\infty \leq \gamma_1$, berechne $\mathbf{u}(x) = A\mathbf{r}(x) = \mathbf{u}_l(x) + 2\gamma_2\mathbf{u}_h(x)$ (Zerlegung bezüglich γ_2) und
$$c(x) = H(m\|\mathbf{u}_h), \qquad \mathbf{v}(x) = \mathbf{r}(x) + c(x)\mathbf{s}(x)$$
Verwerfen: Falls $\|\mathbf{v}(x)\|_\infty \geq \gamma_1 - \beta$ oder $\|(\mathbf{u}(x) - c(x)\mathbf{e}(x))_l\|_\infty \geq \gamma_2 - \beta$ mit $\beta = \|c\mathbf{e}\|_\infty \leq \eta\tau$, wähle ein neues $\mathbf{r}(x)$. Ansonsten:
Signatur: $(\mathbf{v}(x), c(x))$

- **Verifikation**: Berechne $\mathbf{w}(x) = A\mathbf{v}(x) - c(x)\mathbf{b}(x)$ und überprüfe $\|\mathbf{v}(x)\|_\infty < \gamma_1 - \beta$ und $c(x) = H(m\|\mathbf{w}_h)$.

Bei der Verifikation gilt $\mathbf{w}(x) = \mathbf{u}(x) - c(x)\mathbf{e}(x)$ und da während der Signaturerzeugung $\|(\mathbf{u}(x) - c(x)\mathbf{e}(x))_l\|_\infty < \gamma_2 - \beta$ sicher gestellt wurde und $\|c(x)\mathbf{e}(x)\|_\infty \leq \tau\eta = \beta$ gilt, gibt es bei $\mathbf{u}(x) = (\mathbf{u}(x) - c(x)\mathbf{e}(x)) + c\mathbf{e}(x)$ keinen Überlauf der niedrigen Stellen in die höheren Stellen und es gilt $\mathbf{w}_h(x) = \mathbf{u}_h(x)$.

Beispiel 9.34 Wir wählen

$$p = 97, \qquad n = 4, \quad k = l = 2, \quad \gamma_1 = 32,\ \gamma_2 = 8, \eta = 1,\ \tau = 2.$$

Ausserdem sei

$$\begin{aligned} A(x) &= \begin{pmatrix} 79x^3 + 61x^2 + 88x + 66 & 67x^3 + 57x^2 + 8x + 34 \\ 21x^3 + 50x^2 + 85x + 66 & 66x^3 + 95x^2 + 74x + 90 \end{pmatrix} \in \mathbb{Z}_{97}[x]^{2\times 2}_{x^4+1}, \\ \mathbf{s}(x) &= \big(-x^3 + x + 1, -x^3 - x^2 + x - 1\big) \in \mathbb{Z}_p[x]^2_{x^4+1} \end{aligned}$$

und

$$\mathbf{e}(x) = \big(x^3 - x^2 + x - 1, -x^3 + x^2 - 1\big) \in \mathbb{Z}_{97}[x]^2_{x^4+1}.$$

Die Hashfunktion $H : \mathbb{Z}^* \to \mathcal{T}(\tau)$ sei wie folgt gegeben: Die Nachricht $m = (m_0, m_1, \ldots)$ sei als eine endliche Folge von Zahlen gegeben (Buchstaben sind durch ihre ASCII-Codes zu ersetzten) von der zuerst $\tilde{m} = m_0 + m_1 + \ldots \bmod 2^{3\tau}$ zu bilden ist. Nun zerlegen wir die Dualdarstellung von $\tilde{m}$ in τ Blöcke zu je 3 Bit. Wir starten bei $i = 0$ und erhöhen i um die Zahl die den ersten beiden Bits entspricht und setzten $c_i = 1$ falls das dritte Bit gleich 0 und $c_i = -1$ falls das dritte Bit gleich 1. Das wiederholen wir für alle τ Blöcke. Wird $i \geq n$, so reduzieren wir i modulo n und ist $c_i \neq 0$, so erhöhen wir i so lange bis $c_i = 0$.
a) Wie lautet der öffentliche bzw. private Schlüssel?
b) Wie signiert Alice die Nachricht $m = \texttt{FAD}$?
c) Wie prüft Bob die Signatur?

Lösung zu 9.34 a) Es ist (wir rechnen in $\mathbb{Z}_{97}[x]_{x^4-1}$!)

$$\mathbf{b}(x) = A(x)\mathbf{s}(x) + \mathbf{e}(x) = (23x^3 + 17x^2 + 75x + 38, 63x^3 + 15x^2 + 87x + 45)$$

und somit lautet der öffentliche Schlüssel $(A(x), \mathbf{b}(x))$ und der private ist $\mathbf{s}(x)$.
b) Alice wählt $\mathbf{r}(x)$ mit zufälligen Koeffizienten zwischen $-\gamma_1$ und γ_1:

$$\mathbf{r}(x) = (21x^3 - 8x^2 - 27x + 1, 22x^3 + 2x^2 - 8x + 25)$$

und berechnet

$$\mathbf{u}(x) = (30x^3 + 12x^2 + 77x + 92, 67x^3 + 13x^2 + 15x + 14).$$

Nun bestimmt sie die ASCII Werte der Nachricht 70, 65, 68 und die hohen Stellen der Koeffizienten von $\mathbf{u}(x)$: 6, 5, 1, 2, 1, 1, 1, 4. Die Ziffernsumme dieser Zahlen ist 224 und modulo $2^{3\tau} = 64$ haben wir $32 = (100000)_2$. Daraus ergibt sich der Wert unsere Hashfunktion zu

$$c(x) = H(70, 65, 68, 6, 5, 1, 2, 1, 1, 1, 4) = H(32) = 1 - x.$$

Damit berechnen wir

$$\mathbf{v}(x) = (20x^3+88x^2+70x+1, 22x^3+91x+23) = (20x^3-9x^2-27x+1, 22x^3+91x-6)$$

und es gilt $\|\mathbf{v}(x)\|_\infty = 27 < 30 = \gamma_1 - \beta$. Ansonsten hätte Alice ein neues $\mathbf{r}$ wählen müssen. Um auch die zweite Bedingung zu überprüfen benötigen wir

$$\mathbf{u}(x) - c(x)\mathbf{s}(x) = (28x^3 + 14x^2 + 75x + 92, 69x^3 + 12x^2 + 14x + 16).$$

Die niedrigen Stellen sind $-4, -5, -2, -4, 0, -2, -4, 5$ und alle sind kleine $6 = \gamma_2 - \beta$. Also ist die Signatur $(\mathbf{v}(x), c(x))$ gültig.
c) Bob prüft zuerst ob $\|\mathbf{v}(x)\|_\infty < \gamma_1 - \beta$ und berechnet dann

$$\mathbf{w}(x) = 28x^3 + 14x^2 + 75x + 92, 69x^3 + 12x^2 + 14x + 16.$$

Die hohen Stellen von $\mathbf{w}(x)$ sind 6, 5, 1, 2, 1, 1, 1, 4 und damit erhält er

$$c(x) = H(70, 65, 68, 6, 5, 1, 2, 1, 1, 1, 4)$$

und die Signatur ist gültig. ■

Einige Details zur Wahl der Parameter und zur Sicherheit:

- Bei Dilithium ist $p = 8380417 = 2^{23} - 2^{13} + 1$ und $n = 256$.
- Moduldimensionen (k, l): $(4, 4)$, $(6, 5)$, $(8, 7)$
- Die Schlüsselerzeugung erfolgt, abgesehen von den leicht unterschiedlichen Parametern, wie bei Kyber. Allerdings werden $\mathbf{s}$ und $\mathbf{e}$ nun gleichverteilt mit Schranke η erzeugt, da bei der Signatur keine Dekodierfehler auftreten dürfen. Zusammen mit der Schranke τ and c wird das durch die zweite Bedingung beim Verwerfen sicher gestellt.
- Die Hashfunktion H wird aus einer klassischen Hashfunktion erzeugt indem man mit den Bits der klassischen Hashfunktion eine Zufallsfolge aus $\{\pm 1\}$ erzeugt und diese dann durch Mischen auf die Stellen von c verteilt. Die Informationen der klassischen Hashfunktion werden also nur neu verteilt. Natürlich muss τ groß genug sein, damit die ursprüngliche Hashfunktion ihre kryptographischen Eigenschaften nicht verliert. $\tau = 39, 49, 60$. (Entspricht 192, 225, 257 Bit — vgl. Aufgabe 11)
- η (Wertebereich für $\mathbf{s}$, $\mathbf{e}$): Nicht zu klein, damit $\mathbf{s}$, $\mathbf{e}$ nicht über Brute-Force ermittelt werden kann. 2, 4, 2
- $\gamma_1 > \beta$ (Wertebereich für $\mathbf{r}$): Groß damit $\mathbf{r}$ seine Maskierungsaufgabe erfüllen kann aber nicht zu groß damit die Bedingung $\|v\|_\infty \geq \gamma_1 - \beta$ nicht zu einschränkend wird. 2^{17}, 2^{19}, 2^{19}

- $\gamma_2 > \beta$ (Rundungsgrenze) darf nicht zu groß sein, damit genügend Stellen für $\mathbf{u}_h$ übrig bleiben. Typisch $\gamma_2 = (p-1)/88$ bzw. $\gamma_2 = (p-1)/32$.

- Das Verwerfen stellt sicher, dass einerseits $\mathbf{v}$ keine Informationen über $\mathbf{s}$ preisgibt und andererseits beim Überprüfen keine Dekodierfehler passieren. Ist $\mathbf{v}$ zu groß, so ist die Maskierung von $c\mathbf{s}$ durch $\mathbf{r}$ nicht ausreichend. Jeder Koeffizient von $\mathbf{v}(x)$, der größer als $\gamma_1 - \beta$ ist, liefert statistische Informationen über den entsprechenden Koeffizienten von $c\mathbf{s}$.

  Idee: Ist r gleichverteilt im Intervall $[-\gamma_1, \gamma_1]$ und verschieben wir r um einen festen Wert s, dann ist $r + s$ gleichverteilt in $[-\gamma_1 + s, \gamma_1 + s]$. Durch Stichproben, können wir die obere/untere Grenze schätzten und damit auch s. Werden nur Stichproben aus dem Intervall $[-\gamma_1 + \beta, \gamma_1 - \beta]$ bekannt, mit $|s| \leq \beta$, dann sind diese Stichproben gleichverteilt und verraten keine Informationen über s.

  Die zweite Bedingung dient ebenfalls der Sicherheit und stellt sicher, dass bei der Verifikation kein Dekodierfehler auftritt.

- Über die Wahl der Parameter wird sicher gestellt, dass nicht zu viele Signaturen verworfen werden müssen (der Erwartungswert liegt bei ca. 4).

- Beim Standard werden noch weitere Modifikationen zur Effizienzsteigerung gemacht. Wie bei Kyber wird A aus einem Wert erzeugt. Zusätzlich werden bei $\mathbf{b}$ niedrige Bits entfernt, da bei der Verifikation nur die hohen Stellen benötigt werden. Dadurch kann es aber zu einem Übertrag von den niedrigen Stellen in die höheren Stellen kommen. Die Information ob ein Übertrag aufgetreten ist oder nicht muss dann der Signatur hinzugefügt werden, damit die Verifikation weiterhin klappt. Dieser Trick verkleinert den öffentlichen Schlüssel auf Kosten der Signaturlänge.

Für weitere Details siehe [Bai+21].

9.4 Kontrollfragen

Fragen zu Abschnitt 9.1: Gitter

Erklären Sie folgende Begriffe: Gitter, Gitterbasis, unimodular, Fundamentalbereich, Gitterdeterminante, SVP, CVP, Babai-Rundungsalgorithmus

1. Wie unterscheiden sich verschiedene Basen U und V für das gleiche Gitter?

   (Lösung zu Kontrollfrage 1)

2. Was beschreibt die Gitterdeterminante $\det(\mathcal{L})$ geometrisch?

   (Lösung zu Kontrollfrage 2)

3. Wie viele Ecken hat der Fundamentalbereich eines n-dimensionalen Gitters?

   (Lösung zu Kontrollfrage 3)

4. Ist der kürzeste Vektor in einem Gitter eindeutig?

   (Lösung zu Kontrollfrage 4)

5. Wie funktioniert der Babai-Rundungsalgorithmus?

   (Lösung zu Kontrollfrage 5)

6. Liefert der Babai-Rundungsalgorithmus für einen Punkt im Fundamentalbereich immer die näheste Ecke des Fundamentalbereichs?

   (Lösung zu Kontrollfrage 6)

7. Was ist der Unterschied zwischen einer *guten* und einer *schlechten* Basis für das CVP.

   (Lösung zu Kontrollfrage 7)

Fragen zu Abschnitt 9.2: Gitterreduktion

Erklären Sie folgende Begriffe: Gram–Schmidt-Verfahren, Hadamard-Ungleichung, Hadamard-Verhältnis, Lagrange-Algorithmus, längenreduzierte Basis, gewichtsreduzierte Basis, LLL-Algorithmus, Lovász-Bedingung, Hermite-Abschätzung

1. Wann wird das Hadamard-Verhältnis einer Basis maximal?

   (Lösung zu Kontrollfrage 1)

2. Wie unterscheiden sich der Lagrange-Algorithmus und der LLL-Algorithmus in zwei Dimensionen?

   (Lösung zu Kontrollfrage 2)

3. Was liefert der LLL-Algorithmus im Fall $\delta = 0$?

   (Lösung zu Kontrollfrage 3)

4. Löst der LLL-Algorithmus das SVP?

   (Lösung zu Kontrollfrage 4)

Fragen zu Abschnitt 9.3: Gitterbasierte Kryptographie

Erklären Sie folgende Begriffe: GGH, NTRU, LWE

1. Worauf basiert die Sicherheit von GGH?

   (Lösung zu Kontrollfrage 1)

2. Warum wird bei NTRU statt $\mathbb{Z}_q$ der Ring $\mathbb{Z}_q[x]_{x^n-1}$ verwendet?

   (Lösung zu Kontrollfrage 2)

3. Worauf basiert die Sicherheit von LWE?

   (Lösung zu Kontrollfrage 3)

4. Ist der private Schlüssel $\mathbf{s}$ beim LWE-Verfahren eindeutig.

   (Lösung zu Kontrollfrage 4)

5. Welche Rolle spielt bei LWE die Wahrscheinlichkeitsverteilung mit der Nonce bzw. Fehler gewählt werden?

   (Lösung zu Kontrollfrage 5)

6. Wird bei Frodo ein Polynomring verwendet?

   (Lösung zu Kontrollfrage 6)

7. Welcher Ring wird by Kyber verwendet verwendet?

   (Lösung zu Kontrollfrage 7)

8. Warum werden bei Dilithium potentielle Signaturen bei der Erstellung verworfen?

   (Lösung zu Kontrollfrage 8)

Lösungen zu den Kontrollfragen

Lösungen zu Abschnitt 9.1

1. Durch einen ganzzahlige unimodulare Matrix $W = U^{-1}V$.

2. Das Volumen des Fundamentalbereichs.

3. 2^n

4. Nein, in $\mathbb{Z}^n$ haben alle Basisvektoren die gleiche (kürzeste) Länge.

5. Man berechnet die (reellen) Koordinate bezüglich der Gitterbasis und rundet auf ganze Zahlen.

6. Nein. Im Allgemeinen nur wenn die Basis orthogonal ist.

7. Eine Gute Basis ist näherungsweise orthogonal.

Lösungen zu Abschnitt 9.2

1. Wenn die Vektoren der Basis orthogonal sind.
2. Für $\delta = 1$ sind beide identisch. Für $\delta < 1$ stoppt der LLL-Algorithmus eventuell zu früh, da er nicht mehr vertauscht.
3. Für $\delta = 0$ erfolgt kein Vertauschen und der Algorithmus terminiert nach dem ersten Schritt mit einer gewichtsreduzierten Basis.
4. Nein. Er löst es nur bis auf einen Faktor.

Lösungen zu Abschnitt 9.3

1. Auf der Schwierigkeit des CVP.
2. Weil das zugrundeliegende Gitterproblem in $\mathbb{Z}_q$ nur zweidimensional ist und daher keine Sicherheit bietet.
3. Auf der Schwierigkeit des CVP.
4. Nein, es kann mehrere Schlüssel geben mit denen korrekt entschlüsselt werden kann.
5. Durch die Wahrscheinlichkeitsverteilung wird sicher gestellt, dass eine statistische Analyse des Geheimtext keinerlei Informationen über den Klartext verrät.
6. Nein, es wird über $\mathbb{Z}_q$ gerechnet.
7. Der Polynomring $\mathbb{Z}_q[x]_{x^n+1}$.
8. Die Bedingung stellen sicher, dass es zu keinen Decodierfehlern kommt und dass eine statistische Analyse der Signatur nichts über den privaten SChlüssel verrät.

9.5 Übungen

Aufwärmübungen

1. Spannen $\mathbf{u}_1$, $\mathbf{u}_2$ und $\mathbf{v}_1$, $\mathbf{v}_2$ das gleiche Gitter auf?
   a) $\mathbf{u}_1 = (2, 1)$, $\mathbf{u}_2 = (0, 1)$ und $\mathbf{v}_1 = (4, 7)$, $\mathbf{v}_2 = (6, 10)$.
   a) $\mathbf{u}_1 = (2, 1)$, $\mathbf{u}_2 = (0, 1)$ und $\mathbf{v}_1 = (4, 3)$, $\mathbf{v}_2 = (5, 0)$
2. Lösen Sie das CVP für $\mathbf{x} = (119, 64)$ näherungsweise mit dem Rundungsalgorithmus bezüglich beider Basen aus Beispiel 9.5.

Weiterführende Aufgaben

1. Ist die Menge $\mathcal{L} = \{\mathbf{a} \in \mathbb{Z}^2 | a_1 + a_2 = 0 \bmod 2\}$ ein Gitter? Wenn ja, geben Sie eine Basis an.

2. Lösen Sie das CVP für $\mathbf{x} = (29.2, 3.4)$ näherungsweise mit dem Rundungsalgorithmus bezüglich beider Basen aus Beispiel 9.5.

3. Lösen Sie das CVP aus der letzten Aufgabe mit der Basis V näherungsweise mit dem Babai-Näheste-Ebene-Algorithmus.

4. Zeigen Sie, dass die Gitterbasis

$$U = \begin{pmatrix} 1 & 1 \\ \sqrt{3} & -\sqrt{3} \end{pmatrix}$$

längenreduziert ist und Gleichheit in der Ungleichung in Satz 9.10 annimmt.

5. Zeigen Sie, dass der Fehler beim Babai-Rundungsalgorithmus wie folgt abgeschätzt werden kann:

$$\|\mathbf{x} - U\mathbf{k}\| \leq \frac{1}{2}\big(\|\mathbf{u}_1\| + \cdots + \|\mathbf{u}_m\|\big), \qquad \mathbf{k} = \lfloor U^{-1}\mathbf{x} \rceil.$$

6. Zeigen Sie: Gilt für $\mathbf{x} \in \mathbb{R}^n$ und $\mathbf{a} \in \mathcal{L}$

$$\|\mathbf{x} - \mathbf{a}\| \leq \frac{\lambda_1(\mathcal{L})}{2},$$

dann kann es keinen näheren Gittervektor geben und wir haben das CVP gelöst.

7. Zeigen Sie, dass für eine längenreduzierte Gitterbasis in 3 Dimensionen

$$\|\mathbf{u}_1\|\|\mathbf{u}_2\|\|\mathbf{u}_3\| \leq 2\sqrt{2/3}\det(\mathcal{L})$$

und

$$\lambda_1(\mathcal{L}) \leq \frac{\sqrt{2}}{3^{1/6}}\det(\mathcal{L})^{1/3}$$

gilt. Analog gilt in 4 Dimensionen

$$\|\mathbf{u}_1\|\|\mathbf{u}_2\|\|\mathbf{u}_3\|\|\mathbf{u}_4\| \leq 4\sqrt{2/3}\det(\mathcal{L})$$

und

$$\lambda_1(\mathcal{L}) \leq (32/3)^{1/8}\det(\mathcal{L})^{1/4}.$$

8. Zeigen Sie, mit
$$\mathbf{y} = \mathbf{x} - \lfloor \frac{\langle \mathbf{u}, \mathbf{x} \rangle}{\|\mathbf{u}\|^2} \rceil \mathbf{u},$$
dass $\|\mathbf{y}\| \leq \|\mathbf{x}\|$ gilt. Wir nennen einen Vektor $\mathbf{x}$ dementsprechend **längenreduziert** bezüglich einer Gitterbasis U falls
$$|\frac{\langle \mathbf{u}_j, \mathbf{x} \rangle}{\|\mathbf{u}_j\|^2}| \leq \frac{1}{2}, \qquad 1 \leq j \leq m$$
gilt. Geben sie einen Algorithmus an, der zu $\mathbf{x}$ einen längenreduzierten Vektor $\mathbf{y}$ bestimmt.

   Lösen sie damit das CVP aus Beispiel 9.5.

9. Es sei $a(x) \in \mathcal{T}(d_+, d_-)$. Was ist $a(1)$? Kann im Fall $d_+ = d_-$ ein multiplikatives Inverses in $\mathcal{R}$ existieren?

10. Zeigen Sie für $a, b \in R$ die Abschätzung an die Koeffizienten des Produktes
$$|(a \cdot b)_k| \leq \max_{0 \leq j \leq n-1} |a_j| \sum_{j=0}^{n-1} |b_j|$$
gilt. Insbesondere für und $b \in \mathcal{T}(d_+, d_-)$ also
$$|(a \cdot b)_k| \leq (d_+ + d_-) \max_{0 \leq j \leq n-1} |a_j|.$$

11. Wie viele Polynome gibt es in $\mathcal{T}(d_+, d_-)$ bzw. $\mathcal{T}(d)$?

12. Die Komponenten des Fehlervektors $\mathbf{e} \in \mathbb{Z}^n$ werden nach einer symmetrischen Binomialverteilung $\mathcal{B}(\frac{1}{2}, 2\eta) - \eta$ gewählt.
   a) Was ist die Wahrscheinlichkeit, dass $|e_1 + \cdots + e_n| > \delta$?
   b) Was ist der Erwartungswert für die Länge $\|\mathbf{e}\|^2$?

Lösungen zu den Aufwärmübungen

1. a) Ja b) Nein

2. Die Koordinaten von $\mathbf{x}$ bezüglich U sind
$$\mathbf{x} = 34.8\mathbf{u}_1 + 49.4\mathbf{u}_2$$
und durch Runden ergibt sich der näheste Gittervektor zu
$$\mathbf{a} = \lfloor 34.8 \rceil \mathbf{u}_1 + \lfloor 49.4 \rceil \mathbf{u}_2 = 35\mathbf{u}_1 + 50\mathbf{u}_2 = (199, 63).$$

Die Koordinaten von $\mathbf{x}$ bezüglich V sind

$$\mathbf{x} = 14.6\mathbf{v}_1 + 20.2\mathbf{v}_2$$

und durch Runden ergibt sich der vermeintlich näheste Gittervektor zu $\tilde{\mathbf{a}} = (120, 65)$. Dieser hat aber einen größeren Abstand $\|\mathbf{x} - \tilde{\mathbf{a}}\| = \sqrt{2}$ als unser vorheriger Vektor $\|\mathbf{x} - \mathbf{a}\| = 1$.

Lösungen zu ausgewählten Aufgaben

1. $\mathcal{L} = \mathcal{L}((1,1),(2,0))$
2. $\mathbf{a} = (29, 3)$ bzw. $\tilde{\mathbf{a}} = (24, 3)$.
3. $(29, 3)$
4. –
5. –
6. –
7. –
8. –
9. $a(1) = d_+ - d_-$. Nein (da $a(1) = m(1) = 0$).
10. –
11. $\frac{n!}{d_+!d_-!(n-d_+-d_-)!}$ bzw. $2^d \frac{n!}{d!(n-d)!}$
12. a) $2(1 - I_{1/2}(\eta n - \lfloor \delta \rfloor, \eta n + 1 + \lfloor \delta \rfloor))$.
    b) $\frac{n\eta}{2}$.

Literatur

[Alk+21] E. Alkim u. a. *FrodoKEM, Learning With Errors Key Encapsulation: Algorithm Specifications And Supporting Documentation*. Techn. Ber. 2021. URL: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf.

[Ava+21] R. Avanzi u. a. *CRYSTALS-Kyber: Algorithm Specifications And Supporting Documentation*. Techn. Ber. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf.

[Bai+21] S. Bai u. a. *CRYSTALS-Dilithium: Algorithm Specifications And Supporting Documentation*. Techn. Ber. 2021. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[Ber+18] D. J. Bernstein u. a. „NTRU Prime: Reducing Attack Surface at Low Cost“. In: *Selected Areas in Cryptography – SAC 2017*. Hrsg. von C. Adams und J. Camenisch. Cham: Springer, 2018, S. 235–260. DOI: 10.1007/978-3-319-72565-9_12.

[Bre12] M. R. Bremner. *Lattice basis reduction: An introduction to the LLL algorithm and its applications*. Boca Raton: CRC Press, 2012.

[BSI] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. Techn. Ber. 2023. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf (besucht am 09. 01. 2023).

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

[GGH97] O. Goldreich, S. Goldwasser und S. Halevi. „Public-key cryptosystems from lattice reduction problems“. In: *Advances in Cryptology — CRYPTO '97*. Hrsg. von B. S. Kaliski. Berlin: Springer, 1997, S. 112–131. DOI: 10.1007/BFb0052231.

[HPS08] J. Hoffstein, J. Pipher und J. H. Silverman. *An Introduction to Mathematical Cryptography*. New York: Springer, 2008.

[LLL82] A. K. Lenstra, H. W. Lenstra und L. Lovász. „Factoring polynomials with rational coefficients“. In: *Mathematische Annalen* 261 (1982), S. 515–534. DOI: 10.1007/BF01457454.

[MR09] D. Micciancio und O. Regev. „Lattice-based Cryptography“. In: *Post-Quantum Cryptography*. Hrsg. von D. J. Bernstein, J. Buchmann und E. Dahmen. Berlin: Springerg, 2009, S. 147–191. DOI: 10.1007/978-3-540-88702-7_5.

[MV13] D. Micciancio und P. Voulgaris. „A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations“. In: *SIAM Journal on Computing* 42.3 (2013), S. 1364–1391. DOI: 10.1137/100811970.

[Ngu99] P. Nguyen. „Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97“. In: *Advances in Cryptology — CRYPTO' 99*. Hrsg. von M. Wiener. Berlin: Springer, 1999, S. 288–304. DOI: 10.1007/3-540-48405-1_18.

[NR06] P. Q. Nguyen und O. Regev. „Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures“. In: *Advances in Cryptology - EUROCRYPT 2006*. Hrsg. von S. Vaudenay. Berlin: Springer, 2006, S. 271–288. DOI: 10.1007/11761679_17.

[NV10] P. Q. Nguyen und B. Vallée, Hrsg. *The LLL Algorithm; Survey and Applications*. New York: Springer, 2010.

[Reg05] O. Regev. „On Lattices, Learning with Errors, Random Linear Codes, and Cryptography“. In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. New York: Association for Computing Machinery, 2005, S. 84–93. DOI: 10.1145/1060590.1060603.

[Reg10] O. Regev. „The Learning with Errors Problem (Invited Survey)“. In: *2010 IEEE 25th Annual Conference on Computational Complexity*. 2010, S. 191–204. DOI: 10.1109/CCC.2010.26.