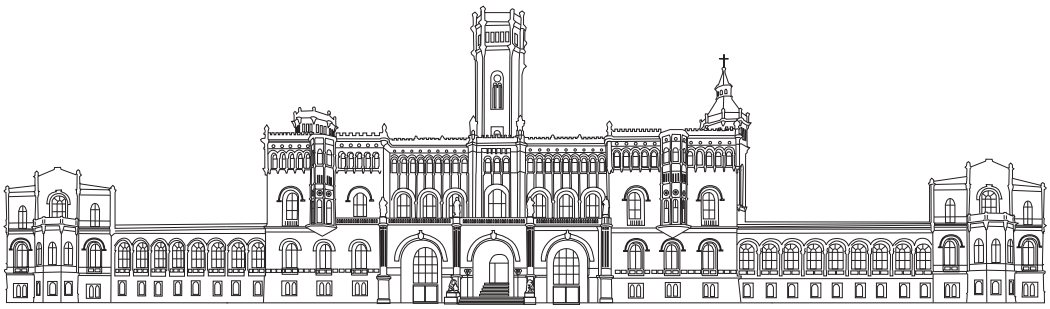


Leibniz Universität Hannover

Fakultät für Elektrotechnik und Informatik

Institut für Theoretische Informatik

Gitterbasierte Post-Quantum-Kryptographie

Masterarbeit

eingereicht von

Jan Eberhardt

am 12. November 2018

Erstprüfer:	Prof. Dr. Heribert Vollmer
Zweitprüfer:	Dr. Arne Meier
Betreuer:	Dr. Arne Meier
Matrikelnr.:	2944210

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

_____

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit, in der weltweit über das Internet kommuniziert wird, spielt die Gewährleistung der Vertraulichkeit von Datenströmen nach wie vor eine entscheidende Rolle. Die dafür eingesetzten kryptographischen Verfahren haben einen langen Reifungsprozess durchlebt und werden regelmäßig gegen neue Angriffsszenarien abgesichert. Auch die zugrundeliegenden mathematischen Probleme, auf denen die Sicherheit heutiger Public-Key-Kryptographie beruht, sind weitgehend erforscht. Mit dem Aufkommen von Quantencomputern droht diese Sicherheit ins wanken zu geraten, weshalb mittelbis langfristig Alternativen für die heutigen Public-Key-Verfahren erarbeitet werden sollten.

Quantencomputer und ihre Bedeutung für die moderne Kryptographie

Quantencomputer sind eines der interessantesten Forschungsgebiete in der Schnittmenge von Quantenphysik und Informatik. Von der Entwicklung von Quantencomputern wird sich insbesondere in Anwendungsbereichen bei denen immense Mengen von Daten verarbeitet werden („Big Data“), ein enormer Vorteil versprochen.

Erste prototypische Quantencomputer mit wenigen Qubits gibt es bereits. Zurzeit arbeiten große Technologiefirmen wie Google und D-Wave Systems zusammen mit der NASA an größeren Quantencomputern und erzielen dabei Stück für Stück Fortschritte[1]. Die kanadische Firma D-Wave Systems baute schon 2007 einen prototypischen Quantencomputer mit „16 Qubit quantum annealing“. 2018 stellte Google einen neuen Quantencomputer mit 72 Qubits vor.

Scott Aaronson, einer der führenden Experten auf dem Gebiet für Quantencomputer und Quantenalgorithmen, beschreibt in seinem Artikel *The Limits of Quantum Computers* [Aar08] welche Erwartungen man realistischer Weise an Quantencomputer stellen kann. In dem Artikel wird eine Einführung in Quantencomputer gegeben und eine Einordnung von Quantenalgorithmen in die moderne Komplexitätstheorie. Aaronson kritisiert die teilweise überzogenen Erwartungen, die in manchen populären Publikationen über Quantencomputer geschürt würden. Quantencomputer seien keine magischen Maschinen, die NP-vollständige Probleme effizient lösen können. Es gebe aber spezielle Anwendungen, bei denen Quantencomputer einen enormen Performance-Gewinn gegenüber klassischen Computern verzeichnen können. Die meisten der aktuellen Forschungen sind aber (noch) rein theoretischer Natur.

Die Sicherheit der Verschlüsselungsmethoden, die heute weltweit im Internet eingesetzt werden um die Vertraulichkeit der Datenströme zu gewährleisten, basiert im Kern

[1] https://ai.googleblog.com/2015/12/when-can-quantum-annealing-win.html

auf der Annahme, dass Primfaktorzerlegung und die Berechnung des diskreten Logarithmus im Allgemeinen schwer lösbare Probleme sind. Kryptosysteme, deren Sicherheit auf dieser Annahme beruhen, sind unter anderem das Public-Key-Verfahren RSA und der Diffie-Hellman-Schlüsselaustausch (sowohl in der klassischen Variante, als auch der auf elliptischen Kurven aufgebaute). Bei in der Praxis eingesetzten Implementierungen von kryptographischen Verfahren wählt man die Parameter (bspw. die Primfaktoren im RSA-Verfahren) so groß, dass selbst heutige Supercomputer astronomische Zeiten zum Lösen der zugrundeliegenden mathematischen Probleme benötigen würden.

Seit Mitte der 90er Jahre ist bekannt, dass diese Probleme (zumindest theoretisch) auch effizient lösbar sind. Peter W. Shor entwickelte einen für Quantencomputer konziperten Polynomial-Zeit-Algorithmus zur Primfaktorzerlegung und Berechnung des diskreten Logarithmus [Sho97]. Dies war ein bahnbrechendes Resultat in der theoretischen Informatik und insbesondere für die Kryptographie bedeutsam, denn damit war gezeigt, dass aktuelle Public-Key-Kryptosysteme mithilfe von Quantenalgorithmen effizient attackierbar sind.

Es scheint nur noch eine Frage der Zeit zu sein, bis Quantencomputer mit ausreichend vielen Qubits entwickelt werden, sodass diese, die heute in breitem Maße eingesetzten Verschlüsselungsverfahren, effizient brechen können. Ziel ist es daher, Kryptosysteme zu entwickeln, die auch in Zeiten von Quantencomputern noch als sicher gelten können.

Gitter

Diese Arbeit soll eine Einführung in Gitter und wichtige Bausteine gitterbasierter Kryptographie geben. Ein Gitter besteht aus allen ganzzahligen Linearkombinationen einer Menge von linear unabhängigen Vektoren im $\mathbb{R}^n$. Diese einfache Beschreibung eines Gitters lässt zunächst nicht erahnen, dass auf diesen geometrischen Strukturen Probleme definiert werden können, die (NP-)schwer zu lösen sind. Die Probleme sind jedoch sogar derart schwer, dass selbst Quantencomputer nach aktuellem Forschungsstand keine nennenswerten Vorteile gegenüber klassischen Computern haben sollen.

Seit gut 200 Jahren beschäftigen sich Mathematiker mit Gittern und den damit zusammenhängenden Fragen. Eines der wichtigsten Probleme, die auf Gittern definiert sind, ist das *Shortest Vektor Problem* (das Problem, den kürzesten Vektor in einem Gitter zu finden), welches in Abschnitt 3.1 definiert wird. Dies wurde schon 1842 von Dirichlet in Form von simultanen diophantischen Approximations-Problemen formuliert. Erwähnenswerte Theoreme erzielten auch Gauss 1801, Hermite 1850 und Minkowski 1896 [Min96]. Besonders die Ergebnisse der Forschung von Minkowski sind heute interessant für gitterbasierte Kryptographie. Gitter wurden jedoch erst in den letzten 30-40 Jahren für kryptographische Zwecke in Betracht gezogen.

Ein Meilenstein ist der LLL-Algorithmus von Arjen Lenstra, Hendrik Lenstra und László Lovász [LLL82], der als erster effizienter Algorithmus zum Finden von relativ kurzen Vektoren in einem Gitter gilt. Die Anwendungsmöglichkeiten von LLL erstrecken sich aber auch auf viele andere Bereiche der Mathematik, Informatik und Kryptologie (dazu mehr in Unterabschnitt 3.3.1).

Ein Durchbruch in der Komplexitätsanalyse von Gitterproblemen gelang Miklós Ajtai im Jahr 1996. Er zeigte eine Worst-case zu Average-case Reduktion für das Problem kurze Vektoren in einem Gitter zu finden [Ajt96]. Damit war der Grundstein der gitterbasierten Kryptographie gelegt, denn nun war ein Weg bekannt, zufällige Instanzen kryptographischer Probleme mit den schwierigsten Fällen gitterbasierter Probleme zu verknüpfen. Eine ausführliche Beschreibung dessen ist in Unterabschnitt 4.1.1 zu finden. Auf der Suche nach neuen kryptographischen Verfahren lenkte dieses Resultat viel Aufmerksamkeit auf Gitter. Im weiteren Verlauf wurden verschiedene Probleme definiert, die direkt oder indirekt mit Problemstellungen auf Gittern zusammenhängen und Grundlage vieler gitterbasierter Kryptosysteme sind. Die beiden wichtigsten Probleme *Short Integer Solutions Problem* und *Learning With Errors Problem* werden in Kapitel 4 vorgestellt.

NIST-PQC-Wettbewerb

Heute gibt es eine Vielzahl an kryptographischen Verfahren die auf Gitterproblemen basieren welche als aussichtsreiche Kandidaten für zukünftige Verschlüsselungsstandards gelten. Dies ist unter anderem darin begründet, dass zurzeit keine Quantenalgorithmen bekannt sind, die gitterbasierte Probleme (bis auf wenige Ausnahmen) signifikant effizienter lösen als klassische Algorithmen. Weitere Vorteile und ein Vergleich zu bereits etablierten kryptographischen Verfahren (vor allem Public-Key-Kryptographie) werden in Abschnitt 5.7 vorgestellt.

Das NIST[2] rief 2016 einen Wettbewerb zum Thema Post-Quantum Kryptographie aus um Vorschläge für Public-Key-Verfahren zu sammeln, die auch gegenüber Algorithmen die auf Quantencomputern implementiert werden können, eine hohe Sicherheit bieten. Der aktuelle Stand des Wettbewerbs und ausgewählte Kryptosysteme, die an dem Wettbewerb teilnehmen werden in Kapitel 5 beschrieben.

[2]National Institute of Standards and Technology – `https://www.nist.gov/`

2 Gitter

In diesem Kapitel werden grundlegende Definitionen und Zusammenhänge bezüglich (mathematischer) Gitter dargestellt. Diese basieren hauptsächlich auf dem Buch „Complexity Of Lattice Problems – A Cryptographic Perspective“ [MG02] von Micciancio und Goldwasser, sowie auf Vorlesungsskripten von Schnorr [Sch18], Peikert [Pei15b], Regev [Reg04] und Vaikuntanathan [Vai15]. Es werden grundlegende Kenntnisse der (linearen) Algebra vorausgesetzt.

2.1 Notation

Es wird folgende Notation verwendet um das Lesen von mathematischen Formeln leichter verständlich zu machen. Vektoren und Skalare sind kleingeschrieben. Zur Unterscheidung dieser werden Vektoren fett dargestellt (bspw. $\mathbf{v}$). Die Elemente eines n-elementigen Vektors $\mathbf{x}$ werden mit x_i für $1 \leq i \leq n \in \mathbb{N}$ indiziert. Matrizen werden ebenfalls fett dargestellt und sind großgeschrieben (bspw. $\mathbf{M}$). Um eine Transposition einer Matrix anzuzeigen, wird ein hochgestelltes T (bspw. $\mathbf{M}^T$) verwendet. Das Sysmbol $\mathbf{E}_n$ wird für die (n-dimensionale) Einheitsmatrix gebraucht. Längen von Vektoren werden (wenn nicht explizit anders angegeben) mit der euklidischen (2-)Norm beschrieben:

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Es wird der üblichen $\mathcal{O}$-Notation zur Beschreibung der asymptotischen Laufzeit von Algorithmen gefolgt. Außerdem wird $\tilde{\mathcal{O}}$ verwendet, wenn logarithmische Faktoren versteckt werden.

2.2 Gitterdefinition

Um sich zunächst eine erste Vorstellung von Gittern zu machen, kann man sie sich als regelmäßige, unendliche Schnittpunktmenge eines (n-dimensionalen) Rasters vorstellen. Im Folgenden wird dies durch verschiedene Grafiken veranschaulicht. Doch zunächst folgt die mathematische Definition eines Gitters.

Definition 2.1. Seien $\mathbf{b}_1, \ldots, \mathbf{b}_m$ linear unabhängige Vektoren im euklidischen Vektorraum $\mathbb{R}^n$, $n \in \mathbb{N}$. Ein n-dimensionales *Gitter* $\mathcal{G}$ bezeichnet alle ganzzahligen Linearkombinationen einer solchen Menge von Vektoren:

$$\mathcal{G}(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \left\{ \sum_{i=1}^{m} x_i \mathbf{b}_i \;\middle|\; x_i \in \mathbb{Z} \right\}.$$

Die Vektoren $\mathbf{b}_i$ mit $i \in \{1, \ldots, m\}$ bilden eine *Basis* des Gitters. Üblicherweise werden die $\mathbf{b}_i$ als Spaltenvektoren einer Matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ repräsentiert, wobei sich folgende äquivalente Gitterdefinition ergibt:

$$\mathcal{G}(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \mathcal{G}(\mathbf{B}) = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^m\}.$$

Aus dieser Darstellung heraus wird ersichtlich, dass jedes Gitter aus einer bijektiven linearen Transformation des Gitters $\mathbb{Z}^m$ hervorgeht (siehe Abbildung 2.1 als Beispiel).

Die m Basisvektoren $\mathbf{b}_i \in \mathbb{R}^n$ eines Gitters spannen einen Untervektorraum von $\mathbb{R}^n$ auf, wobei n die Dimension des Gitters bezeichnet und m den Rang des Gitters.

Definition 2.2. Der *Rang* m eines Gitters $\mathcal{G} \subset \mathbb{R}^n$ ist die Dimension seiner linearen Hülle, also $m = \dim(\text{span}(\mathcal{G}))$ mit $\text{span}(\mathcal{G}) := \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{R}^n\} = \sum_{i=1}^{n} \mathbf{b_i}\mathbb{R}$. Ist $m = n$, so hat $\mathcal{G}$ vollen Rang und wird auch als *vollständig* bezeichnet.

Ein Beispiel für ein nicht-vollständiges Gitter ist $\mathcal{G}((1,1)^T)$, welches im $\mathbb{R}^2$ liegt, jedoch Rang 1 hat (siehe Abbildung 2.2). Der allgemeine Fall ($m \leq n$) wird in diesem Kapitel der Einfachheit halber nicht betrachtet. Die meisten Resultate lassen sich aber auch für den allgemeineren Fall zeigen, oder auf den Unterraum $\text{span}(\mathcal{G}) \cong \mathbb{R}^m$ beziehen. Da Gitter im Hinblick auf kryptographische Verfahren eingeführt werden, die auf Computersystemen implementiert werden, ist es sinnvoll, sich auf Gitter mit rationalen bzw. ganzzahligen

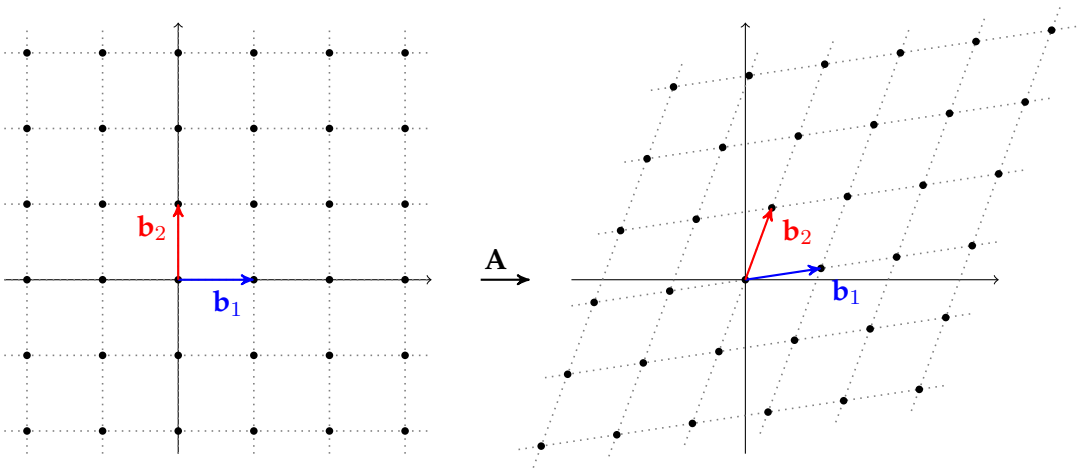


Abbildung 2.1: Transformiertes Gitter $\mathcal{G} = \{\mathbf{Ax} \mid \mathbf{x} \in \mathbb{Z}^2\}$ mit $\mathbf{A} = \begin{pmatrix} 1 & 0{,}35 \\ 0{,}15 & 0{,}95 \end{pmatrix}$.

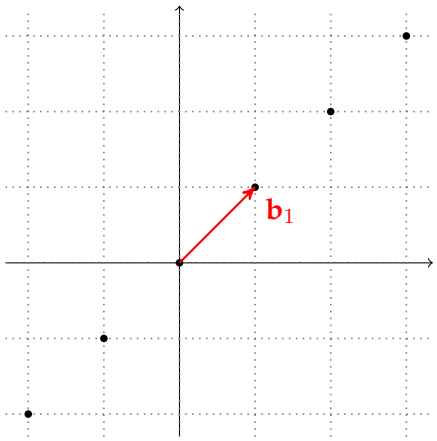


Abbildung 2.2: Nicht-vollständiges Gitter $\mathcal{G}'$ mit $\text{Rang}(\mathcal{G}') = 1$ und $\dim(\mathcal{G}') = 2$.

Basen (also Gitter die Teilmengen von $\mathbb{Q}^n$ bzw. $\mathbb{Z}^n$ sind) zu beschränken. Rationale Gitter sind durch geeignete Skalierung in ganzzahlige Gitter überführbar. Daher können ohne Beschränkung der Allgemeinheit ganzzahlige Gitter betrachtet werden. Die Basismatrix eines Gitters dieser Art ist also quadratisch und invertierbar (da voller Rang) und somit Element der *allgemeinen linearen Gruppe* $\text{GL}_n(\mathbb{R})$, mit rationalen bzw. ganzzahligen Einträgen. Die präsentierten mathematischen Resultate gelten (außer es wird explizit angegeben) dennoch auch für allgemeine Gitter $\subset \mathbb{R}^n$.

2.3 Gitterbasen

Die Basis eines Gitters ist nicht eindeutig bestimmt. Für jedes Gitter gibt es sogar unendlich viele Basen. In Abbildung 2.3 und Abbildung 2.4 ist das Gitter $\mathbb{Z}^2$ mit unterschiedlichen Basisvektoren dargestellt. Basen die das selbe Gitter erzeugen, lassen sich durch *unimodulare* Matrizen ineinander umrechnen.

Definition 2.3. Eine Matrix $\mathbf{U} \in \mathbb{Z}^{n\times n}$ heißt *unimodular*, wenn $\det(\mathbf{U}) = \pm 1$ gilt. Solche Matrizen bilden mit der Matrizenmultiplikation die allgemeine lineare Gruppe $\text{GL}_n(\mathbb{Z})$ über den ganzen Zahlen.

Lemma 2.4. *$\mathcal{G}(\mathbf{B}_1) = \mathcal{G}(\mathbf{B_2})$ genau dann wenn es eine ganzzahlige unimodulare Matrix $\mathbf{U}$ gibt, so dass $\mathbf{B}_2 = \mathbf{B}_1\mathbf{U}$.*

Beweis. „⇒“: Seien $\mathbf{B}_1$ und $\mathbf{B}_2$ Basismatrizen desselben Gitters $\mathcal{G}$, also $\{\mathbf{B}_1\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\} = \{\mathbf{B}_2\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$, kurz $\mathbf{B}_1 \cdot \mathbb{Z}^n = \mathbf{B}_2 \cdot \mathbb{Z}^n$. Dann gibt es ein $\mathbf{U} \in \mathbb{Z}^{n\times n}$ mit $\mathbf{B}_2 = \mathbf{B}_1\mathbf{U}$, da jeder Spaltenvektor von $\mathbf{B}_2$ eine ganzzahlige Linearkombination der Spaltenvektoren von $\mathbf{B}_1$ ist. Umgekehrt gibt es ebenso ein $\mathbf{V} \in \mathbb{Z}^{n\times n}$, sodass $\mathbf{B}_1 = \mathbf{B}_2\mathbf{V}$ gilt. Zusammen also $\mathbf{B}_2 = \mathbf{B}_1\mathbf{U} = \mathbf{B}_2\mathbf{V}\mathbf{U}$. Da $\mathbf{B}_2$ invertierbar ist, folgt $\mathbf{E}_n = \mathbf{V}\mathbf{U}$ und $\det(\mathbf{E}_n) = 1 = \det(\mathbf{V}) \cdot \det(\mathbf{U})$ und somit $|\det(\mathbf{V})| = |\det(\mathbf{U})| = 1$. Also sind $\mathbf{U}$ und $\mathbf{V}$ unimodular.

„⇐“: Auf der anderen Seite sei $\mathbf{B}_2 = \mathbf{B}_1\mathbf{U}$. Weil $\mathbf{U}$ und $\mathbf{U}^{-1}$ ganzzahlig unimodular sind, folgt aus $\mathbf{U}\mathbf{x} = \mathbf{y} \Leftrightarrow \mathbf{x} = \mathbf{U}^{-1}\mathbf{y}$, dass $\mathbf{U} \cdot \mathbb{Z}^n = \mathbb{Z}^n$ und daraus wiederum die

Behauptung:

$$\mathcal{G}(\mathbf{B}_1) = \mathbf{B}_1 \cdot \mathbb{Z}^n = \mathbf{B}_1\mathbf{U} \cdot \mathbb{Z}^n = \mathbf{B}_2 \cdot \mathbb{Z}^n = \mathcal{G}(\mathbf{B}_2)$$

□

Aus diesem Lemma lässt sich folgern, dass $\mathbf{B} \in \mathbb{Z}^{n \times n}$ genau dann eine Basis von $\mathbb{Z}^n$ darstellt, wenn $\mathbf{B}$ unimodular ist.

Korollar 2.4.1. *Die Basen von $\mathbb{Z}^n$ sind genau die unimodularen Matrizen $\mathbf{U} \in \mathbb{Z}^{n \times n}$.*

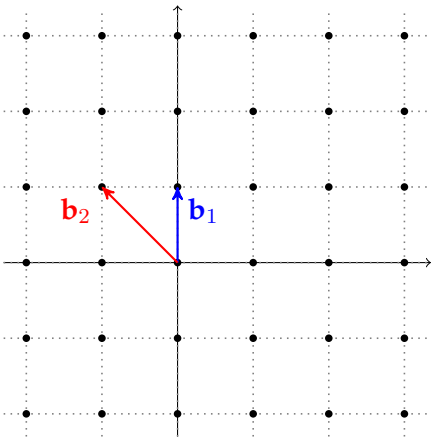


Abbildung 2.3: Gitter $\mathbb{Z}^2$ mit Basisvektoren $\mathbf{b}_1 = (0,1)^T$ und $\mathbf{b}_2 = (1,1)^T$.

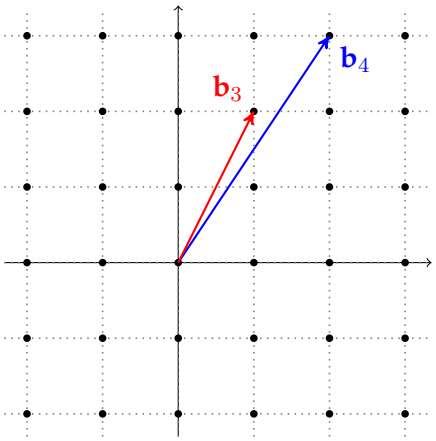


Abbildung 2.4: Gitter $\mathbb{Z}^2$ mit Basisvektoren $\mathbf{b}_3 = (1,2)^T$ und $\mathbf{b}_4 = (2,3)^T$.

Verschiedene Basen eines Gitters sind unterschiedlich gut zum Lösen bestimmter Probleme im jeweiligen Gitter geeignet. Kurze und annähernd orthogonale Basisvektoren (genauere Angaben folgen später) lassen sich beispielsweise in effizienten Algorithmen zur Bestimmung des nächstgelegenen Gitterpunktes zu einem beliebigen Raumpunkt verwenden (*Closest Vector Problem* – CVP). Die Bestimmung wird ungleich schwerer, wenn nur eine „schlechte" Basis (mit langen Basisvektoren) desselben Gitters bekannt ist. Dieses und andere Probleme, und die Beziehung zu unterschiedlichen Gitterbasen werden in Kapitel 3 ausführlich beschrieben.

2.4 Grundmasche eines Gitters

Gitter können durch Angabe einer Basis eindeutig charakterisiert werden. Damit zusammenhängend ist das *grundlegende Parallelepiped*, das von den Basisvektoren aufgespannt wird. Ein Parallelepiped ist ein dreidimensionales Parallelogramm. H. S. M. Coxeter definierte die n-dimensionale Verallgemeinerung von Parallelogrammen als *Parallelotop* [Cox73]. In der Literatur wird jedoch oft die Bezeichnung (n-dimensionales) Parallelepiped genutzt, die auch in dieser Arbeit verwendet wird. Das grundlegende Parallelepiped wird im Bezug auf Gitter auch *Grundmasche* genannt; abgeleitet von der Darstellung des

Parallelogramms, das von den Basisvektoren eines zweidimensionalen Gitters aufgespannt wird und an einen Maschendrahtzaun erinnern kann. *Grundlegend* bedeutet hier, dass das Parallelepiped auf dem Ursprungspunkt **0** steht.

Definition 2.5 (Grundmasche)**.** Sei $\mathcal{G}$ ein Gitter mit Basis $\mathbf{B}$. Dann ist

$$\mathcal{P} := \mathcal{P}(\mathbf{B}) := \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \;\middle|\; \alpha_i \in [0,1) \right\} \subset \text{span}(\mathcal{G})$$

das grundlegende Parallelepiped, bzw. die Grundmasche des Gitters $\mathcal{G}$.

In Abbildung 2.5 und Abbildung 2.6 ist exemplarisch jeweils das Gitter $\mathbb{Z}^2$ mit unterschiedlichen Basen und somit anderen Grundmaschen dargestellt. Aus der linearen Algebra ist bekannt, dass das Volumen der Grundmasche durch die Determinante der aufspannenden Vektoren (repräsentiert durch Basismatrix $\mathbf{B}$) bestimmt ist. Der Wert des Volumens entspricht gleichzeitig der Definition der Determinante des Gitters:

Definition 2.6. Sei $\mathcal{G}$ ein Gitter mit Basis $\mathbf{B}$. Die Determinante des Gitters $\mathcal{G}$ ist definiert als

$$\det(\mathcal{G}(\mathbf{B})) := |\det(\mathbf{B})| = \text{vol}(\mathcal{P}(\mathbf{B})).$$

Das Volumen der Grundmasche, bzw. der Betrag von $\det(\mathcal{G}(\mathbf{B}))$ ist invariant in Bezug auf die Wahl der Basis $\mathbf{B}$. Dies ist leicht anhand der Multiplikativität der Determinante und Lemma 2.4 zu sehen. Aus $\mathbf{B} = \mathbf{B}'\mathbf{U}$ mit $|\det(\mathbf{U})| = 1$ folgt $|\det(\mathbf{B})| = |\det(\mathbf{B}'\mathbf{U})| = |\det(\mathbf{B}')| \cdot |\det(\mathbf{U})| = |\det(\mathbf{B}')|$.

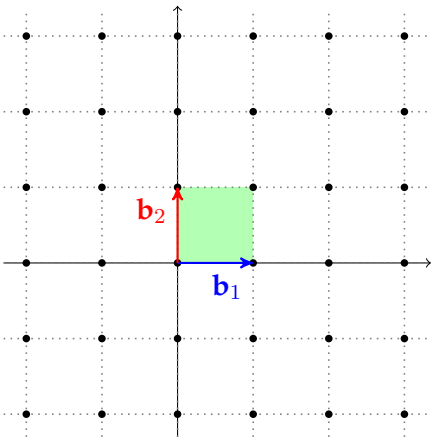


Abbildung 2.5: Gitter $\mathbb{Z}^2$ mit Basisvektoren $(0,1)^T$ und $(1,0)^T$ und eingefärbter Grundmasche.

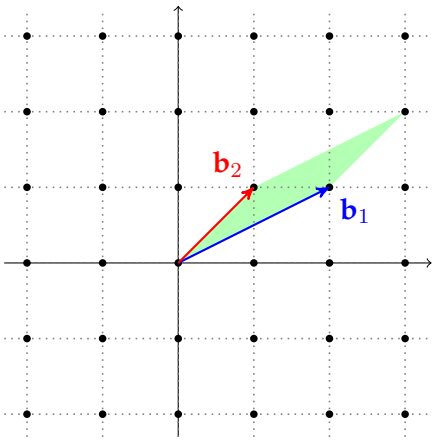


Abbildung 2.6: Gitter $\mathbb{Z}^2$ mit Basisvektoren $(1,1)^T$ und $(2,1)^T$ und eingefärbter Grundmasche.

Alternativ lässt sich die Determinante eines Gitters als Volumen pro Gitterpunkt innerhalb einer Kugel mit Radius r definieren:

Definition 2.7. Eine Menge $M \subseteq \mathbb{R}^n$ heißt **0**-symmetrisch oder zentral-symmetrisch wenn für alle $\mathbf{x} \in M$ ebenfalls $-\mathbf{x} \in M$ ist. Sei $\overline{\mathcal{K}}_n(\mathbf{0}, r) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq r\}$ die n-dimensionale, **0**-symmetrische Kugel mit Radius r. Dann lässt sich die Gitterdeterminante so definieren:

$$\det(\mathcal{G}) = \lim_{r \to \infty} \frac{\mathrm{vol}(\mathcal{K}_n(\mathbf{0}, r))}{\mathcal{G} \cap \mathcal{K}_n(\mathbf{0}, r)}$$

Aufgrund dieser Definitionen der Gitterdeterminante wird intuitiv klar, dass die Determinante mit der Dichte der Gitterpunkte zusammenhängt. Je kleiner die Gitterdeterminante, die dem Volumen der Grundmasche entspricht, desto enger liegen die Gitterpunkte beieinander.

Definition 2.8 (Dichte eines Gitters)**.** Die (globale) Dichte der Punkte in einem Gitter ergibt sich aus der Reziproken des n-dimensionalen Volumens des grundlegenden Parallelepipeds eines Gitters, welches mit der Determinante des Gitters ausgedrückt wird.

$$\mathcal{D}(\mathcal{G}) := \frac{1}{\det(\mathcal{G})}$$

2.4.1 Partitionierung des Raums $\mathrm{span}(\mathcal{G})$

Gitter sind unendliche, periodische Punkteraster im $\mathbb{R}^n$ und korrespondieren mit einer periodischen Partitionierung des Raums $\mathrm{span}(\mathcal{G}(\mathbf{B})) = \mathbb{R}^n$, den die Basisvektoren aus $\mathbf{B}$ aufspannen. Die Partitionierung besteht aus gleichförmigen n-dimensionalen Gebieten, welche *grundlegende Gebiete* eines Gitters genannt werden.

Definition 2.9. Ein grundlegendes Gebiet eines vollständigen Gitters $\mathcal{G} \subset \mathbb{R}^n$ ist eine Teilmenge $\mathcal{F} \subset \mathrm{span}(\mathcal{G}) \simeq \mathbb{R}^n$, dessen Verschiebungen $\mathbf{g} + \mathcal{F} = \{\mathbf{g} + \mathbf{x} \mid \mathbf{x} \in \mathcal{F}\}$ über alle Gitterpunkte $\mathbf{g} \in \mathcal{G}$ eine Partition von $\mathbb{R}^n$ ergeben.

Die Grundmasche $\mathcal{P}$ eines Gitters ist das natürliche grundlegende Gebiet eines Gitters (vgl. Abbildung 2.8). Jeder Punkt $\mathbf{x}$ hat die eindeutige Zerlegung $\mathbf{x} = \mathbf{g} + \mathbf{p}$ mit $\mathbf{g} \in \mathcal{G}$ und $\mathbf{p} \in \mathcal{P}$. Somit wird $\mathrm{span}(\mathcal{G})$ durch das Anlegen der Grundmasche an jeden Gitterpunkt abgedeckt, also $\mathrm{span}(\mathcal{G}) = \bigcup_{\mathbf{g} \in \mathcal{G}} \mathbf{g} + \mathcal{P} := \{\mathbf{g} + \mathbf{p} \mid \mathbf{g} \in \mathcal{G} \text{ und } \mathbf{p} \in \mathcal{P}\}$.

2.4.2 Voronoi-Zellen als grundlegendes Gebiet eines Gitters

Ein Beispiel für ein anderes grundlegendes Gebiet ist die *Voronoi-Zelle* (vgl. Abbildung 2.7). Sie ist die Menge aller Punkte in $\mathbb{R}^n$, die näher am Ursprung als an irgendwelchen anderen Gitterpunkten liegen:

$$\mathcal{V}(\mathcal{G}) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| < \|\mathbf{x} - \mathbf{g}\| \ \ \forall \mathbf{g} \in \mathcal{G} \setminus \{\mathbf{0}\}\}.$$

Anders als bei der Grundmasche liegen die Gitterpunkte jeweils im Zentrum einer Zelle und nicht an einer Ecke. Die Verschiebung von Voronoi-Zellen bildet eine Abdeckung des

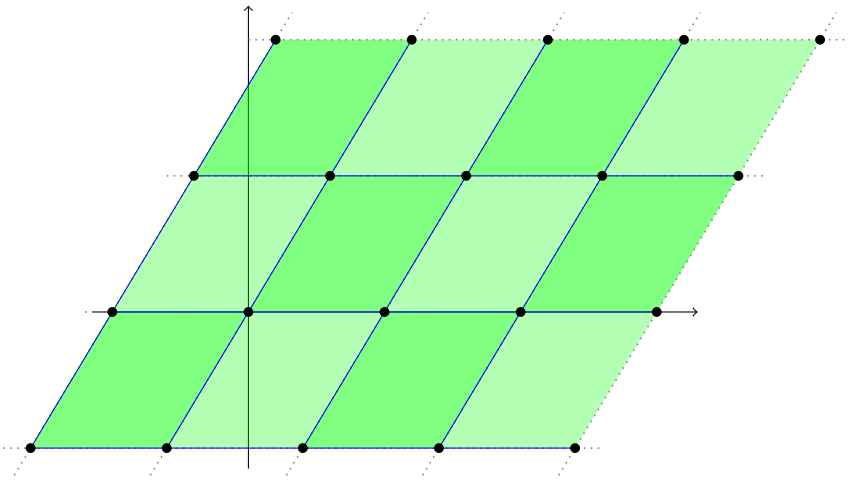


Abbildung 2.8: span($\mathcal{G}$) partitioniert durch die Grundmasche des Gitters.

$\mathbb{R}^n$, bis auf die Punkte, die auf den Grenzen zwischen den Zellen liegen und den selben Abstand zu je zwei Gitterpunkten haben. Um ein echtes grundlegendes Gebiet zu sein, müssten die Voronoi-Zellen halb-offen definiert werden, was an dieser Stelle zu viele technische Details benötigen würde. Voronoi-Zellen sind Grundlage einiger Exponentialzeit-Algorithmen zur Lösung gitterbasierter Probleme, beispielsweise dem *Closest Vector Problem* (CVP), welches in Abschnitt 3.2 definiert wird. Mehr zu Voronoi-basierten Algorithmen ist in Abschnitt 3.3 zu finden.

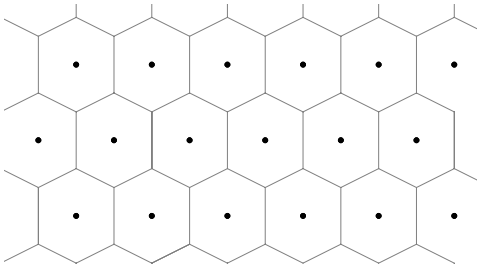


Abbildung 2.7: Voronoi-Zellen als grundlegendes Gebiet eines Gitters.

Grundlegende Gebiete müssen nicht konvex sein und können sogar aus unzusammenhängenden Teilen bestehen. Auch alle volumenerhaltenden Verformungen eines grundlegenden Gebiets sind ebenfalls solche. Wichtig ist nur, dass sie pro Gitterpunkt genau so viel Volumen einnehmen, dass eine Partitionierung durch Verschiebung, bzw. Kopie zu jedem Gitterpunkt hin, entsteht. Zusammenhängend mit der Gitterdeterminante ist somit das Volumen eines grundlegenden Gebiets eine Invariante eines Gitters.

Da die Grundmasche eines Gitters $\mathcal{G}$ per Definition nur genau einen Gitterpunkt enthält, lässt sich damit auch geometrisch bestimmen, ob eine Menge linear unabhängiger Gittervektoren $\mathbf{V} = \{\mathbf{v}_i \mid \mathbf{v}_i \in \mathcal{G} \subset \mathbb{R}^n; 1 \leq i \leq n\}$ eine Basis des Gitters $\mathcal{G}$ bildet. Betrachtet man das Parallelepiped $\mathcal{P}(\mathbf{V})$, dass die Vektoren $\mathbf{v}_i$ aufspannen, ist $\mathbf{V}$ genau dann eine Basis des Gitters $\mathcal{G}$ wenn $\mathcal{P}(\mathbf{V})$ außer dem Nullvektor keinen anderen Gitterpunkt enthält. Wären anderenfalls Gitterpunkte von $\mathcal{P}(\mathbf{V})$ eingeschlossen, so wären diese nicht mit einer ganzzahligen Linearkombination der $\mathbf{v}_i$ erzeugbar (siehe Abbildung 2.9). Als Teilmenge von $\mathcal{G}$ ist $\mathbf{V}$ jedoch eine Basis für ein Teil- bzw. Untergitter von $\mathcal{G}$, wie in Unterabschnitt 2.5.1 näher beschrieben wird. Es ist nun leicht zu sehen, dass, je länger die

Basisvektoren sind, sie umso näher beieinander liegen müssen um auf zwei benachbarte Punkte im Gitter zu zeigen, ohne einen Gitterpunkt einzuschließen.

2.4.3 Modulo-Operation bzgl. der Grundmasche

Mit dem grundlegenden Parallelepiped (Grundmasche) kann eine Operation definiert werden, die einen beliebigen Punkt $\mathbf{x} = a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n \in \text{span}(\mathcal{G}(\mathbf{B}))$ mit $a_i \in \mathbb{R}$ und Gitterbasis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ bezüglich der Grundmasche reduziert. Man bezeichnet

$$\mathbf{x}' = \mathbf{x} \bmod \mathcal{P}(\mathbf{B}) \coloneqq (a_1 \bmod 1)\mathbf{b}_1 + \cdots + (a_n \bmod 1)\mathbf{b}_n \tag{2.4.1}$$

reduziert modulo der Grundmasche $\mathcal{P}(\mathbf{B})$. Ein Punkt $\mathbf{x}$ wird damit auf einen Punkt $\mathbf{x}'$ innerhalb der Grundmasche abgebildet, der die selbe relative Position zum Gitter hat wie $\mathbf{x}$. Das heißt, $\mathbf{x}'$ hat entsprechend den selben Abstand zu den umgebenden Gitterpunkten wie $\mathbf{x}$. Mit dieser Operation lässt sich sehr einfach bestimmen, ob ein Punkt Teil eines Gitters ist. Dies ist genau dann der Fall, wenn der reduzierte Punkt $\mathbf{x}'$ gleich dem Nullpunkt ist – dem einzigen Gitterpunkt innerhalb der Grundmasche. Die eigentliche Arbeit, herauszufinden ob ein Punkt zu einem Gitter gehört, besteht daher im Grunde genommen in der Darstellung des Punkts in der jeweiligen Gitterbasis. Sind in dieser Darstellung alle Koeffizienten des Punkts ganzzahlig (also $a_i \equiv 0 \mod 1,\ \forall i$), gehört der Punkt zum Gitter. Darüber hinaus findet diese Operation in Algorithmen zur Lösung verschiedener Gitterprobleme Anwendung.

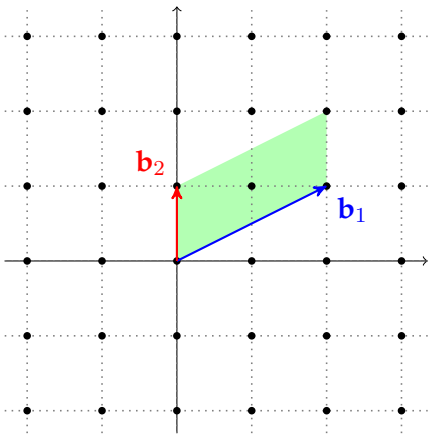


Abbildung 2.9: Die Vektoren $\mathbf{b}_1$ und $\mathbf{b}_2$ bilden keine Basis für das Gitter $\mathbb{Z}^2$, jedoch für das Untergitter $\mathcal{G}' \subset \mathbb{Z}^2$ bei dem die Punkte der Spalten mit ungeradem x-Wert fehlen.

2.5 Gitter als Untergruppe des $\mathbb{R}^n$

Gitter können auch als additive Untergruppe des reellen Vektorraums $\mathbb{R}^n$ definiert werden. Diese Betrachtungsweise erleichtert das Verständnis der später folgenden Sätze über Gitter, in denen gruppentheoretische Eigenschaften von Gittern genutzt werden.

In der eingangs gegebenen Definition liegt einem Gitter eine Basis aus linear unabhängigen Vektoren $\mathbf{v}_i \in \mathbb{R}^n$ zugrunde, dessen ganzzahlige Linearkombinationen das Gitter ergeben ($\Lambda = \mathbb{Z}\mathbf{v}_1 + \cdots + \mathbb{Z}\mathbf{v}_m$ (wobei allgemein $m \leq n$)). Gitter lassen sich allerdings auch ohne Angabe einer Basis als diskrete, nicht-leere Untergruppe des $\mathbb{R}^n$, die unter Subtraktion abgeschlossen ist, definieren. *Diskret* meint hier die Eigenschaft, dass je zwei Punkte $\mathbf{x} \neq \mathbf{y} \in \Lambda$ einen reellwertigen Mindestabstand $\epsilon > 0$ haben. Das heißt, dass jeder Punkt im Gitter von einer (n-dimensionalen) Kugel umgeben ist, in der kein weiterer Gitterpunkt liegt. Diese Eigenschaft ist notwendig, damit eine endlich erzeugte Untergruppe von $\mathbb{R}^n$ ein Gitter ist, denn nicht jede Untergruppe des Vektorraums $\mathbb{R}^n$ ist ein Gitter, wie Neukirch in [Neu92, Kap. 1, S. 26] beweist:

Satz 2.10. *Eine Untergruppe $\Lambda \subset \mathbb{R}^n$ ist genau dann ein Gitter, wenn sie diskret ist.*

Ein Beispiel für eine Gitterdefinition ohne Angabe einer konkreten Basis ist

$$\Lambda = \left\{ \mathbf{x} \in \mathbb{Z}^n \;\middle|\; \sum_{i=1}^{n} x_i \in 2\mathbb{Z} \right\}.$$

Die zweidimensionale Form dieses Gitters ist in Abbildung 2.10 dargestellt. Auch wenn Gitter ohne Angabe einer Basis definiert werden, lässt sich dennoch immer eine Basis zu einem Gitter finden (vgl. [Sch18, Kap. 1.3]).

Die Betrachtung von Gittern als Untergruppe lässt unmittelbar die folgenden Eigenschaften erkennen. Der Ursprungspunkt ist immer in einem Gitter enthalten ($\mathbf{0} = \mathbf{x} - \mathbf{x}$). Des Weiteren ist für jedes $\mathbf{x} \in \mathcal{G}$ auch $-\mathbf{x} \in \mathcal{G}$ enthalten. Daraus folgt, dass es immer mindestens zwei Vektoren gleicher Länge in einem Gitter geben muss.

2.5.1 Untergitter und Teilgitter

Die Betrachtung von Gittern als Gruppe, legt nahe, dass es auch Untergruppen bzw. Untergitter von Gittern gibt. Dabei können Untergitter als eine Teilmenge eines dichteren Gitters identifiziert werden. Die Grundmasche eines solchen Untergitters ist dementsprechend größer, da mehr Raum zwischen den Punkten besteht.

Definition 2.11. Seien $\mathcal{G}$ und $\mathcal{G}'$ Gitter mit $\mathcal{G}' \subseteq \mathcal{G} \subset \mathbb{R}^n$. $\mathcal{G}'$ heißt Untergitter von $\mathcal{G}$ wenn $\mathcal{G}$ und $\mathcal{G}'$ denselben Rang haben und Teilgitter von $\mathcal{G}$ wenn $\text{Rang}(\mathcal{G}') < \text{Rang}(\mathcal{G})$.

Abbildung 2.10 zeigt beispielhaft ein Untergitter von $\mathbb{Z}^2$. Es handelt sich um das sogenannte Schachbrettgitter, bei dem im Vergleich zu $\mathbb{Z}^2$ jedem Gitterpunkt die direkten Nachbarn in den Richtungen oben, unten, rechts und links fehlen. Genauer gesagt sind die Punkte, deren Summe der Koordinaten ungerade sind, nicht enthalten.

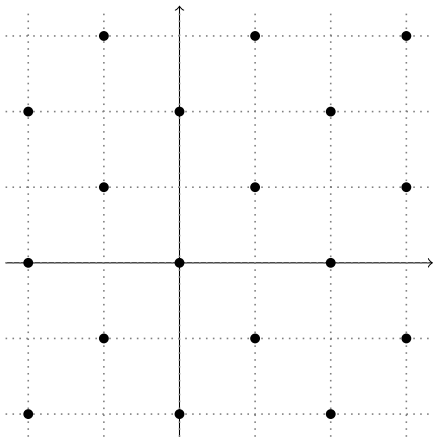


Abbildung 2.10: Untergitter $\mathcal{G}' \subset \mathbb{Z}^2$ mit vollem Rang. Für alle $\mathbf{g} \in \mathcal{G}'$ gilt $g_1 + g_2 \in 2\mathbb{Z}$.

2.6 Gram-Schmidt-Orthogonalisierung

Das Gram-Schmidt-Orthogonalisierungsverfahren hat im Bezug auf Gitter nützliche Eigenschaften, die manche Berechnungen erleichtern und eine untere Abschätzung der kürzesten Vektoren in einem Gitter liefern (siehe Unterabschnitt 2.7.2). Außerdem ist es Grundlage von Algorithmen zur Approximierung von Gitterproblemen, wie in Abschnitt 3.3 erklärt wird. Bei diesem Verfahren wird aus einer Menge linear unabhängiger Vektoren $\mathbf{b}_1, \ldots, \mathbf{b}_n$ eine Menge orthogonaler Vektoren $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n$ generiert.

Definition 2.12 (Gram-Schmidt-Orthogonalisierung)**.** Für eine Menge linear unabhängiger Vektoren $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^n$ ist ihre Gram-Schmidt-Orthogonalisierung $(\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n)$ folgendermaßen definiert:

$$\tilde{\mathbf{b}}_1 = \mathbf{b}_1$$
$$\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{\mathbf{b}}_j \quad \text{wobei } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle}$$

Die Koeffizienten $\mu_{i,j}$ werden Gram-Schmidt-Koeffizienten genannt.

Mit dem Verfahren erhält man schrittweise zueinander orthogonale Vektoren, wobei die Vektoren nacheinander auf den Raum projiziert werden, der orthogonal zur linearen Hülle der vorherigen Vektoren ist. Der erste Vektor $\mathbf{b}_1$ wird dabei als Startreferenz gewählt. Generell kann die Reihenfolge verändert werden, was hingegen zu einer anderen Orthogonalisierung führt. Die nicht-senkrechten Anteile von $\mathbf{b}_i$ bezogen auf den $(i-1)$-dimensionalen Raum, den die schon orthogonaliserten Vektoren $\mathbf{b}_j$ mit $1 \leq j \leq i-1$ aufspannen, werden im i-ten Schritt des Gram-Schmidt-Verfahrens subtrahiert. Somit erhält man jeweils den Anteil $\tilde{\mathbf{b}}_i$ von $\mathbf{b}_i$, welcher orthogonal zu $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_{i-1}$ ist.

Anzumerken ist, dass für alle $1 \leq i \leq n$, $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_i$ den selben Raum aufspannen wie $\mathbf{b}_1, \ldots, \mathbf{b}_i$. Also bilden die $\tilde{\mathbf{b}}_i$ eine Orthogonalbasis von $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, jedoch im

Allgemeinen keine Gitterbasis des Gitters $\mathcal{G}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Sie liegen darüber hinaus nicht unbedingt in $\mathcal{G}$, sind also keine Gitterpunkte. Abbildung 2.11 zeigt dies beispielhaft.

Zusätzlich zur Orthogonalisierung können die Gram-Schmidt-Vektoren $\tilde{\mathbf{b}}_i$ normiert werden ($\tilde{\mathbf{b}}_i / \|\tilde{\mathbf{b}}_i\|$). Der gesamte Orthogonalisierungsprozess kann nun in Matrix-Form dargestellt werden. Hierbei werden die Basisvektoren $\mathbf{b}_i$ wie üblich als Spaltenvektoren der Matrix $\mathbf{B}$ geschrieben.

$$\begin{aligned}
\mathbf{B} &= \begin{pmatrix} b_{1,1} & & b_{n,1} \\ \vdots & \cdots & \vdots \\ b_{1,n} & & b_{n,n} \end{pmatrix} = \begin{pmatrix} \tilde{b}_{1,1} & & \tilde{b}_{n,1} \\ \vdots & \cdots & \vdots \\ \tilde{b}_{1,n} & & \tilde{b}_{n,n} \end{pmatrix} \cdot \begin{pmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \cdots & \mu_{n,1} \\ 0 & 1 & \mu_{3,2} & \cdots & \mu_{n,2} \\ 0 & 0 & 1 & \cdots & \mu_{n,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} \frac{\tilde{b}_{1,1}}{\|\tilde{\mathbf{b}}_1\|} & & \frac{\tilde{b}_{n,1}}{\|\tilde{\mathbf{b}}_n\|} \\ \vdots & \cdots & \vdots \\ \frac{\tilde{b}_{1,n}}{\|\tilde{\mathbf{b}}_1\|} & & \frac{\tilde{b}_{n,n}}{\|\tilde{\mathbf{b}}_n\|} \end{pmatrix}}_{\mathbf{Q}} \cdot \underbrace{\begin{pmatrix} \|\tilde{\mathbf{b}}_1\| & \mu_{2,1}\|\tilde{\mathbf{b}}_1\| & \mu_{3,1}\|\tilde{\mathbf{b}}_1\| & \cdots & \mu_{n,1}\|\tilde{\mathbf{b}}_1\| \\ 0 & \|\tilde{\mathbf{b}}_2\| & \mu_{3,2}\|\tilde{\mathbf{b}}_2\| & \cdots & \mu_{n,2}\|\tilde{\mathbf{b}}_2\| \\ 0 & 0 & \|\tilde{\mathbf{b}}_3\| & \cdots & \mu_{n,3}\|\tilde{\mathbf{b}}_3\| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \|\tilde{\mathbf{b}}_n\| \end{pmatrix}}_{\mathbf{R}}
\end{aligned} \tag{2.6.1}$$

Man erhält mit dem Gram-Schmidt-Verfahren somit auch eine **QR**-Zerlegung der Basismatrix $\mathbf{B} = \mathbf{QR}$, mit orthogonaler Matrix $\mathbf{Q}$ und oberer Dreiecksmatrix $\mathbf{R}$. Dabei entspricht $\mathbf{R}$ der Darstellung der Gitterbasisvektoren $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in der Orthonormalbasis (ONB) $\mathbf{Q} = (\tilde{\mathbf{b}}_1/\|\tilde{\mathbf{b}}_1\|, \ldots, \tilde{\mathbf{b}}_n/\|\tilde{\mathbf{b}}_n\|)$. Wie in Beispiel 2.1 beschrieben, kann man diese Darstellung der Gitterbasis als Drehung und/oder Spiegelung des Gitters im Raum betrachten.

Beispiel 2.1. Abbildung 2.11 verdeutlicht anschaulich, dass die Darstellung der Gittervektoren $\mathbf{b}_1$ und $\mathbf{b}_2$ in einer ONB, einer Drehung des Koordinatensystems entspricht. Für die Gitterbasis $\mathbf{B} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ ist ihre Orthogonalisierung $\tilde{\mathbf{B}} = \left(\begin{smallmatrix} 1 & -1/2 \\ 1 & 1/2 \end{smallmatrix}\right)$ und die entsprechende ONB $\mathbf{Q} = \left(\begin{smallmatrix} 1/\sqrt{2} & -\sqrt{2}/2 \\ 1/\sqrt{2} & \sqrt{2}/2 \end{smallmatrix}\right)$. Dann ist

$$\begin{aligned}
\mathbf{B} &= \mathbf{QR} \\
\Leftrightarrow \mathbf{R} &= \mathbf{Q}^{-1}\mathbf{B} \\
&= \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} \sqrt{2} & \frac{3}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \end{pmatrix}.
\end{aligned}$$

Eine Drehung des Raums verändert die Längen und Volumen von Vektoren nicht, was sich unter anderem für die Berechnung der Gitterdeterminante und die Verwendung

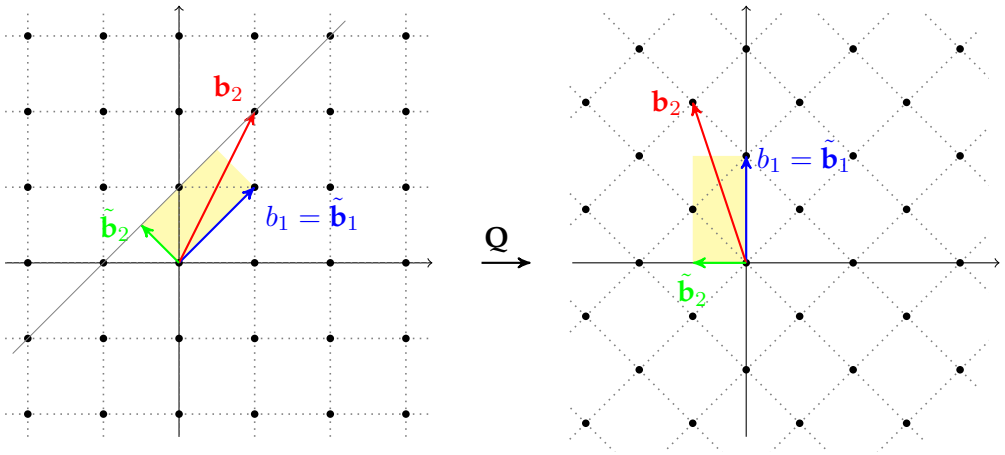


Abbildung 2.11: Gram-Schmidt-Orthogonalisierung der Basisvektoren $\mathbf{b}_1$ und $\mathbf{b}_2$. Die Orthogonalisierung liefert ein rechteckiges grundlegendes Gebiet des Gitters. Die Darstellung der ursprünglichen Gitterbasis in der zugehörigen ONB kann als Drehung des Gitters verstanden werden.

in grundlegenden Sätzen über Gitter (siehe Abschnitt 2.7) als nützlich erweist. Da die Vektoren $(\tilde{\mathbf{b}}_i/\|\tilde{\mathbf{b}}_i\|)$ orthonormal sind, ist der Wert der Determinante von Matrix $\mathbf{Q}$ gleich 1. Daraus folgt, dass die Determinante der Matrix $\mathbf{B}$ gleich der Determinante der Matrix $\mathbf{R}$ in Gleichung 2.6.1 ist. Also gilt

$$\det(\mathcal{G}(\mathbf{B})) = \prod_{i=1}^{n} \|\tilde{\mathbf{b}}_i\|. \tag{2.6.2}$$

Der Orthogonalisierungsprozess erhält somit das Volumen des grundlegenden Parallelepipeds $\mathcal{P}(\mathbf{B})$ und liefert eine orthogonalisierte Form desselben, welches ebenfalls ein grundlegendes Gebiet des Gitters ist.

2.7 Kürzeste Vektoren und sukzessive Minima eines Gitters

Die Frage, welche Vektoren, abgesehen vom Nullvektor, in einem Gitter die kürzesten sind (*Shortest Vector Problem* (*SVP*)), ist von fundamentaler Bedeutung in der gitterbasierten Kryptographie. Viele gitterbasierte kryptographische Verfahren basieren auf diesem Problem. Dieses und weitere ähnliche Probleme werden im nachfolgenden Kapitel 3 dargelegt. Zunächst sollen allgemeine, grundlegende Erkenntnisse zu kurzen Vektoren in Gittern vorgestellt werden.

Schon lange bevor man über kryptographische Anwendungen von Gittern nachdachte, versuchten Mathematiker Schranken für die Länge der kürzesten Vektoren in einem Gitter zu finden. Dies gelang Hermann Minkowski 1896 in seinem Werk *Geometrie der Zahlen* [Min96]. Dafür führte er das Konzept der sukzessiven Minima ein. Dies sind grundlegende Konstanten eines Gitters, die die minimalen Längen von Gittervektoren

beschreiben.

Definition 2.13 (Sukzessive Minima)**.** Sei $\mathcal{G}$ ein vollständiges Gitter mit Rang $n \in \mathbb{N}$ und $\mathcal{K}_n(\mathbf{0}, r) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| < r\}$ die n-dimensionale, offene, $\mathbf{0}$-symmetrische Kugel mit Radius r. Das i-te sukzessive Minimum des Gitters $\mathcal{G}$ ist der Radius der kleinsten Kugel $\mathcal{K}_n(\mathbf{0}, r)$ mit Mittelpunkt $\mathbf{0}$, welche i linear unabhängige Gittervektoren enthält [MG02].

$$\lambda_i(\mathcal{G}) = \inf\{r > 0 \mid \dim(\operatorname{span}(\mathcal{G} \cap \mathcal{K}_n(\mathbf{0}, r))) \geq i\},$$

beziehungsweise, äquivalent [Sch18]

$$\lambda_i(\mathcal{G}) = \inf\left\{ r > 0 \;\middle|\; \begin{array}{l} \exists \text{ linear unabhängige } \mathbf{g}_1, \ldots, \mathbf{g}_i \in \mathcal{G} \\ \text{mit } \|\mathbf{g}_1\|, \ldots, \|\mathbf{g}_i\| \leq r \end{array} \right\}.$$

Es kann gezeigt werden, dass es in jedem Gitter Vektoren gibt, die den sukzessiven Minima entsprechen.

Theorem 2.14. *Sei $\mathcal{G}$ ein vollständiges n-dimensionales Gitter mit sukzessiven Minima $\lambda_1, \ldots, \lambda_n$. Dann gibt es linear unabhängige $\mathbf{g}_i \in \mathcal{G}$ mit $\|\mathbf{g}_i\| = \lambda_i(\mathcal{G})$ für alle $i \in \{1, \ldots, n\}$.*

Beweis. Dies folgt aus der Charakterisierung von Gittern als additive Untergruppe des $\mathbb{R}^n$. Ein ausführlicher Beweis ist in [MG02, S. 9f] zu finden. □

Somit ist das Infimum in Lemma 2.13 praktisch ein Minimum, wenn man die offene Kugel $\mathcal{K}_n(\mathbf{0}, r)$ mit der geschlossenen Kugel $\overline{\mathcal{K}}_n(\mathbf{0}, r) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq r\}$ ersetzt. Das erste sukzessive Minimum $\lambda_1(\mathcal{G})$ beschreibt also die Länge der kürzesten, von $\mathbf{0}$ verschiedenen Vektoren in $\mathcal{G}$. Sei $\mathbf{g}_1 \in \mathcal{G}$ mit $\|\mathbf{g}_1\| = \lambda_1$, so gilt also $\|\mathbf{g}_1\| \leq \|\mathbf{g}\|\ \forall \mathbf{g} \neq \mathbf{0} \in \mathcal{G}$. Wie schon erwähnt, gibt es davon mindestens zwei, da ein Gitter als additive Untergruppe zu jedem $\mathbf{g} \in \mathcal{G}$ auch $-\mathbf{g}$ enthält, welches dieselbe Länge hat. Abbildung 2.12 verdeutlicht dies. Es kann aber auch bis zu 2^n kürzeste Vektoren in einem Gitter geben, sodass also $\lambda_1 = \lambda_2 = \cdots = \lambda_n$ gilt. So zum Beispiel in $\mathbb{Z}^n$, wo die kürzesten Vektoren in Richtung der i-ten Dimension jeweils dieselbe Länge haben wie in allen n Dimensionen.

Als Länge der kürzesten Vektoren in $\mathcal{G}$ ist λ_1 ebenfalls der minimale Abstand zwischen zwei Vektoren:

$$\lambda_1(\mathcal{G}) = \min_{\mathbf{g} \in \mathcal{G} \setminus \{\mathbf{0}\}} \|\mathbf{g}\| = \min_{\mathbf{g} \neq \mathbf{h} \in \mathcal{G}} \|\mathbf{g} - \mathbf{h}\|.$$

2.7.1 Reduzierte Gitterbasen

Im nachfolgenden Kapitel werden Algorithmen zur Basisreduktion von Gittern beschrieben. Ziel ist es, dabei möglichst kurze Basisvektoren eines Gitters zu erhalten. Die sukzessiven Minima sind dabei ein Maß für die „Reduziertheit“ einer Basis. Je kleiner der Quotient $\|\mathbf{b}_i\|/\lambda_i$ (für $i = 1, \ldots, n$) ist, desto reduzierter ist die Basis. Reduzierte Basen bestehen aus nahezu orthogonalen Vektoren. Als Maß für die Orthogonalität, bzw. den sogenannten orthogonalen Defekt einer Basis kann die Hadamard-Ungleichung heran gezogen werden.

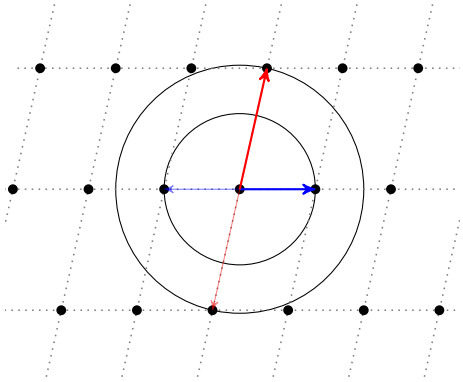


Abbildung 2.12: Sukzessive Minima $\lambda_1(\mathcal{G}) = 1$ (blau) und $\lambda_2(\mathcal{G}) = 1,64$ (rot).

Definition 2.15 (Hadamard-Ungleichung)**.** Sei $\mathbf{M}$ eine $n \times n$ Matrix mit Koeffizienten aus den komplexen Zahlen und Spaltenvektoren $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Dann gilt für die euklidische Norm:

$$|\det(\mathbf{M})| \leq \prod_{i=1}^{n} \|\mathbf{v}_i\|. \tag{2.7.1}$$

Bilden die Vektoren $\mathbf{v}_i, \ldots, \mathbf{v}_n$ eine orthogonale Basis des $\mathbb{R}^n$, so gilt Gleichheit in 2.7.1 (siehe dazu auch Gleichung 2.6.2). Wenn man also das Produkt der Längen der Basisvektoren mit dem Volumen des von ihnen aufgespannten Parallelepipeds vergleicht (vgl. Abschnitt 2.4), erhält man ein Maß für den orthogonalen Defekt der Basis. Umso näher der Quotient

$$\frac{\prod_{i=1}^{n} \|\mathbf{v}_i\|}{|\det(\mathbf{M})|}$$

der 1 kommt, desto geringer ist der orthogonale Defekt von $\mathbf{M}$.

Ist die Basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ eines Gitters eine Orthogonalbasis, so gilt außerdem $\|\mathbf{b}_i\| = \lambda_i$. An dem Gitter in Abbildung 2.12 lässt sich erkennen, dass die Umkehrung dieser Aussage jedoch nicht gilt.

Interessanterweise gibt es Gitter, für die keine Basis existiert bei der für alle $i \in \{1, \ldots, n\}$ gilt, dass $\mathbf{b}_i = \lambda_i$. Zwar gibt es in solchen Gittern n linear unabhängige $\mathbf{g}_i$ mit den Längen der sukzessiven Minima, aber keine Basis die nicht mindestens einen Vektor enthält, der strikt länger ist als λ_n. Ein anschauliches Beispiel dafür wird in [MG02, S. 126] gegeben.

2.7.2 Untere Schranke für λ_1

Da ein Gitter eine diskrete Untergruppe ist, muss es eine gitterspezifische untere Schranke für die Länge der kürzesten Vektoren λ_1 geben.

Theorem 2.16. *Sei $\mathcal{G} \subset \mathbb{R}^n$ ein vollständiges Gitter mit Basis* $\mathbf{B}$ *und entsprechender Gram-Schmidt-Orthogonalisierung* $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n)$. *Dann gilt*

$$\lambda_1(\mathcal{G}(\mathbf{B})) \geq \min_{i=1,\ldots,n} \left\|\tilde{b_i}\right\| > 0.$$

Beweis. Sei $\mathbf{x} \neq \mathbf{0} \in \mathbb{Z}^n$ ein beliebiger ganzzahliger Vektor. Zu zeigen ist, dass für alle Gittervektoren $\mathbf{Bx} \in \mathcal{G}(B)$ gilt, dass $\|\mathbf{Bx}\| \geq \min\left\|\tilde{\mathbf{b}}_i\right\|$. Sei $j \in \{1, \ldots, n\}$ das größte j, sodass $x_j \neq 0$. Dann gilt

$$\begin{aligned}
\left|\left\langle \mathbf{Bx}, \tilde{\mathbf{b}}_j \right\rangle\right| &= \left|\left\langle \sum_{i=1}^{j} x_i \mathbf{b}_i, \tilde{\mathbf{b}}_j \right\rangle\right| \\
&= \left|\sum_{i=1}^{j} x_i \left\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \right\rangle\right| \\
&= |x_j| \cdot \left\langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \right\rangle \quad \left(\text{da } \left\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \right\rangle = 0 \text{ für alle } i < j \text{ und } \left\langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \right\rangle = \left\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \right\rangle\right) \\
&= |x_j| \cdot \left\|\tilde{\mathbf{b}}_j\right\|^2.
\end{aligned}$$

Die *Cauchy-Schwarz Ungleichung* liefert

$$\left|\left\langle \mathbf{Bx}, \tilde{\mathbf{b}}_j \right\rangle\right| \leq \|\mathbf{Bx}\| \cdot \left\|\tilde{\mathbf{b}}_j\right\|$$

und zusammen erhält man

$$\|\mathbf{Bx}\| \geq \frac{\left|\left\langle \mathbf{Bx}, \tilde{\mathbf{b}}_j \right\rangle\right|}{\left\|\tilde{\mathbf{b}}_j\right\|} \geq |x_j| \cdot \left\|\tilde{\mathbf{b}}_j\right\| \geq \min_{i=1,\ldots,n} \left\|\tilde{b_i}\right\| > 0.$$

Die Länge jedes Gittervektors ist also mindestens $\min_i \|\tilde{\mathbf{b}}_i\|$ und da die Vektoren $\mathbf{b}_i, \ldots, \mathbf{b}_n$ und somit auch $\tilde{\mathbf{b}}_i, \ldots, \tilde{\mathbf{b}}_n$ linear unabhängig sind, ist sie strikt positiv. □

2.7.3 Obere Schranke für λ_1

Nun soll eine obere Schranke für die Länge der kürzesten Vektoren in einem Gitter angegeben werden. Minkowski beschäftigte sich mit der Frage, wie groß das Volumen eines n-dimensionalen kompakten und konvexen Körpers im $\mathbb{R}^n$ sein kann, dass nicht zwingend ein weiterer Gitterpunkt (außer dem Nullpunkt) des ganzzahligen Gitters $\mathbb{Z}^n$ enthalten sein muss. Er kam zunächst zu dem leicht nachvollziehbaren Ergebnis, dass ein (offener) Würfel mit Kantenlänge 2, der sich symmetrisch zum Nullpunkt befindet, keinen Gitterpunkt außer $\mathbf{0}$ enthält. Dieses Ergebnis entwickelte er weiter und formulierte zwei Theoreme, die heute in der Komplexitätsanalyse von Gittern im Bezug auf Kryptographie eine wichtige Rolle spielen. Das Theorem von H. F. Blichfeldt über Gitter erleichtert das Verständnis der nachfolgenden Theoreme von Minkowski.

Lemma 2.17 (Blichfeldt 1914 nach [Sch18]). *Für ein beliebiges vollständiges Gitter $\mathcal{G}$ und eine kompakte Menge $S \subseteq \text{span}(\mathcal{G})$ mit Volumen $\text{vol}(S) > \det(\mathcal{G})$ gibt es zwei verschiedene Punkte $s_1, s_2 \in S$, sodass $s_1 - s_2 \in \mathcal{G}$.*

Beweis. Sei $\mathcal{G}$ ein Gitter mit Basis $\mathbf{B}$ und $S \subset \text{span}(\mathcal{G})$ eine kompakte Menge mit $\text{vol}(S) > \det(\mathcal{G})$. Mit der Grundmasche $\mathcal{P}(\mathbf{B})$ des Gitters lässt sich der Raum $\text{span}(\mathcal{G}) \simeq \mathbb{R}^n$ partitionieren. D.h. $\mathbb{R}^n = \bigcup_{\mathbf{g}\in\mathcal{G}} \mathbf{g} + \mathcal{P}(\mathbf{B}) = \{\mathbf{g} + \mathbf{p} \mid \mathbf{g} \in \mathcal{G} \text{ und } \mathbf{p} \in \mathcal{P}(\mathbf{B})\}$. Sei nun S_g der Anteil von S, der in dem Parallelepiped $\mathbf{g} + \mathcal{P}(\mathbf{B})$ liegt, also $S_g = S \cap (\mathbf{g} + \mathcal{P}(\mathbf{B}))$. Da $S = \bigcup_{\mathbf{g}\in\mathcal{G}} S_g$, folgt $\text{vol}(S) = \sum_{\mathbf{g}\in\mathcal{G}} \text{vol}(S_g)$. Sei $\widehat{S}_g = S_g - \mathbf{g}$. Dann ist $\widehat{S}_g \subseteq \mathcal{P}(\mathbf{B})$ und $\text{vol}(\widehat{S}_g) = \text{vol}(S_g)$ und man schließt

$$\sum_{\mathbf{g}\in\mathcal{G}} \text{vol}(\widehat{S}_g) = \sum_{\mathbf{g}\in\mathcal{G}} \text{vol}(S_g) = \text{vol}(S) > \text{vol}(\mathcal{P}(\mathbf{B})).$$

Es müssen somit zwei $\mathbf{g}, \mathbf{h} \in \mathcal{G}, \mathbf{g} \neq \mathbf{h}$ existieren, sodass $\widehat{S}_g \cap \widehat{S}_h \neq \emptyset$. Für einen Punkt $\mathbf{s} \in \widehat{S}_g \cap \widehat{S}_h$ gilt nun $\mathbf{s}+\mathbf{g} \in S_g \subseteq S$ und $\mathbf{s}+\mathbf{h} \in S_h \subseteq S$. Die Differenz $(\mathbf{s}+\mathbf{g})-(\mathbf{s}+\mathbf{h}) = \mathbf{g}-\mathbf{h}$ ist damit, aufgrund der Gruppeneigenschaften von $\mathcal{G}$ ebenfalls im Gitter enthalten. □

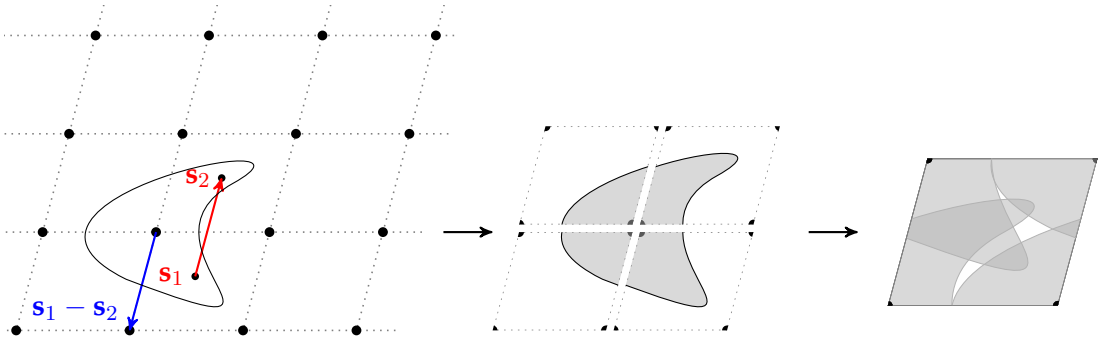


Abbildung 2.13: Theorem von Blichfeldt (1914). Beweisidee: Zerlegung von S und Übereinanderlegen auf die Grundmasche.

Das Argument von Lemma 2.17 ist, dass eine kompakte Menge, die ein größeres Volumen als die Grundmasche eines Gitters hat, zwei Punkte enthalten muss, deren Differenz ein Gitterpunkt ist. Im Beweis wird S auf die Grundmasche projiziert und somit ersichtlich, dass sich Teile von S überlappen müssen, da $\text{vol}(S) > \det(\mathcal{G}) = \text{vol}(\mathcal{P}(\mathbf{B}))$ (siehe Abbildung 2.13). Daraus folgt wiederum die Aussage. Wie Abbildung 2.14 veranschaulicht, muss die Menge S dabei selbst keinen Gitterpunkt enthalten.

Minkowskis Theoreme

Aus Blichfeldts Lemma 2.17 kann leicht auf *Minkowskis Gitterpunktsatz* geschlossen werden und wiederum auf sein sogenanntes erstes und zweites Theorem.

Definition 2.18. Eine Menge S heißt *konvex*, wenn für jedes $x, y \in S$ und $\lambda \in [0, 1]$ gilt, dass $\lambda x + (1 - \lambda)y \in S$. Wenn also zwei Punkte in einem konvexen S liegen, sind auch

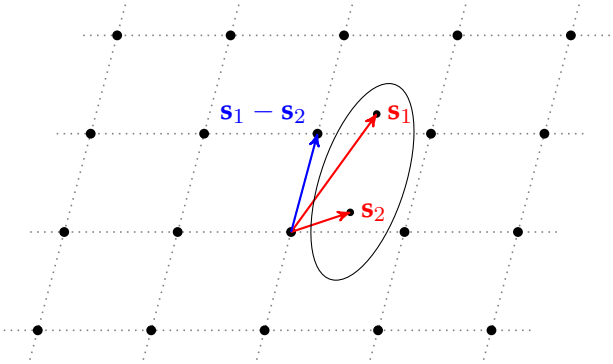


Abbildung 2.14: Theorem von Blichfeldt (1914): S selbst muss keinen Gitterpunkt enthalten.

alle Punkte auf der Verbindungsstrecke zwischen den beiden Punkten in S enthalten.

Theorem 2.19 (Minkowskis Gitterpunktsatz)**.** *Sei $\mathcal{G}$ ein vollständiges Gitter mit Rang n und S eine Menge, die beschränkt, $\mathbf{0}$-symmetrisch und konvex ist. Wenn* $\mathrm{vol}(S) > 2^n \det(\mathcal{G})$ *ist, dann enthält S mindestens zwei von $\mathbf{0}$ verschiedene Gitterpunkte.*

Beweis. Sei $S' = \frac{1}{2}S \coloneqq \{x \mid 2x \in S\}$. $\mathrm{vol}(S') = 2^{-n} \mathrm{vol}(S) > \det(\mathcal{G})$. Nach Blichfeldts Lemma 2.17 gibt es $\mathbf{s}_1, \mathbf{s}_2 \in S'$, sodass $\mathbf{s}_1 - \mathbf{z}_2 \in \mathcal{G}$ ein von $\mathbf{0}$ verschiedener Gitterpunkt ist. Per Definition sind $2\mathbf{s}_1$ und $2\mathbf{s}_2$ in S enthalten und da S $\mathbf{0}$-symmetrisch ist, liegt auch $-2\mathbf{z}_2$ in S. Da S zusätzlich konvex ist, gilt $\frac{2\mathbf{z}_1 - 2\mathbf{z}_2}{2} = \mathbf{z}_1 - \mathbf{z}_2 \in S$ (siehe Abbildung 2.15). Ebenso ist aufgrund der Symmetrie $-(\mathbf{z}_1 - \mathbf{z}_2) \in S$. □

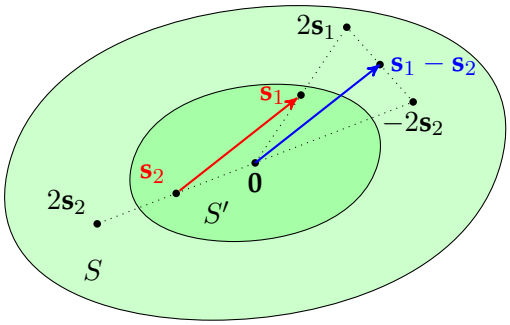


Abbildung 2.15: Minkowskis Gitterpunktsatz

Aus Minkowskis Gitterpunktsatz lässt sich nun ein Korollar ableiten, was als *Minkowskis erstes Theorem* bekannt ist.

Korollar 2.19.1 (Minkowskis erstes Theorem)**.** *Für ein beliebiges vollständiges Gitter $\mathcal{G}$ mit Rang n gilt*

$$\lambda_1(\mathcal{G}) \leq \sqrt{n} \cdot \det(\mathcal{G})^{1/n}.$$

Beweis. Die n-dimensionale Kugel $\overline{\mathcal{K}}_n(\mathbf{0}, r)$ mit Mittelpunkt $\mathbf{0}$ und Radius r hat mindestens ein Volumen $\text{vol}(\mathcal{K}_n(\mathbf{0}, r)) \geq \left(\frac{2r}{\sqrt{n}}\right)^n$, denn sie enthält einen Würfel mit Seitenlänge $(2r/\sqrt{n})^n$:

$$\left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \forall i, |x_i| < \frac{r}{\sqrt{n}} \right\} \subseteq \mathcal{K}_n(\mathbf{0}, r).$$

Betrachtet man nun die offene Kugel $\mathcal{K}_n(\mathbf{0}, \lambda_1(\mathcal{G})) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| < \lambda_1(\mathcal{G})\}$, die per Definition keinen Gitterpunkt außer $\mathbf{0}$ enthält, erhält man

$$\left(\frac{2\lambda_1(\mathcal{G})}{\sqrt{n}}\right)^n \leq \text{vol}(\tilde{\mathcal{K}}_n(\mathbf{0}, \lambda_1(\mathcal{G}))) \leq 2^n \det(\mathcal{G}).$$

Die letzte Ungleichung folgt hierbei aus Minkowskis Gitterpunktsatz (2.19). Durch Umstellen erhält man die zu beweisende Aussage:

$$\begin{aligned} & \left(\frac{2\lambda_1(\mathcal{G})}{\sqrt{n}}\right)^n && \leq 2^n \det(\mathcal{G}) \\ \Leftrightarrow \quad & \frac{2\lambda_1(\mathcal{G})}{\sqrt{n}} && \leq 2 \det(\mathcal{G})^{1/n} \\ \Leftrightarrow \quad & \lambda_1(\mathcal{G}) && \leq \sqrt{n} \cdot \det(\mathcal{G})^{1/n}. \end{aligned}$$

□

Man beachte, dass diese Eigenschaft korrekt skaliert. Ein skaliertes Gitter $c\mathcal{G}$ mit erstem sukzessiven Minimum $c\lambda_1(\mathcal{G})$ hat die Determinante $c^n \det(\mathcal{G})$, sodass die Abschätzung ebenfalls mit c skaliert. Diese Abschätzung der kürzesten Länge von Gittervektoren in einem Gitter kann sehr grob sein. Dazu kann beispielsweise ein zweidimensionales Gitter $\mathcal{G}(\mathbf{B})$ mit $\mathbf{B} = \begin{pmatrix} \epsilon & 0 \\ 0 & 1/\epsilon \end{pmatrix}$ und kleinem $\epsilon > 0$, etwa $\epsilon = 2^{-42}$, betrachtet werden. Dann ist $\det(\mathcal{G}) = 1$ und Minkowskis Theorem liefert die obere Schranke $\sqrt{2}$ für das erste sukzessive Minimum. Jedoch ist in diesem Beispiel $\lambda_1(\mathcal{G}) = \epsilon = 2^{-42} \ll \sqrt{2}$.

Andererseits kann im Allgemeinen, bis auf konstante Faktoren, keine kleinere Schranke für λ_1 gefunden werden, denn es gibt unendlich viele n-dimensionale Gitter mit Determinante 1 für die $\lambda_1(\mathcal{G}) \geq C\sqrt{n}$ gilt, wobei $C \approx \sqrt{1/\pi e} \approx 0,3422$ (vgl. [Pei15b, Lec. 1]).

Minkowski erweiterte Korollar 2.19.1 und zeigte, dass $\sqrt{n} \cdot \det(\mathcal{G})^{1/n}$ nicht nur obere Schranke für $\lambda_1(\mathcal{G})$ ist, sondern auch für das (geometrische) Mittel aller λ_i.

Theorem 2.20 (Minkowskis zweites Theorem)**.**

$$\left(\prod_{i=1}^{n} \lambda_i(\mathcal{G})\right)^{1/n} \leq \sqrt{n} \cdot \det(\mathcal{G})^{1/n}$$

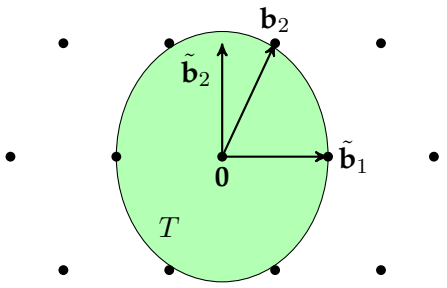


Abbildung 2.16: Darstellung des Ellipsoiden T. Die eingefärbte Fläche gehört zu T, nicht jedoch der schwarze Rand. $\mathbf{b}_1$ liegt genau auf dem Rand um T und $\mathbf{b}_2$ liegt außerhalb des Randes.

Beweis (*nach* [Reg04]). Seien $\mathbf{b}_1, \ldots, \mathbf{b}_n$ linear unabhängige Gittervektoren in einem vollständigen Gitter $\mathcal{G}$, die jeweils die Länge der sukzessiven Minima haben: $\|\mathbf{b}_i\| = \lambda_i(\mathcal{G})$ für $1 \leq i \leq n$. Seien $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n$ die entsprechend Gram-Schmidt-orthogonalisierten Vektoren. Man betrachte den Ellipsoid

$$T = \left\{ \mathbf{y} \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\| \cdot \lambda_i} \right)^2 < 1 \right\}.$$

Zu zeigen ist, dass T keinen von $\mathbf{0}$ verschiedenen Gitterpunkt enthält (siehe dazu Abbildung 2.16). Sei nun $\mathbf{y} \neq \mathbf{0} \in \mathcal{G}$ und $k \in \{1, \ldots, n\}$ maximal, sodass $\|\mathbf{y}\| \geq \lambda_k(\mathcal{G})$. $\mathbf{y}$ muss in $\operatorname{span}(\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_k) = \operatorname{span}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ liegen, da ansonsten $\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{y}$ $k+1$ linear unabhängige Vektoren wären, deren Länge kleiner als $\lambda_{k+1}(\mathcal{G})$ wäre. Dies widerspräche der Definition der sukzessiven Minima $\lambda_i(\mathcal{G})$. Jetzt gilt

$$\sum_{i=1}^{n} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\| \cdot \lambda_i} \right)^2 = \sum_{i=1}^{k} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\| \cdot \lambda_i} \right)^2 \geq \frac{1}{\lambda_k^2} \sum_{i=1}^{k} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\|} \right)^2 = \frac{\|\mathbf{y}\|^2}{\lambda_k^2} \geq 1, \qquad (2.7.2)$$

und daher $\mathbf{y} \notin T$. Die Summanden mit $i > k$ fallen weg, da die $\tilde{\mathbf{b}}_i$ orthogonal zu $\operatorname{span}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ sind, in dem auch $\mathbf{y}$ enthalten ist. Die letzte Gleichung in 2.7.2 ergibt sich dadurch, dass $\left\{ \tilde{\mathbf{b}}_i / \|\tilde{\mathbf{b}}_i\| \right\}$, mit $1 \leq i \leq k$, ein Orthonormalsystem ist und $\mathbf{y}$ deshalb folgendermaßen in diesem System ausgedrückt werden kann:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} y_i \frac{\tilde{\mathbf{b}}_i}{\|\tilde{\mathbf{b}}_i\|} = \sum_{i=1}^{n} \left\langle \mathbf{y}, \frac{\tilde{\mathbf{b}}_i}{\|\tilde{\mathbf{b}}_i\|} \right\rangle \frac{\tilde{\mathbf{b}}_i}{\|\tilde{\mathbf{b}}_i\|}.$$

Nach Minkowskis Gitterpunktsatz ist $\text{vol}(T) \leq 2^n \det(\mathcal{G})$. Außerdem ist

$$\text{vol}(T) = \left(\prod_{i=1}^{n} \lambda_i\right) \cdot \text{vol}(\mathcal{K}_n(\mathbf{0}, 1)) \geq \left(\prod_{i=1}^{n} \lambda_i\right) \cdot \left(\frac{2}{\sqrt{n}}\right)^n.$$

Die Kombination der beiden Volumengrenzen liefert das gewünschte Resultat:

$$\left(\prod_{i=1}^{n} \lambda_i\right)^{1/n} \leq \sqrt{n} \det(\mathcal{G})^{1/n}$$

□

3 Problemstellungen auf Gittern

In diesem Kapitel werden die für die kryptographische Anwendung wichtigsten Probleme beschrieben, die im Bezug auf Gitter definiert sind. Das gewissermaßen offensichtlichste Problem ist das Finden des kürzesten Vektors (bzw. einer der kürzesten Vektoren) in einem Gitter („Shortest Vector Proplem“ (SVP)). Hiermit nah verwandt ist das Problem den Gitterpunkt zu finden, der einem bestimmten Punkt im Raum $\mathbb{R}^n$ am nächsten ist. Dies ist bekannt unter dem Namen „Closest Vector Problem“ (CVP). Beide Probleme werden auch mit einem Näherungsfaktor γ definiert (SVP_γ und CVP_γ), bei dem die Länge des gesuchten Vektors um den Faktor γ länger sein darf. Nach der Definition der Probleme folgt jeweils eine Komplexitätsanalyse und unterschiedliche algorithmische Ansätze die Probleme zu lösen.

3.1 Shortest Vector Problem (SVP)

Definition 3.1 (Shortest Vector Problem (SVP))**.** Das kürzeste Vektoren Problem in einem Gitter gibt es in drei Varianten. Gegeben sei jeweils eine beliebige Gitterbasis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ eines Gitters $\mathcal{G} = \mathcal{G}(\mathbf{B})$.

Suche Finde einen kürzesten nicht-trivialen Gittervektor, also ein $\mathbf{g} \neq \mathbf{0} \in \mathcal{G}(\mathbf{B})$, sodass $\|g\| = \lambda_1(\mathcal{G}(\mathbf{B}))$.

Entscheidung (dSVP) Gegeben sei zusätzlich ein $d > 0 \in \mathbb{Q}$. Entscheide ob $\lambda_1(\mathcal{G}(\mathbf{B})) \leq d$ oder $\lambda_1(\mathcal{G}(\mathbf{B})) > d$.

Berechnung Berechne $\lambda_1(\mathcal{G}(\mathbf{B}))$.

Für die drei Varianten von SVP existieren jeweils (Polynomialzeit-) Reduktionen aufeinander, was sie im wesentlichen zu äquivalenten Problemen macht.

3.1.1 Approximiertes SVP_γ

Das kürzeste Vektoren Problem kann auch in abgeschwächter Form, mit einem Näherungsfaktor definiert werden. Die Entscheidungsvariante dieser abgeschwächten Form wird üblicherweise als Promise-Problem definiert. Diese werden wiederum folgendermaßen charakterisiert.

Definition 3.2 (Promise Problem)**.** Promise-Probleme sind eine Verallgemeinerung der Entscheidungsprobleme. Wie bei einem klassischen Entscheidungsproblem gibt es disjunkte Mengen Π_{JA} und Π_{NEIN} von JA- und NEIN-Instanzen, also Eingaben bei denen

ein Algorithmus entweder akzeptiert oder verwirft. Allerdings kann es bei einem Promise-Problem Eingaben geben, die weder JA- noch NEIN-Instanz sind. Einem Algorithmus, der ein Promise-Problem löst wird sozusagen versprochen, dass die Eingabe entweder eine JA- oder NEIN-Instanz ist. Falls dies nicht der Fall ist, ist die Ausgabe des Algorithmus undefiniert.

Definition 3.3. Das approximierte kürzeste Vektoren Problem in einem Gitter ist mit einem Näherungsfaktor $\gamma \geq 1$ definiert. Hier können ebenfalls drei Varianten unterscheiden werden. Gegeben sei jeweils eine beliebige Gitterbasis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ eines Gitters $\mathcal{G} = \mathcal{G}(\mathbf{B})$.

Suche (SVP_γ) Finde einen kurzen nicht-trivialen Gittervektor, also ein $\mathbf{g} \neq \mathbf{0} \in \mathcal{G}(\mathbf{B})$, sodass $\|g\| \leq \gamma \cdot \lambda_1(\mathcal{G}(\mathbf{B}))$.

Entscheidung (Promise) (GapSVP_γ) Gegeben sei zusätzlich ein $d > 0 \in \mathbb{Q}$. Entscheide für ein Paar $(\mathbf{B}, d)$ ob $\lambda_1(\mathcal{G}(\mathbf{B})) \leq d$ (JA-Instanz) oder $\lambda_1(\mathcal{G}(\mathbf{B})) > \gamma d$ (NEIN-Instanz).

Abschätzung (EstSVP_γ) Berechne $\lambda_1(\mathcal{G}(\mathbf{B}))$, bis auf einen Faktor γ. Gesucht ist also ein $d \in [\lambda_1(\mathcal{G}(\mathbf{B})), \gamma \cdot \lambda_1(\mathcal{G}(\mathbf{B}))]$.

Abbildung 3.1 und Abbildung 3.2 veranschaulichen das kürzeste Vektoren Problem.

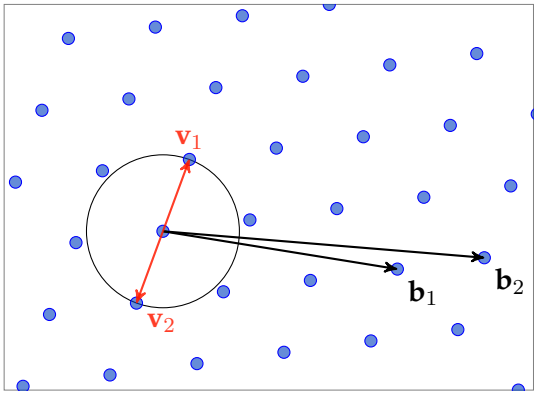


Abbildung 3.1: Exaktes Shortest Vector Problem. Bei gegebener Basis $\{\mathbf{b}_1, \mathbf{b}_2$ ist einer der kürzesten Gittervektoren ($\mathbf{v}_1$ oder $\mathbf{v}_2$) gesucht.

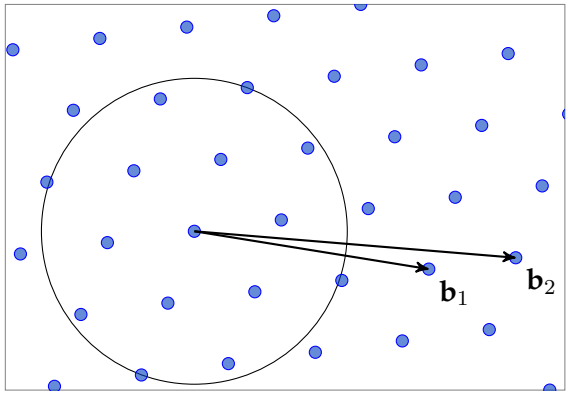


Abbildung 3.2: Approximiertes Shortest Vector Problem: Gesucht sind Gitterpunkte $\mathbf{g} \in \mathcal{G}$ mit $\|\mathbf{g}\| \leq \gamma\lambda_1(\mathcal{G})$; (hier: $\gamma = 2$).

Es ist leicht ersichtlich, dass ein Algorithmus, der das Suchproblem SVP_γ für ein bestimmtes γ löst, einfach erweitert werden kann um das Promise-Problem GapSVP_γ zu lösen. Sei A der Algorithmus, der einen näherungsweise kurzen Vektor im Gitter $\mathcal{G}(\mathbf{B})$ findet. Zur Entscheidung einer GapSVP_γ-Instanz $(\mathbf{B}, d)$ muss A nur noch um folgende Schritte erweitert werden. Die Länge l des von A gefundenen Vektors liegt im Intervall von $[\lambda_1, \gamma\lambda_1]$. Ist $l > \gamma d$, so ist $\lambda_1 > d$. Somit ist $(\mathbf{B}, d)$ keine JA-Instanz von GapSVP_γ. Unter dem Versprechen, dass $(\mathbf{B}, d) \in \Pi_{JA} \cup \Pi_{NEIN}$ ist, muss $(\mathbf{B}, d)$ eine NEIN-Instanz sein.

Wenn andersherum $l \leq \gamma d$ ist, so folgt $\lambda_1 \leq \gamma d$ und unter dem erwähnten Versprechen muss $(\mathbf{B}, d)$ eine JA-Instanz sein.

Außerdem kann gezeigt werden, dass die Abschätzungsvariante EstSVP_γ nicht schwerer als die Entscheidungsvariante GapSVP_γ ist. Sei ein Entscheidungsorakel O gegeben, dass GapSVP_γ für ein $(\mathbf{B}, d)$ löst. (Zur Erinnerung: Falls die Eingabe für O nicht dem Versprechen (Promise) des GapSVP_γ-Problems entspricht, kann O eine beliebige Antwort geben.) Sei $z \in \mathbb{Z}$ eine obere Grenze für $\lambda_1(\mathbf{B})^2$ (bspw. durch $z = \|\mathbf{b}_i\|^2$ für ein $\mathbf{b}_i$ aus der Basis $\mathbf{B}$). Dann gibt O bei der Eingabe $(\mathbf{B}, \sqrt{z})$ immer JA zurück und für die Eingabe $(\mathbf{B}, 0)$ immer NEIN zurück. Nun kann mittels binärer Suche ein $s \in \{0, \ldots, z\}$ gefunden werden, sodass $(\mathbf{B}, \sqrt{s})$ eine JA-Antwort ist und $(\mathbf{B}, \sqrt{s-1})$ eine NEIN-Antwort. Dann liegt λ_1 im Intervall von $[\sqrt{s}, \gamma\sqrt{s}]$.

Insgesamt sind folgende Reduktionen bekannt:

$$\mathsf{GapSVP}_\gamma \leq_m^p \mathsf{EstSVP}_\gamma \leq_m^p \mathsf{SVP}_\gamma$$
$$\mathsf{EstSVP}_\gamma \leq_m^p \mathsf{GapSVP}_\gamma$$

Eine Reduktion von SVP_γ zu GapSVP_γ für bestimmte (oder alle) „interessanten“ $\gamma > 1$ zu finden, oder die Existenz einer solchen Reduktion zu widerlegen, ist ein offenes Problem. Die Einschränkung auf interessante Werte von γ ist darin begründet, dass beide Probleme für $\gamma \approx 2^n$ in polynomieller Zeit lösbar sind.

3.1.2 Komplexitätsanalyse von SVP_γ

In diesem Abschnitt sollen einige Resultate aus der Komplexitätsanalyse der Gitterprobleme SVP_γ bzw. GapSVP_γ präsentiert werden. Diese Probleme sind maßgeblich für heutige gitterbasierte Kryptographie. Daher wird ihnen in der Forschung viel Aufmerksamkeit gewidmet, um Vertrauen in gitterbasierte Kryptographie aufzubauen.

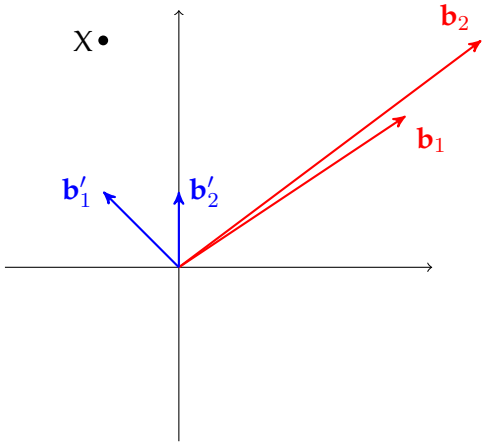


Abbildung 3.3: Vergleich zwischen „guter“ und „schlechter“ Basis desselben Gitters.

Die Schwierigkeit von SVP ist nicht direkt aus den obigen Grafiken nachzuvollziehen. Im Zweidimensionalen ist dieses Problem in der Tat nicht besonders schwer zu lösen. Für kryptographische Anwendungen werden allerdings Gitter in viel höheren Dimensionen verwendet, etwa $n = 500$. Was außerdem einen erheblichen Anteil an der Schwierigkeit von SVP_γ ausmacht, ist die gegebene Basis. Sind die Basisvektoren bereits vergleichsweise kurz, und damit annähernd orthogonal (vgl. Unterabschnitt 2.7.1), ist die geometrische Struktur des Gitters klarer zu erkennen. Um eine Intuition dafür zu gewinnen kann Abbildung 3.3 betrachtet werden. Hier sind jeweils zwei Basisvektoren verschiedener Basen desselben Gitters dargestellt. Außerdem

ist ein Punkt X des Gitters abgebildet. Es ist sofort ersichtlich, dass eine Linearkombination der blauen Basisvektoren $\mathbf{b}'_1$ und $\mathbf{b}'_2$, die zum Punkt X führt, deutlich einfacher zu finden ist, als eine entsprechende Linearkombination der roten Vektoren $\mathbf{b}_1$ und $\mathbf{b}_2$. Eine bessere Kenntnis der geometrischen Struktur des Gitters macht es auch einfacher dessen kürzeste Gittervektoren zu bestimmen. Eine gegebene Basis für ein Gitter indem SVP gelöst werden soll kann aus sehr langen Vektoren bestehen, was das Problem umso schwieriger macht. Daher werden in den meisten Algorithmen zur Lösung gitterbasierter Probleme Basisreduzierungsverfahren eingesetzt. Die verschiedenen Ansätze solcher Algorithmen werden in Abschnitt 3.3 beschrieben.

In dieser Arbeit beziehen sich die Aussagen über die Schwere von Berechnungsproblemen auf Gittern jeweils auf die euklidische (l_2) Norm. Für andere Normen (bspw. l_p-Norm mit $1 \leq p \leq \infty$) gibt es in manchen Fällen leicht abweichende Resultate. In allen Komplexitätsanalysen entspricht n üblicherweise der Dimension des Gitters, in dem ein Problem gelöst werden soll.

NP-schwere von SVP_γ

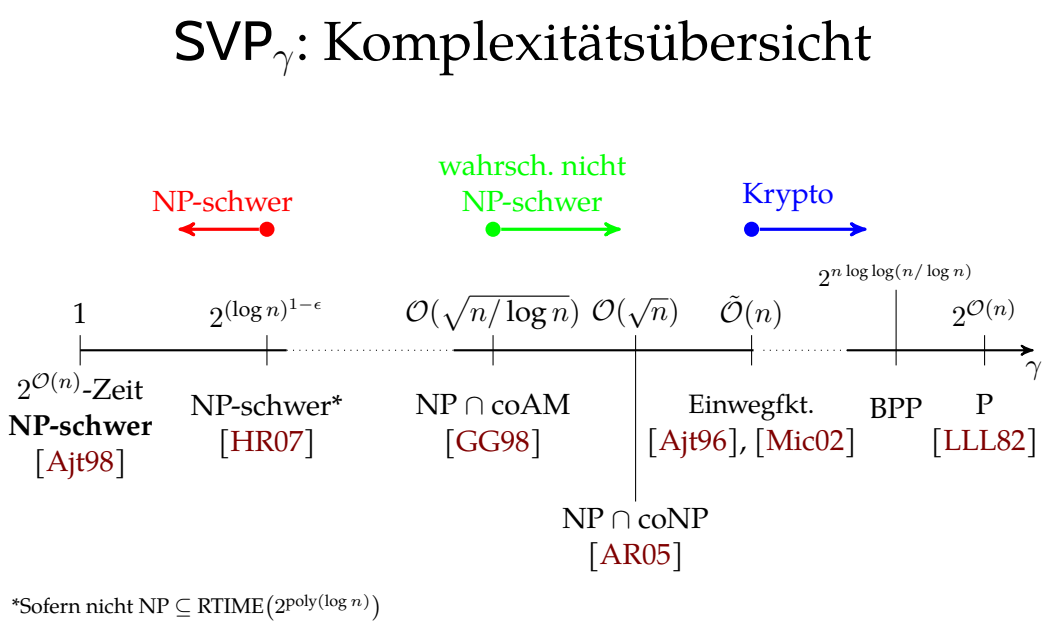


Abbildung 3.4: Komplexitätsübersicht von SVP_γ[1]

Abbildung 3.4 zeigt eine grobe Übersicht der wichtigsten komplexitätstheoretischen Resultate des approximierten Shortest Vector Problems (SVP_γ). Auf der x-Achse sind Werte für den Näherungswert γ aufgetragen, für die konkrete Resultate vorliegen. Zunächst ist leicht ersichtlich, dass das Berechnungsproblem SVP_γ mit steigenden Werten

[1] Adaptiert von `https://simons.berkeley.edu/talks/vinod-vaikuntanathan-2015-05-18a`.

für γ immer einfacher zu lösen ist. Angefangen beim schwerst möglichen Problem zeigte Ajtai [Ajt98] einen der wichtigsten Meilensteine in der Komplexitätsanalyse von SVP_γ.

Satz 3.4 (Ajtai, 1998 [Ajt98])**.** SVP $(\gamma = 1)$ *ist* ***NP****-schwer* (*unter randomisierter Polynomialzeit-Reduktion*)*. Das korrespondierende Entscheidungsproblem* dSVP *ist* ***NP****-vollständig.*

Bald darauf wurde von Micciancio auch für kleine konstante Werte $\gamma \leq \sqrt{2}$ gezeigt, dass SVP_γ weiterhin NP-schwer (unter randomisierter Polynomialzeit-Reduktion) ist [Mic98]. Micciancio führt weiterhin einen NP-schwere-Nachweis unter deterministischer Karp-Reduktion für SVP an, der auf folgender zahlentheoretischen Annahme fußt.

Vermutung 3.5. *Für jedes $\epsilon > 0$ existiert ein d, sodass es für alle ausreichend großen n eine ungerade Zahl im Intervall $[n, n + n^\epsilon]$ gibt, die quadratfrei und $(\log^d n)$-glatt ist (d.h. alle Primfaktoren haben den Exponent 1 und sind kleiner als $\log^d n$).*

Die Vermutung hält Micciancio für sehr plausibel, auch wenn heutige mathematische Techniken nicht ausreichen um diese zu beweisen.

Die Grenze der NP-schwere wurde durch nachfolgende Forschungen für immer größere Werte von γ nachgewiesen (auf Grundlage komplexitätstheoretischer Annahmen). Die bisher besten Resultate zur quasi-NP-schwere (d.h. mittels probabilistischer, quasi-polynomieller Reduktion (s.u.)) von SVP_γ stammen von Haviv und Regev in [HR07]. Zunächst wird darin die Komplexitätsklasse RTIME definiert.

Definition 3.6. Die Komplexitätsklasse RTIME ist ein randomisiertes Analogon zu DTIME, mit einseitiger Fehlerwahrscheinlichkeit. Für eine Funktion f ist $\mathsf{RTIME}(f)$ die Klasse der Probleme, für die es einen probabilistischen Algorithmus gibt, der bei Eingaben der Größe n mit einer Laufzeit von $\mathcal{O}(f(n))$ läuft. Bei Eingabe einer JA-Instanz akzeptiert ein solcher Algorithmus mit einer Wahrscheinlichkeit von mindestens $2/3$ und verwirft Eingaben von NEIN-Instanzen in jedem Fall.

[HR07]s Hauptaussage ist nun in Satz 3.7 angegeben.

Satz 3.7.

(1) *Sofern nicht* $\mathsf{NP} \subseteq \mathsf{RTIME}\left(2^{poly(\log n)}\right)$ *gilt, gibt es keinen Polynomialzeit-Algorithmus der* SVP *in einem n-dimensionalen Gitter bis auf einen Näherungsfaktor $\gamma = 2^{(\log n)^{1-\epsilon}}$ (für alle $\epsilon > 0$) approximiert.*

(2) *Unter der Annahme, dass* NP *nicht Teilmenge von* $\mathsf{RSUBEXP} = \bigcap_{\delta>0} \mathsf{RTIME}(2^{n^\delta})$ *ist, existiert ein $c > 0$, sodass es keinen Polynomialzeit-Algorithmus gibt, der* SVP *in einem n-dimensionalen Gitter bis auf einen Näherungsfaktor $\gamma = n^{c/\log\log n}$ approximiert.*

Unter den gegebenen Annahmen bedeutet der Satz, dass γ „nahezu polynomiell“ in n sein kann, um die (quasi-)NP-schwere von SVP_γ aufrecht zu erhalten. Der Beweis gelang durch eine probabilistische quasi-polynomielle Reduktion (mit einer Laufzeit von $2^{\text{poly}(\log n)}$) von SAT auf SVP_γ. Micciancio verbesserte den Beweis von Satz 3.7, indem er einen Schritt in Richtung deterministischer NP-schwere-Reduktion für SVP_γ erbrachte

[Mic12]. Während die Reduktion in [HR07] zweiseitig probabilistisch ist, ist die Reduktion in [Mic12] in einer Richtung deterministisch (NEIN-Instanzen werden sicher verworfen).

Die Grenze, bis zu welchem Wertebereich von γ eine randomisierte Polynomialzeit-Reduktion von NP-schweren Problemen auf SVP_γ gezeigt werden konnte, ist mit dem roten Punkt markiert. Die nächsten Ergebnisse für größere γ, die in Abbildung 3.4 dargestellt sind, markieren eine (angenommene) Grenze für NP-schwere Nachweise von SVP_γ (grüner Punkt). In [AR05] wurde gezeigt, dass SVP_γ (sowie auch CVP_γ) für einen Näherungswert von $\gamma \in \mathcal{O}(\sqrt{n})$ in $\mathsf{NP} \cap \mathsf{coNP}$ liegt. Da mehrheitlich davon ausgegangen wird, dass $\mathsf{NP} \neq \mathsf{coNP}$ gilt, lässt dieses Resultat vermuten, dass $\mathsf{SVP}_{\sqrt{n}}$ nicht NP-schwer ist. Wäre dies doch der Fall, wäre $\mathsf{SVP}_{\sqrt{n}}$ NP-vollständig und ebenfalls im Schnitt von NP und coNP enthalten, was eine Gleichheit der beiden Komplexitätsklassen implizieren würde. Das wiederum ist gleichbedeutend mit dem Kollabieren der Polynomialzeithierarchie bis zur zweiten Stufe ($\Sigma_1 = \Pi_1$).

Schon früher wurde von Goldreich und Goldwasser gezeigt, dass das Approximieren von SVP bei einem Näherungsfaktor $\gamma = \mathcal{O}(\sqrt{n/\log n})$ in $\mathsf{NP} \cap \mathsf{coAM}$ liegt. Wäre dies NP-schwer, würde folgen, dass coNP in AM enthalten ist, was wiederum einen Kollaps der Polynomialzeithierarchie zur Folge hätte (vgl. [MG02, Kap. 9]).

Für größere Näherungswerte ($\gamma = 2^{\mathcal{O}(n)}$) gibt es SVP_γ-Algorithmen, die in Polynomialzeit laufen. Der erste dieser Algorithmen war der LLL-Algorithmus, welcher in Unterabschnitt 3.3.1 vorgestellt wird. Aufbauend auf LLL wurden Algorithmen entwickelt, die noch etwas bessere Approximationen für kurze Vektoren in Gittern liefern. Dabei gibt es unterschiedliche Ansätze, die noch in Abschnitt 3.3 dargelegt werden.

Zusammenfassend lässt sich sagen, dass NP-schwere-Nachweise für SVP_γ, bei Werten für γ im Bereich von $\mathcal{O}(\sqrt{n/\log n})$ und größer, mit hoher Wahrscheinlichkeit nicht gelingen werden. Der rote Punkt in Abbildung 3.4 wird also nicht über den grünen Punkt hinaus zu verschieben sein. Deshalb wird primär daran geforscht, kryptographische Verfahren auf Gitterproblemen mit möglichst niedrigem Näherungswert γ aufzubauen, also den blauen Punkt weiter nach Links zu bewegen. Bislang ist es allerdings nicht gelungen Kryptographie-Primitive (wie z.B. Einweg- oder Hashfunktionen) auf SVP_γ mit $\gamma \leq n$ zu konstruieren.

Die nächsten beiden Probleme sind Abwandlungen des kürzeste Vektoren Problems und dienen in verschiedenen kryptographischen Konstruktionen (siehe Kapitel 4) als Grundlage, auf der die Sicherheit der Verfahren beruht.

3.1.3 Shortest Independent Vector Problem

Eine erweiterte Variante des Problems den kürzesten Vektor in einem Gitter zu finden, ist das *Shortest Independent Vector Problem* (SIVP) bei dem nicht nur ein Vektor $\mathbf{v}$ mit $\|\mathbf{v}\| = \lambda_1$ gesucht ist, sondern n linear unabhängige Vektoren mit der maximalen Länge des n-ten sukzessiven Minimums. Hierbei kann wiederum ein Näherungswert γ angegeben werden.

Definition 3.8 (Shortest Independent Vector Problem)**.** Gegeben sei jeweils ein vollständiges n-dimensionales Gitter $\mathcal{G}(\mathbf{B})$, durch eine Basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ und außerdem ein $\gamma \geq 1$.

Suche (SIVP$_\gamma$) Finde n linear unabhängige Vektoren $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathcal{G}(\mathbf{B})$, für die $\|\mathbf{v}_i\| \leq \gamma\lambda_n(\mathcal{G})$ gilt.

Entscheidung (Promise) (GapSIVP$_\gamma$) Gegeben sei zusätzlich ein reelles $d > 0$. Entscheide ob $\lambda_n(\mathcal{G}) \leq d$ (JA-Instanz) oder $\lambda_n(\mathcal{G}) > \gamma d$ (NEIN-Instanz).

Wie bei SVP$_\gamma$ ist die Angabe einer Reduktion vom Entscheidungsproblem GapSIVP$_\gamma$ auf das Suchproblem SIVP$_\gamma$ leicht möglich (dies wird an dieser Stelle dem Leser überlassen). Die umgekehrte Reduktion ist nach wie vor ein offenes Problem.

Blömer und Seifert wiesen nach, dass das exakte Lösen von SIVP NP-vollständig ist [BS99]. Außerdem konnten sie eine Grenze von $\gamma = n^{1/\log\log n}$ bestimmen, bis zu der SIVP$_\gamma$ NP-schwer ist.

Ebenso ist GapSIVP$_\gamma$ für konstante γ ein NP-schweres Problem (vgl. [Pei08]) Guruswami u.a. zeigen in [GMR05] zusätzlich einige komplexitätstheoretische Zusammenhänge, u.a.

- GapSIVP$_{\sqrt{n/\log n}} \in$ NP $\cap$ coAM
- GapSIVP$_{\sqrt{n}} \in$ NP $\cap$ coNP.

SIVP$_\gamma$ spielt eine große Rolle in Ajtais Worst-case zu Average-case Reduktion (siehe Unterabschnitt 4.1.1) und wird bspw. bei der Konstruktion von kollisionsresistenten kryptographischen Hashfunktionen verwendet [MR07].

3.1.4 Unique Shortest Vector Problem

Der Konstruktion von Public-Key-Kryptosystemen liegt häufig eine spezielle Form von SVP zugrunde. Bei dieser Variante wird zusätzlich noch eine Bedingung an das Gitter gestellt in dem ein kürzester Vektor gefunden werden soll. Die sukzessiven Minima des Gitters sollen nicht zu nah aneinander liegen dürfen. Konkret wird gefordert, dass ein Vektor mit Länge des zweiten sukzessiven Minimums um einen Faktor γ länger sein soll, als ein Vektor mit Länge des ersten sukzessiven Minimums. Ein Gitter mit dieser Eigenschaft ($\lambda_2(\mathcal{G}) > \gamma\lambda_1(\mathcal{G})$) heißt γ-*eindeutig*. Dies schließt eine Klasse von Gittern aus, in der im Extremfall alle sukzessiven Minima den selben oder einen ähnlichen Wert haben (vgl. Abschnitt 2.7). Das Unique Shortest Vector Problem kann sowohl als Such-, als auch als Entscheidungsproblem definiert werden. Beide Varianten des Problems können als Promise-Problem gesehen werden, bei dem die γ-Eindeutigkeit versprochen wird.

Definition 3.9 (Unique Shortest Vector Problem)**.**

Suche (uSVP$_\gamma$) Gegeben sei ein γ-eindeutiges Gitter $\mathcal{G}$ mit Basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$. Gesucht ist ein von $\mathbf{0}$ verschiedener Gitterpunkt $\mathbf{g} \in \mathcal{G}$, sodass $\|g\| = \lambda_1(\mathcal{G}(\mathbf{B}))$.

Entscheidung (duSVP$_\gamma$) Gegeben sei ein γ-eindeutiges Gitter $\mathcal{G}$ mit Basis $\mathbf{B} \in \mathbb{Q}^{n\times n}$ und ein $d > 0$. Entscheide, ob $\lambda_1(\mathcal{G}) \leq d$ (JA-Instanz) oder $\lambda_1(\mathcal{G}) > d$ (NEIN-Instanz).

In dieser Problemdefinition, kann γ sowohl als Eindeutigkeits- als auch als Näherungsfaktor verstanden werden. Kumar und Sivakumar zeigten aufbauend auf Ajtais NP-schwere Resultat für SVP, dass auch duSVP$_\gamma$ NP-schwer ist (für $\gamma = 1 + 2^{-\mathcal{O}(n^2)}$)[KS01]. Dies gelang durch eine randomisierte Polynomialzeit-Reduktion von SVP auf duSVP. In [AD16] gelang es Aggarwal und Dubey dieses Resultat zu verbessern, indem sie eine deterministische Reduktion angaben und die NP-schwere von duSVP$_\gamma$ für $\gamma = 1+1/\mathrm{poly}(n)$ nachwiesen. Diese Grenze konnte in [Ste15] noch leicht auf $\gamma = 1+\mathcal{O}(\log n/n)$ verbessert werden. Außerdem fanden sie Komplexitätsabschätzungen für duSVP$_\gamma$ für verschiedene γ und zeigten, dass die Entscheidungsvariante ähnlich schwer wie die Suchvariante ist. Dafür entwickelten sie einen Algorithmus, der uSVP$_\gamma$ mithilfe eines duSVP$_{\gamma/2}$-Orakels in polynomieller Zeit löst.

3.2 Closest Vector Problem (CVP)

Das zweite wichtige Gitterproblem ist es zu einem gegeben Punkt im Raum, den am nächsten (bzw. nah) gelegenen Gitterpunkt zu finden. Dabei ist noch folgende Definition wichtig.

Definition 3.10. Die Distanz eines Punktes $\mathbf{x} \in \mathbb{R}^n$ im Raum zum Gitter $\mathcal{G}$ ist definiert als $\mathrm{dist}(\mathbf{x}, \mathcal{G}) := \min_{g\in\mathcal{G}} \|\mathbf{x} - \mathbf{g}\|$.

Definition 3.11 (Closest Vector Problem (CVP))**.** Das Problem des nächstgelegenen Gitterpunkts kann ebenfalls in drei Varianten definiert werden. Gegeben sei jeweils eine beliebige Gitterbasis $\mathbf{B} \in \mathbb{Q}^{n\times n}$ eines Gitters $\mathcal{G} = \mathcal{G}(\mathbf{B})$ und ein Zielvektor $\mathbf{z} \in \mathbb{Q}^n$.

Suche Finde einen Gitterpunkt $\mathbf{Bx}$, der $\mathbf{z}$ am nächsten ist, also $\|\mathbf{Bx} - \mathbf{z}\|$ minimal ist.

Entscheidung Gegeben sei zusätzlich ein $r \in \mathbb{Q}$. Entscheide ob $\mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B})) \leq r$ oder $\mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B})) > r$.

Berechnung Berechne die Distanz $\mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B}))$ von $\mathbf{z} \in \mathbb{Q}^n$ zum Gitter $\mathcal{G}$, also zum nächstgelegenen Gitterpunkt in $\mathcal{G}$.

3.2.1 Approximiertes CVP$_\gamma$

Definition 3.12 (Approximiertes Closest Vector Problem (CVP$_\gamma$))**.** Gegeben sei jeweils eine beliebige Gitterbasis $\mathbf{B} \in \mathbb{Q}^{n\times n}$ eines Gitters $\mathcal{G} = \mathcal{G}(\mathbf{B})$ und ein Zielvektor $\mathbf{z} \in \mathbb{Q}^n$. Bei der approximierten Variante von CVP gibt es zusätzlich einen Näherungsfaktor $\gamma \geq 1$.

Suche Finde einen Gitterpunkt $\mathbf{g} \in \mathcal{G}(\mathbf{B})$, sodass $\|\mathbf{g} - \mathbf{z}\| \leq \gamma \cdot \mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B}))$.

Entscheidung (Promise) GapCVP$_\gamma$ Gegeben sei zusätzlich ein $r \in \mathbb{Q}$. Entscheide ob $\mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B})) \leq r$ (JA-Instanz) oder $\mathrm{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B})) > \gamma r$ (NEIN-Instanz).

Abschätzung Berechne ein $r \in \mathbb{Q}$, sodass $\text{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B})) \leq r \leq \gamma \cdot \text{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B}))$.

Abbildung 3.5 und Abbildung 3.6 dienen zur Anschauung des Closest Vector Problems.

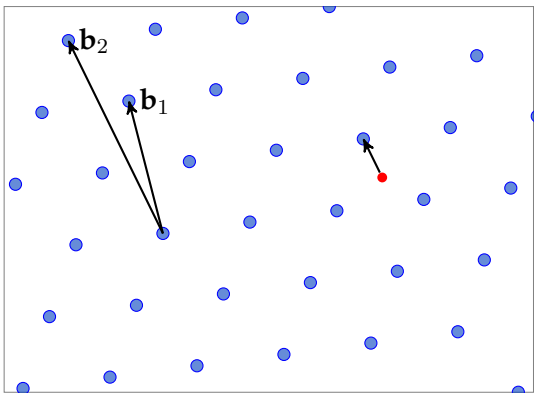


Abbildung 3.5: Closest Vector Problem

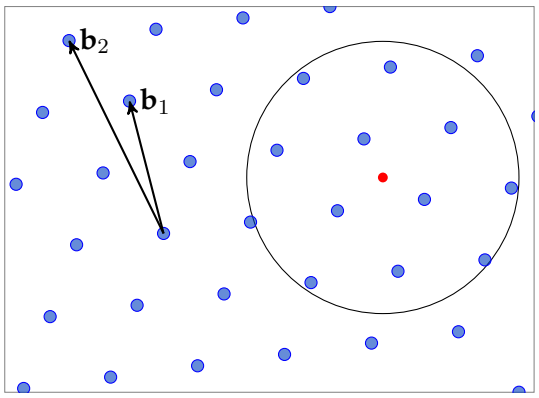


Abbildung 3.6: Approximiertes Closest Vector Problem: Gesucht sind zu gegebenem $\mathbf{z} \in \mathbb{Q}^n$, Gitterpunkte $\mathbf{g} \in \mathcal{G}$ mit $\|\mathbf{g} - \mathbf{z}\| \leq \gamma \cdot \text{dist}(\mathbf{z}, \mathcal{G}(\mathbf{B}))$.

Ähnlich wie bei SVP_γ und GapSVP_γ kann man Reduktionen von GapCVP_γ zu CVP_γ zeigen. Die umgekehrte Reduktion ist wie bei SVP ein offenes Problem.

3.2.2 Komplexitätsanalyse von CVP_γ

Aus den Abbildungen zum Closest Vector Problem ist aufgrund der niedrigen Dimensionalität nicht unmittelbar ersichtlich, dass es sich um ein schweres (sogar NP-vollständiges) Problem handelt. Folgende Überlegung hilft, um davon einen Eindruck zu bekommen. Die Schwierigkeit von CVP, also zu einem Punkt im Raum den nächstgelegenen Gitterpunkt zu finden, steigt mit der Anzahl der Punkte, die sich in unmittelbarer Nähe zum gegebenen Punkt befinden können. Mit steigender Dimension wächst die Zahl der Gitterpunkte, die sich in direkter Umgebung zu einem Raumpunkt befinden, exponentiell an. Folglich wird es umso schwerer herauszufinden, welchem dieser Gitterpunkte der gegebene Punkt am nächsten ist.

Für die Komplexität von CVP_γ zeigt sich ein ähnliches Bild wie für SVP_γ (vgl. Abbildung 3.4). Da CVP_γ ein etwas schwereres Problem als SVP_γ ist, können mitunter aber bessere Schranken für die NP-schwere von CVP_γ gefunden, oder diese auf stärkeren Annahmen aufgebaut werden. So kann zum Beispiel gezeigt werden, dass CVP_γ mit $\gamma = n^{c/\log\log n}$ NP-schwer ist, allein auf der Annahme basierend, dass $\mathsf{P} \neq \mathsf{NP}$ ist. Bei SVP_γ ist dieselbe Grenze für die NP-schwere nur unter der Annahme dass $\mathsf{NP} \not\subseteq \mathsf{RSUBEXP} = \bigcap_{\delta>0} \mathsf{RTIME}(2^{n^\delta})$ gezeigt worden (vgl. [HR07]). Im Gegensatz zu SVP kann für CVP die NP-vollständigkeit unter einer *deterministischen* Polynomialzeit-Reduktion gezeigt werden. In [MG02, Kap 3]) wird eine solche Reduktion vom Teilsum-

menproblem zu CVP angegeben. Unter deterministischer Reduktion bleibt CVP_γ auch für beliebige konstante γ-Werte NP-schwer.

Wie SVP und CVP zusammenhängen wird im nächsten Abschnitt näher beleuchtet. Es ist relativ leicht nachzuweisen, dass SVP nicht schwerer als CVP ist.

3.2.3 SVP ist nicht schwerer als CVP

O. Goldreich, D. Micciancio, S. Safra und J.-P. Seifert untersuchten 1999 die Verbindung der Gitterprobleme CVP und SVP und bewiesen folgendes Theorem.

Theorem 3.13. [*GMSS99*]
Sei $\gamma > 1$ beliebig. Mithilfe eines CVP_γ Orakels lässt sich SVP_γ in polynomieller Laufzeit lösen (Cook-Reduktion).

Beweis. Angegeben wird ein Beweis für $\gamma = 1$, der für $\gamma > 1$ generalisiert werden kann. Abbildung 3.7 dient dabei als Veranschaulichung des Beweises. Für eine Gitterbasis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ und $i \in \{1, \ldots, n\}$ sei $\mathbf{B}_i = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$, sodass das Gitter $\mathcal{G}(\mathbf{B}_i) \subset \mathcal{G}(\mathbf{B})$ den Vektor $\mathbf{b}_i$ nicht enthält. Der folgende Algorithmus löst SVP mithilfe eines CVP-Orakels.

Eingabe: Gitterbasis $\mathbf{B}$
for all $i \in \{1, \ldots, n\}$ **do**
  Generiere $\mathbf{B}_i$
  $\mathbf{v}_i \leftarrow \mathsf{CVP}(\mathbf{B}_i, \mathbf{b}_i)$
end for
Wähle $i^* \leftarrow \arg\min_i \|\mathbf{v}_i - \mathbf{b}_i\|$
return $\mathbf{v}_{i^*} - \mathbf{b}_{i^*}$

Für die Korrektheit des Algorithmus dient folgende Betrachtung. Sei $\mathbf{g} = \sum_{i=1}^{n} c_i \mathbf{b}_i$ der kürzeste Vektor in $\mathcal{G}(\mathbf{B})$, den der Algorithmus zurück geben soll. Im Fall, dass ein Koeffizient c_j von $\mathbf{g}$ ungerade ist, ist $\mathbf{g} + \mathbf{b}_j$ der nächstgelegene Vektor zu $\mathbf{b}_j$ in $\mathcal{G}(\mathbf{B}_j)$. Zunächst ist es leicht zu sehen, dass $\mathbf{g} + \mathbf{b}_j \in \mathcal{G}(\mathbf{B}_j)$. Durch einen Widerspruchsbeweis lässt sich nachvollziehen, dass $\mathbf{g} + \mathbf{b}_j$ den geringsten Abstand zu $\mathbf{b}_j \notin \mathcal{G}(\mathbf{B}_j)$ hat. Angenommen, und es gäbe ein $\mathbf{g}' \in \mathcal{G}(\mathbf{B}_j)$ mit $\|\mathbf{g}' - \mathbf{b}_j\| < \|(\mathbf{g} + \mathbf{b}_j) - \mathbf{b}_j\| = \|\mathbf{g}\|$. Dann wäre $\mathbf{g}' - \mathbf{b}_j$ ein Gitterpunkt in $\mathcal{G}(\mathbf{B})$ und würde dem widersprechen, dass $\mathbf{g}$ der kürzeste Vektor in $\mathcal{G}(\mathbf{B})$ ist.

Mindestens einer der Koeffizienten c_i von $\mathbf{g}$ muss nun ungerade sein, da anderenfalls $\mathbf{g}/2$ ein kürzerer Vektor in $\mathcal{G}(\mathbf{B})$ wäre. Sei bspw. c_j ein solcher Koeffizient. Dann setzt der Algorithmus $\mathbf{v}_j = \mathbf{g} + \mathbf{b}_j$ mithilfe des CVP-Orakels und gibt $\mathbf{g}$ aus. □

Die Reduktion lässt sich leicht für die Entscheidungsvarianten GapSVP_γ und GapCVP_γ erweitern, indem der GapSVP_γ-Algorithmus für Eingaben $(\mathbf{B}, d)$ genau dann akzeptiert, wenn das GapCVP_γ-Orakel für mindestens eine der Anfragen $(\mathbf{B}_i, \mathbf{b}_i, d)$ JA zurück gibt. Für weitere Details sei auf [GMSS99] verwiesen.

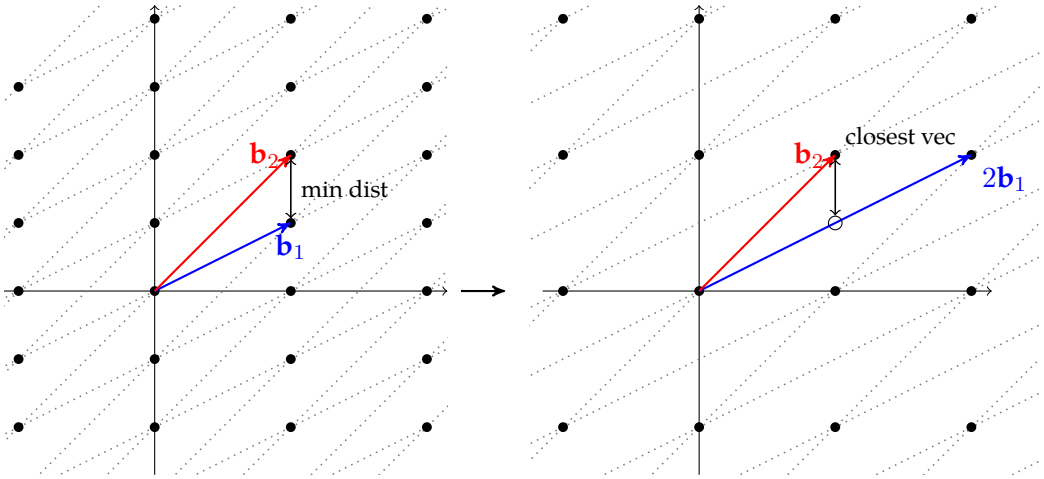


Abbildung 3.7: Reduktion von SVP zu CVP.

3.3 Algorithmen für SVP und CVP

Grundsätzlich lassen sich zwei Arten von Algorithmen zur Lösung von Gitterproblemen wie SVP und CVP unterscheiden. Diejenigen, die eine exakte Lösung der Probleme berechnen, benötigen eine exponentielle Laufzeit (in der Dimension n des Gitters). Dazu zählen Enumerationsverfahren, bei denen im Fall von SVP eine Suche unter allen möglichen kurzen Vektoren eines Gitters durchgeführt wird. Die Anzahl kurzer Vektoren steigt im schlimmsten Fall allerdings exponentiell mit der Dimension des Gitters an. Um Enumerationsalgorithmen zu optimieren, werden unterschiedliche Methoden angewandt, die Auswahl der für die Suche gewählten Gitterpunkte einzuschränken (Pruning genannt). Dabei muss beachtet werden, eine zu große Fehlerwahrscheinlichkeit zu vermeiden. Eine andere Klasse exakter Gitteralgorithmen sind Siebverfahren, welche eine etwas bessere Laufzeit erreichen (jedoch immer noch exponentiell), aber zusätzlich exponentiell viel Platz benötigen. Außerdem sind Siebverfahren randomisierte Algorithmen. Der zurzeit beste Sieb-Algorithmus zum Lösen von SVP [BDGL15] erreicht eine Laufzeit von $2^{0,292n+o(n)}$.

Andererseits gibt es Polynomialzeit-Algorithmen, die bspw. SVP_γ bis auf asymptotisch exponentielle Werte für γ approximieren. Die meisten Approximations-Algorithmen basieren auf Verfahren, die eine gegebene Gitterbasis reduzieren. Dabei wird eine andere Basis für das selbe Gitter gefunden, die aus kürzeren Vektoren besteht. Oft werden auch beide Ansätze kombiniert. Blockweise Gitterreduzierungsalgorithmen, wie BKZ (siehe unten) benutzen Subroutinen um kurze Vektoren in niedrig-dimensionalen Untergittern des Eingabegitters zu finden. Diese Subroutinen werden oft durch Enumeration, teilweise auch von Siebverfahren, implementiert. Die Laufzeit von Enumerationsalgorithmen hängt allerdings enorm von der „Qualität“ der gegebenen Gitterbasis ab (es werden vergleichsweise kurze, nahezu orthogonale Basisvektoren benötigt). Daher werden die Gitterbasen zunächst von einem Basisreduzierungsalgorithmus reduziert.

Nachfolgend wird der erste allgemeine Basisreduzierungsalgorithmus, der LLL-Algorithmus, beschrieben.

3.3.1 LLL

Der LLL-Algorithmus von Arjen Lenstra, Hendrik Lenstra und László Lovász [LLL82] ist ein Meilenstein gitterbasierter Algorithmen. Das Ziel des Algorithmus ist das Faktorisieren von Polynomen mit rationalen Koeffizienten. Dies hängt insofern mit Gittern zusammen, dass beim Berechnen der irreduziblen Faktoren eines rationalen Polynoms gleichzeitig relativ kurze Vektoren in einem assoziierten Gitter gefunden werden. Um dieses Ziel zu erreichen reduziert LLL eine gegebene Gitterbasis, d.h. er findet eine Basis mit (relativ) kurzen Basisvektoren. Dabei ist LLL der erste effiziente Algorithmus dieser Art. Eine ausführliche Beschreibung mit historischen Hintergründen und Entwicklungen rund um den LLL-Algorithmus kann in [Ngu10] gelesen werden.

Die Laufzeit des Algorithmus ist polynomiell in der Dimension n eines Gitters $\mathcal{G}$ und liefert eine Basis, dessen kürzester Vektor, eine maximale Länge von $2^{\mathcal{O}(n/2)}\lambda_1(\mathcal{G})$ hat. In anderen Worten, LLL löst $\mathsf{SVP}_{2^{\mathcal{O}(n/2)}}$ in P. Bemerkenswert ist, dass die Laufzeit des Algorithmus nur geringfügig von der Länge (Bitlänge) der gegebenen Basisvektoren abhängt, sondern maßgeblich von der Dimension des Gitters.

Neben den oben erwähnten Anwendungen, kann LLL in weiteren Bereichen der Mathematik, theoretischen Informatik und Kryptologie eingesetzt werden, beispielsweise

- Faktorisieren von rationalen Polynomen
- Approximierung der kürzesten Vektoren in einem Gitter
- Lösen von Integer Programmierungs-Problemen in fester Anzahl von Variablen
- Finden von Relationen zwischen Zahlen (z.B. $6,73205080756887... = \sqrt{3}+5$)
- Kryptoanalyse (Angriffe gegen bestimmte RSA-Varianten, NTRU[2], ...)
- Diophantische Approximation

Nachfolgend wird LLL in der Funktion, näherungsweise kurze Vektoren in einem Gitter zu finden, beschrieben. Der Algorithmus berechnet zu einer gegebenen Gitterbasis eine reduzierte Basis folgender Form.

Definition 3.14 (δ-LLL-reduzierte Basis)**.** Es sei $1/4 < \delta < 1$. Eine Gitterbasis $\mathbf{B} \in \mathbb{R}^{n\times n}$ mit zugehöriger Gram-Schmidt-Orthogonalisierung $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n]$ (siehe Abschnitt 2.6) heißt *δ-LLL-reduziert*, wenn für alle $1 \leq j < i \leq n$,

$$|\mu_{i,j}| = \left|\frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j\rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j\rangle}\right| \leq \frac{1}{2} \tag{3.3.1}$$

[2] siehe Abschnitt 5.3

und für alle $1 \leq i < n$,

$$\delta \left\| \tilde{\mathbf{b}}_i \right\|^2 \leq \left\| \mu_{i+1,i} \tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1} \right\|^2 \tag{3.3.2}$$

gilt.

Für δ wird üblicherweise der Wert $3/4$ gewählt (dieser Wert wurde von den Ursprünglichen Autoren gewählt und erleichtert die Analyse des Algorithmus). Für höhere Werte von δ werden bessere (aber immer noch exponentiell große) Näherungen für den kürzesten Vektor eines Gitters gefunden. Jedoch benötigt der Algorithmus dafür mehr Iterationen (die Laufzeit bleibt für $\delta < 1$ polynomiell). Eine derart reduzierte Basis eines Gitters $\mathcal{G}$ enthält mindestens einen Vektor, der höchstens eine Länge von $2^{(n-1)/2} \lambda_1(\mathcal{G})$ hat. Dies lässt sich aus folgendem Lemma schließen.

Lemma 3.15. *Für eine δ-LLL-reduzierte Basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ mit zugehöriger Gram-Schmidt-Orthogonalisierung $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n]$ gilt $\left\| \tilde{\mathbf{b}}_{i+1} \right\|^2 \geq \frac{1}{2} \left\| \tilde{\mathbf{b}}_i \right\|^2$ für alle $1 \leq i < n$.*

Beweis. Die zweite Eigenschaft aus Lemma 3.14 kann umgeschrieben werden zu

$$\delta \left\| \tilde{\mathbf{b}}_i \right\|^2 \leq \left\| \mu_{i+1,i} \tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1} \right\|^2 = \mu_{i+1,i}^2 \left\| \tilde{\mathbf{b}}_i \right\|^2 + \left\| \tilde{\mathbf{b}}_{i+1} \right\|^2$$

da $\tilde{\mathbf{b}}_i$ und $\tilde{\mathbf{b}}_{i+1}$ orthogonal zueinander sind. Aufgrund der ersten LLL-Eigenschaft (3.3.1) folgt daraus

$$\left\| \tilde{\mathbf{b}}_{i+1} \right\|^2 \geq (\delta - \mu_{i+1,i}^2) \left\| \tilde{\mathbf{b}}_i \right\|^2 \geq \left(\delta - \frac{1}{4} \right) \left\| \tilde{\mathbf{b}}_i \right\|^2.$$

□

Das bedeutet, dass in einer LLL-reduzierten Basis $\tilde{\mathbf{b}}_{i+1}$ nicht viel kürzer als $\tilde{\mathbf{b}}_i$ ist. Dies lässt folgende Aussage über den ersten Vektor $\mathbf{b}_1$ einer LLL-Reduzierten Basis zu.

Korollar 3.15.1. *Sei $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ eine δ-LLL-reduzierte Basis. Dann gilt für die Länge des ersten Basisvektors $\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}} \right)^{n-1} \lambda_1(\mathcal{G}(\mathbf{B}))$.*

Beweis. Lemma 3.15 liefert folgende Ungleichungskette.

$$\left\| \tilde{\mathbf{b}}_n \right\|^2 \geq \left(\delta - \frac{1}{4} \right) \left\| \tilde{\mathbf{b}}_{n-1} \right\|^2 \geq \cdots \geq \left(\delta - \frac{1}{4} \right)^{n-1} \left\| \tilde{\mathbf{b}}_1 \right\|^2 = \left(\delta - \frac{1}{4} \right)^{n-1} \left\| \mathbf{b}_1 \right\|^2$$

Somit gilt für alle i

$$\left\| \mathbf{b}_1 \right\| \leq \left(\delta - \frac{1}{4} \right)^{-(i-1)/2} \left\| \tilde{\mathbf{b}}_i \right\| \leq \left(\delta - \frac{1}{4} \right)^{-(n-1)/2} \left\| \tilde{\mathbf{b}}_i \right\|$$

Aufgrund von Theorem 2.16 gilt für jede Gitterbasis $\lambda_1(\mathcal{G}(\mathbf{B})) \geq \min_i \left\| \tilde{\mathbf{b}}_i \right\|$ und folglich

$$\left\| \mathbf{b}_1 \right\| \leq \left(\delta - \frac{1}{4} \right)^{-(n-1)/2} \min_i \left\| \tilde{\mathbf{b}}_i \right\| \leq \left(\delta - \frac{1}{4} \right)^{-(n-1)/2} \lambda_1(\mathcal{G}(\mathbf{B}))$$

□

Für $\delta = 3/4$ wird somit eine Näherung zu den kürzesten Gittervektoren von $2^{(n-1)/2}\lambda_1(\mathcal{G})$ erreicht. Der nachfolgende Algorithmus berechnet zu einer gegebenen Basis eine LLL-reduzierte Basis (zur Anschauung dient Abbildung 3.8).

Algorithmus 1 LLL

1: EINGABE: Gitterbasis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{Z}^{n \times n}$
2: AUSGABE: LLL-reduzierte Basis des Gitters $\mathcal{G}(\mathbf{B})$
3:
4: Berechne $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n]$ ▷ Gram-Schmidt-orthogonalisierte Vektoren von $\mathbf{B}$
5: REDUZIERE$(\mathbf{B}, \tilde{\mathbf{B}})$
6: **if** $\frac{3}{4}\|\tilde{\mathbf{b}}_i\|^2 > \|\mu_{i+1,i}\tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1}\|^2$ für ein $i \in \{1, \ldots, n-1\}$ **then**
7: Vertausche $\mathbf{b}_i$ und $\mathbf{b}_{i+1}$ in $\mathbf{B}$
8: Gehe zurück zu Zeile 4
9: **else**
10: **return B**
11: **end if**

1: **procedure** REDUZIERE$(\mathbf{B}, \tilde{\mathbf{B}})$
2: **for** $i = 2, \ldots, n$ **do**
3: **for** $j = i-1, \ldots, 1$ **do**
4: $\mathbf{b}_i \leftarrow \mathbf{b}_i - c_{i,j}\mathbf{b}_j$ ▷ $c_{i,j} = \lfloor \mu_{i,j} \rceil = \left\lfloor \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \right\rceil$
5: **end for**
6: **end for**
7: **end procedure**

$\lfloor \cdot \rceil$ rundet zur nächsten ganzen Zahl.

Es ist wichtig zu beachten, dass beim Reduzieren der Basisvektoren, deren Gram-Schmidt-orthogonalisierung erhalten bleibt, also die zugehörigen $\tilde{\mathbf{b}}_i$ nicht verändert werden. Dies geschieht jedoch beim Vertauschen zweier Vektoren in Zeile 7. Daher müssen die Gram-Schmidt-Vektoren neu berechnet werden, wobei es insbesondere auf die veränderten Gram-Schmidt-Koeffizienten ankommt.

Nach einem Reduktionsschritt in der Prozedur REDUZIERE ist für ein $j < i$ die erste Eigenschaft (3.3.1) einer LLL-reduzierten Basis, also $|\mu_{i,j}| \leq 1/2$ erfüllt. Dies folgt aus der Tatsache, dass der gerundete Teil von $\mu_{i,j}\tilde{\mathbf{b}}_j$ beim Reduzieren vom Basisvektor $\mathbf{b}_i$ abgezogen wird und sich $\mu_{i,j}$ somit betragsmäßig höchstens um $1/2$ vom ursprünglichen Wert unterscheidet. Konkret ist nach einem Reduzierungsschritt

$$|\mu_{i,j}| = \left| \frac{\langle \mathbf{b}_i - c_{i,j}\mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \right| = \left| \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} - \left\lfloor \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \right\rceil \cdot \frac{\langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \right| \leq \frac{1}{2}.$$

Nach Ablauf von REDUZIERE gilt dies für alle $\mu_{i,j}$ mit $i > j$. Die erste Gleichung folgt aus der Definition des Reduktionsschritts, der eine leicht angepasste Form eines Schritts

im Gram-Schmidt-Verfahren darstellt. Die Ungleichung ergibt sich dann daraus, dass $\langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle = \langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle$ ist.

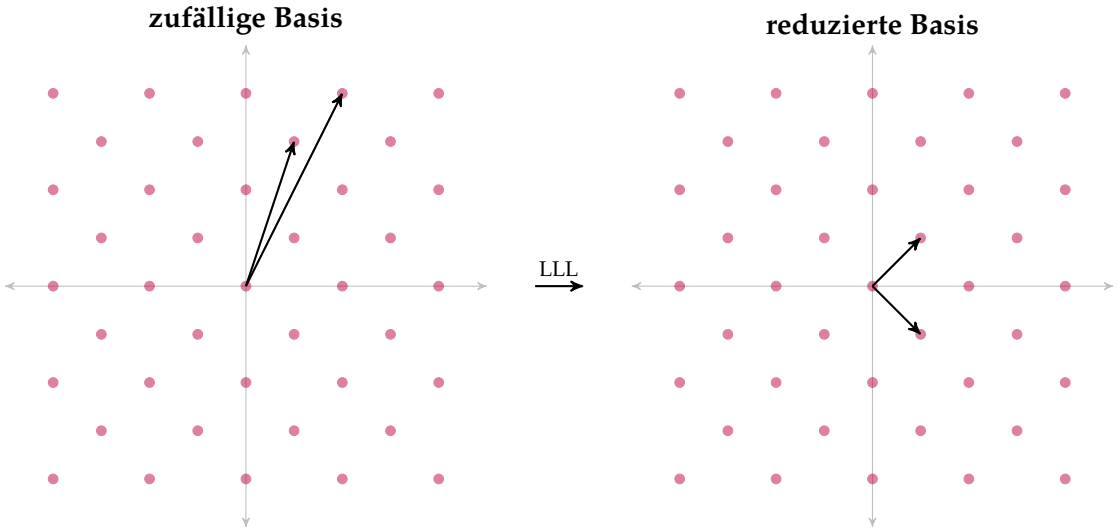


Abbildung 3.8: LLL-Basisreduktion[3]

Der Algorithmus terminiert nur, wenn die reduzierte Basis zusätzlich der zweiten LLL-Eigenschaft (3.3.2) genügt, sodass die Ausgabe immer eine LLL-reduzierte Basis ist. Dies wird nach polynomiell vielen Schritten (maßgeblich abhängig von der Dimension des Gitters) erreicht. Die Argumentation über die Terminierung des Algorithmus und eine ausführliche Analyse der Laufzeit von LLL und seiner Korrektheit ist in [MG02, Kap. 2] zu finden.

3.3.2 BKZ

Nach dem LLL-Algorithmus wurden weitere Algorithmen zur Lösung von SVP (und CVP) entwickelt. Einer der bekanntesten ist der BKZ-Algorithmus (Block Korkin Zolotarev), der ausgehend von einer LLL-reduzierten Basis, noch kürzere Vektoren in einem Gitter findet. Dabei arbeitet er mit Unteraufrufen von Emumerations- und Siebalgorithmen.

Unterdessen stellten Chen und Nguyen [CN11] eine verbesserte Variante des Algorithmus namens BKZ 2.0 vor. Darin wurden mittlerweile bekannt gewordene Verbesserungen bei Gitter-Enumerationsverfahren implementiert, sodass ein Performancegewinn beim Lösen von Gitterproblemen erzielt wurde. Mithilfe von BKZ 2.0 konnten infolgedessen genauere Sicherheitsabschätzungen für gitterbasierte Kryptosysteme aufgestellt werden.

[3] In leicht angepasster Form von [Jea16] übernommen.

3.3.3 Voronoi-basierte Algorithmen

Neben Sieb- und Enumerationsverfahren zur Lösung von Gitterproblemen gibt es auch Algorithmen, die mit Voronoi-Zellen (siehe Unterabschnitt 2.4.2) arbeiten. Ein intuitiver Anwendungsfall von Voronoi-Zellen liegt in Algorithmen zur Lösung des Closest Vector Problems. Der Raum $\mathbb{R}^n$, der durch die Basisvektoren eines vollständigen, n-dimensionalen Gitters aufgespannt wird, kann mithilfe von Voronoi-Zellen partitioniert werden. Damit ist für jeden Gitterpunkt die Menge der ihm am nächsten gelegenen Raumpunkte definiert. So ist das Problem, den nächstgelegenen Gitterpunkt zu einem Zielvektor $\mathbf{t} \in \mathbb{R}^n$ zu bestimmen, auf das Finden der Voronoi-Zelle in der $\mathbf{t}$ liegt, reduziert (siehe Abbildung 3.9).

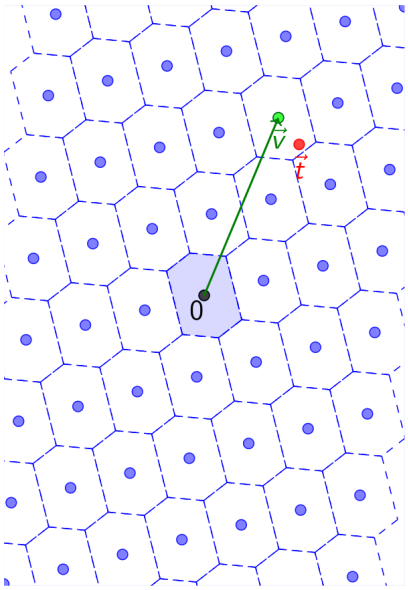


Abbildung 3.9: Lösen von CVP mithilfe von Voronoi-Zellen.[4]

Daniele Micciancio führt eine Liste[5] Voronoi-basierter Algorithmen. Einer der ersten dieser Algorithmen zur Lösung von CVP wurde in [MV13] angegeben und hat eine Laufzeit von $\tilde{\mathcal{O}}(2^{2n})$, was eine Verbesserung zur bisherigen Laufzeit von $n^{\mathcal{O}(n)}$ darstellte. Obwohl es theoretisch bislang der schnellste deterministische Algorithmus zum exakten lösen von CVP ist, gibt es hingegen (Sieb-, bzw. Enumerations-) Techniken, die in der Praxis besser geeignet sind. Dies liegt unter anderem an der hohen räumlichen Komplexität des Algorithmus ($\mathcal{O}(2^n)$).

3.3.4 Lattice Challenge

Die TU Darmstadt betreibt die Webseite `https://latticechallenge.org`, auf der verschiedene Herausforderungen, unter anderem zur Lösung von SVP gestellt werden. Es werden Definitionen für Gitter unterschiedlicher Dimension zum Download bereit gestellt, zu denen man eine Lösung (kurzer Vektor im Gitter) einreichen kann. Ziel ist es unterschiedliche Algorithmen für Gitterprobleme zu benchmarken. Die besten Lösungen werden mit grober Angabe des verwendeten Algorithmus, der verwendeten Hardware und der benötigten Zeit, in einer Hall of Fame veröffentlicht.

3.3.5 Alternativer Lösungsansatz

Neben den obigen Ansätzen gibt es auch etwas exotischere Methoden zum Lösen von SVP, wie bspw. der Evolutionäre Algorithmus (EA) von Ding et. al [DZW15]. Die Autoren verwendeten den Algorithmus im Zusammenspiel mit BKZ und verglichen die

[4] Quelle: `http://cseweb.ucsd.edu/~daniele/papers/Voronoi-slides.pdf`
[5] `http://cseweb.ucsd.edu/~daniele/LatticeLinks/Voronoi.html`

Laufzeit für zufällig erzeugte Gitter aus der SVP-Challenge der TU Darmstadt. Dabei konnte der Algorithmus für kleinere Dimensionen bis $n = 118$ bessere Ergebnisse erzielen (kürzere Laufzeit), als eine Kombination von BKZ und Enumeration-Verfahren. Für Gitter dieser Dimensionen sind auch einige Lösungen, die mithilfe eines EA gefunden wurden, in der Hall of Fame der SVP-Challenge zu finden.

3.4 Offene Probleme

Micciancio betreibt eine Webseite[6], auf der er neben dem Fortschritt in vielfältigen Einsatzbereichen von Gittern in der Kryptographie, auch die Entwicklung von Algorithmen zur Lösung von Gitterproblemen darstellt. Auf der Webseite sind auch Listen offener Probleme in verschiedenen Bereichen zu finden, wovon hier eine Auswahl wiedergegeben wird.

- Ein offenes Problem ist weiterhin die NP-schwere von SVP_γ in der euklidischen Norm unter *deterministischer* Polynomialzeit-Reduktion zu zeigen. Es wurden zwar schon derartige Reduktionen angegeben, bspw. in [Mic98]; diese kommen jedoch nicht ohne unbewiesene Zahlentheoretische Annahmen aus. Dies ist bisher auch nicht für das exakte lösen von SVP ($\gamma = 1$) gelungen.
- Es gibt bisher keinen Algorithmus, der SVP exakt löst und in einfacher exponentieller Zeit ($2^{\mathcal{O}(n)}$ im Gegensatz zu $n^{\mathcal{O}(n)}$) läuft **und** (nur) polynomiell viel Platz benötigt.

Vinod Vaikuntanathan pflegt ebenfalls eine Liste[7] offener Probleme bzgl. der Komplexität verschiedener Gitterprobleme, zugehöriger Algorithmen und offener Fragen in der gitterbasierten Kryptographie.

[6] `http://cseweb.ucsd.edu/~daniele/LatticeLinks/index.html`
[7] `http://people.csail.mit.edu/vinodv/6876-Fall2017/OpenProblems/projectideas.html`

4 Gitterbasierte Kryptographie

Auf dem Weg in Richtung gitterbasierter Kryptographie gibt es zwei prominente Problemdefinitionen, auf dessen Grundlage kryptographische Verfahren konstruiert werden können. Diese Probleme sind das Bindeglied zwischen den allgemeinen Gitterproblemen, die im letzten Kapitel definiert wurden und kryptographischen Konstruktionen wie kollisionsresistente Hashfunktionen und Public-Key Kryptosysteme. Miklós Ajtai gelang 1996 ein Durchbruch für die gitterbasierte Kryptographie, indem er eine Reduktion von verschiedenen approximierten allgemeinen Gitterproblemen zu dem Problem, was später als „Short Integer Solutions Problem“ (SIS) formalisiert wurde, zeigte. Vor diesem Meilenstein wurden Gitter hauptsächlich zur Kryptoanalyse genutzt (bspw. im LLL-Algorithmus), um kryptographische Verfahren zu brechen. Ajtais Entdeckung, dass Gitter auch zur Konstruktion von Kryptosystemen genutzt werden können, war ein Startschuss für die gitterbasierte Kryptographie. Seid dem wurden auch andere Problemstellungen definiert, die in ähnlicher Weise mit Gitterproblemen wie SVP_γ und CVP_γ zusammenhängen und Grundlage verschiedener Kryptosysteme sind. Die prominentesten dieser Probleme sind das „Small Integer Solutions Problem“ und das „Learning with Errors Problem“, welche nachfolgend vorgestellt werden.

4.1 Small Integer Solutions (SIS)

Das Small Integer Solutions Problem ist bereits 1996 indirekt von Ajtai in seiner wegweisenden Arbeit „Generating Hard Instances of Lattice Problems“ [Ajt96] beschrieben worden, welche im nächsten Abschnitt erläutert wird. Hier ist es in einer etwas allgemeineren Form wiedergegeben, wie es in einer nachfolgenden Arbeit von Daniele Micciancio und Oded Regev in [MR07] definiert ist.

Definition 4.1 (SIS)**.**
Gegeben seien die Parameter $n, m, q \in \mathbb{N}$, eine Matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ und ein reelles β.
Gesucht ist ein Vektor $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$, sodass folgende Bedingungen gelten:

$$\mathbf{Az} \equiv 0 \pmod{q} \text{ und } \|\mathbf{z}\| \leq \beta.$$

Dieses Problem ist ein Gitterproblem, denn es wird nach einem kurzen Vektor in einem bestimmten Gitter $\mathcal{G}_q(\mathbf{A})$ gefragt, wobei

$$\mathcal{G}_q(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{Az} = \mathbf{0} \pmod{q}\}. \tag{4.1.1}$$

In der Problemdefinition von SIS wird davon ausgegangen, dass es eine Lösung $\mathbf{z}$ mit $\|\mathbf{z}\| \leq \beta$ gibt. Damit dies garantiert werden kann, muss β eine Mindestgröße haben.

Lemma 4.2. *Für alle q, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ und $\beta \geq \sqrt{m} \cdot q^{n/m}$ gibt es zu einer* SIS*-Instanz $(q, \mathbf{A}, \beta)$ eine Lösung.*

Man beachte, dass SIS nur durch die Längenbeschränkung der Lösungen $\mathbf{z}$ zu einem schwierigen Problem wird. Ohne diese Einschränkung ist das Problem beispielsweise mit dem Gauß-Algorithmus effizient lösbar.

Der Vollständigkeit halber sei angemerkt, dass das SIS-Problem auch genereller über abelsche (kommutative) Gruppen definiert werden kann.

Definition 4.3 (Allgemeines SIS)**.** Sei G eine kommutative Gruppe. Gegeben ist ein Vektor aus Gruppenelementen $g_1, \ldots, g_m \in G^m$ und ein $\beta \geq 1$. Gesucht ist ein Vektor $\mathbf{z} \in \mathbb{Z}^m$, sodass

$$\sum_{i=1}^{m} z_i g_i = 0 \in G \text{ (neutrales Element der Gruppe)}$$
$$\text{und } \|\mathbf{z}\| \leq \beta.$$

4.1.1 Worst-Case zu Average-Case Reduktion

Miklós Ajtai erzielte 1996 ein bemerkenswertes Ergebnis im Bezug auf gitterbasierte Problemstellungen [Ajt96].

Er zeigte, dass kryptographische Verfahren entwickelt werden können, deren Sicherheit auf der Schwierigkeit beruhen, dass Worst-case-Instanzen bestimmter Gitterprobleme (u.a. SIVP_γ, vgl. Unterabschnitt 3.1.3) schwer zu lösen sind. Dafür definierte er ein Problem, das später unter der Bezeichnung Short Integer Solutions Problem formalisiert wurde (siehe obige Definition). Ajtais Hauptresultat besteht in der Verbindung von SIS im Average-case zu schwierigen Gitterproblemen (u.a. SIVP_γ) im Worst-case.

Theorem 4.4 (Ajtais Theorem (vereinfacht))**.** *Es seinen die Parameter $n, m, q \in \mathbb{N}$ für das* SIS*-Problem so gewählt, dass $n \log q < m \leq \frac{q}{2n^4}$ und $q = \mathcal{O}(n^c)$, für ein konstantes $c > 0$. Außerdem sei $\beta = n$. Angenommen es gibt einen probabilistischen Polynomialzeit-Algorithmus A, der eine zufällige Instanz (gleichverteilt) des Problems* SIS *mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2}$ lösen kann. Dann gibt es einen probabilistischen Algorithmus B, der für* jede *n-dimensionale Instanz von SIVP_γ mit $\gamma = poly(n)$ eine Lösung mit nahezu hundertprozentiger Wahrscheinlichkeit findet.*

In leicht abgewandelter Form seiner Reduktion, kann die Grenze für eine Lösung von SIS auch auf $\beta = n^c$ geweitet werden, sodass auch Lösungen $\mathbf{x}$ mit $\|\mathbf{x}\| \leq n^{c'}$ für ein $c' > 1$ erlaubt sind. Außerdem kann die Wahrscheinlichkeit $\frac{1}{2}$ in der Annahme bzgl. Algorithmus A durch n^{-c} ersetzt werden. Dabei muss allerdings ein Faktor von n^c bei der Laufzeit von Algorithmus B in Kauf genommen werden.

Das besondere an Ajtais Reduktion ist, dass sie sich auf die Lösung von SIVP_γ u.a. in jedem Gitter bezieht. Damit hatte er eine Klasse von Gittern spezifiziert, aus der beliebig gewählt und Einwegfunktionen abgeleitet werden können, wobei deren Sicherheit auf der selben Annahme beruht. Generell zeigte er, dass es möglich ist, kryptographische Verfahren zu konstruieren, die auf der Worst-case Schwierigkeit gewisser Gitterprobleme basieren. Diese herausragende Eigenschaft macht gitterbasierte Kryptographie so interessant. Sie hat damit im Vergleich zu kryptographischen Verfahren, die auf dem Faktorisierungsproblem aufbauen (z.B. RSA), einen entscheidenden Vorteil. Denn bei solchen Verfahren wird auf die Schwierigkeit des Faktorisierens im Durchschnittsfall gebaut. Dafür werden Zahlen aus einer bestimmten Verteilung gewählt, für die angenommen wird, dass diese schwer zu faktorisieren sind. Für die „richtige“ Wahl der Verteilung gibt es jedoch keinen Beweis. Bei Kryptosystemen, die auf der Schwierigkeit beruhen den diskreten Logarithmus in einer Gruppe zu berechnen, ist es ähnlich. Nicht alle Gruppen haben die Eigenschaft, dass in ihnen das Problem des diskreten Logarithmus schwer zu lösen ist. Es gibt auch Gruppen, in denen dies effizient machbar ist. Auch hier ist sorgfältig eine gute Wahl zu treffen. Bei Kryptosystemen, die auf der Worst-Case Schwierigkeit gewisser Gitterprobleme basieren, stellt sich die Frage nach dem „richtigen“ Gitter jedoch nicht (mehr).

4.1.2 Ajtai-GGH Hashfunktion

Aufbauend auf der Worst-case/Average-case Reduktion, konstruierte Ajtai eine Familie von Einwegfunktionen. Da die verwendeten Gitterprobleme als schwer zu lösen gelten, folgt daraus auch die Schwierigkeit der Umkehrbarkeit dieser Einwegfunktionen. Kurze Zeit später fanden Goldreich et al. heraus, dass Ajtais Konstruktion sogar eine Familie kollisionsresistenter kryptographischer Hashfunktionen liefert [GGH96]. Dafür benutzen sie die von Ajtai selbst angemerkte Eigenschaft, dass in SIS auch polynomiell (in n) lange Gittervektoren als Lösungen akzeptiert werden können (s.o.). So konnten sie in einer leicht veränderten Konstruktion Lösungen für SIS der Form $\mathbf{x} \in \{-1, 0, 1\}^m \setminus \{\mathbf{0}\}$ mit $\|\mathbf{x}\| \leq m$ fordern. Daraus folgerten sie wiederum die verbesserte Eigenschaft der Funktion (Kollisionsresistenz).

Algorithmus 2 Hashfunktion nach Ajtais Konstruktion

Parameter: ganzzahlige $n, m, q \geq 1$
Schlüssel: Matrix $\mathbf{A}$, zufällig gleichverteilt gewählt aus $\mathbb{Z}_q^{n \times m}$.
Hashfunktion: $f_\mathbf{A} : \{0,1\}^m \to \mathbb{Z}_q^n$ definiert durch $f_\mathbf{A}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$.

Der Parameter n ist bei dieser Hashfunktion der Sicherheitsparameter. Damit die Funktion die Eingabe komprimiert muss $m > n \log q$ gewählt werden, da m Bits auf $n \log q$ Bits abgebildet werden. Beachtenswert ist, dass eine Kollision $f_\mathbf{A}(\mathbf{x}) = f_\mathbf{A}(\mathbf{x}')$ für $\mathbf{x} \neq \mathbf{x}'$ unmittelbar einen kurzen Vektor $\mathbf{x} - \mathbf{x}'$ in dem assoziierten Gitter $\mathcal{G}_q(\mathbf{A})$ liefert. Denn aus $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \pmod q$ und $\mathbf{x} - \mathbf{x}' \in \{-1, 0, 1\} \subset \mathbb{Z}^m$ folgt $\mathbf{A}(\mathbf{x} - \mathbf{x}') = 0 \pmod q \in \mathcal{G}_q(\mathbf{A})$ und $\|\mathbf{x} - \mathbf{x}'\| \leq \sqrt{m}$. Die Kollisionsresistenz der Hashfunktion ist daher

unmittelbar an das Problem, kurze Vektoren in einem Gitter zu finden, gekoppelt.

Die Hashfunktion kommt mit sehr effizient berechenbaren und leicht zu implementierenden Operationen aus, da nur Integer-Additionen und Multiplikationen modulo q verwendet werden. Diese vorteilhafte Eigenschaft ist typisch für gitterbasierte Kryptographie. Dennoch sind diese Hashfunktionen nicht sehr effizient, weil ihre Schlüsselgröße mindestens quadratisch in der Gitterdimension n ist. Folgendes konkretes Beispiel verdeutlicht dies.

Beispiel 4.1. Seien die Parameter für Ajtais Hashfunktion $q = n^2$ und $m = 2n \log q = 4n \log n$. Der entsprechende Schlüssel (Matrix $\mathbf{A}$) hat dann $m \cdot n = 4n^2 \log n$ Elemente aus $\mathbb{Z}_q$. Ungefähr ebenso viele arithmetische Operationen werden beim Berechnen der Hashfunktion benötigt. Die Komplexität, eine Kollision zu finden, kann mit etwa $L = 2^{m/10}$ abgeschätzt werden (vgl. [DJE09, Kap 5.3]). Um daher eine 100-Bit-Sicherheit zu erreichen (also $L \approx 2^{100}$) muss $m = 4n \log n \approx 1000$ und $n \geq 46$ gesetzt werden. Somit ergibt sich für die Schlüsselgröße der Hashfunktion $mn \log q \approx 500000$bits und es werden ca. $m \cdot n \approx 50000$ arithmetische Operationen zum Berechnen eines Hashwerts gebraucht. Diese Werte werden, im Vergleich zu anderen Hashfunktionen in der Praxis, als nicht akzeptabel angesehen.

Nachfolgende Arbeiten

Regev und Micciancio gelang es in [MR07] Ajtais Resultat zu verbessern, indem sie SIS formalisierten und eine niedrigere Näherungsgrenze für die allgemeinen Gitterprobleme nachwiesen. Sie zeigten, dass SIS im Durchschnitt mindestens so schwer zu lösen ist, wie das Lösen verschiedener Gitterprobleme (GapSVP_γ, SIVP_γ u.a.) für Näherungsfaktoren von $\gamma = n \log^{\mathcal{O}(1)} n$ im schwersten Fall.

Zusammen mit Cynthia Dwork, gelang Ajtai schließlich die Konstruktion eines gitterbasierten Public-Key Kryptosystems, welches auf der (Worst-case) Schwierigkeit von uSVP (vgl. Unterabschnitt 3.1.4) beruht. Dieses Kryptosystem kann als Prototyp gesehen werden. Es hatte hohen theoretischen Wert, da dadurch die Entwicklung gitterbasierter Public-Key-Kryptographie erst richtig begann. Für die Praktische Nutzung ist das Verfahren allerdings nicht gut geeignet, da es im Hinblick auf die Schlüsselgrößen zu ineffizient ist (diese liegen im Bereich $\tilde{\mathcal{O}}(n^4)$ in der Gitterdimension n).

Heute gibt es eine Vielzahl von kryptographischen Verfahren, die auf diese Art von „Sicherheitsbeweis“ bauen. Eine Auswahl davon wird in Kapitel 5 vorgestellt. Nachfolgend wird zunächst ein weiteres Problem präsentiert, auf dem gitterbasierte Kryptographie aufgebaut werden kann.

4.2 Learning With Errors (LWE)

Das Learning with Errors Problem wurde 2005 von Oded Regev definiert [Reg09]. Das Problem besteht darin, ein Geheimnis $\mathbf{s} \in \mathbb{Z}_q^n$ aus einer Sequenz von zufälligen

„verrauschten“ linearen Gleichungen über **s** wiederherzustellen. Eine solche Sequenz kann folgendermaßen aussehen:

$$\begin{aligned} s_1 + 14s_2 + 7s_3 + 3s_4 &\approx 3 \pmod{17} \\ 4s_1 + 6s_2 + 9s_3 + 13s_4 &\approx 6 \pmod{17} \\ 2s_1 + 16s_2 + 7s_3 + 7s_4 &\approx 4 \pmod{17} \\ 15s_1 + 9s_2 + 2s_3 + 8s_4 &\approx 8 \pmod{17} \\ 11s_1 + 5s_2 + 2s_3 + 3s_4 &\approx 5 \pmod{17} \\ &\vdots \end{aligned}$$

Dabei ist jede Gleichung bis auf einen kleinen Fehler korrekt. Durch den eingeführten Fehler ist es schwer den Vektor **s** zu finden, wohingegen die Lösung für das Gleichungssystem ohne den Fehler effizient durch das Gaußverfahren berechenbar ist.

Die formale Definition des Problems lautet wie folgt. Dabei bedeutet die Notation $s \leftarrow S$, dass s zufällig gleichverteilt aus einer Menge S gewählt wird.

Definition 4.5 (LWE)**.** Es seinen folgende Parameter gegeben: Die Dimension des Problems, $n \in \mathbb{N}$ und ein $q \geq 2$, welches als Modulus genutzt wird (q ist üblicherweise polynomiell in n). Zusätzlich wird eine Wahrscheinlichkeitsverteilung $\mathcal{X}$ über $\mathbb{Z}/q\mathbb{Z}$ gewählt. Mit $\mathcal{X}$ soll ein Rauschen erzeugt werden, wobei eine Auswahl $e \leftarrow \mathcal{X}$ möglichst kleine Werte $\sqrt{n} \leq e \ll q$ haben soll. In der Praxis wird üblicherweise die Normalverteilung mit Standardabweichung $q\alpha$ für ein $\alpha > 0$ verwendet. Um die Verteilung zu diskretisieren wird dabei zur nächsten ganzen Zahl gerundet und modulo q gerechnet.

LWE wird in zwei Varianten definiert.

Suche Sei $\mathcal{O}_{\mathbf{s}}^n$ ein Orakel, welches Proben der Form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ ausgibt. Dabei wird $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ für jede Probe zufällig neu gewählt. Das Geheimnis $\mathbf{s} \in \mathbb{Z}_q^n$ bleibt jedes mal gleich. Der Fehler $e \leftarrow \mathcal{X}$ wird jeweils zufällig aus $\mathcal{X}$ entnommen.

Das Such-LWE-Problem besteht darin, das Geheimnis **s** mit Zugriff auf das Orakel $\mathcal{O}_{\mathbf{s}}^n$ heraus zu finden. Die Anzahl der Orakelzugriffe kann durch einen Parameter $m \in \mathbb{N}$ beschränkt werden.

Entscheidung Sei $\mathcal{O}_{\mathbf{s}}^n$ ein Orakel wie oben. Zusätzlich sei $\mathcal{R}$ ein Orakel, dass zufällig gleichverteilte Proben der Form $(\mathbf{a}, b) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$ ausgibt.

Das Entscheidungsproblem LWE besteht darin, zwischen beiden Orakeln zu unterscheiden. Eine Interaktion mit einem fixen, unbekanntem Orakel ($\mathcal{O}_{\mathbf{s}}^n$ oder $\mathcal{R}$) kann wiederum quantitativ auf m begrenzt werden.

Eine Anzahl von m Proben des Orakels $\mathcal{O}_{\mathbf{s}}^n$ der Form

$$(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + e_1), \ldots, (\mathbf{a}_m, \langle \mathbf{a}_m, \mathbf{s} \rangle + e_m)$$

kann kompakt in Matrixschreibweise dargestellt werden:

$$\mathbf{A}, \mathbf{As} + \mathbf{e}$$

für eine Matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ mit Zeilen $\mathbf{a}_1, \ldots, \mathbf{a}_m$ und $\mathbf{e} \leftarrow \mathcal{X}^m$.

Die Entscheidungsvariante von LWE besteht somit darin, für ein gegebenes Paar $(\mathbf{A}, \mathbf{v})$ zu entscheiden, ob $\mathbf{v}$ entweder zufällig gleichverteilt aus $\mathbb{Z}_q^m$ gewählt wurde, oder $\mathbf{As} + \mathbf{e}$ mit zufällig gleichverteilt gewähltem $\mathbf{s} \in \mathbb{Z}_q^n$ und $\mathbf{e} \in \mathbb{Z}_q^m$ aus $\mathcal{X}^m$ entspricht.

Für kryptographische Anwendungen wird die Entscheidungsvariante von LWE verwendet. Regev gab bei der Definition des Problems auch eine (probabilistische) Reduktion von der Such-Variante von LWE zur Entscheidungsvariante an. Mittlerweile konnte durch nachfolgende Arbeiten von [Pei09] bis hin zu [BLPRS13] eine Äquivalenz beider Varianten, bis auf einen polynomiellen Anstieg in der Anzahl der benötigten Proben m gezeigt werden.

Regev gab bei der Vorstellung von LWE eine Worst-case/Average-case Reduktion von Gitterproblemen wie GapSVP_γ und SIVP_γ zu LWE an. Diese Reduktion geschieht allerdings mithilfe von Quantenalgorithmen. Ein effizientes Lösen einer LWE-Instanz würde also (nur) einen Quantenalgorithmus zum Lösen von GapSVP_γ und SIVP_γ implizieren. Mittlerweile wurden auch „klassische“ Reduktionen gefunden, erstmals von Peikert in [Pei09] (mit der Einschränkung auf GapSVP_γ). Daher wird LWE ebenso wie SIS als (sehr) schweres Problem eingestuft. Außerdem existiert eine (quantenalgorithmische) Reduktion von SIS zu LWE [SSTX09].

4.2.1 LWE-basiertes Kryptosystem

LWE stellte sich als sehr vielseitige Grundlage verschiedener kryptographischer Anwendungen heraus. So gibt es eine Menge von Kryptosystemen unterschiedlicher Art, die auf dem Problem basieren. Nachfolgend soll Regevs prototypisches LWE-Kryptosystem aus [Reg10] vorgestellt werden.

Algorithmus 3 LWE-Kryptosystem

- **Parameter:** Sicherheitsparameter n, Anzahl an Gleichungen m, Modulus q und ein reelles $\alpha > 0$ (Rauschparameter). Um semantische Sicherheit und Korrektheit gewährleisten zu können, sollte q eine Primzahl zwischen n^2 und $2n^2$ sein. Außerdem sollten $m = 1.1 \cdot n \log q$ und $\alpha = 1/\sqrt{n} \log^2 n$ gesetzt werden. Nachfolgend werden alle Additionen modulo q ausgeführt.
- **Private Key:** Der Private-Key ist ein zufällig gleichverteilt gewähltes $\mathbf{s} \leftarrow \mathbb{Z}_q^n$.
- **Public Key:** Der Public-Key besteht aus m Proben $(\mathbf{a}_i, b_i)$ einer LWE-Verteilung, mit $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e$ und $1 \leq i \leq m$ (wie oben beschrieben).
- **Verschlüsselung:** Jedes Bit einer Nachricht wird folgendermaßen verschlüsselt. Wähle eine Menge S aus der 2^m-elementigen Potenzmenge von $\{1, \ldots, m\}$ aus. Die Verschlüsselung entspricht $(\sum_{i \in S} \mathbf{a}_i, \sum_{i \in S} b_i)$, falls das Eingabebit 0 ist und $(\sum_{i \in S} \mathbf{a}_i, \lfloor \frac{q}{2} \rfloor + \sum_{i \in S} b_i)$, falls es 1 ist.
- **Entschlüsselung:** Ein Paar $(\mathbf{a}, b)$ wird zu 0 entschlüsselt, wenn $b - \langle \mathbf{a}, \mathbf{s} \rangle$ näher an 0 liegt, als an $\lfloor \frac{q}{2} \rfloor$ (modulo q). Ansonsten wird $(\mathbf{a}, b)$ zu 1 entschlüsselt.

Korrektheit. Ohne den eingefügten Fehler in LWE wäre die Zuordnung beim Entschlüsseln eindeutig ($b - \langle \mathbf{a}, \mathbf{s} \rangle$ wäre entweder 0 oder $\lfloor \frac{q}{2} \rfloor$). Entschlüsselungsfehler treten auf, wenn die Summe der Fehlerterme über alle S größer als $q/4$ ist. Es werden höchstens m normalverteilte Fehlerterme mit Standardabweichung αq hinzugerechnet. Daher ist die Standardabweichung der Summe höchstens $\sqrt{m} \cdot \alpha q < q/\log n$. Das macht es sehr unwahrscheinlich, dass dies den Wert $q/4$ übersteigt. Solche Entschlüsselungsfehler können ein Sicherheitsrisiko darstellen, da dies spezielle Angriffe ermöglicht (siehe dazu Abschnitt 5.5). Daher ist es wichtig die Wahrscheinlichkeit für solche Fehler durch entsprechende Wahl der Parameter gering zu halten.

Der Vorteil von Regevs Kryptosystem, gegenüber dem von Ajtai und Dwork besteht in zwei Punkten. Erstens baut es durch LWE auf den Worst-case Gitterproblemen GapSVP_γ und SIVP_γ auf. Diese weisen „weniger Struktur“ (vgl. [Pei15a, Kap. 4.2.3]) als uSVP_γ auf und sind daher schwieriger zu Lösen. Zweitens ist es bezogen auf Public-Key- und Geheimtextgröße effizienter. Der Public-Key ist $\tilde{\mathcal{O}}(n^2)$ Bits anstatt $\tilde{\mathcal{O}}(n^4)$ Bits groß und der Geheimtext hat eine Größe von $\tilde{\mathcal{O}}(n)$, anstatt $\tilde{\mathcal{O}}(n^2)$ Bits pro verschlüsseltem Klartext-Bit. Diese Werte sind immer noch hoch, sodass verschiedene Methoden entwickelt wurden LWE-basierte Kryptosysteme effizienter zu machen. Außerdem ist Regevs Kryptosystem nicht gegen Chosen Ciphertext Angriffe geschützt.

Peikert entwickelte 2009 ein verbessertes Public-Key-Kryptosystem [Pei09], dass den Anspruch erhebt, sowohl effizient als auch sicher gegen Chosen Ciphertext Angriffe zu sein. In den Nachfolgenden Jahren wurden weitere Verbesserungen und Variationen von LWE-basierten Kryptosystemen entworfen. Eine Übersicht über diese Entwicklung

bietet [Pei15a].

4.3 Weitere Möglichkeiten gitterbasierter Kryptogaphie: (Voll)-homomorphe Verschlüsselung

Eine bemerkenswerte Möglichkeit, die Gitter in der Kryptographie bieten, ist homomorphe Verschlüsselung. Die Idee dahinter ist, dass Berechnungen auf verschlüsselten Daten ausgeführt werden können, sodass diese einer Operation auf dem Klartext entsprechen. So soll es ermöglicht werden, verschlüsselte Daten auf (Cloud-)Servern zu speichern und (rechenintensive) Funktionen auf den Daten berechnen zu lassen, ohne, dass der Server den Klartext kennen muss. Anstatt lokal für ein x eine Funktion $f(x)$ zu berechnen, kann dies leistungsstarken Rechnern eines Anbieters überlassen werden. Da mitunter aber sensible Daten verarbeitet werden, sollen diese verschlüsselt übertragen und weiterverarbeitet werden. Ein homomorphes Verschlüsselungsverfahren $(\text{enc}(x))$ ermöglicht es, dass ein externer (möglicherweise nicht-vertrauenswürdiger) Anbieter $f(\text{enc}(x))$ berechnen kann. Das Ergebnis wird nun zurück übertragen und lokal entschlüsselt, sodass $f(x)$ zurückgewonnen wird, also $\text{dec}(f(\text{enc}(x))) = f(x)$.

Craig Gentry entwickelte erstmals ein voll-homomorphes Verschlüsselungsverfahren, welches er 2009 in seiner Dissertation [Gen09] veröffentlichte. Vor dieser Arbeit gab es homomorphe Verschlüsselungsverfahren, bei denen jedoch eine Beschränkung auf bestimmte Funktionen f besteht, die auf verschlüsselten Daten evaluiert werden können. Bei voll-homomorpher Verschlüsselung besteht keine solche Beschränkung der Funktionen. Gentrys Arbeit war ein Durchbruch auf dem Gebiet der homomorphen Verschlüsselung, denn bis zu dieser Veröffentlichung war es eine offene Frage in der Kryptographie, ob man ein voll-homomorphes Verschlüsselungsverfahren entwickeln kann. Demnach zog dieses Resultat die Aufmerksamkeit von vielen Forschern auf sich, die beständig an Verbesserungen und praktischer Nutzbarkeit eines solchen Verschlüsselungssystems arbeiten. Peikert bietet in [Pei15a, Kap. 6] eine Übersicht über die Entwicklung voll-homomorpher Verschlüsselung in den letzten Jahren. Ein aktueller Vergleich homomorpher Kryptosysteme wird in [CS16] gegeben.

5 Aktuelle Entwicklungen gitterbasierte Kryptosysteme

In diesem Kapitel soll ein kurzer Ausschnitt über aktuelle Entwicklungen gitterbasierter Kryptosysteme gegeben werden. Ein großer Teil der Analyse und Verbesserung von Post-Quantum-Kryptosystemen geschieht im Zuge des aktuellen Standardisierungsprozess vom National Institute of Standards and Technology. Dieser wird nachfolgend, inklusive zweier kandidierender Kryptosysteme, vorgestellt. Anschließend werden mögliche Angriffe und Risiken beim Einsatz solcher Systeme thematisiert.

5.1 NIST-Wettbewerb

In dem in der Einleitung erwähnten Post-Quantum-Kryptographie Wettbewerb (kurz PQC-Wettbewerb) des NIST (Abschnitt 1) werden zur Zeit eine Vielzahl an kryptographischen Verfahren analysiert und bewertet. Dabei geht es vor allem um die Sicherheit der Verfahren gegenüber Angriffen von Quantencomputern. Ausgerufen wurde der Wettbewerb im Jahr 2016. Die Deadline für das Einreichen von Public-Key-Kryptosystemen (asymmetrische Verschlüsselungsalgorithmen, Signaturverfahren und Schlüsselaustauschverfahren) endete am 30. November 2017. Laut einer Übersicht[1] eines NIST-Mitarbeiters wurden hauptsächlich gitterbasierte, Code-basierte, Multivariate und Hash-basierte Verfahren eingereicht, wobei die meisten (28) der eingereichten Verfahren gitterbasiert sind. Derzeit werden 69 eingegangene Vorschläge[2] überprüft und ausgewertet. Nach mehreren Auswahlrunden sollen nach aktuellem Zeitplan[3] zwischen 2022 und 2024 Entwürfe für standardisierte Post-Quantum Kryptosysteme vorliegen.

Eine erste Analyse der eingereichten Implementierungen von Post-Quantum-Kryptographie zeigt, dass gitterbasierte Verfahren sowohl bei der Performance (etwa der Ver- und Entschlüsselungszeit) als auch bei der Schlüsselgröße vorteilhafter sind als andere Verfahren. In Abbildung 5.1 werden Ausschnitte aus dieser vorläufigen Analyse gezeigt, die auf der PQCRYPTO-2018 vom NIST präsentiert wurden.

Zur Zeit (2018/2019) läuft die zweite Runde des Wettbewerbs, in der die bisher angenommenen Einreichungen näher untersucht werden. Bis zur zweiten PQC-Standardisierungskonferenz, die voraussichtlich im August 2019 stattfinden wird, werden dann die vielversprechendsten Kandidaten für die dritte (und letzte) Runde ausgewählt.

[1] https://twitter.com/DemocraticLuntz/status/937702005631586304
[2] https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
[3] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Workshops-and-Timeline

Disskussion

Wichtigster Bestandteil des NIST-Wettbewerbs ist der Review-Prozess der eingereichten Kryptosysteme. Die Details jedes Kryptosystems sind offen einsehbar und es musste jeweils eine Referenz-Implementierung mit abgegeben werden. Somit wird es allen teilnehmenden Kryptographen ermöglicht die Arbeiten der konkurrierenden Kryptosysteme zu analysieren und in einem Forum[4] zu kommentieren. Dieser offene Diskurs führte schon kurz nach Ende der Einreichungsfrist dazu, dass mehrere Kandidaten die erste Runde des Wettbewerbs nicht überstanden. Es wurden teilweise signifikante Schwächen in den Kryptosystemen gefunden oder sie wurden sogar komplett gebrochen und somit für die nächste Bewertungsrunde aussortiert. In drei Fällen wurde die Einreichung bereits zurückgezogen.

Nicht immer sind die Schlussfolgerungen der Kryptoanalysten im Review-Prozess unumstritten. Vor Allem bei der Bewertung der (theoretischen) Sicherheit eines Kryptosystems kommt es teilweise zu hitzigen Diskussionen[5] unter den Forschern.

Alle Informationen zum weiteren Verlauf des Wettbewerbs lassen sich unter `csrc.nist.gov/projects/post-quantum-cryptography` finden.

5.2 Google-Experiment

Schon im Juli 2016 experimentierte Google mit Implementierungen von gitterbasierten Kryptosystemen und baute ein Schlüsselaustausch-Verfahren namens „NewHope“ [**newhope:cryptoeprint:2015:1092**] in eine Entwickler-Version von Google Chrome (Canary) ein[6]. Mit dieser Chrome-Version konnte man auf manchen Google-Webseiten schon den neuen Schlüsselaustausch ausprobieren. NewHope hielten die Google-Forscher Ende 2015 für den vielversprechendsten Kandidaten für Post-Quantum-Schlüsselaustausch-Verfahren. Google wollte mit seiner Wahl aber „explizit keinen de facto Standard setzen“ und will auch andere Verfahren wie bspw. „NTRUprime“ [BCLV16] oder „Frodo“ [Bos+16] näher auf Praxistauglichkeit untersuchen. All diese Verfahren sind auch Kandidaten beim laufenden NIST-Wettbewerb für Post-Quantum-Kryptographie.

5.3 NTRU

NTRU [HPS98] ist ein Public-Key-Kryptosystem und wurde von Hoffstein, Pipher und Silverman entwickelt. Es zählt zu den ersten Konstruktionen gitterbasierter Kryptographie und enthält Verfahren für asymmetrische Verschlüsselung (NTRUEncrypt) und zur Erstellung von Signaturen (NTRUSign). NTRU steht dabei für „Nth Degree Truncated Polynomial“.

Nachfolgend soll NTRUEncrypt kurz skizziert werden (vgl. [Pei15a, Kap. 3.2]).

[4] `https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum`
[5] `https://groups.google.com/d/topic/cryptanalytic-algorithms/BoSRL0uHIjM/discussion`
[6] `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`

Algorithmus 4 NTRU-Kryptosystem

- **Parameter:** Wie auch Ring-LWE-basierte Kryptosysteme verwendet NTRU bestimmte Ringstrukturen um das Kryptosystem effizienter zu machen. Bei NTRU wird hierzu ein Ring der Form $R = \mathbb{Z}[X]/f(X)$ genutzt. Beispielsweise kann $f(X) = (X^n - 1)$ für eine Primzahl n oder $f(X) = (X^n + 1)$ für $n = 2^d$ verwendet werden. Für ein ausreichend großen und ungeraden Modulus q wird nun der Quotientenring $R_q = R/qR$ gebildet. Die Ringelemente können so leicht als ganzzahlige Vektoren in $\mathbb{Z}^n$ dargestellt werden.
- **Public-Key:** Der Public-Key ist $h = 2g \cdot s^{-1} \in R_q$ für zwei „kurze“ Polynome $g, s \in R$ (mit relativ kleinen ganzzahligen Koeffizienten).
- **Private-Key:** s ist der Private-Key, welcher invertierbar modulo q und 2 sein muss.
- **Verschlüsselung:** $c = h \cdot r + e \in R_q$. Hier ist $r \in R$ ein „Rauschfaktor“ und in den Koeffizienten (modulo 2) von $e \in R$ werden die zu verschlüsselnden Nachrichtenbits codiert.
- **Entschlüsselung:** Für die Entschlüsselung wird mit dem Private-Key multipliziert: $c \cdot s = 2g \cdot r + e \cdot s \in R_q$. Das Ergebnis kann als „kleines“ Element von R interpretiert werden, da g, r, e und s kleine Koeffizienten haben. So kann man $e \cdot s$ modulo 2 gewinnen und daraus die Nachricht dekodieren.

Im Gegensatz zu modernen Gitter-Kryptosystemen hat das ursprüngliche NTRU keinen semantischen Sicherheitsbeweis durch eine Worst-case/Average-case Reduktion auf schwierige Gitterprobleme. Mittlerweile konnte für eine NTRU-Variante ein Sicherheitsbeweis, der auf der Schwierigkeit von Ring-LWE basiert gezeigt werden [SS13]. Die Verbindung zu LWE-Kryptosystemen gelang aufgrund dessen semantischer Ähnlichkeit.

Daniel J. Bernstein, Nadia Heninger, und Tanja Lange geben auf der Webseite `https://latticehacks.cr.yp.to` eine niedrigschwellige Einführung in das NTRU-Kryptosystem. Dort ist eine Anleitung zum Ausprobieren von NTRU in *sagemath*[7] im Zusammenspiel mit ihrem Vortrag[8] auf dem 34. Chaos Communication Congress zu finden. Dabei können die mathematischen Funktionen, die in NTRU benötigt werden, sowie die Ver- und Entschlüsselung selbst implementiert werden und an kleinen Beispielen nachvollzogen werden. Die Autoren weisen darauf hin, dass die angegebene Implementierung nur der Veranschaulichung dient und nicht in praktischen Implementierungen eingesetzt werden sollte, da sie anfällig gegen diverse Angriffe ist. Dies führen Bernstein und co. auch in anschaulicher Form in ihrem Vortrag vor. Außerdem vermitteln sie einen Eindruck, wie Gitter zur Kryptoanalyse von RSA genutzt werden können. Sie zeigen eingängig auf, dass eine Implementierung von „Schulbuch“-RSA keine ausreichende Sicherheit gegen praktische Angriffe bietet.

[7] Freies Open Source Mathematiksystem mit Python-basierter Benutzerschnittstelle
[8] `https://www.youtube.com/watch?v=QCsvBFFLsqE`

Seit der Vorstellung des NTRU-Kryptosystems vor gut 20 Jahren, wurde es vielfach auf Schwächen untersucht. In Folge dessen wurden eine Reihe von Verbesserungen und angepasste Parametersets entwickelt. Zum Beispiel besteht beim NTRU-Kryptosystem eine gewisse Fehlerwahrscheinlichkeit bei der Entschlüsselung von Daten. Die Wahrscheinlichkeit dafür konnte mittlerweile jedoch durch umfassende Verbesserungen und entsprechende Wahl der Parameter deutlich verringert werden. Im neuesten Technischen Report[9] der NTRU-Entwickler wird für die Wahrscheinlichkeit eines Entschlüsselungsfehlers, bei empfohlenen Parametern, der Wert 2^{-100} angegeben.

Es gibt allerdings auch grundsätzliche Kritik an NTRU. Die Strukturen, die in dem Kryptosystem für Effizienzsteigerungen sorgen (allen voran die Verwendung des Polynomrings $R = \mathbb{Z}[X]/(X^n - 1)$), bieten eine (potentielle) Angriffsfläche. Daher wurde die Erweiterung NTRU-Prime entwickelt.

5.3.1 NTRU-Prime

Aufbauend auf ihrer Analyse des NTRU-Kryptosystems haben Bernstein, Chuengsatiansup, Lange und van Vredendaal eine verbesserte Version namens NTRU-Prime [BCLV16] entwickelt. Ziel dieser NTRU-Variante ist es, bekannte Schwächen und mögliche Angriffspunkte des originalen NTRU-Kryptosystems zu vermeiden. Auf der Webseite `https://ntruprime.cr.yp.to` stellen sie eine Familie von NTRU-Prime-Kryptosystemen mit „high-security post-quantum“ Parametern vor.

Unterschiede zu NTRU

Der Hauptunterschied zwischen NTRU und NTRU-Prime ist der verwendete Polynom-Ring, der in den Ver- und Entschlüsselungsfunktionen genutzt wird. NTRU-Prime verwendet einen Quotientenkörper.

Der mathematische Unterbau von NTRU-Prime ist mit folgenden Parametern gegeben:

- Wähle Primzahl p und definiere $F = X^p - X - 1$.
- Wähle Primzahl q, sodass F mod q irreduzibel ist.
- Damit ist $\mathbb{Z}_q[X]/(X^p - X - 1)$ ein Körper mit Galoisgruppe S_p der Größe p.

5.4 Potentielle Schwächen von gitterbasierten Kryptosystemen

Die Entwickler der NTRU-Prime-Kryptosysteme warnen auf ihrer Projektwebseite[10], davor sich voreilig auf die Sicherheit von aktueller gitterbasierter Kryptographie zu verlassen. Sie weisen dabei auf die Risiken weiteren Fortschritts unter anderem in folgenden Bereichen hin:

[9] `https://www.onboardsecurity.com/products/ntru-crypto/ntru-resources`
[10] `https://ntruprime.cr.yp.to/warnings`

- Die Entwicklung der Algorithmen zum Lösen von SVP in den letzten 10–15 Jahren. Während vor ca. 15 Jahren der beste SVP-Algorithmus eine Laufzeit von $2^{\Theta(n \log n)}$ hatte, wurde zwischenzeitlich eine Verbesserung auf $2^{(c+o(1))n}$ mit $c \approx 0,415$ erreicht. Mittlerweile (2015) konnte c auf einen Wert von ungefähr $0,292$ optimiert werden [BDGL15].

- Algorithmen, die die allgemeine Struktur von kryptographischen Problemen wie LWE ausnutzen.

- Algorithmen, die die Struktur bestimmter Ringe, die in Ring-LWE-basierten Kryptosystemen verwendet werden, ausnutzen [CDW16].

- Angriffe auf gitterbasierte Kryptosysteme, die ohne das Brechen von bestimmten Gitterproblemen auskommen [HGS99].

Aufgrund diesem Verlauf und der noch vergleichsweise jungen Entwicklungsstufe gitterbasierter Kryptographie mahnen die Forscher zur Vorsicht und stellen fest, dass es noch weiterer Analyse diverser Angriffsvektoren auf gitterbasierte Kryptosysteme bedarf. Auch ihre eigenen NTRU-Prime-Kryptosysteme sind erst kürzlich veröffentlicht worden (Streamlined NTRU Prime Mai 2016 und NTRU LPRime Dezember 2017). Daher halten es die Entwickler für unbedingt notwendig, dass die NTRU-Prime-Kryptosysteme eine umfassende Sicherheitsüberprüfung erhalten. Dies geschieht unter anderem durch den Review-Prozess im NIST PQC-Wettbewerb, bei dem auch die NTRU-Prime Varianten beteiligt sind.

Kurz vor der Fertigstellung dieser Arbeit wurde ein Paper [Li18] veröffentlicht, in dem der Autor einen Angriff auf das Streamlined NTRU Prime Kryptosystem beschreibt. Dabei will er gewisse Eigenschaften des im Kryptosystem verwendeten Rings ausgemacht haben, welche das Kryptosystem entscheidend schwächen sollen. In dem Diskussionsforum[11] des NIST PCQ-Wettbewerbs wies Bernstein diese Behauptung allerdings zurück. Bernstein könne keinen konkreten Angriff auf Streamlined NTRU Prime in dem Paper erkennen und sehe nicht, dass Schwächen in dem bei NTRU-Prime verwendeten Ring aufgezeigt würden. Auch ein weiterer Kommentator hält den Angriff für inkorrekt.

Eine Forschergruppe hat es unternommen alle LWE-basierten Kryptosysteme und NTRU-Varianten, die am NIST PQC-Wettbewerb teilnehmen, zu untersuchen [Alb+18]. Dabei stellten sie die prognostizierte Sicherheit der Kryptosysteme den asymptotischen Laufzeiten gängiger Basisreduktions-Algorithmen für Gitter und anderer Angriffsmethoden gegenüber. Eine Übersicht ihrer Ergebnisse kann man unter `https://estimate-all-the-lwe-ntru-schemes.github.io/docs/` finden. Über die Aussagekraft und die Schlussfolgerungen, die aus den Ergebnissen gezogen werden können, wird ebenfalls im NIST-PQC-Forum diskutiert[12,13].

[11] `https://groups.google.com/a/list.nist.gov/d/topic/pqc-forum/W_3OPJYTDsU/discussion`
[12] `https://groups.google.com/a/list.nist.gov/d/topic/pqc-forum/C9erDb3QG6U/discussion`
[13] `https://groups.google.com/a/list.nist.gov/d/topic/pqc-forum/l5IaJTe_pUI/discussion`

5.5 Angriffe gegen gitterbasierte Public-Key-Kryptosysteme

In [HGS99] beschreiben Hall, Goldberg, und Schneier Attacken auf das Code-basierte McEliece Public-Key-Kryptosystem und die gitterbasierten Public-Key-Kryptosysteme von Ajtai und Dwork [AD97] sowie das darauf aufbauend, verbesserte Kryptosystem von Goldreich, Goldwasser und Halevi [GGH97]. Die dort beschriebenen Angriffe sind „Reaktions-Angriffe“ und erlauben es dem Angreifer bei den gitterbasierten Kryptosystemen sogar an den Private-Key zu gelangen. Bei einem Reaktions-Angriff kann ein Angreifer dem System speziell zusammengesetzte Geheimtexte (*chosen-ciphertext*) zum Entschlüsseln übergeben und aus der Reaktion des Systems Informationen über den entsprechenden Klartext oder gar den Private-Key erhalten. Die analysierten Kryptosysteme haben gemeinsam, dass sie als „closest-point cryptosystems“ eingeordnet werden können. In solch einem System hängt die Fähigkeit einen Geheimtext entschlüsseln zu können unmittelbar damit zusammen, den nächstgelegenen „Punkt“ zu dem Geheimtext in einem bestimmten linearen Raum ermitteln zu können. Im Fall von gitterbasierten Systemen ist dies das *Closest Vector Problem* (siehe Abschnitt 3.2). Die Autoren weisen darauf hin, dass prinzipiell alle Kryptosysteme, die auf einem solchen „closest-point cryptosystem“ beruhen, für ihre Attacken anfällig sein könnten. In ihrem Fazit [HGS99, S. 9] schreiben sie:

> „We feel that the existance of these attacks effectively limits these ciphers to theoretical considerations only. That is, any implementation of the ciphers will be subject to the attacks we present and hence not safe. We should point out that we have not broken these cryptosystems in the sense that we are able to mathematically derive the private key. Instead our attacks rely upon the fact that someone within the system knows the private key (and hence the answer to the intractible problem).“

In [HS00b] adaptierten Hoffstein und Silverman diese Art von Angriff auf das NTRU-Kryptosystem. Sie geben jedoch auch Möglichkeiten an Reaktions-Angriffe zu erkennen und diese zu vereiteln. In dem Nachfolgepapier [HS00a] wird außerdem eine Padding Technik von Fujisaki und Okamoto [FO13] für NTRU erläutert, die Public-Key-Kryptosysteme gegen aktive Angreifer absichern soll. Dies hat jedoch den Nachteil, dass das System zusätzlich komplexer wird und die Performance beeinträchtigt wird. Außerdem schützt die Modifizierung nicht vor allen Attacken, denn in [How+03] konnte gezeigt werden, dass valide Geheimtexte zu Entschlüsselungsfehlern führen können, woraus sich wiederum Informationen gewinnen lassen.

Wie bei anderen Public-Key-Verfahren (z.B. RSA) auch, folgt also aus der theoretischen Sicherheit, die auf den zugrundeliegenden mathematischen bzw. algorithmischen Probleme basiert, nicht automatisch eine praktisch anwendbare Sicherheit (bspw. CPA[14]- oder CCA[15]-Sicherheit) der gitterbasierten Kryptosysteme.

[14] Chosen Plaintext Attack
[15] Chosen Ciphertext Attack

5.5.1 Angriffe mit Quantenalgorithmen

Seit mehr als 20 Jahren wird daran gearbeitet Quantenalgorithmen zu entwerfen, die Gitterprobleme effizient lösen können. Dies ist bisher nicht gelungen und lässt Entwickler von Kryptosystemen, die auf Gittern basieren, Hoffnung schöpfen, dass diese auch in Zukunft als sicher gelten können. Gitterbasierte Kryptosysteme sind unter anderem deshalb die vielversprechendsten Kandidaten für Post-Quantum-Kryptographie. Um die Robustheit von gitterbasierten Kryptosystemen gegenüber Quantencomputern zu belegen, bedarf es allerdings noch viel Forschungsarbeit. Nach heutigem Stand haben Angriffe mit Quantenalgorithmen zwar einen gewissen Vorteil gegenüber klassischen Angriffen, es können allerdings immer noch hohe Sicherheitsabschätzungen gegeben werden. So beispielsweise in [GVW17], wo u.a. die Sicherheit der LWE-basierten Kryptosysteme New Hope und Frodo mit einem neuartigen Quantenalgorithmus analysiert wurde. Die Autoren geben für den dualen Logarithmus der benötigten Laufzeit eines Quantenangriffs auf New Hope einen Wert von 384 an. Diese Einschätzung liegt damit noch Höher als die von den New Hope Autoren selbst, welche einen konservativen Schätzwert von mindestens 200 (2er Logarithmus der Laufzeit) nennen [ADPS16].

Es gibt jedoch immer wieder Ansätze, die die Sicherheit gitterbasierter Kryptographie gegenüber Quantenalgorithmen in Frage stellen könnten. Lior Eldar und Peter W. Shor veröffentlichten 2016 ein Paper, das einen Quantenalgorithmus für eine Variante des Closest Vector Problems zeigt[16]. Dieser hat eine polynomielle Laufzeit und würde somit eine der Sicherheitsannahmen von LWE-Kryptosystemen im Bezug auf Quantencomputer in Frage stellen. In Folge dessen wäre die Eignung gitterbasierter Kryptosysteme als Kandidaten für Post-Quantum Kryptographie stark geschwächt. In dem Algorithmus wurden allerdings Fehler gefunden, was zu einem Widerruf des Papers seitens der Autoren geführt hat. Bisher konnten diese Fehler nicht behoben werden, aber die Autoren arbeiten weiterhin an einer Lösung des Problems. Die Frage, ob gitterbasierte Kryptosysteme auch Attacken von zukünftigen Quantencomputern standhalten können, bleibt daher vorerst offen.

5.6 Implementierungen

Mittlerweile existieren mehrere Software-Bibliotheken, die hoch optimierte Operationen für kryptographische Anwendungen bereitstellen. Für Ideal-Gitter-Kryptosysteme gibt es die quelloffene NFLlib[17] [Agu+16].

Das Projekt „Open Quantum Safe“[18] entwickelt Prototypen von Post-Quantum-Kryptosystemen in C und stellt diese öffentlich auf `github.com` zur Verfügung. Es wird außerdem eine API angeboten und die implementierten Kryptoverfahren lassen sich schon in geforkten Varianten von OpenSSL und OpenSSH ausprobieren.

[16] `https://arxiv.org/abs/1611.06999`
[17] `https://github.com/quarkslab/NFLlib`
[18] `https://openquantumsafe.org/`

Ein weiteres Post-Quantum-Kryptographie-Projekt ist PQCRYPTO[19], welches von der EU finanziert wird. Mitglieder sind acht (hauptsächlich europäische) Universitäten, sowie drei Industriepartner. Zusammen reichten die 11 Organisationen 22 Kryptosysteme beim NIST PQC-Wettbewerb ein. Das Projekt bietet eine Software-Bibliothek[20] der entwickelten Kryptosysteme an und stellt u.a. Python- und C-APIs für diese bereit. Außerdem werden im Zuge des Projekts Konferenzen zum Thema Post-Quantum-Kryptographie organisiert.

5.7 Vergleich zu etablierter Public-Key-Kryptographie

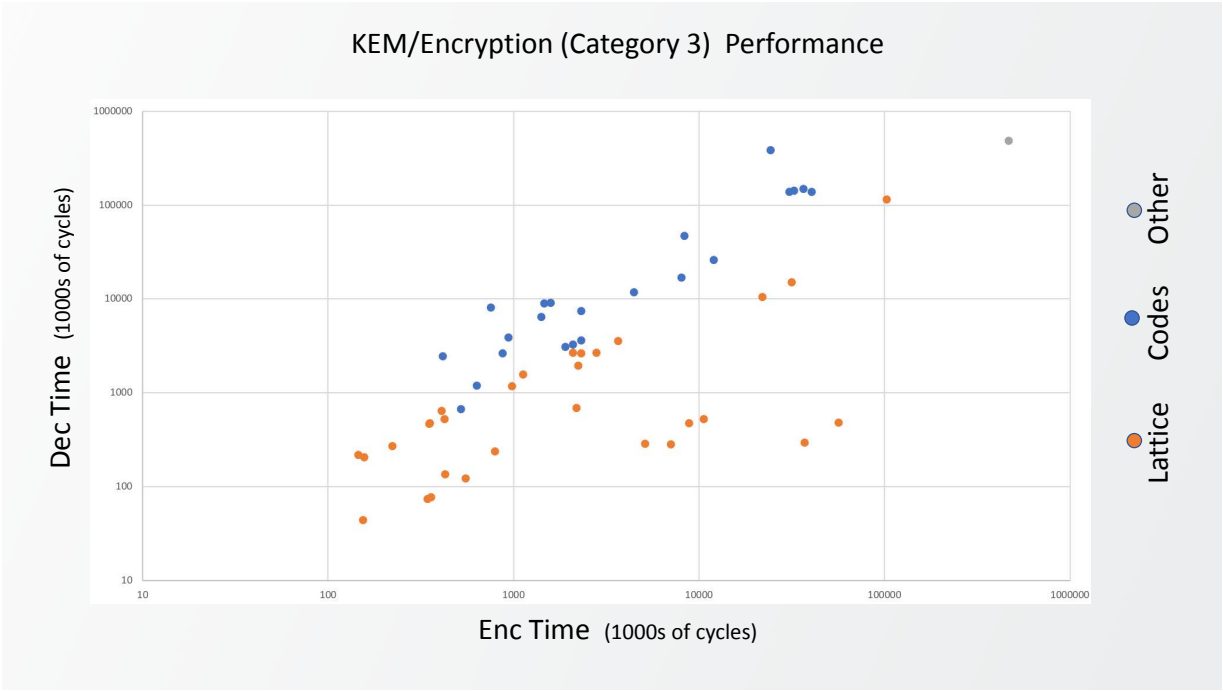
In einem Vortrag auf der „Winter School on Cryptography" 2012 an der Bar Ilan University[21] beschreibt Oded Regev die wesentlichen Vorteile von gitterbasierter Kryptographie gegenüber heutzutage etablierten Public-Key-Verfahren wie RSA, Diffie-Hellman Schlüsselaustausch und Verfahren die auf elliptischen Kurven basieren.

- Einfache und effiziente Implementationen, da hauptsächlich Integer-Addition und Multiplikation (modulo q) die grundlegenden Operationen sind.
- Die Sicherheit bestimmter gitterbasierter Kryptosysteme kann nachweislich auf die Worst-case Schwierigkeit von Gitterproblemen zurückgeführt werden.
- Mithilfe von Gittern ist es möglich voll-homomorphe Verschlüsselungssysteme zu konstruieren.
- Es gibt (bisher) keine Quantenalgorithmen, die grundlegende Gitterprobleme erheblich schneller lösen können als klassische Algorithmen.

5.8 Offene Probleme bezüglich aktueller Kryptosysteme

Der NIST-Mitarbeiter Jacob Alperin-Sheriff stellte auf einem Workshop für gitterbasierte Kryptographie (LACTA) eine Liste von 10 Fragen[22] auf, die bzgl. der allgemeinen Sicherheit von gitterbasierten Kryptosystemen diskutiert werden sollten. Auch Léo Ducas und Damien Stehlé veröffentlichen eine Liste von Fragen[23], die teilweise mit denen von Alperin-Sheriff übereinstimmen. Als Co-Autoren von Kryptosystemen, die am NIST PQC-Wettbewerb teilnehmen, haben sie jedoch eine etwas andere Sicht auf die offenen Fragen.

[19] http://pqcrypto.eu.org/index.html
[20] https://libpqcrypto.org/index.html
[21] https://cyber.biu.ac.il/event/the-2nd-biu-winter-school
[22] http://crypto-events.di.ens.fr/LATCA/program/alperin-sheriff.pdf
[23] http://www.h2020prometheus.eu/2018-06-04.assessing-security.html



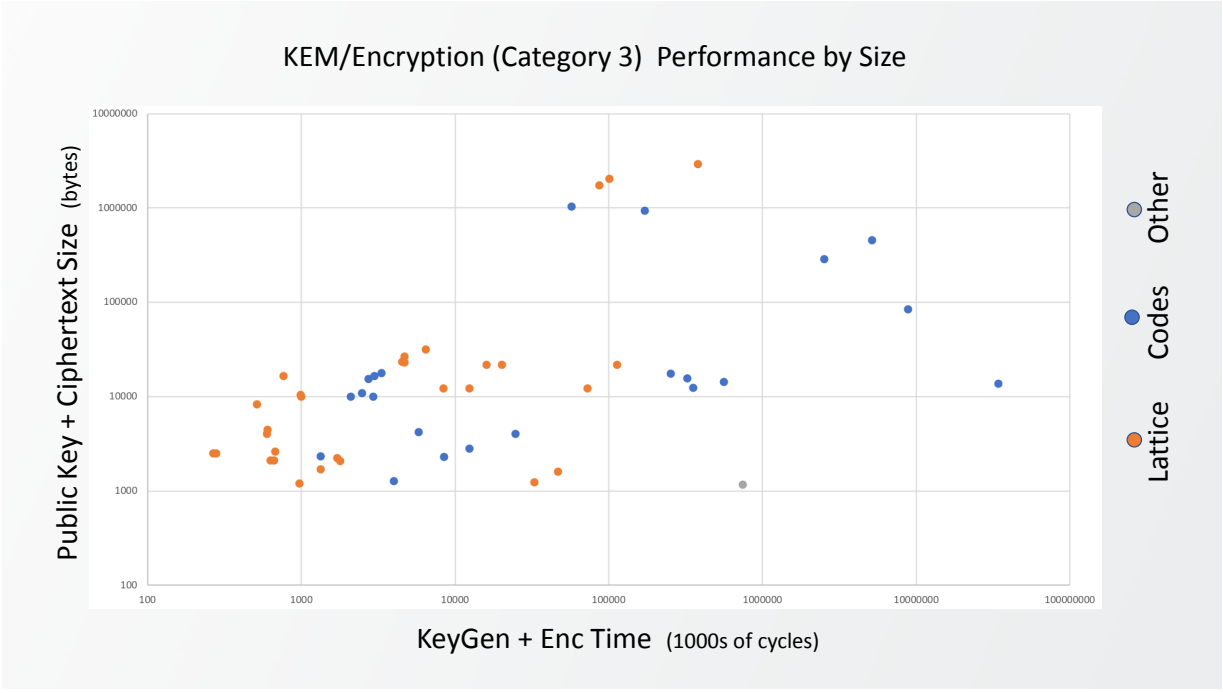


Abbildung 5.1: Vergleich der Post-Quantum-Verfahren im aktuellen NIST PQC Wettbewerb.

6 Fazit und Ausblick

Das Forschungsfeld zu gitterbasierten Problemen und kryptographischen Anwendungen ist in den letzten 20 Jahren enorm gewachsen. Aufbauend auf dem grundlegenden Meilenstein von Miklós Ajtais Worst-case/Average-case Reduktion für Gitterprobleme [Ajt96] ist eine Fülle von kryptographischen Anwendungen entstanden. Weitere Forschungen führten zur Entwicklung des Learning with Errors Problems LWE, woraus wiederum eine Vielzahl neuer Ideen und Varianten entsprungen sind. So bietet LWE eine breite Grundlage für verschiedene kryptographische Verfahren. Auch wenn es eine gewisse Entwicklungszeit gebraucht hat, gibt es heutzutage Systeme die „High Performance Lattice-based CCA-secure Encryption“ versprechen [EB15]. Die Entwicklung gitterbasierter Kryptographie in den Jahren 2005-2015 skizziert Peikert in [Pei15a]. Angesichts der vielen Vorteile und Möglichkeiten, die Gitter in der Kryptographie bieten (siehe Abschnitt 5.7) ist diese Entwicklung nicht verwunderlich. Nach wie vor sind viele der beim NIST PQC-Wettbewerb teilnehmenden Kryptosysteme relativ jung und es gibt eine Reihe offener Fragen (siehe Abschnitt 5.8).

Außerdem besteht die Frage nach der Eignung gitterbasierter Kryptographie in Post-Quantum Zeiten. Dies werden mitunter weitere Forschungen in den nächsten Jahren zeigen. Überstürzen muss man den Umstieg auf Post-Quantum-Kryptographie jedoch nicht. Der renommierte Experte für Quantenalgorithmen Scott Aaronson merkt in seinem Blog[1] an, dass die Gefahr von Quantencomputern für aktuelle Public-Key-Verschlüsselungsverfahren eher theoretischer Natur ist. Hacker und Geheimdienste hätten hundert andere Wege an geschützte (verschlüsselte) Daten zu gelangen, als die Verschlüsselung zu brechen. Es genüge zum Beispiel Fehler in der Implementierung von Kryptosystemen oder andere menschliche Fehlerquellen auszunutzen. Dies ist mit weit weniger Aufwand verbunden, als einen Quantencomputer zum Brechen der eigentlichen Verschlüsselung zu entwickeln. Aaronson hat auch Einblick in die aktuelle Entwicklung von Quantencomputern bei Google und kann nachvollziehen wie schwer es ist einen effektiv nutzbaren Quantencomputer zu bauen. Anhand des Fortschritts den Google und andere Unternehmen (z.B. Intel, IBM, Microsoft, ...) in der Entwicklung von Quantencomputern in den letzten Jahren gemacht haben, geht er davon aus, dass Quantencomputer für spezielle Anwendungen im Bereich „Quantum Simulation“, in der Größe von ca. 50-200 Qubits, erst in zehn Jahren zu erwarten sind. Dass man mit Quantencomputern effektiv heutige Public-Key-Kryptographie brechen kann, liegt in noch weiterer Ferne, schätzt Aaronson.

Auch im Hinblick auf die Umstellung auf Post-Quantum-Kryptographie im großen Stil (internetweit) sieht Aaronson das Problem, dass auf kurze Sicht die Sicherheit durch sehr

[1] https://www.scottaaronson.com/blog/?p=3848

junge Verfahren und wenig gereifte Implementierungen zunächst sogar beeinträchtigt werden könnte. Die vorherrschende Meinung unter Sicherheitsexperten ist jedoch, dass man lieber früher als (zu spät) auf Quantencomputer-sichere Verschlüsselungsverfahren umstellen sollte, zumal ein globaler Umstellungsprozess auch einige Jahre dauern wird. Wie es auch gängige Praxis bei der Einführung neuer Verfahren (z.B. Diffie-Hellman-Schlüsselaustausch mit elliptischen Kurven (ECDHE)) ist, sollte man vernünftigerweise Post-Quantum-Verfahren parallel zu den etablierten Public-Key-Verfahren einführen. So können Erfahrungen damit in der Praxis gesammelt werden, ohne sich schon frühzeitig allein darauf zu verlassen.

Im weiteren Verlauf des NIST PQC-Wettbewerbs wird sich nach und nach herauskristallisieren, inwiefern gitterbasierte Kryptographie reif für den Praxiseinsatz ist. Dies wird noch bis ca. 2024 dauern. Bis dahin werden die teilnehmenden Gitter-Kryptosysteme weiterhin analysiert und müssen zeigen, dass sie sich auch gegenüber alternativen Post-Quantum-Kryptosystemen behaupten können.

Abbildungsverzeichnis

Literatur

[Aar08] S. Aaronson. „The Limits of Quantum Computers“. In: *Scientific American* (298 2008), S. 62–69.

[AD16] D. Aggarwal und C. Dubey. „Improved hardness results for unique shortest vector problem“. In: *Information Processing Letters* 116.10 (2016), S. 631–637. ISSN: 0020-0190. URL: http://www.sciencedirect.com/science/article/pii/S0020019016300746.

[AD97] M. Ajtai und C. Dwork. „A Public-key Cryptosystem with Worst-case/Average-case Equivalence“. In: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*. STOC '97. El Paso, Texas, USA: ACM, 1997, S. 284–293. ISBN: 0-89791-888-6. URL: http://doi.acm.org/10.1145/258533.258604.

[ADPS16] E. Alkim, L. Ducas, T. Pöppelmann und P. Schwabe. „Post-quantum key exchange – a new hope“. In: *Proceedings of the 25th USENIX Security Symposium*. Document ID: 0462d84a3d34b12b75e8f5e4ca032869, http://cryptojedi.org/papers/#newhope. USENIX Association, 2016.

[Agu+16] C. Aguilar-Melchor u. a. „NFLlib: NTT-Based Fast Lattice Library“. In: *Topics in Cryptology - CT-RSA 2016*. Hrsg. von K. Sako. Cham: Springer International Publishing, 2016, S. 341–356. ISBN: 978-3-319-29485-8.

[Ajt96] M. Ajtai. „Generating Hard Instances of Lattice Problems (Extended Abstract)“. In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: ACM, 1996, S. 99–108. ISBN: 0-89791-785-5. URL: http://doi.acm.org/10.1145/237814.237838.

[Ajt98] M. Ajtai. „The Shortest Vector Problem in L2 is NP-hard for Randomized Reductions (Extended Abstract)“. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Dallas, Texas, USA: ACM, 1998, S. 10–19. ISBN: 0-89791-962-9. URL: http://doi.acm.org/10.1145/276698.276705.

[Alb+18] M. R. Albrecht u. a. „Estimate All the {LWE, NTRU} Schemes!“ In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*. Hrsg. von D. Catalano und R. D. Prisco. Bd. 11035. Lecture Notes in Computer Science. Springer, 2018, S. 351–367. ISBN: 978-3-319-98112-3. URL: https://doi.org/10.1007/978-3-319-98113-0_19.

[AR05] D. Aharonov und O. Regev. *Lattice Problems in NP ∩ coNP*. Techn. Ber. 2005. URL: https://cims.nyu.edu/~regev/papers/cvpconp.pdf (besucht am 13. 10. 2018).

[BCLV16] D. J. Bernstein, C. Chuengsatiansup, T. Lange und C. van Vredendaal. *NTRU Prime: reducing attack surface at low cost*. Cryptology ePrint Archive, Report 2016/461. https://eprint.iacr.org/2016/461. 2016. URL: https://ntruprime.cr.yp.to/index.html.

[BDGL15] A. Becker, L. Ducas, N. Gama und T. Laarhoven. *New directions in nearest neighbor searching with applications to lattice sieving*. Cryptology ePrint Archive, Report 2015/1128. 2015. URL: https://eprint.iacr.org/2015/1128 (besucht am 31. 10. 2018).

[BLPRS13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev und D. Stehlé. „Classical Hardness of Learning with Errors“. In: *CoRR* abs/1306.0281 (2013). arXiv: 1306.0281. URL: http://arxiv.org/abs/1306.0281.

[Bos+16] J. Bos u. a. *Frodo: Take off the ring! Practical, Quantum-Secure Key Exchange from LWE*. Cryptology ePrint Archive, Report 2016/659. https://eprint.iacr.org/2016/659. 2016.

[BS99] J. Blömer und J.-P. Seifert. „On the Complexity of Computing Short Linearly Independent Vectors and Short Bases in a Lattice“. In: *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*. STOC '99. Atlanta, Georgia, USA: ACM, 1999, S. 711–720. ISBN: 1-58113-067-8. URL: http://doi.acm.org/10.1145/301250.301441.

[CDW16] R. Cramer, L. Ducas und B. Wesolowski. *Short Stickelberger Class Relations and application to Ideal-SVP*. Cryptology ePrint Archive, Report 2016/885. https://eprint.iacr.org/2016/885. 2016.

[CN11] Y. Chen und P. Q. Nguyen. „BKZ 2.0: Better Lattice Security Estimates“. In: *Advances in Cryptology – ASIACRYPT 2011*. Hrsg. von D. H. Lee und X. Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 1–20. ISBN: 978-3-642-25385-0.

[Cox73] H. S. M. Coxeter. *Regular Polytopes*. New York: Dover, 1973.

[CS16] A. Costache und N. P. Smart. „Which Ring Based Somewhat Homomorphic Encryption Scheme is Best?“ In: *Topics in Cryptology - CT-RSA 2016*. Hrsg. von K. Sako. Cham: Springer International Publishing, 2016, S. 325–340. ISBN: 978-3-319-29485-8.

[DJE09] Daniel J. Bernstein, Johannes Buchmann und Erik Dahmen. *Post-Quantum Cryptography*. Berlin Heidelberg: Springer-Verlag, 2009. URL: https://www.springer.com/de/book/9783540887010.

[DZW15] D. Ding, G. Zhu und X. Wang. „A Genetic Algorithm for Searching the Shortest Lattice Vector of SVP Challenge“. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15. Madrid, Spain: ACM, 2015, S. 823–830. ISBN: 978-1-4503-3472-3. URL: http://doi.acm.org/10.1145/2739480.2754639.

[EB15] R. El Bansarkhani und J. Buchmann. *High Performance Lattice-based CCA-secure Encryption*. Cryptology ePrint Archive, Report 2015/042. https://eprint.iacr.org/2015/042. 2015.

[FO13] E. Fujisaki und T. Okamoto. „Secure Integration of Asymmetric and Symmetric Encryption Schemes“. In: *Journal of Cryptology* 26.1 (Jan. 2013), S. 80–101. ISSN: 1432-1378. URL: https://doi.org/10.1007/s00145-011-9114-1.

[Gen09] C. Gentry. „A fully homomorphic encryption scheme“. crypto.stanford.edu/craig. Diss. Stanford University, 2009.

[GG98] O. Goldreich und S. Goldwasser. „On the Limits of Non-approximability of Lattice Problems“. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Dallas, Texas, USA: ACM, 1998, S. 1–9. ISBN: 0-89791-962-9. URL: http://groups.csail.mit.edu/cis/pubs/shafi/1998-stoc.pdf (besucht am 03. 11. 2018).

[GGH96] O. Goldreich, S. Goldwasser und S. Halevi. *Collision-Free Hashing from Lattice Problems*. 1996. URL: http://www.wisdom.weizmann.ac.il/~oded/COL/cfh.pdf (besucht am 30. 10. 2018).

[GGH97] O. Goldreich, S. Goldwasser und S. Halevi. „Eliminating decryption errors in the Ajtai-Dwork Cryptosystem“. In: *Advances in Cryptology — CRYPTO '97*. Hrsg. von B. S. Kaliski. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, S. 105–111. ISBN: 978-3-540-69528-8.

[GMR05] V. Guruswami, D. Micciancio und O. Regev. „The complexity of the covering radius problem“. In: *computational complexity* 14.2 (Juni 2005), S. 90–121. ISSN: 1420-8954. URL: https://doi.org/10.1007/s00037-005-0193-y.

[GMSS99] O. Goldreich, D. Micciancio, S. Safra und J.-P. Seifert. „Approximating shortest lattice vectors is not harder than approximating closest lattice vectors“. In: *Information Processing Letters* 71.2 (1999), S. 55–61. ISSN: 0020-0190. URL: http://www.sciencedirect.com/science/article/pii/S0020019099000836.

[GVW17] F. Göpfert, C. van Vredendaal und T. Wunderer. „A Hybrid Lattice Basis Reduction and Quantum Search Attack on LWE“. In: *Post-Quantum Cryptography*. Hrsg. von T. Lange und T. Takagi. Cham: Springer International Publishing, 2017, S. 184–202. ISBN: 978-3-319-59879-6.

[HGS99] C. Hall, I. Goldberg und B. Schneier. „Reaction Attacks against Several Public-Key Cryptosystem“. In: *Information and Communication Security*. Hrsg. von V. Varadharajan und Y. Mu. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, S. 2–12. ISBN: 978-3-540-47942-0.

[How+03] N. Howgrave-Graham u. a. „The Impact of Decryption Failures on the Security of NTRU Encryption“. In: *Advances in Cryptology - CRYPTO 2003*. Hrsg. von D. Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, S. 226–246. ISBN: 978-3-540-45146-4.

[HPS98] J. Hoffstein, J. Pipher und J. H. Silverman. „NTRU: A Ring-Based Public Key Cryptosystem“. In: *Lecture Notes in Computer Science*. Springer-Verlag, 1998, S. 267–288.

[HR07] I. Haviv und O. Regev. „Tensor-based Hardness of the Shortest Vector Problem to Within Almost Polynomial Factors“. In: *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*. STOC '07. San Diego, California, USA: ACM, 2007, S. 469–477. ISBN: 978-1-59593-631-8. URL: http://doi.acm.org/10.1145/1250790.1250859.

[HS00a] J. Hoffstein und J. H. Silverman. *Protecting NTRU Against Chosen Ciphertext and Reaction Attacks*. Techn. Ber. 2000. URL: https://web.archive.org/web/20000819042400/http://www.ntru.com:80/NTRUFTPDocsFolder/NTRUTech016.pdf.

[HS00b] J. Hoffstein und J. H. Silverman. *Reaction Attacks Against the NTRU Public Key Cryptosystem*. Techn. Ber. 2000. URL: https://web.archive.org/web/20000914041434/http://www.ntru.com:80/NTRUFTPDocsFolder/NTRUTech015.pdf.

[Jea16] J. Jean. *TikZ for Cryptographers*. https://www.iacr.org/authors/tikz/. 2016.

[KS01] S. R. Kumar und D. Sivakumar. „On the unique shortest lattice vector problem“. In: *Theoretical Computer Science* 255.1 (2001), S. 641–648. ISSN: 0304-3975. URL: http://www.sciencedirect.com/science/article/pii/S030439750000387X.

[Li18] C. Li. *A Key Recovery Attack on Streamlined NTRU Prime*. Cryptology ePrint Archive, Report 2018/998. https://eprint.iacr.org/2018/998. 2018.

[LLL82] A. Lenstra, H. Lenstra und L. Lovász. „Factoring polynomials with rational coefficients“. In: *Mathematische Annalen* (1982), S. 515–534. URL: https://doi.org/10.1007/BF01457454 (besucht am 19. 10. 2018).

[MG02] D. Micciancio und S. Goldwasser. *Complexity of Lattice Problems – A Cryptographic Perspective*. Springer US, 2002.

[Mic02] D. Micciancio. „Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions from Worst-Case Complexity Assumptions“. In: *Proceedings of the 43rd Symposium on Foundations of Computer Science*. FOCS '02. Washington, DC, USA: IEEE Computer Society, 2002, S. 356–365. ISBN: 0-7695-1822-2. URL: http://dl.acm.org/citation.cfm?id=645413.652130.

[Mic12] D. Micciancio. „Inapproximability of the Shortest Vector Problem: Toward a Deterministic Reduction“. In: *Theory of Computing* 8.22 (2012), S. 487–512. URL: http://www.theoryofcomputing.org/articles/v008a022. I. Haviv und O. Regev. „Tensor-based Hardness of the Shortest Vector Problem to within Almost Polynomial Factors“. In: *Theory of Computing* 8.23 (2012), S. 513–531. URL: http://www.theoryofcomputing.org/articles/v008a023.

[Mic98] D. Micciancio. „The Shortest Vector in a Lattice is Hard to Approximate to Within Some Constant“. In: *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*. 1998, S. 92–98. URL: https://doi.org/10.1109/SFCS.1998.743432.

[Min96] H. Minkowski. *Geometrie der Zahlen*. Leipzig, Berlin: B. G. Teubner, 1896.

[MR07] D. Micciancio und O. Regev. „Worst-Case to Average-Case Reductions Based on Gaussian Measures“. In: *SIAM Journal on Computing* 37.1 (2007). Ursprünglich veröffentlicht in FOCS 2004, S. 267–302. eprint: https://doi.org/10.1137/S0097539705447360.

[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. eprint: https://doi.org/10.1137/100811970.

[Neu92] J. Neukirch. *Algebraische Zahlentheorie*. Berlin Heidelberg: Springer-Verlag, 1992.

[Ngu10] P. Q. Nguyen. *The LLL Algorithm: Survey and Applications*. Hrsg. von P. Q. Nguyen und B. Vallée. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-02295-1. URL: https://doi.org/10.1007/978-3-642-02295-1.

[Pei08] C. Peikert. „Limits on the Hardness of Lattice Problems in ℓpNorms“. In: *computational complexity* 17.2 (Mai 2008), S. 300–351. ISSN: 1420-8954. URL: https://doi.org/10.1007/s00037-008-0251-3.

[Pei09] C. Peikert. „Public-key Cryptosystems from the Worst-case Shortest Vector Problem: Extended Abstract“. In: *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*. STOC '09. Bethesda, MD, USA: ACM, 2009, S. 333–342. ISBN: 978-1-60558-506-2. URL: http://doi.acm.org/10.1145/1536414.1536461.

[Pei15a] C. Peikert. *A Decade of Lattice Cryptography*. Cryptology ePrint Archive, Report 2015/939. https://eprint.iacr.org/2015/939. 2015.

[Pei15b] C. Peikert. *Lattices in Cryptography*. http://web.eecs.umich.edu/~cpeikert/lic15/index.html. 2015.

[Reg04] O. Regev. *Lattices in Computer Science*. https://cims.nyu.edu/~regev/teaching/lattices_fall_2004. 2004.

[Reg09] O. Regev. „On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *J. ACM* 56.6 (Sep. 2009), 34:1–34:40. ISSN: 0004-5411. URL: http://doi.acm.org/10.1145/1568318.1568324.

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

[Sch18] C.-P. Schnorr. *Gitter und Kryptographie*. https://www.math.uni-frankfurt.de/~dmst/teaching/SS2018/vorlesung.html. 2018.

[Sho97] P. W. Shor. „Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM J. Comput.* 26.5 (Okt. 1997), S. 1484–1509. ISSN: 0097-5397. URL: http://dx.doi.org/10.1137/S0097539795293172.

[SS13] D. Stehlé und R. Steinfeld. *Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices*. Cryptology ePrint Archive, Report 2013/004. https://eprint.iacr.org/2013/004. 2013.

[SSTX09] D. Stehlé, R. Steinfeld, K. Tanaka und K. Xagawa. „Efficient Public Key Encryption Based on Ideal Lattices". In: *Advances in Cryptology – ASIACRYPT 2009*. Hrsg. von M. Matsui. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 617–635. ISBN: 978-3-642-10366-7.

[Ste15] N. Stephens-Davidowitz. „Search-to-Decision Reductions for Lattice Problems with Approximation Factors (Slightly) Greater Than One". In: *CoRR* abs/1512.04138 (2015). arXiv: 1512.04138. URL: http://arxiv.org/abs/1512.04138.

[Vai15] V. Vaikuntanathan. *6.876J Advanced Topics in Cryptography: Lattices*. http://people.csail.mit.edu/vinodv/6876-Fall2015/index.html. 2015.