TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät für Informatik

Professur Theoretische Informatik und Informationssicherheit

Bachelorarbeit

Gitterbasierte Kryptographie

Martin Wild

Chemnitz, den 25. Februar 2019

Gutachter: Prof. Dr. Hanno Lefmann
Dr. Knut Odermann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

1 Einführung

Kryptographische Verfahren erfahren im Laufe der Zeit immer wieder Anpassungen, um ihre praktische Sicherheit und Anwendbarkeit zu erhalten. Auch wenn das Grundprinzip eines Verfahrens als sicher gilt, bleibt die Wahl der Verfahrensparameter, sowie die konkrete Implementierung des Verfahrens Gegenstand stetiger Weiterentwicklung. Ursächlich hierfür ist zum einen die Entdeckung neuer und die Verbesserung bestehender Angriffsmöglichkeiten. Zum anderen erfordert die durch das Mooresche Gesetz beschriebene Zunahme der Rechenleistung moderner Computer eine stufenweise Vergrößerung relevanter Sicherheitsparameter in einem Abstand von etwa zehn Jahren. Entsprechende Empfehlungen konkreter Werte werden beispielsweise durch die ECRYPT-Projekte der Europäischen Union herausgegeben[1].

Durch die Entwicklung von Quantencomputern erhält ein großer Teil der heutigen Verfahren jedoch ein unbestimmtes aber finales Ablaufdatum. Verglichen mit einem klassischen Digitalrechner ermöglicht das Konzept des Quantencomputers eine deutliche Erweiterung der algorithmischen Möglichkeiten. Denn während klassische Computer mit Bits arbeiten, die immer genau einen von zwei möglichen Zuständen annehmen können, operieren Quantencomputer auf sogenannten *Qubits*, die für eine gewisse Zeitspanne auch einen Zwischenzustand ermöglichen[31]. Dieses als Superposition bezeichnete Phänomen ist eine von mehreren quantenmechanischen Eigenschaften, die sowohl das Verhalten einzelner Qubits als auch das Verbundverhalten mehrerer Qubits, die zu sogenannten *Quantenregistern* zusammengefasst werden, bestimmen. Ein Verbund von Qubits verfügt also über eine Reihe zusätzlicher Möglichkeiten im Vergleich zu klassischen Bits. Darauf aufbauend lassen sich neuartige Algorithmen beschreiben, die auch als Quantenalgorithmen bezeichnet werden. Der bekannteste dieser Algorithmen geht auf die Arbeit von Shor zurück[67] und erlaubt die Faktorisierung großer Zahlen mit polynomiellen Aufwand. Weiterhin beschreibt Shor darin einen Quantenalgorithmus zur effizienten Berechnung des diskreten Logarithmus. Ein Quantencomputer von hinreichender Qubit-Anzahl könnte damit die meisten der aktuell verwendeten asymmetrischen Kryptosysteme, wie beispielsweise RSA, effizient angreifen, ohne dass dies durch eine Vergrößerung der Sicherheitsparameter wirkungsvoll kompensiert werden kann[31]. Für symmetrische Verfahren wie AES existieren ebenfalls bereits auf Quantenalgorithmen gestützte Angriffsmöglichkeiten[36]. Jedoch geht man hier nur von einer Halbierung der in Bit gemessenen

[1] http://www.ecrypt.eu.org/

Sicherheit aus, welche sich durch eine Erhöhung der Schlüsselgröße vergleichsweise einfach ausgleichen ließe.

Auch wenn die technische Realisierung von Quantencomputern relevanter Größe erst in eher ferner Zukunft erwartet wird, beschäftigt man sich mit alternativen Kryptosystemen, deren Sicherheit auch im Angesicht von Quantenalgorithmen Bestand hat. Eine vielversprechende Gruppe dieser Verfahren basiert auf der Schwierigkeit verschiedener Vektorprobleme in Gittern. Als Gitter wird dabei eine regelmäßige Menge von Vektoren bezeichnet, die eine diskrete Untergruppe des Euklidischen Vektorraumes bildet. Beispielsweise bildet der ganzzahlige Lösungsraum eines linearen Gleichungssystems ein Gitter. Frühe Beschreibungen von Gittern als quadratische metrische Formen gehen bereits auf Gauß und Hermite zurück. Als Punktmengen wurden Gitter um 1900 Teil der Geometrie der Zahlen von Hermann Minkowski. Heute werden Gitter vor allem mit den Mitteln der Vektorgeometrie und der linearen Algebra beschrieben. Bedeutung für die Kryptographie erlangten Gitter durch die Formulierung gitterbasierter Angriffe auf verschiedene kryptographische Verfahren, wie beispielsweise RSA und DSA[24][42]. Es zeigte sich, dass diese Verfahren unter bestimmten Umständen mithilfe sogenannter Gitterreduktionsalgorithmen erfolgreich angegriffen werden können. Diesen Angriffen kann jedoch durch eine geeignete Auswahl der Verfahrensparameter begegnet werden. Gitter bilden darüber hinaus die Grundlage für eine Reihe von mathematischen Problemstellungen mit exponentieller Komplexität. Es wird angenommen, dass diese auch für Quantenalgorithmen nicht signifikant schneller zu lösen sind. Die ersten gitterbasierten Kryptosysteme galten hinsichtlich Speicherbedarf und Effizienz als unpraktikabel, da relativ schnell Angriffsmöglichkeiten gefunden wurden, welche die Verwendung sehr großer Sicherheitsparameter notwendig machten. Moderne gitterbasierte Verfahren verfügen dagegen über starke Sicherheitsgarantien und versprechen darüber hinaus eine mit gegenwärtigen Verfahren vergleichbare Effizienz. Hierfür werden neue Problemstellungen formuliert, die auf der Schwierigkeit der bekannten Probleme in Gittern beruhen, sich jedoch effizienter darstellen lassen. Auf diese Weise hat sich der Bereich der gitterbasierten Kryptographie zu einem komplexen Fachgebiet mit vielversprechenden Ansätzen zur Entwicklung von Kryptosystemen für das Quantenzeitalter entwickelt.

Ziel der vorliegenden Arbeit ist es, die Grundlagen gitterbasierter Kryptosysteme zu erläutern und einige repräsentative Ansätze für verschiedene Anwendungen vorzustellen und zu bewerten. Abschließend wird ein Vergleich mit derzeit verbreiteten kryptographischen Verfahren durchgeführt.

2 Grundlagen

In diesem Kapitel sollen die zum Verständnis der Arbeit erforderlichen mathematischen Grundlagen behandelt werden. Dabei soll der Fokus auf den verwendeten Methoden liegen. Für eine detaillierte Darstellung der jeweiligen algebraischen Zusammenhänge wird auf die Fachliteratur verwiesen[43][16]. Eine ausführlichere Einführung in die hier vorgestellten Konzepte liefert das Grundlagenwerk von Buchmann[16], auf dem dieses Kapitel beruht.

2.1 Division ganzer Zahlen

Die Division ganzer Zahlen wird, insofern das Ergebnis der Operation in jedem Fall wieder eine ganze Zahl sein soll, als Division mit Rest durchgeführt. Für kryptographische Anwendungen ist hierbei die Bestimmung des Divisionsrestes von Interesse. Dazu wird die sogenannte *Modulo*-Funktion definiert.

Definition 2.1.1 *Für zwei ganze Zahlen a und b mit $b \neq 0$ sei der Quotient der ganzzahligen Division durch $q = \lfloor a/b \rfloor$ gegeben. Dann gibt $r = a - qb$ den ganzzahligen Divisionsrestrest an. Man sagt r ist gleich a* modulo *b und schreibt*

$$r = a \bmod b. \tag{2.1}$$

Weiterhin heißt jede ganze Zahl c, für die gilt $(a - c) \bmod b = 0$, kongruent *zu a modulo b. Man schreibt*

$$c \equiv a \bmod b. \tag{2.2}$$

Aus der Division mit Rest ergibt sich weiterhin das Prinzip der Teilbarkeit.

Definition 2.1.2 *Für zwei ganze Zahlen a und b gelte $b = na$ mit einem $n \in \mathbb{Z}$. Dann heißt a Teiler von b und b Vielfaches von a. Man schreibt*

$$a \mid b. \tag{2.3}$$

Ist a kein Teiler von b, gilt die Schreibweise

$$a \nmid b. \tag{2.4}$$

Damit lässt sich der größte gemeinsame Teiler zweier ganzer Zahlen definieren.

Definition 2.1.3 *Für zwei Zahlen $a, b \in \mathbb{Z}$ ist der* größte gemeinsame Teiler *definiert als*

$$gcd\,(a, b) = max\,(t \in \mathbb{N} : t \mid a \wedge t \mid b). \tag{2.5}$$

Ist der größte gemeinsame Teiler zweier Zahlen 1, dann heißen sie *teilerfremd*. Die *Eulersche Funktion* oder auch *Phi-Funktion* gibt für eine Zahl $n \in \mathbb{N}$ an, wie viele zu n teilerfremde Zahlen es gibt, die kleiner oder gleich n sind.

Definition 2.1.4 *Für eine natürliche Zahl n, ist die* Phi-Funktion *definiert als*

$$\varphi(n) := |\{a \in \mathbb{N} : 1 \leq a \leq n \wedge gcd\,(a, n) = 1\}| \tag{2.6}$$

Ist n eine Primzahl, gilt also $\varphi(n) = n - 1$.

2.2 Restklassenringe

Kongruenz modulo einer ganzen Zahl m ist eine Äquivalenzrelation auf der Menge der ganzen Zahlen. Die entsprechende Äquivalenzklasse einer ganzen Zahl a heißt *Restklasse* von $a \bmod m$ und hat die Schreibweise $a + m\mathbb{Z}$. Die Menge aller Restklassen ganzer Zahlen modulo m wird mit $\mathbb{Z}/m\mathbb{Z}$ oder auch $\mathbb{Z}_m$ bezeichnet. Das Tripel $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ bildet einen kommutativen Ring mit Einselement $1 + m\mathbb{Z}$ und heißt *Restklassenring* modulo m. Abkürzend wird hierfür anstelle des Tripels einfach $\mathbb{Z}/m\mathbb{Z}$ oder $\mathbb{Z}_m$ geschrieben.

Eine Restklasse $a+m\mathbb{Z}$, für die $\gcd(a, m) = 1$ gilt, heißt *prime Restklasse*. Die Menge aller primen Restklassen modulo m wird mit $(\mathbb{Z}/m\mathbb{Z})^*$ oder $\mathbb{Z}^*$ notiert. Das Paar $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$ bildet eine endliche Abelsche Gruppe und heißt *prime Restklassengruppe*. Dies beinhaltet die Eigenschaft, dass jedes von null verschiedene Element invertierbar ist. Die Ordnung einer primen Restklassengruppe ist durch die Phi-Funktion $\varphi(m)$ gegeben. Ist m eine Primzahl, dann bilden die von 0 verschiedenen Elemente eines Restklassenringes $\mathbb{Z}/m\mathbb{Z}$ bezüglich der Multiplikation eine prime Restklassengruppe. Somit ist der Restklassenring $\mathbb{Z}/m\mathbb{Z}$ genau dann ein Körper, wenn m eine Primzahl ist. Als *Primitivwurzel* wird ein besonderes Element einer primen Restklassengruppe bezeichnet. Diese hat die Eigenschaft, dass jedes Element der Gruppe als Potenz der Primitivwurzel dargestellt werden kann. Wenn also eine Zahl $a \in \mathbb{Z}$ eine Primitvwurzel modulo m ist, lassen sich alle Elemente der primen Restklassengruppe $(\mathbb{Z}/m\mathbb{Z})^*$ als Ausdruck

$$a^k \bmod m \tag{2.7}$$

darstellen mit $k \in \mathbb{N}$. Primitivwurzeln modulo m existieren unter anderem dann, wenn m eine Primzahl ist[16].

2.3 Polynomringe

Es sei R ein unitärer kommutativer Ring. Ein Polynom über R ist ein Ausdruck

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{2.8}$$

mit x als Variablen und Koeffizienten $a_0 \cdots a_n \in R$. Der Wert der größten Potenz $n \in \mathbb{N}$ mit $a_n \neq 0$ gibt dabei den *Grad* des Polynoms an. Man schreibt $\operatorname{grad} f(x) = n$. Ein einzelnes Glied $a_i x^i$ eines Polynoms wird als *Monom* bezeichnet. Ein Polynom heißt *monisch*, wenn der Leitkoeffizient, also der Koeffizient a_n des Monoms $a_n x^n$ gleich 1 ist.

Die Menge aller Polynome über R mit der Variablen x wird als *Polynomring* $R[X]$ bezeichnet. Der Polynomring $R[X]$ ist *nullteilerfrei*[16], d.h. es existiert kein Element $a \in R[X]$, für das sich ein vom Nullelement verschiedenes Element $b \in R[X]$ finden lässt mit $ab = 0$. Im Polynomring $R[X]$ ist die Division mit Rest analog zum Ring der ganzen Zahlen möglich. Sind also $f(x)$ und $g(x)$ Polynome aus dem Polynomring $R[X]$ mit $g(x) \neq 0$, dann ist das Ergebnis der Polynomdivision $f(x) : g(x)$ gegeben durch die Polynome $s(x)$ und $r(x)$ aus $R[X]$. Es gilt also

$$f(x) : g(x) = s(x) + \frac{r(x)}{g(x)} \tag{2.9}$$

mit $\operatorname{grad} r(x) < \operatorname{grad} g(x)$. Das Polynom $s(x)$ bezeichnet dabei das Ergebnis der Division und das Polynom $r(x)$ den Divisionsrest. Dieser kann nun auch wie bei der Division ganzer Zahlen mithilfe der Modulo-Funktion beschrieben werden. Es gilt also

$$r(x) = f(x) \bmod g(x). \tag{2.10}$$

Während der Polynomring $R[X]$ grundsätzlich Polynome beliebigen Grades enthält, sind für kryptografische Anwendungen sogenannte *Quotientenringe* von Interesse, die Polynome begrenzten Grades enthalten. Diese können analog den Restklassenringen modulo einer ganzen Zahl m beschrieben werden. Hierzu wird zunächst der Begriff des *Ideals* definiert.

Definition 2.3.1 *Eine Teilmenge I eines Ringes R, die*

1. *das Nullelement von R enthält,*
2. *abgeschlossen bezüglich der Addition mit Elementen aus I ist,*
3. *abgeschlossen bezüglich der (beidseitigen) Multiplikation mit beliebigen Elementen aus R ist,*

heißt (beidseitiges) Ideal *von R.*

Betrachtet wird nun ein Ring R und ein beidseitiges Ideal $I \subset R$. Die Menge

$$R/I = \{a + I : a \in R\} \tag{2.11}$$

der Äquivalenzklassen von R modulo I zusammen mit den Verknüpfungen

$$(a + I) + (b + I) = (a + b) + I \tag{2.12a}$$
$$(a + I) \cdot (b + I) = (a \cdot b) + I \tag{2.12b}$$

bildet einen Ring. Dieser heißt *Faktorring* R modulo I oder *Quotientenring*. Es sei nun ein Polynom $f \in R[X]$ vom Grad n gegeben. Dann ist $(f) = R[X] \cdot f$ die Menge aller Polynomvielfachen von f und ein Ideal im Polynomring $R[X]$. Der Ring

$$R[X]/(f) = \{a + (f) : a \in R[X]\} \tag{2.13}$$

bildet somit einen Quotientenring und enthält Polynome mit einem Grad kleiner n.

2.4 Relevante Problemstellungen

2.4.1 Disktreter Logarithmus

Das Problem des *diskreten Logarithmus* (DLP) ist sicherheitsbestimmend für eine Reihe kryptographischer Verfahren, wie z.B. dem Diffie-Hellman-Verfahren. Der diskrete Logarithmus ist die Umkehrfunktion der diskreten Exponentiation über einer zyklischen Gruppe. In einem Ring $\mathbb{Z}_n$ wird für gegebene Werte der Basis a und des Exponenten x das Ergebnis m der diskreten Exponentiation mithilfe der Gleichung

$$m = a^x \bmod n \tag{2.14}$$

berechnet. Bezogen auf obige Gleichung, soll der diskrete Logarithmus also den kleinst möglichen Wert des Exponenten x ermitteln für gegebene Werte von a, m und n. Diese Berechnung lässt sich mit derzeitigen Verfahren nicht effizient durchführen[16]. Aus diesem Grund stellt die diskrete Exponentiation eine geeignete Einwegfunktion für kryptographische Verfahren dar, also eine Funktion, die einfach zu berechnen ist, aber deren Umkehrfunktion nicht direkt berechenbar ist, sondern in der Regel ein Suchproblem bildet.

2.4.2 Primzahlfaktorisierung

Das *Faktorisierungsproblem* für ganze Zahlen geht auf die Aufgabe zurück, nichttriviale Teiler zu einer gegebenen zusammengesetzten Zahl zu ermitteln. In Kombination mit Primzahltests kann damit die Primfaktorzerlegung einer natürlichen

Zahl bestimmt werden. Die Schwierigkeit der Primfaktorzerlegung ist sicherheitsbestimmend für das in Abschnitt 3.2 erläuterte RSA-Verfahren. Auch wenn derzeit kein effizientes klassisches Verfahren zur Primfaktorzerlegung bekannt ist, besteht für kryptographische Verfahren die Notwendigkeit, möglichst schwierige Instanzen des Problems zu erzeugen, da nicht jede Instanz des Problems hinreichend schwer zu berechnen ist. Die Erzeugung möglichst geeigneter Instanzen des Problems wird in Abschnitt 3.5 näher beschrieben.

2.4.3 Learning With Errors

Learning with Errors (LWE) ist der Name einer Problemstellung aus dem Bereich des Machine Learning, die von Oded Regev beschrieben wurde[65]. Es wird auf Grundlage eines Parameters $n \in \mathbb{N}$, einer Primzahl q und einer Wahrscheinlichkeitsverteilung χ über $\mathbb{Z}/q\mathbb{Z}$ angegeben. Die Verteilung χ wird auch als *Fehlerverteilung* bezeichnet und soll mit großer Wahrscheinlichkeit kleine Elemente bezüglich einer Norm $\|\cdot\|$ ausgeben. Für Fehler $e \leftarrow \chi$ soll dabei gelten

$$\|e\| \ll q. \tag{2.15}$$

In der Praxis wird hierfür oft eine diskrete Gaußverteilung über $\mathbb{Z}$ verwendet.

LWE ist als Suchproblem und als Entscheidungsproblem formuliert. Ausgangspunkt des Suchproblems bildet eine Menge zufällig gewählter Proben als Paare der Form $(a, \langle a, s\rangle + e)$. Hierbei beschreibt $a \in \mathbb{Z}_q^n$ einen für jede Probe zufällig gewählten Vektor. Der Vektor $s \in \mathbb{Z}_q^n$ ist fest und bildet das Geheimnis. Die Fehler $e \leftarrow \chi$ werden für jedes Tupel zufällig auf Grundlage der Fehlerverteilung erzeugt. Das innere Produkt der Vektoren a und s ist gegeben durch

$$\langle a, s\rangle = \sum_{i=1}^{n} a_i s_i. \tag{2.16}$$

Der Term $\langle a, s\rangle + e$ wird als *gestörtes inneres Produkt* bezeichnet. Ziel ist es nun, anhand einer in n polynomiellen Anzahl von zufällig generierten Proben, den geheimen Vektor s zu finden.

Beim Entscheidungsproblem ist es das Ziel, Proben der Form $(a, \langle a, s\rangle + e)$ von weiteren zufällig erzeugten Proben (a, b) mit $b \leftarrow \mathbb{Z}_q$ zu unterscheiden. Es soll also erkannt werden, welche Proben mithilfe eines festen Geheimnisses s und einer Fehlerverteilung χ erzeugt wurden und welche nur zufällige Paare aus $\mathbb{Z}_q^n \times \mathbb{Z}_q$ sind. Ausgangspunkt hierfür ist eine in n polynomielle Anzahl von zufällig generierten Proben beider Typen.

Learning with Errors over Rings

Learning with Errors over Rings (Ring-LWE) heißt eine Variante des Problems LWE, die analog beschrieben ist, aber anstelle von Vektoren mit Polynomen aus bestimmten Quotientenringen arbeitet. Hierfür sei der Ring $R = \mathbb{Z}_q[X]/f(x)$ definiert mit einem Polynom $f(x) \in \mathbb{Z}_q[X]$. Für das Polynom $f(x)$ wird gefordert, dass es *irreduzibel* ist, d.h. es kann nicht als nichttriviales Produkt von kleineren Polynomen aus R angegeben werden. Die Proben haben damit die Form $(a(x), (a(x) \cdot s(x)) + e(x))$ mit Polynomen $a(x) \in R$, einem festen Geheimnis $s(x) \in R$ und Fehlerpolynomen $e(x) \in R$. Die Fehlerverteilung χ soll dabei Fehlerpolynome mit kleinen Koeffizienten erzeugen. Für das Entscheidungsproblem werden zudem Tupel $(a(x), b(x))$ erzeugt mit zufälligen Polynomen $b(x) \in R$.

Die Probleme LWE und Ring-LWE haben einen engen Bezug zu bestimmten Problemstellungen in Gittern und bilden die Grundlage für eine Reihe von gitterbasierten Kryptosystemen, worauf in Kapitel 5 ausführlicher eingegangen wird.

3 Kryptographische Verfahren

In diesem Kapitel werden grundlegende kryptographische Definitionen und ausgewählte Verfahren vorgestellt. Der Schwerpunkt liegt dabei auf heute weit verbreiteten Kryptosystemen, deren Sicherheit mit hinreichend großen Quantencomputern vollständig gebrochen werden könnte. Das bedeutet, dass das sicherheitsbestimmende Problem mithilfe von Quantenalgorithmen effizient gelöst werden kann, weshalb eine Vergrößerung der jeweiligen Sicherheitsparameter, also z.B. der Schlüssellänge, keine ausreichende Kompensation liefern kann. Eine ausführliche Behandlung der hier vorgestellten Grundlagen und Verfahren findet sich bei Buchmann[16], an dessen Darstellungen sich dieses Kapitel orientiert. Darüber hinaus wird an entsprechender Stelle auf die jeweiligen Originalveröffentlichungen verwiesen.

3.1 Definitionen

Kryptographische Verfahren werden in *symmetrische* und *asymmetrische* Verfahren unterschieden. Symmetrische Verfahren zeichnen sich dadurch aus, dass Ver- und Entschlüsselung einer Nachricht mit ein und demselben Schlüssel erfolgen. Asymmetrische Verfahren dagegen verwenden zwei Schlüssel, einen zum Verschlüsseln und einen anderen zum Entschlüsseln einer Nachricht. Wie oben erläutert, sind für diese Arbeit ausschließlich asymmetrische Verfahren von Bedeutung, insbesondere die sogenannten *Public-Key-Verschlüsselungsverfahren* und *digitalen Signaturverfahren*. Erstere sind häufig Teil sogenannter *hybrider Verfahren*, bei denen ein Public-Key-Verfahren verwendet wird, um einen für eine Session gültigen gemeinsamen Schlüssel sicher zwischen zwei Kommunikationspartnern auszutauschen. Für die eigentliche Kommunikation kommt dann ein symmetrisches Verschlüsselungsverfahren zum Einsatz, unter Verwendung des zuvor ausgetauschten Schlüssels.

3.1.1 Public-Key-Verfahren

Definition 3.1.1 *Ein* Public-Key-Kryptosystem *wird durch ein Tupel* $(K, P, C,$ KeyGen, Enc, Dec$)$ *angegeben. Dabei bildet* K *den* Schlüsselraum *mit Elementen* $(e, d) \in K$*, wobei* e *der* öffentliche Schlüssel *und* d *der* geheime Schlüssel *eines zusammengehörigen Schlüsselpaares ist.* P *ist der* Klartextraum *und* C *der* Chiffretextraum*. Ihre Elemente heißen* Klartexte *bzw.* Chiffretexte*.*

KeyGen *ist der* Schlüsselerzeugungsalgorithmus. *Er liefert in Abhängigkeit eines* Sicherheitsparameters $n \in \mathbb{N}$ *ein Schlüsselpaar* $(e, d) \in K$. *Man schreibt*

$$(e, d) \leftarrow \text{KeyGen}(n). \tag{3.1}$$

Die Menge aller Schlüsselpaare, die mit einem festen n *erzeugt werden können, wird als* $K(n)$ *bezeichnet.*

Enc *bezeichnet den* Verschlüsselungsalgorithmus *und* Dec *den* Entschlüsselungsalgorithmus. *Für einen gegebenen Sicherheitsparameter* $n \in \mathbb{N}$, *einen öffentlichen Schlüssel* e *aus einem Schlüsselpaar* $(e, d) \in K(n)$ *und einen Klartext* $p \in P$ *liefert* Enc *einen Chiffretext* $c \in C$. *Es gilt also*

$$c \leftarrow \text{Enc}(n, e, p). \tag{3.2}$$

Dec *entschlüsselt für einen gegebenen Sicherheitsparameter* $n \in \mathbb{N}$ *und mithilfe des geheimen Schlüssels* d *eines Schlüsselpaares* $(e, d) \in K(n)$ *einen Chiffretext* $c \in C$ *in den zugehörigen Klartext* $p \in P$. *Der Chiffretext* c *muss dafür unter Verwendung von Gleichung 3.2 mit dem zugehörigen öffentlichen Schlüssel* e *verschlüsselt worden sein. Man schreibt*

$$p \leftarrow \text{Dec}(n, d, c). \tag{3.3}$$

Public-Key-Verfahren lassen sich im Wesentlichen also durch die jeweilige Methodik der Schlüsselerzeugung, Verschlüsselung und Entschlüsselung beschreiben. Weitere Aspekte sind natürlich die Sicherheit des Verfahrens, die im Wesentlichen durch die Schwierigkeit des mathematischen Problems bestimmt ist, das gelöst werden muss, um das Verfahren zu brechen. Weiterhin existieren zu verbreiteten Verfahren häufig theoretische und praktische Angriffsmöglichkeiten, die eine ungünstige Wahl der Verfahrensparameter oder Schwachstellen einzelner technischer Implementierungen ausnutzen. Die im Sinne der Sicherheit korrekte Wahl der Verfahrensparameter ist also in der Regel von zentraler Bedeutung. Darüber hinaus ist die Effizienz eines Public-Key-Verfahrens von Interesse, also die Laufzeit der verwendeten Algorithmen und der Speicherbedarf des Verfahrens. Diese werden in der Regel in Abhängigkeit vom Sicherheitsparameter n betrachtet. Dadurch besteht ein direkter Zusammenhang zwischen der Sicherheit und der Effizienz eines Verfahrens. Werden beispielsweise für eine ausreichende Sicherheit sehr große Werte für n benötigt, kann ein Verfahren in Anbetracht der dadurch verbrauchten Ressourcen unpraktikabel werden. Dieser Zusammenhang gilt auch für die im folgenden Abschnitt behandelten digitalen Signaturverfahren.

3.1.2 Digitale Signaturverfahren

Eine Signatur im Allgemeinen hat die Aufgabe, den Urheber einer Nachricht eindeutig zu identifizieren. Eine digitale Signatur adaptiert diese Anforderung für Nachrichten

in digitaler Form und erweitert sie, indem zusätzlich die Integrität der Nachricht sichergestellt werden soll.

Definition 3.1.2 *Ein* digitales Signaturverfahren *ist durch ein Tupel* $(K, M, S,$ KeyGen, Sign, Ver$)$ *gegeben. Die Menge* K *bildet den* Schlüsselraum *mit Paaren* $(e, d) \in K$*, wobei* e öffentlicher Schlüssel *und* d geheimer Schlüssel *genannt wird. Die Menge* M *aller signierbaren Nachrichten heißt* Nachrichtenraum *und die Menge* S *aller möglichen Signaturen heißt* Signaturraum.

KeyGen *ist der* Schlüsselerzeugungsalgorithmus*. In Abhängigkeit eines Sicherheitsparameters* $n \in \mathbb{N}$ *liefert er ein Schlüsselpaar* $(e, d) \in K(n)$*, wobei* $K(n)$ *der durch* n *erzeugte Schlüsselraum ist. Man schreibt*

$$(e, d) \leftarrow \text{KeyGen}(n). \tag{3.4}$$

Sign *ist der* Signieralgorithmus*. Für ein gegebenes* $n \in \mathbb{N}$ *als Sicherheitsparameter, einen geheimen Schlüssel* d *eines Schlüsselpaares* $(e, d) \in K$ *und einer Nachricht* $m \in M$ *liefert er eine Signatur* $s \in S$*. Es gilt also*

$$s \leftarrow \text{Sign}(n, d, m). \tag{3.5}$$

Der Verifikationsalgorithmus Ver *gibt bei Eingabe des Sicherheitsparameters* n*, des öffentlichen Schlüssels* e *aus* $(e, d) \in K$*, einer Nachricht* $m \in M$ *und einer Signatur* $s \in S$ *einen Wahrheitswert* $b \in \{true, false\}$ *zurück. Im Falle einer gültigen Signatur gilt* $b \rightarrow true$ *und wenn die Signatur ungültig ist, gilt* $b \rightarrow false$*. Man schreibt*

$$b \leftarrow \text{Ver}(n, e, m, s). \tag{3.6}$$

Eine Signatur s *einer Nachricht* m *soll gültig sein, wenn sie für ein festes Schlüsselpaar* (e, d) *und einen festen Parameter* n *mit Gleichung 3.5 unter Verwendung des geheimen Schlüssels* d *signiert wurde. Die Signatur soll dann mithilfe des zugehörigen öffentlichen Schlüssels* e *verifiziert werden können. Ein digitales Signaturverfahren ist vollständig, wenn für alle* $(e, d) \in K(n)$ *und alle Nachrichten* $m \in M$ *gilt:*

$$\forall s \leftarrow \text{Sign}(n, d, m): \quad \text{Ver}(n, e, m, s) \rightarrow true \tag{3.7}$$

Viele digitale Signaturverfahren sind in der Lage, Nachrichten von beliebiger Länge zu signieren[16]. Für andere Verfahren ist es jedoch erforderlich, die Nachricht auf eine Menge mit Elementen von begrenzter Länge abzubilden, bevor sie signiert werden kann. Hierfür werden sogenannte *Hashfunktionen* verwendet, die im folgenden Abschnitt vorgestellt werden.

3.1.3 Hashfunktionen

Definition 3.1.3 *Es sei K die Menge aller möglichen Eingangsdaten und V die Menge der möglichen* Hashwerte*. Dann ist eine* Hashfunktion *h eine Abbildung*

$$h : K \to V \tag{3.8}$$

mit $|K| \geq |V|$.

Eine Hashfunktion bildet häufig Elemente von beliebiger Länge auf Elemente fester Länge ab und ist daher in der Regel nicht injektiv.

3.2 RSA

Das RSA-Verfahren gilt als das erste und noch immer das wichtigste Public-Key-Verfahren[16]. Es wurde von Ronald Rivest, Adi Shamir und Leonard Adleman entwickelt und basiert wesentlich auf dem Problem, große Zahlen in Primfaktoren zu zerlegen[66]. Im Folgenden sollen die grundlegenden Aspekte des Verfahrens behandelt werden.

3.2.1 Schlüsselerzeugung

Ein $k \in \mathbb{N}$ bezeichne den Sicherheitsparameter. Zunächst werden vom Algorithmus zwei zufällige Primzahlen p und q gewählt mit

$$n = p \cdot q, \tag{3.9}$$

wobei n eine k-Bit-Zahl sein soll. Weiterhin wird eine natürliche Zahl e gewählt mit $1 < e < \varphi(n)$, welche die Bedingung

$$\gcd(e, \varphi(n)) = 1. \tag{3.10}$$

erfüllt. Daraufhin wird mithilfe des *Erweiterten Euklidischen Algorithmus* die natürliche Zahl d ermittelt[16]. Dabei gilt: wenn Gleichung 3.10 erfüllt ist, existiert ein $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$ und

$$d \cdot e \equiv 1 \bmod \varphi(n). \tag{3.11}$$

Der öffentliche Schlüssel ist nun durch das Paar (n, e) gegeben und d bildet den geheimen Schlüssel. Somit wird der Schlüsselraum durch Paare $((n, e), d)$ gebildet. Es ist üblich, dass für den Sicherheitsparameter k gerade Zahlen verwendet werden, damit p und q als $k/2$-Bit Zahlen gewählt werden können[16].

3.2.2 Verschlüsselung

Der Klartextraum sei $\mathbb{Z}_n$ mit Nachrichten $m \in \mathbb{Z}_n$. Der Verschlüsselung einer Nachricht erfolgt mithilfe des öffentlichen Schlüssels (n, e). Dabei wird der Chiffretext $c \in \mathbb{Z}_n$ berechnet durch

$$c = m^e \bmod n. \tag{3.12}$$

3.2.3 Entschlüsselung

Ein gegebener Chiffretext $c \in \mathbb{Z}_n$ sei wie in Gleichung 3.12 angegeben mit dem öffentlichen Schlüssel (n, e) des Schlüsselpaares $((n, e), d)$ verschlüsselt worden. Die Entschlüsselung erfolgt nun mit dem geheimen Schlüssel d über

$$m = c^d \bmod n. \tag{3.13}$$

Der Korrektheit des Verfahrens lässt sich mithilfe des Satzes von Euler-Fermat zeigen. Es wird hierfür auf die Originalpublikation verwiesen[66].

3.3 Diffie-Hellman-Schlüsselaustausch

Eine weitere Möglichkeit, einen gemeinsamen Schlüssel zwischen zwei Kommunikationspartnern auf sichere Weise auszutauschen, bieten sogenannte *Schlüsselaustauschverfahren*. Hierbei wird davon ausgegangen, dass der Kanal, über den der Schlüssel ausgetauscht werden soll, unsicher ist, also die zur Vereinbarung des gemeinsamen Schlüssels übermittelten Informationen von einem Angreifer mitgehört werden können. Das bekannteste Schlüsselaustauschverfahren ist das *Diffie-Hellman-Verfahren*, das auf Whitfield Diffie und Martin Hellman zurückgeht[26].

3.3.1 Funktionsweise

Zwei Kommunikationspartner, im Folgenden als Alice und Bob bezeichnet, möchten sich über einen unsicheren Kanal auf einen gemeinsamen Schlüssel k einigen. Zunächst wählen sie gemeinsam eine Primzahl p und eine Primitivwurzel $g \bmod p$. Wie in Abschnitt 2.2 erläutert, existieren Primitivwurzeln in $(\mathbb{Z}/p\mathbb{Z})^*$, wenn p eine Primzahl ist. Alice wählt nun eine natürliche Zahl $a \in \{0, \ldots, p-2\}$ und berechnet

$$A = g^a \bmod p. \tag{3.14}$$

Alice sendet A an Bob. Bob seinerseits wählt ebenfalls eine Zahl $b \in \{0, \ldots, p-2\}$, berechnet damit

$$B = g^b \bmod p \tag{3.15}$$

und sendet B an Alice. Aus den übermittelten Werten A und B können Alice und Bob nun jeweils den gemeinsamen Schlüssel k berechnen über

$$k = A^b \bmod p = B^a \bmod p = g^{ab} \bmod p. \tag{3.16}$$

Zur Berechnung des Schlüssels werden also die ausgetauschten Werte A und B und die geheimen Werte von a und b benötigt, eine direkte Berechnung des Schlüssels ausschließlich anhand der ausgetauschten Informationen ist nicht möglich.

3.4 DSA

Der *Digital Signature Algorithm* (DSA) ist ein digitales Signaturverfahren, das 1991 vom US-amerikanischen *National Institute of Standards and Technology* (NIST) vorgeschlagen und später zum Standard erklärt wurde[16].

3.4.1 Schlüsselerzeugung

Der Sicherheitsparameter wird durch das Paar (k, l) gebildet mit $(k, l) \in \{(1024, 160), (2048, 256), (3072, 256)\}$. Es werden nun eine l-Bit Primzahl q und eine k-Bit Primzahl p mit $q \mid p-1$ zufällig gewählt. Des Weiteren wird eine Zahl $x \in \{2, \ldots, p-1\}$ gewählt, um eine Zahl g zu berechnen.

$$g = x^{(p-1)/q} \bmod p \tag{3.17}$$

Dabei soll der Zusammenhang $g \neq 1 \bmod p$ erfüllt sein. Da q wie oben gefordert Teiler von $p-1$ ist, ist der Exponent $(p-1)/q$ immer eine natürliche Zahl. Zuletzt wird eine Zahl $a \in \{2, \ldots, p-1\}$ gewählt und damit

$$A = g^a \bmod p \tag{3.18}$$

berechnet. Das Tupel (p, q, g, A) bildet den öffentlichen Schlüssel und a den geheimen Schlüssel.

3.4.2 Signierung

Eine binär vorliegende Nachricht $m \in \{0, 1\}^*$ soll signiert werden. Es wird zunächst eine bekannte Hashfunktion h gewählt mit

$$h \colon \mathbb{Z}_2^* \to \{1, \ldots, q-1\} \tag{3.19}$$

und damit der Hashwert m_h der Nachricht m berechnet.

$$m_h = h(m) \tag{3.20}$$

Es folgt die zufällige Auswahl einer Zahl $n \in \{1, \ldots, q-1\}$ und die Berechnung von

$$r = (g^n \bmod p) \bmod q \tag{3.21}$$

und

$$s = n^{-1}(m_h + ar) \bmod q, \tag{3.22}$$

wobei n^{-1} das multiplikative Inverse von $n \bmod q$ und a der geheime Schlüssel ist. Da q eine Primzahl ist, ist jede Zahl $n \in \{1, \ldots, q-1\}$ in $\mathbb{Z}_q$ invertierbar (vgl. Abschnitt 2.2). Damit s invertierbar ist, muss $m_h + ar \neq 0 \bmod q$ gelten. Das Paar (r, s) bildet die Signatur der Nachricht m.

3.4.3 Verifikation

Wenn eine Nachricht unter Verwendung des geheimen Schlüssels a signiert wurde, kann die Verifikation der Signatur (r, s) über den zugehörigen öffentlichen Schlüssel, also das Tupel (p, q, g, A) erfolgen. Dieser wird auch *Verifikationsschlüssel* genannt. Zunächst wird geprüft, ob die folgenden Bedingungen durch die Parameter r und s der Signatur erfüllt werden:

$$1 \leq r \leq q-1 \tag{3.23a}$$

$$1 \leq s \leq q-1 \tag{3.23b}$$

Ist dies nicht der Fall, gilt die Signatur als ungültig. Andernfalls wird mithilfe der Parameter p, q, g und A des öffentlichen Schlüssels und des mittels Gleichung 3.20 erlangten Nachrichten-Hashes m_h der Zusammenhang

$$r = ((g^{(s^{-1}m) \bmod q} A^{(rs^{-1}) \bmod q}) \bmod p) \bmod q \tag{3.24}$$

überprüft. Ist Gleichung 3.24 erfüllt, ist die Signatur gültig. Dies lässt sich unmittelbar aus dem Zusammenhang

$$g^{(s^{-1}m) \bmod q} A^{(rs^{-1}) \bmod q} \equiv g^{s^{-1}(m+ra)} \equiv g^n \bmod p, \tag{3.25}$$

ableiten, der erfüllt ist, wenn die Signatur korrekt unter Verwendung von Gleichung 3.21 und Gleichung 3.22 konstruiert wurde.

3.5 Sicherheit und Effizienz

In diesem Abschnitt soll die Sicherheit und die Effizienz der vorgestellten kryptographischen Verfahren diskutiert werden. Beim RSA- und dem DSA-Verfahren hat ein Angreifer grundsätzlich die Parameter des öffentlichen Schlüssels zur Verfügung. Für das RSA-Verfahren gilt: Ein Angreifer kann den privaten Schlüssel d mithilfe

des öffentlichen Schlüssels e berechnen, wenn er die Primfaktoren p und q kennt. In diesem Falle ist lediglich die Gleichung

$$1 \equiv de \bmod (p-1)(q-1) \tag{3.26}$$

zu lösen. Eine Möglichkeit das RSA-Verfahren zu brechen, besteht also darin, die Primfaktoren p und q des RSA-Moduls n zu finden, was eine spezifische Instanz des Faktorisierungsproblems darstellt. Da n aus nur zwei Primfaktoren besteht, ist das Finden eines einzigen nichttrivialen Teilers bereits ausreichend. Um die Faktorisierung des RSA-Moduls n hinreichend schwer zu gestalten, existieren Richtwerte für die Mindestgröße von n in Bit, die im Rahmen des EU-Projektes ECRYPT II ermittelt wurden. Hierbei wurde die durch das Mooresche Gesetz beschriebene Zunahme der Rechengeschwindigkeit von Computern berücksichtigt[16]. In Tabelle 3.1 sind die empfohlenen Mindestgrößen für das RSA-Modul für unterschiedliche Sicherheitslevel aufgeführt. Die Größe des Moduls wird Verfahrens-seitig sichergestellt, indem die

Sicherheitslevel	Mindestgröße [Bit]
veralteter Standard	1024
sicher in näherer Zukunft (2019 - 2028)	3072
sicher in absehbarer Zukunft (2019 - 2068)	15360

Tabelle 3.1: Empfohlene Mindestgrößen für das RSA-Modul[6].

Primzahlen p und q solange als zufällige $k/2$-Bit Zahlen gewählt werden, bis n eine k-Bit Zahl ist. Die zufällige Wahl von p und q stellt dabei sicher, dass keine zusätzlichen Informationen über die Struktur der gewählten Primfaktoren durch ein Faktorisierungsverfahren genutzt werden können. Der Exponent e sollte von hinreichender Größe sein, um die Möglichkeit eines *Low-Exponent*-Angriffs zu verhindern, aber klein genug, um noch eine effiziente Verschlüsslung zuzulassen[16]. In der Praxis ist $e = 2^{16} + 1$ üblich. Kleinere Werte des Parameters d wirken sich vorteilhaft auf Speicherbedarf und Effizienz des Verfahrens aus, jedoch wird bei Boneh und Durfee gezeigt, dass RSA für Parameter $d < n^{0,292}$ gebrochen werden kann. Ver- und Entschlüsselung erfordern beim RSA-Verfahren jeweils eine Exponentiation modulo n. Je kleiner dabei der Exponent ist, umso effizienter lässt sich die Berechnung durchführen. Wenn $e = 2^{16} + 1$ ist, ist d in der Größenordnung von n. Die Verschlüsselung erfolgt damit also in der Regel schneller als die Entschlüsselung.

Das DSA-Verfahren ist eine Weiterentwicklung des El-Gamal Signaturverfahrens mit verbesserter Effizienz und einer genaueren Beschreibung der Parameterwahl[16]. Prinzipiell gilt für die Sicherheit des Verfahrens: Ist ein Angreifer in der Lage, diskrete

Logarithmen in der von $g + p\mathbb{Z}$ erzeugten Untergruppe von $(\mathbb{Z}/p\mathbb{Z})^*$ zu berechnen, dann kann er den geheimen Schlüssel a berechnen und somit selbst valide Signaturen erzeugen (vgl. Unterabschnitt 2.4.1). Dies bildet also das sicherheitsbestimmende Problem des Verfahrens. Weiterhin ist die Auswahl eines neuen Parameters n für jede Signatur und die Verwendung einer Hashfunktion zwingend erforderlich, da anderenfalls die Möglichkeit zu existenziellen Fälschungen besteht, also der Erzeugung einer gültigen neuen Signatur zu einer beliebigen Nachricht[16]. Für den Parameter p gelten die gleichen Größenempfehlungen wie für das RSA-Modul[6] (vgl. Tabelle 3.1). Für die Effizienz des DSA Verfahrens ist es von Vorteil, dass die Berechnungen in der kleineren Untergruppe $g + p\mathbb{Z}$ der Gruppe $(\mathbb{Z}/p\mathbb{Z})^*$ durchgeführt werden. Für alle derzeitigen Verfahren zur Berechnung diskreter Logarithmen stellt dieser Umstand jedoch keinen Vorteil dar[16]. Die Performanz des Verfahrens profitiert weiterhin von der Möglichkeit, Vorberechnungen des Parameters r durchzuführen und somit die Signierung zu beschleunigen. Weiterhin erlaubt die Struktur der Verifikationsberechnung eine simultane Exponentiation. Grundsätzlich lässt sich feststellen, dass die bei Signierung und Verifizierung verwendeten Exponenten mit einer Größe von 160 bzw. 256 Bit deutlich kleiner sind, als beispielsweise beim RSA-Verfahren, was einen weiteren Effizienzvorteil bedeutet.

Die Sicherheit des Diffie-Hellman-Verfahrens beruht darauf, dass ein Angreifer zwar über die Parameter p und q sowie A und B verfügt, um den geheimen Schlüssel K zu berechnen, nicht jedoch über die diskreten Logarithmen a von A und b von B. Diese Konstellation wird auch als Diffie-Hellman-Problem (DH) bezeichnet. Ist ein Angreifer in der Lage, diskrete Logarithmen modulo p zu berechnen, dann kann er das Diffie-Hellman Problem lösen. Es ist allerdings nicht bewiesen, dass dies die einzige Möglichkeit zur Lösung von DH ist. Dies würde bedeuten, dass die Fähigkeit, DH effizient berechnen zu können auch impliziert, DLP effizient lösen zu können. Für die Größe des Parameters q gelten die gleichen Empfehlungen wie für das RSA Modul[6](vgl. Tabelle 3.1).

Hinsichtlich Sicherheit und Effizienz besteht eine Weiterentwicklung der hier vorgestellten asymmetrischen Kryptosysteme in einer Gruppe von Verfahren, die unter dem Begriff der *Elliptische-Kurven-Kryptographie* (ECC) zusammengefasst werden.

Definition 3.5.1 *Für zwei feste Parameter $a, b \in \mathbb{R}$ bilden alle Punkte $(x, y) \in \mathbb{R}^2$, die die Gleichung*

$$y^2 = x^3 + ax + b \tag{3.27}$$

erfüllen, eine elliptische Kurve *über dem Körper der reellen Zahlen. Für a und b soll dabei der Zusammenhang*

$$-4a^3 - 27b^2 \neq 0 \tag{3.28}$$

gelten, um Singularitäten auszuschließen.

Eine elliptische Kurve ist also eine Menge von Punkten über einem Körper. Für die Punkte einer elliptischen Kurve sei weiterhin eine Addition definiert. Damit kann eine additive zyklische Gruppe erzeugt werden[72]. Mit einem Kurvenpunkt P als Erzeuger sind die Elemente der Gruppe die Vielfachen des Punktes P auf der Kurve. Ein beliebiges Element Q lässt sich also als Vielfaches nP angeben. Das Problem des diskreten Logarithmus ist analog für elliptische Kurven definiert. Die Berechnung des diskreten Logarithmus in dieser Gruppe entspricht dem Finden einer natürlichen Zahl n zu einem gegebenen Element Q mit

$$Q = nP. \tag{3.29}$$

Die Sicherheit entsprechender Kryptosysteme beruht auf der Schwierigkeit, diskrete Logarithmen in dieser Gruppe von Punkten einer elliptischen Kurve zu berechnen. Da dies schwerer ist, als die Berechnung diskreter Logarithmen über endlichen Körpern oder die Faktorisierung großer Zahlen, kommen diese Verfahren mit erheblich kürzeren Schlüsseln aus[72].

4 Gitter

Dieses Kapitel behandelt allgemeine Gitter, deren Darstellung und Eigenschaften, sowie spezielle Klassen von Gittern, die aufgrund ihrer Eigenschaften interessant für kryptographische Anwendungen sind. Darüber hinaus werden die wichtigsten Problemstellungen in Gittern vorgestellt und ein Einblick in das Gebiet der Gitterreduktion gewährt. Es handelt sich hierbei um einen grundlegenden Ausschnitt eines weitaus umfangreicheren Themengebietes. Die Auswahl der vorgestellten Elemente erfolgte in Hinblick auf die in Kapitel 5 behandelten Verfahren und Konzepte.

4.1 Darstellung von Gittern

Definition 4.1.1 *Mit $b_1, b_2, \ldots, b_m \in \mathbb{R}^n$ seien m linear unabhängige Vektoren mit $m \leq n$ gegeben. Dann ist*

$$\Lambda := \langle b_1, b_2, \ldots, b_m \rangle_{\mathbb{Z}} := \sum_{i=1}^{m} \mathbb{Z} b_i \tag{4.1}$$

definiert als ein Gitter vom Rang m *oder auch m-dimensionales Gitter. Im Falle $m = n$ heißt das Gitter* volldimensional *oder* vollständig. *Die Vektoren $b_1, b_2, \ldots, b_m$ heißen* Basisvektoren *oder auch* Basis von Λ.

Ein Gitter ist somit eine diskrete additive Untergruppe des $\mathbb{R}^n$. Das heißt auch, dass $\mathbb{Z}^n$ selbst ein Gitter bildet. Ein Gitter wird in der Regel durch Angabe einer *Basismatrix* beschrieben.

Definition 4.1.2 *Durch die Basisvektoren $b_1, b_2, \ldots, b_m \in \mathbb{R}^n$ sei ein Gitter Λ vom Rang $m \leq n$ gegeben. Die Matrix*

$$B := (b_1, b_2, \ldots, b_m) \in \mathbb{R}^{n \times m} \tag{4.2}$$

heißt dann Basismatrix *von Λ. Somit lässt sich das Gitter schreiben als*

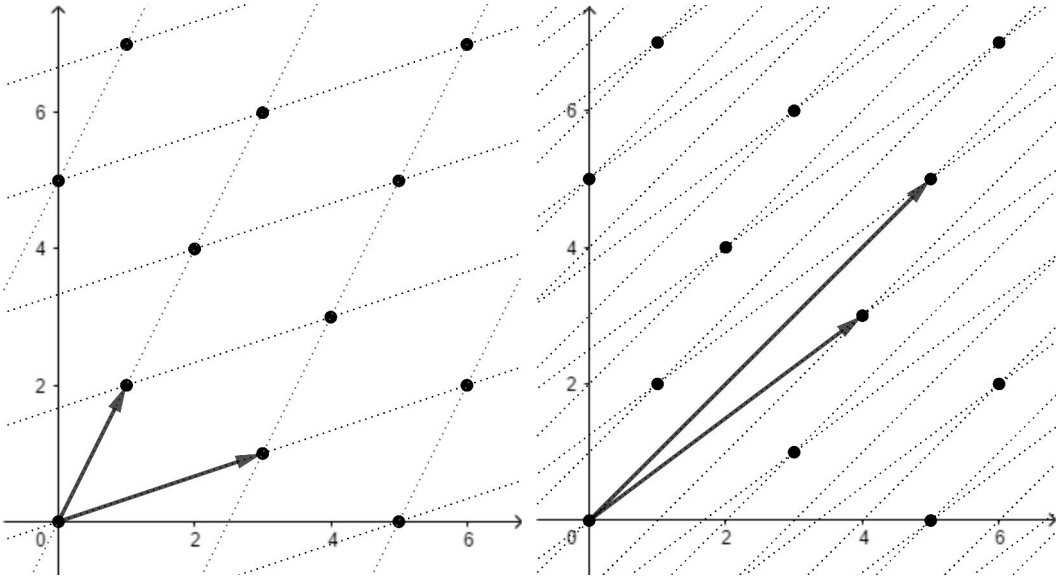
$$\Lambda = \Lambda(B) = \{Bx \mid x \in \mathbb{Z}^m\}. \tag{4.3}$$

Die Elemente eines Gitters werden als *Gitterpunkte* oder *Gittervektoren* bezeichnet. Ein beliebiges Element eines Gitters Λ vom Rang m mit $B \in \mathbb{R}^{n \times m}$ lässt sich sowohl

als Vektor $y \in \mathbb{R}^n$ angeben oder auch als Vektor $x \in \mathbb{Z}^m$ mit $y = Bx$. Jedes Gitter kann durch eine Menge unterschiedlicher Basen beschrieben werden, worauf im Verlauf dieses Abschnitts noch näher eingegangen wird. Offenbar bilden beispielsweise die Matrizen

$$B_1 = \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 4 & 5 \\ 3 & 5 \end{pmatrix}$$

Basen ein und desselben Gitters, wie Abbildung 4.1 veranschaulicht. Es gilt also für



(a) Gitter mit Basis B_1 (b) Gitter mit Basis B_2

Abbildung 4.1: Unterschiedliche Basen desselben zweidimensionalen Gitters.

das Gitter $\Lambda(B_1) = \Lambda(B_2)$, was sich dadurch zeigen lässt, dass beide Vektoren der Basis B_1 als Linearkombination der Vektoren von B_2 darstellbar sind.

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} = -\begin{pmatrix} 4 \\ 3 \end{pmatrix} + \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$
$$\begin{pmatrix} 3 \\ 1 \end{pmatrix} = 2\begin{pmatrix} 4 \\ 3 \end{pmatrix} - \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$

Um den Zusammenhang zwischen unterschiedlichen Basen eines Gitters allgemein zu beschreiben, wird der Begriff der *unimodularen Matrix* definiert.

Definition 4.1.3 *Eine quadratische Matrix $U \in \mathbb{Z}^{n \times n}$, für deren Determinante gilt* $\det U \in \{-1, 1\}$*, heißt* unimodular*.*

Daraus folgt, dass jede unimodulare Matrix regulär ist und dass wenn $U \in \mathbb{Z}^{n\times n}$ unimodular ist, ihre Inverse Matrix U^{-1} ebenfalls ganzzahlig und unimodular ist, da für die Inverse Matrix $U^{-1} = (\hat{u}_{ij})$ gilt

$$\hat{u}_{ij} = \frac{(-1)^{i+j} \det U_{ji}}{\det U} \tag{4.4}$$

mit der Matrix U_{ji}, die durch Streichung der j-ten Zeile und i-ten Spalte aus der Matrix U entsteht.

Satz 4.1.1 *Für unterschiedliche Gitterbasen $B, B' \in \mathbb{R}^{n\times m}$ eines m-dimensionalen Gitters $\Lambda(B) = \Lambda(B') \subset \mathbb{R}^n$ besteht der Zusammenhang: es existiert eine unimodulare Matrix $U \in \mathbb{Z}^{n\times n}$, sodass gilt*

$$B' = BU. \tag{4.5}$$

Beweis 4.1.1 *Es soll gezeigt werden, dass B' eine Basis des Gitters $\Lambda(B)$ ist, wenn $U \in \mathbb{Z}^{n\times n}$ unimodular ist. Aus der Gleichung $B' = BU$ mit den Spaltenvektoren von $B = (b_1, \ldots, b_m)$ und $B' = (b'_1, \ldots, b'_m)$, sowie den Einträgen der Matrix $U = (u_{ij})$ folgt der Zusammenhang*

$$b'_j = \sum_{i=1}^{n} u_{ij} b_i. \tag{4.6}$$

Das heißt, jeder Spaltenvektor b'_j ist eine ganzzahlige Linearkombination der Basisvektoren $b_1, \cdots, b_m$, da U ganzzahlig ist. Damit sind die Vektoren $b'_1, \ldots, b'_m$ Elemente des Gitters $\Lambda(B)$. Wenn U regulär ist, folgt daraus

$$\begin{aligned} B'U^{-1} &= BUU^{-1} \\ B'U^{-1} &= B. \end{aligned} \tag{4.7}$$

Wenn U ganzzahlig und unimodular ist, ist U^{-1} ebenfalls ganzzahlig und unimodular, also lassen sich die Spaltenvektoren $b_1, \ldots, b_m$ mithilfe des oben gezeigten Zusammenhangs als ganzzahlige Linearkombination der Spaltenvektoren $b'_1, \ldots, b'_m$ darstellen. Somit ist B' eine Basis von $\Lambda(B)$, wenn U ganzzahlig und unimodular ist.

Wie oben gezeigt, lässt sich die Basismatrix B_1 durch ganzzahlige Linearkombination der Spaltenvektoren von B_2 bilden. Die dabei verwendeten Koeffizienten seien in der Matrix

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

zusammengefasst, womit der Zusammenhang

$$B_1 = B_2 C$$

erfüllt ist. Wie bekannt ist, sind B_1 und B_2 Basen desselben Gitters und die Matrix C ist mit $\det C = -1$ erwartungsgemäß unimodular. Es lässt sich leicht überprüfen, dass die Matrix

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

mit $\det C^{-1} = -1$ ebenfalls unimodular ist und den Zusammenhang $B_1 C^{-1} = B_2$ erfüllt.

Die Determinante eines Gitters ist eine Größe, die invariant gegenüber der gewählten Gitterbasis ist. Für ein Gitter $\Lambda(B)$ vom Rang m hat sie den gleichen Wert wie das Euklidische Volumen $\mathrm{vol}\,(\mathcal{P})$ des m-dimensionalen Parallelepipeds $\mathcal{P}$, das durch die Basisvektoren des Gitters aufgespannt wird. Diese sogenannte *Grundmasche* oder *Fundamentalmasche* entspricht der beschränkten Menge

$$F_\Lambda = \{Bx \mid x \in \mathbb{R}^m, 0 \leq x_i < 1\} \tag{4.8}$$

und ist in Abbildung 4.2 exemplarisch für ein zweidimensionales Gitter dargestellt.

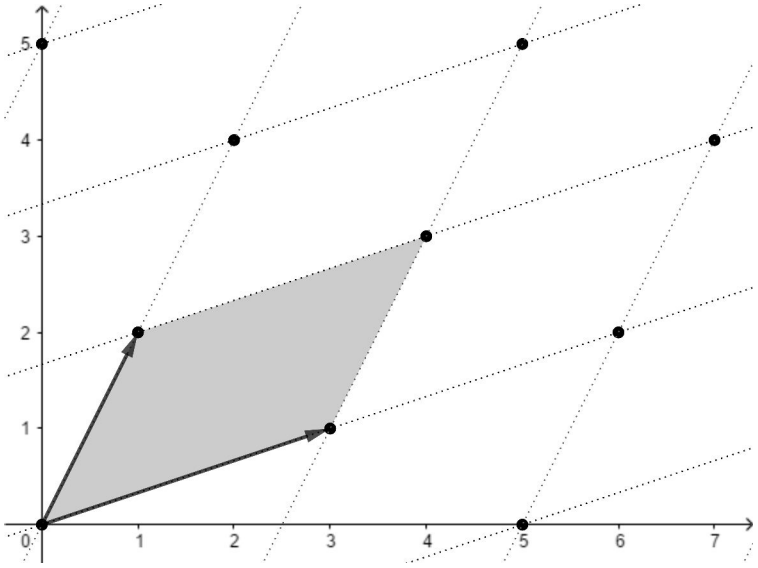


Abbildung 4.2: Grundmasche eines zweidimensionalen Gitters.

Definition 4.1.4 *Die Determinante eines Gitters $\Lambda(B)$ ist definiert als*

$$\det(\Lambda(B)) := vol(\mathcal{P}) = \sqrt{\det(B^T B)} \tag{4.9}$$

und wird auch als Gitterdeterminante *bezeichnet.*

Aus Satz 4.1.1 folgt, dass die Determinante eines Gitters nicht von der gewählten Basis abhängt, weil sich für die Gitterdeterminante

$$\begin{aligned} \det(\Lambda(B')) &= \sqrt{\det({B'}^T B')} \\ &= \sqrt{\det(U^T B^T B U)} \\ &= \sqrt{\det(B^T B)} = \det(\Lambda(B)) \end{aligned} \tag{4.10}$$

zeigen lässt. Dabei sind B und B' zwei beliebige Basen desselben Gitters und U ist die unimodulare Matrix, die den Zusammenhang $B' = BU$ erfüllt. Das heißt auch, dass alle möglichen Grundmaschen eines Gitters das gleiche euklidische Volumen besitzen.

4.2 Spezielle Klassen von Gittern

Neben den im Vorangegangenen beschriebenen allgemeinen Gittern, existieren spezielle Klassen von Gittern, die sich auf eine andere Weise definieren lassen. Im Kontext dieser Arbeit sind dabei vor allem modulare und ideale Gitter von Bedeutung.

4.2.1 Modulare Gitter

Definition 4.2.1 *Mit einem Modul $q \in \mathbb{Z}$ und einer Matrix $A \in \mathbb{Z}_q^{n \times m}$ bilden alle Vektoren $x \in \mathbb{Z}_q^n$, die der Gleichung*

$$Ax \equiv 0 \bmod q \tag{4.11}$$

genügen, ein m-dimensionales modulares Gitter $\Lambda_{A,q} \subseteq \mathbb{Z}^n$.

Es gilt also $\Lambda_{A,q} = \{x \in \mathbb{Z}^n : Ax \bmod q = 0\}$. Die Spaltenvektoren der Matrix A sind demnach keine Gittervektoren, weshalb A auch nicht als Basis zu verstehen ist. Modulare Gitter sind im kryptographischen Kontext im Zusammenhang mit dem *Shortest Integer Solution* Problem (SIS) (s. Unterabschnitt 4.3.3) von Interesse.

4.2.2 Ideale Gitter

Definition 4.2.2 *Gegeben sei der Ring $R = \mathbb{Z}[X]/f(x)$ mit einem monischen Polynom $f(x)$ vom Grad n. Ein Gitter $\Lambda_{R,I}$ heißt* ideales Gitter*, wenn ein Ideal $I \subseteq R$ existiert, sodass gilt*

$$I = \{a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} : a \in \Lambda_{R,I}\} \tag{4.12}$$

mit Koeffizientenvektoren $a = (a_0, \ldots, a_{n-1})$.

Jedes Ideal $I \subseteq R$ definiert also ein Gitter $\Lambda_{R,I}$, die Koeffizientenvektoren $a \in \mathbb{Z}^n$ der Polynome $a(x) \in I$ bilden dabei die Gittervektoren.

Die idealen Gitter des Rings $R = \mathbb{Z}[X]/(x^n - 1)$ heißen *zyklische Gitter*. Sie verfügen über die Eigenschaft, dass jeder zyklische Shift der Koeffizienten eines Gittervektors wieder einen Gittervektor liefert. Ist beispielsweise der Vektor $(1, 2, 3, 4)^T$ ein Element eines zyklischen Gitters, dann ist auch der Vektor $(4, 1, 2, 3)^T$ Element desselben Gitters. Es lässt sich leicht nachvollziehen, dass der zyklische Rechts-Shift im Ring $R = \mathbb{Z}[X]/(x^n - 1)$ der Multiplikation eines Elementes mit x entspricht. Die Vorteile zyklischer Gitter für kryptographische Anwendungen werden ausführlich bei Lyubashevsky, Peikert und Regev dargestellt[50]. So lassen sich n-dimensionale zyklische Gitter beispielsweise unter Angabe eines einzigen Vektors mit n Einträgen darstellen, während allgemeine Gitter eine Basismatrix mit mindestens $n \times n$ Einträgen benötigen.

4.3 Probleme in Gittern

In diesem Abschnitt werden die für diese Arbeit relevanten Problemstellungen in Gittern behandelt. Hierfür erfolgt die Festlegung einer Norm zur Bestimmung der Länge von Gittervektoren und zur Ermittlung von Abständen innerhalb eines Gitters.

Definition 4.3.1 (Euklidische Vektornorm) *Für einen Vektor $x \in \mathbb{R}^n$ sei die* Euklidische Vektornorm *gegeben durch*

$$\|x\| = \sqrt{\sum_{i=1}^{n} (x_i)^2} \tag{4.13}$$

Damit lässt sich Abstand zwischen zwei Vektoren definieren. Dieser wird als *Euklidischer Abstand* oder auch *Euklidische Distanz* bezeichnet.

Definition 4.3.2 (Euklidischer Abstand) *Der Euklidische Abstand zweier Vektoren $x, y \in \mathbb{R}^n$ sei definiert als*

$$dist\,(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.14}$$

Unter Verwendung der euklidischen Vektornorm lassen sich für ein m-dimensionales Gitter genau m *sukzessive Minima* beschreiben. Das i-te sukzessive Minimum λ_i eines Gitters entspricht dabei dem Radius einer Sphäre um den Koordinatenursprung, der gerade so klein ist, dass sich genau i linear unabhängige Vektoren auf dem Rand bzw. innerhalb der Sphäre befinden. In Abbildung 4.3 sind die sukzessiven Minima λ_1 und λ_2 eines zweidimensionalen Gitters dargestellt.

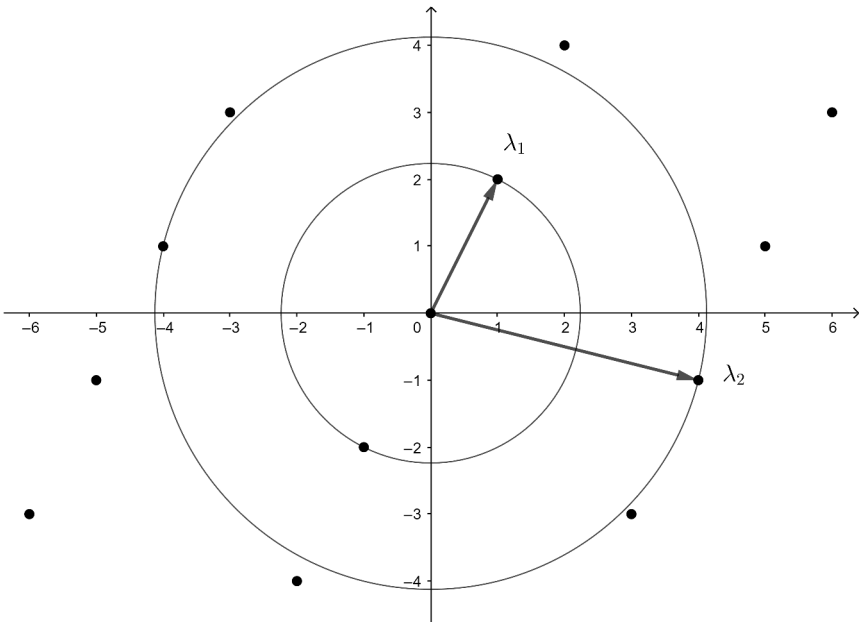


Abbildung 4.3: Sukzessive Minima eines zweidimensionalen Gitters.

Definition 4.3.3 *Für ein m-dimensionales Gitter Λ im n-dimensionalen Euklidischen Raum $\mathbb{R}^n$ sind die* sukzessiven Minima λ_i *mit $i = 1, \ldots, n$ definiert als die jeweils kleinste reelle Zahl r für die gilt: es existieren i linear unabhängige Vektoren $x_1, x_2, \ldots, x_i \in \Lambda$, sodass $\|x_1\|, \|x_2\|, \ldots, \|x_i\| \leq r$. Das heißt, dass*

$$\lambda_i = \min_{x_1, x_2, \ldots, x_i \in \Lambda} max(\|x_1\|, \|x_2\|, \ldots, \|x_i\|) \tag{4.15}$$

für alle möglichen Sätze von i linear unabhängigen Vektoren $x_1, x_2, \ldots, x_i \in \Lambda$.

Damit entspricht das *erste sukzessive Minimum* λ_1 eines Gitters Λ der Länge eines kürzesten Vektors $x_1 \in \Lambda$. Aus der Definition der sukzessiven Minima lässt sich also direkt das *Shortest Vektor Problem* (SVP) ableiten als die Suche nach einem kürzesten Vektor x_1 mit $\|x_1\| = \lambda_1$ in einem gegebenen Gitter.

4.3.1 Shortest Vector Problem

Definition 4.3.4 *Für ein durch $B \in \mathbb{Z}^{n \times m}$ gegebenes m-dimensionales Gitter im Euklidischen Raum $\mathbb{R}^n$ wird ein Vektor $x \in \mathbb{Z}^m \setminus \{0\}$ von kürzester Länge gesucht, sodass*

$$\lambda_1 = \|Bx\| \leq \|By\| \tag{4.16}$$

für alle Vektoren $y \in \mathbb{Z}^m \setminus \{0\}$.

Eine Variante des SVP, bei dem ein kürzester Vektor nur annähernd bestimmt werden soll, heißt *Approximate Shortest Vector Problem* (App-SVP). Für praktische Belange, wie z.B. die in Kapitel 5 vorgestellten Verfahren ist in der Regel das Problem App-SVP von Interesse. Ausgehend von Definition 4.3.4 ist es hierbei ausreichend, einen Vektor zu finden, der der Gleichung

$$\lambda_1 = \|Bx\| \leq \gamma \|By\| \quad \forall y \in \mathbb{Z}^m \backslash \{0\} \tag{4.17}$$

genügt, mit $\gamma \in \mathbb{R}$ als Approximationsfaktor. Man schreibt hierfür auch γ-SVP, wenn ein fester Approximationsfaktor angegeben werden soll. In der Regel wird der Approximationsfaktor als $\gamma(n)$, also in Abhängigkeit von der Gitterdimension n angegeben.

Aktuell ist kein Polynomialzeitalgorithmus bekannt für $\gamma(n) = n^{O(1)}$. Der beste Polynomialzeitalgorithmus erreicht $\gamma(n) = 2^{O(n \log\log n / \log n)}$[4]. App-SVP ist NP-schwer für konstante und in der Gitterdimension polynomielle Approximationsfaktoren[1][2][56]. Die besten Ansätze zur Berechnung des App-SVP gründen sich auf die Reduktion der Gitterbasis (s. Abschnitt 4.4), beispielsweise unter Verwendung des LLL-Algorithmus (s. Unterabschnitt 4.4.3). Dieser ermöglicht eine effiziente Approximierung eines kürzesten Gittervektors mit Faktoren $\gamma(n) = 2^{O(n)}$. Es zeigt sich, dass der LLL-Algorithmus und seine Weiterentwicklungen im durchschnittlichen Fall (Average Case) deutlich besser in arbeiten, als im Worst Case, also dem Fall, in dem der jeweilige Algorithmus zwar terminiert, aber dafür die maximal mögliche Zahl an Schritten benötigt[34][58]. Für kryptographische Anwendungen ist es also interessant, wenn Angreifer einen kürzesten Vektor mit kleinen polynomiellen Faktoren oder im Worst Case approximieren müssen, um ein entsprechendes Kryptosystem zu brechen. Es wird angenommen, dass SVP in idealen Gittern genauso schwer ist, wie in allgemeinen Gittern, jedoch steht ein Beweis oder eine Widerlegung dieser Annahme noch aus[50][64].

4.3.2 Closest Vector Problem

Eng verwandt mit dem Shortest Vector Problem ist das *Closest Vector Problem* (CVP). Hierbei ist zusätzlich zu einem Gitter $\Lambda(B) \subseteq \mathbb{Z}^n$ ein Vektor t gegeben, der kein Element des Gitters $\Lambda(B)$ ist. Gesucht wird nun derjenige Gitterpunkt mit dem kürzesten Abstand zu t.

Definition 4.3.5 *Für ein durch $B \in \mathbb{Z}^{n \times m}$ gegebenes m-dimensionales Gitter im Euklidischen Raum $\mathbb{R}^n$ ist weiterhin ein Vektor $t \in \mathbb{Z}^m$ gegeben. Gesucht wird ein Vektor $x \in \mathbb{Z}^m \backslash \{0\}$, sodass*

$$\|Bx - t\| \leq \|By - t\| \tag{4.18}$$

für alle Vektoren $y \in \mathbb{Z}^m \backslash \{0\}$.

Abbildung 4.4 veranschaulicht das Closest Vector Problem in einem zweidimensionalen Gitter. Wäre t selbst ein Gittervektor, wäre die Lösung trivial. Die Suche nach einem Vektor y, der den kürzesten Abstand zu einem gegebenen Gittervektor x hat, entspricht dem Problem SVP, da der Koordinatenursprung auch ein Punkt in jedem Gitter $\Lambda(B) \subseteq \mathbb{Z}^n$ ist.

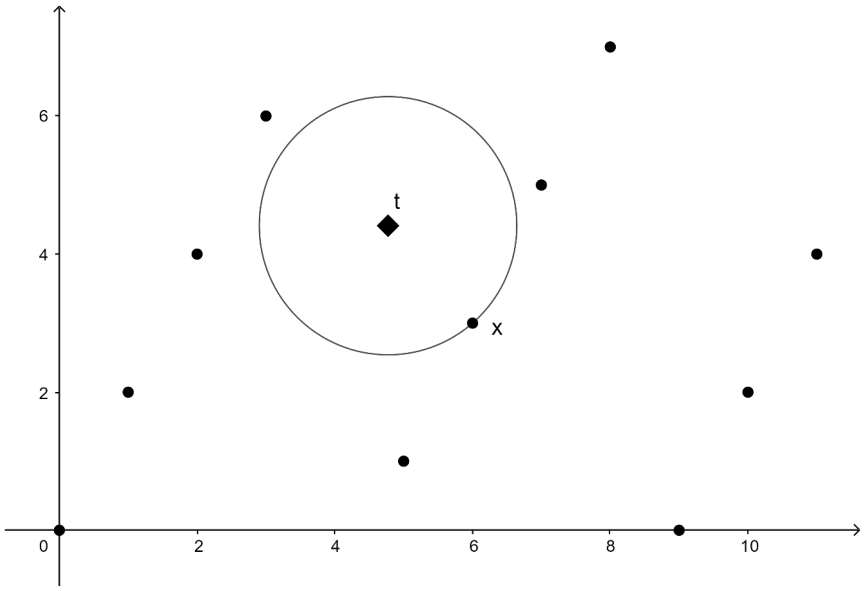


Abbildung 4.4: Minimaler Abstand eines Punktes t zu einem Gitterpunkt x.

Frühe Arbeiten zeigen bereits, dass das Finden exakter Lösungen für CVP NP-schwer ist[30]. Die Suche nach einer Näherungslösung unter Verwendung eines Approximationsfaktors heißt *Approximate Closest Vector Problem* (App-CVP). Ausgehend von obiger Definition muss der gesuchte Vektor x dann der Gleichung

$$\|Bx - t\| \leq \gamma \|By - t\| \quad \forall y \in \mathbb{Z}^m \setminus \{0\} \tag{4.19}$$

genügen, mit einem Approximationsfaktor $\gamma \in \mathbb{R}$. Für Approximationsfaktoren in der Größenordnung von n ist App-CVP immer noch NP-schwer[9]. Goldreich et al. zeigen, dass die Approximation eines kürzesten Vektors (App-SVP) nicht schwerer ist als App-CVP[33].

4.3.3 Shortest Integer Solution Problem

Das *Shortest Integer Solution Problem* (SIS) geht auf die Arbeit von Miklós Ajtai zurück[1]. Ausgangspunkt für eine Instanz $SIS_{n,m,q,\beta}$ sind die Parameter $n, m, q \in \mathbb{N}$ und ein Parameter $\beta \in \mathbb{R}$. Gegeben sei nun eine Matrix $A \in \mathbb{Z}_q^{n \times m}$. Ziel ist es, einen

nichttrivialen Vektor $x \in \mathbb{Z}^m$ zu finden, der die Bedingungen

$$\|x\| \leq \beta \tag{4.20a}$$
$$Ax = 0 \bmod q \tag{4.20b}$$

erfüllt. SIS stellt also eine Variante des Shortest Vector Problems in einem modularen Gitter dar (vgl. Unterabschnitt 4.2.1). Dabei muss in jedem Fall $\beta < q$ gelten, da sonst triviale Lösungen wie der Vektor $(q, 0, \cdots, 0)^T$ existieren. Weiterhin werden für die Parameter die Bedingungen

$$\beta \geq \sqrt{n \log q} \tag{4.21a}$$
$$m \geq n \log q \tag{4.21b}$$

vorgegeben, damit eine kurze, nichttriviale Lösung für x existiert. Weiterhin wird für das Modul q eine Größenordnung von $q \approx n^3$ empfohlen. Ajtai zeigt, dass SIS für hinreichend große Parameter im durchschnittlichen Fall (Average Case) so schwer ist, wie App-SVP im Worst Case mit polynomiellen Approximationsfaktoren[1].

Ring-SIS

Es existiert eine ringbasierte Variante von SIS, die als Ring-SIS bezeichnet wird und unter anderem von Micciancio beschrieben wird[53]. Eine Instanz Ring-$SIS_{m,q,\beta}$ des Problems verwendet die Parameter $m, n \in \mathbb{N}$ und $\beta \in \mathbb{R}$. Ausgangspunkt ist beispielsweise der Quotientenring $R = \mathbb{Z}_q/(x^n - 1)$, wobei für n gefordert wird, dass es gerade ist. Für einen gegebenen Vektor $a \in R^m$ wird ein Vektor $z \in R^m$ gesucht, der die Bedingungen

$$\|z\| \leq \beta \tag{4.22a}$$
$$a^T z = \sum_{i=1}^{m} a_i p_i = 0 \bmod q \tag{4.22b}$$

erfüllt. Die hierbei für den Vektor aus Polynomen verwendete Norm $\|z\|$ ist durch

$$\|z\| = \sqrt{\sum_{i=1}^{m} \|z_i\|^2} \tag{4.23}$$

gegeben, wobei $\|z_i\|$ die euklidische Norm des Koeffizientenvektors des jeweiligen Polynoms z_i darstellt. Lyubashevsky und Micciancio zeigen eine modifizierte Variante von Ring-SIS, die im Average Case so schwer ist, wie App-SVP im Worst Case in idealen Gittern und mit polynomiellen Approximationsfaktoren[48].

4.4 Reduktion der Gitterbasis

Da ein Gitter wie oben gezeigt durch mehrere Gitterbasen dargestellt werden kann, ist die Frage von Interesse, welche Gitterbasen hierbei günstig bzw. ungünstig im Hinblick auf die vorgestellten Gitterprobleme sind. Eine orthogonale Gitterbasis $B_\perp$ stellt hierbei den Idealfall dar, denn hier lassen sich die Problemfamilien SVP und CVP einfach lösen. In einem orthogonalen Gitter kann jeder Basisvektor einer Raumdimension zugewiesen werden. Durch Rotation des Koordinatensystems kann ggf. eine Basis $\tilde{B} = (\tilde{b}_1, \ldots, \tilde{b}_n)$ hergestellt werden, in der jeder Basisvektor $\tilde{b}_i$ nur an i-ter Stelle einen Wert ungleich 0 hat. Das bedeutet, dass jeder Gittervektor der kürzeste Vektor seiner jeweiligen Raumdimension ist. Aus dem Satz des Pythagoras folgt, dass jeder aus zwei orthogonalen Vektoren durch ganzzahlige Linearkombination gebildete Vektor, eine größere Länge als die beiden Ausgangsvektoren hat. Daraus folgt, dass sich aus den Vektoren einer orthogonalen Basis kein Gittervektor bilden lässt, der kürzer als der kürzeste Basisvektor ist. Das heißt, der kürzeste Vektor einer orthogonalen Gitterbasis $B_\perp$ entspricht dem kürzesten Vektor des Gitters $\Lambda(B_\perp)$. Weiterhin folgt damit aus der Definition der sukzessiven Minima, dass die der Länge nach sortierten Basisvektoren die sukzessiven Minima des Gitters bilden mit $\|b_{\perp,i}\| = \lambda_i$. Damit lassen sich alle Varianten des Problems SVP in einem orthogonalen Gitter einfach lösen. Das Problem CVP lässt sich im orthogonalen Gitter ebenfalls leicht lösen. Denn ein gegebener Punkt P, der nicht Element von $\Lambda(B_\perp)$ ist, liegt in jeder Dimension zwischen zwei diskreten Gitterkoordinaten. Es muss nun also für jede Gitterdimension nur ein Vergleich durchgeführt werden, welche der zwei benachbarten Gitterkoordinaten näher an P liegt. Der Gittervektor mit dem geringsten Abstand ergibt sich aus Kombination der jeweils ermittelten Koordinaten. Das Finden einer orthogonalen Basis für ein gegebenes Gitter ermöglicht also sofort die Lösung der vorgestellten Gitterprobleme. Auch wenn ein Gitter im Allgemeinen keine orthogonale Basis aufweist, erscheint es daher zweckmäßig, eine möglichst orthogonale Basis mit möglichst kurzen Basisvektoren zu finden, um so zumindest Näherungslösungen für die oben genannten Probleme zu finden. Dieser Vorgang wird als Gitterreduktion bezeichnet. Eine orthogonale Gitterbasis kann somit als vollständig reduzierte Gitterbasis betrachtet werden. Die Abwesenheit einer orthogonalen Basis im allgemeinen Fall unterscheidet Gitter von Vektorräumen und ist ursächlich für die Schwierigkeit von Gitterproblemen[55].

4.4.1 Gram-Schmidt-Orthogonalisierung

Für zwei Vektoren $a, b \in \mathbb{R}^n$ bezeichne $\langle a, b \rangle$ das Innere Produkt mit

$$\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i. \tag{4.24}$$

Damit lässt sich die *Gram-Schmidt-Orthogonalisierung* einer Basis definieren.

Definition 4.4.1 *Durch die Vektoren $b_1, b_2, \ldots, b_n$ sei eine Basis des $\mathbb{R}^n$ gegeben. Dann heißt die Basis $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$* Gram-Schmidt-Orthogonalisierung *zur Basis b_1, $b_2, \ldots, b_n$ und ist gegeben durch*

$$\hat{b}_1 := b_1 \tag{4.25a}$$

$$\hat{b}_i := b_i - \sum_{j=1}^{i-1} \mu_{ij} \hat{b}_j \quad (2 \leq i \leq n), \quad \mu_{ij} = \frac{\langle b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle} \quad (1 \leq j < i \leq n). \tag{4.25b}$$

Dabei heißen die Koeffizienten μ_{ij} Gram-Schmidt-Koeffizienten *der Basis $b_1, b_2, \ldots, b_n$.*

Mit $\mu_{ij} := 1$ für $i = j$ und $\mu_{ij} := 0$ für $i < j$ lassen sich die Gram-Schmidt-Koeffizienten als folgende Matrix schreiben:

$$M := (\mu_{ij}) \in \mathbb{Q}^{n \times n} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ \mu_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \mu_{n1} & \ldots & \mu_{nn-1} & 1 \end{pmatrix} \tag{4.26}$$

Es ist zu sehen, dass die Matrix M eine untere Dreiecksmatrix bildet mit $\mu_{ii} = 1$ für alle i. Dadurch gilt $\det M = 1$ für alle möglichen Einträge von μ_{ij} mit $1 \leq j < i \leq n$, also ist die Matrix M regulär. Die zu einer Basismatrix $B = (b_1, b_2, \ldots, b_n)$ ermittelte Basis $\hat{B} = \hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$ heißt *Orthogonalsystem*. Die Matrix M wird auch als *Transformationsmatrix* bezeichnet und erfüllt den Zusammenhang

$$B^T = M\hat{B} \tag{4.27}$$

Die Gram-Schmidt-Orthogonlasierung von Gitterbasen ist von Interesse für die Gitterreduktion. Das Orthogonalsystem einer gegebenen Gitterbasis bildet im Allgemeinen natürlich selbst keine Basis des Gitters. Mit Gleichung 4.27 lässt sich für die Determinante eines Gitters $\Lambda(B)$ der Zusammenhang

$$\begin{aligned} \det \Lambda &= \sqrt{\det B^T B} \\ &= \sqrt{\det M \hat{B} \hat{B}^T M^T} \\ &= \sqrt{\det \hat{B}^T \hat{B}} \end{aligned} \tag{4.28}$$

zeigen, mit $\det M = 1$ und

$$\det \hat{B}\hat{B}^T = \det \hat{B} \det \hat{B}^T = \det \hat{B}^T \hat{B}. \tag{4.29}$$

Da die Determinante eines Gitters auch dem Volumen des durch die Basisvektoren aufgespannten Parallelepipeds entspricht, lässt sich sich diese als

$$\det \Lambda = \mathrm{vol}\,(\mathcal{P}) = \prod_{i=1}^{n} \|\hat{b}_i\| \tag{4.30}$$

schreiben. Mit $\|\hat{b}_i\| \leq \|b_i\|$ für $1 \leq i \leq n$ lässt sich daraus die Hadamard-Ungleichung ableiten.

Satz 4.4.1 (Hadamard-Ungleichung) *Es sei $B = (b_1, \cdots, b_m)$ die Basis eines Gitters $\Lambda(B) \subseteq \mathbb{Z}^n$ mit $m \leq n$, dann gilt die Ungleichung*

$$\det \Lambda \leq \prod_{i=1}^{m} \|b_i\|. \tag{4.31}$$

Auf Grundlage der Hadamard-Ungleichung und Gleichung 4.30 lässt sich der *Orthogonalitätsdefekt* einer Gitterbasis B angeben.

Definition 4.4.2 *Für ein gegebenes Gitter $\Lambda(B) \subseteq \mathbb{Z}^n$ mit der Basis $B = (b_1, \cdots, b_m)$ wird der Quotient*

$$\delta = \frac{\prod_{i=1}^{m} \|b_i\|}{\det \Lambda} = \frac{\prod_{i=1}^{m} \|b_i\|}{\prod_{i=1}^{m} \|\hat{b}_i\|} \tag{4.32}$$

als Orthogonalitätsdefekt *von B bezeichnet.*

Wie im Folgenden zu sehen sein wird, sind neben dem Orthogonalitätsdefekt auch die Gram-Schmidt-Koeffizientenein Ausdruck für die Reduziertheit einer Gitterbasis.

4.4.2 Längenreduktion

In diesem Abschnitt wird ein einfacher Algorithmus zur Längenreduzierung einer Gitterbasis erläutert. Dieser ist ein wichtiger Bestandteil der LLL-Reduktion, welche im darauffolgenden Abschnitt behandelt wird.

Definition 4.4.3 *Eine geordnete Gitterbasis $(b_1, b_2, \ldots, b_m) \in \mathbb{R}^n$ heißt* längenreduziert*, wenn die Gram-Schmidt-Koeffizienten die Bedingung*

$$|\mu_{ij}| \leq \frac{1}{2} \tag{4.33}$$

für alle i, j mit $1 \leq j < i \leq m$ erfüllen.

Algorithmus 4.4.1 k-Reduce

Eingabe: eine Gitterbasis $b_1, b_2, \ldots, b_m \in \mathbb{Z}^n$,
die Matrix der Gram-Schmidt-Koeffizienten $(\mu_{ij}) \in \mathbb{Q}^{m \times m}$,
ein $k \in \mathbb{N}$ mit $1 < k \leq m$

Ausgabe: eine Gitterbasis $b_1, b_2, \ldots, b_m$ mit $|\mu_{ik}| \leq 1/2$ für $j = 1, \ldots, k-1$,
die aktualisierte Matrix (μ_{ij})

1: **for** $j = k-1$ to 1 **do**
2: **if** $|\mu_{kj}| > 1/2$ **then**
3: $b_k = b_k - \lceil \mu_{ij} \rfloor b_j$
4: **for** $i = 1$ to $m - 1$ **do**
5: $\mu_{ki} = \mu_{ki} - \lceil \mu_{kj} \rfloor \mu_{ji}$
6: **end for**
7: **end if**
8: **end for**
9: **return** $b_1, b_2, \ldots, b_m$, (μ_{ij})

Der hier gezeigte Algorithmus kann als diskrete Adaption der Gram-Schmidt-Orthogonalisierung angesehen werden. Die Schreibweise $\lceil \cdot \rfloor$ bezeichnet hierbei das kaufmännische Runden. Die Gram-Schmidt-Koeffizienten werden als gegeben betrachtet und sind in Form der Matrix M eine Eingabegröße des Algorithmus. Weiterhin sind die zu reduzierende Gitterbasis B und ein Parameter k Teil der Eingabe. Der Parameter k erlaubt eine teilweise Reduzierung der Gitterbasis. Angefangen bei $k = 2$ kann durch wiederholtes Aufrufen von k-Reduce und Inkrementierung von k bis hin zur Gitterdimension n ein spaltenweises Reduzieren der Gitterbasis erfolgen. Diese Eigenschaft wird im LLL-Algorithmus (Unterabschnitt 4.4.3) benötigt. Wie bei der Gram-Schmidt-Orthogonaliseriung wird der erste Basisvektor b als erster Vektor der reduzierten Basis übernommen. Ausgehend davon erfolgt für jeden weiteren Basisvektor die gleiche Linearkombination wie bei der Gram-Schmidt-Orthogonalisierung mit dem Unterschied, dass der jeweilige Gram-Schmidt-Koeffizient kaufmännisch gerundet wird. Damit wird sichergestellt, dass der so erzeugte Vektor wieder ein Gittervektor ist, da er durch ganzzahliger Linearkombination zweier Basisvektoren erzeugt wird. Im letzten Schritt wird die Matrix der Gram-Schmidt-Koeffizienten auf Grundlage der Vektoren der reduzierten Basisvektoren aktualisiert. Auf diese Weise kann das Reduktionskriterium überprüft werden. Ergebnis ist eine bis zur k-ten Spalte längenreduzierte Gitterbasis. Die Korrektheit des Algorithmus ergibt sich daraus, dass nach jedem Schritt gilt

$$\mu_{ki}^{neu} = \mu_{ki}^{alt} - \lceil \mu_{kj}^{alt} \rfloor \mu_{ji} \tag{4.34}$$

für $i = 1, \cdots, m-1$. Daraus folgt

$$\mu_{ki}^{neu} \leq \frac{1}{2}. \tag{4.35}$$

Dabei bleiben alle μ_{ki} mit $i \geq k$ unverändert. Am Ende des Verfahrens gilt somit $|\mu_{ij}| \leq 1/2$ für $1 \leq j < i \leq m$. Da die Reihenfolge der Basisvektoren $b_1, \ldots, b_m$ hierbei erhalten bleibt, wird der Vektor $b_1 = \hat{b}_1$ nicht verändert. Dadurch bleibt auch das Orthogonalsystem $\hat{b}_1, \ldots, \hat{b}_m$ unverändert.

In Betrachtung des obigen Beispielgitters (vgl. Abbildung 4.1) mit den Basen

$$B_1 = \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 4 & 5 \\ 3 & 5 \end{pmatrix}$$

ergibt sich für die Gram-Schmidt-Koeffizienten der Gitterbasen

$$M_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 \\ 7/10 & 1 \end{pmatrix}$$

mit $M_i = (\mu_{ij})$. Zunächst ist zu erkennen, dass keine der Basen längenreduziert nach Gleichung 4.33 ist. Weiterhin besteht die Besonderheit, dass die Gram-Schmidt-Koeffizienten der Basis B_1 alle ganzzahlig sind, was bedeutet, dass das Gitter eine orthogonale Basis $B_\perp$ hat, die dem Orthogonalsystem $\hat{B}_1$ entspricht.

$$B_\perp = \hat{B}_1 = \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix}$$

Anhand der Basis B_2 soll nun der Algorithmus zur Längenreduzierung veranschaulicht werden. Das Resultat ist in Abbildung 4.5 dargestellt. In zweidimensionalen Gittern wird jede Schleife genau einmal durchlaufen. Als erstes wird der neue Vektor

$$b_2^{neu} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} - \left\lceil \frac{7}{10} \right\rfloor \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

berechnet. Anschließend wird der relevante Gram-Schmidt-Koeffizient μ_{21} mit

$$\mu_{21} = \frac{7}{10} - \left\lceil \frac{7}{10} \right\rfloor \cdot 1 = -\frac{3}{10}$$

aktualisiert, wodurch das Reduktionskriterium nun erfüllt wird. Wie leicht zu überprüfen ist, bleibt das Orthogonalsystem

$$\hat{B}_1 = \begin{pmatrix} 4 & 11/5 \\ 3 & 29/10 \end{pmatrix}$$

unverändert, da keine Vertauschung von Spaltenvektoren vorgenommen. Analog der Gram-Schmidt-Orthogonalisierung ist das Ergebnis des Algorithmus abhängig von der gewählten Reihenfolge der Vektoren der Ausgangsbasis.

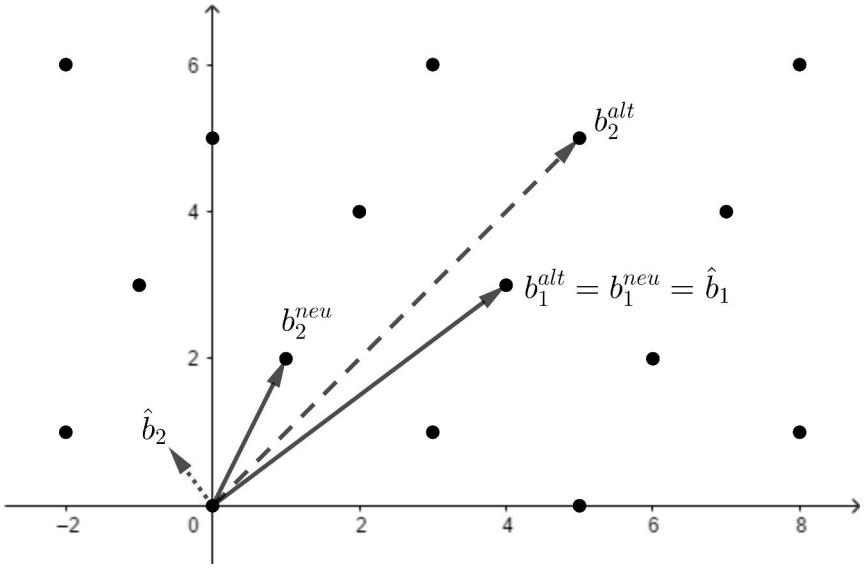


Abbildung 4.5: Längenreduzierung der Gitterbasis B_2.

4.4.3 LLL-Reduktion

Die auf die Gebrüder Lenstra und László Lovász zurückggehende LLL-Reduktion erweitert die Längenreduktion um ein weiteres Kriterium, das auch als LLL-Kriterium bezeichnet wird[45]. Ziel der LLL-Reduktion ist es, eine möglichst orthogonale Basis mit möglichst kurzen Vektoren zu ermitteln. Diese stellt eine Approximation der sukzessiven Minima dar mit einem Approximationsfaktor, der exponentiell in der Dimension des Gitters ist.

Definition 4.4.4 *Gegeben sei eine geordnete Gitterbasis* $B = (b_1, b_2, \ldots, b_m) \in \mathbb{R}^{n \times m}$ *und das Orthogonalsystem* $\hat{B} = (\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m)$ *zur Basis* B*. Für einen Parameter* α *mit* $1/4 \leq \alpha \leq 1$ *heißt die Basis* B α-reduziert *oder* LLL-reduziert mit α*, wenn gilt*

$$|\mu_{ij}| \leq \frac{1}{2} \qquad (1 \leq j < i \leq m) \tag{4.36a}$$

$$\alpha \|\hat{b}_{k-1}\|^2 \leq \|\hat{b}_k\|^2 + \mu_{kk-1}^2 \|\hat{b}_{k-1}\|^2 \qquad (k = 2, \ldots, m) \tag{4.36b}$$

mit μ_{ij} *als den Gram-Schmidt-Koeffizienten zur Basis* B*.*

Das LLL-Kriterium begrenzt dynamisch das Verhältnis der Längen zweier aufeinanderfolgender Vektoren des Orthogonalsystems unter Einbeziehung eines Parameters α mit $1/4 \leq \alpha \leq 1$. Je größer der Wert von α ist, desto stärker ist die Reduktion. Lenstra und Lovász zeigen die LLL-Reduktion für $\alpha = 3/4$. Weiterhin beeinflusst der

jeweilige Basisvektor b_k die Stärke des Kriteriums für das entsprechende Vektorpaar. Der enthaltene Gram-Schmidt-Koeffizient

$$\mu_{kk-1}^2 = \frac{\langle b_k, \hat{b}_{k-1}\rangle^2}{\langle \hat{b}_{k-1}, \hat{b}_{k-1}\rangle^2} \tag{4.37}$$

hat nach Gleichung 4.36a eine Größe von $0 \leq \mu_{kk-1}^2 \leq 1/4$. Um das LLL-Kriterium herzustellen, werden zusätzlich zur Längenreduktion Basisvektoren vertauscht, wodurch sich das Orthogonalsystem ändert. Ziel ist es, dadurch ein Orthogonalsystem zu finden, dessen Vektoren gemäß des LLL-Kriteriums möglichst kurz sind. Die Längenreduktion sorgt hierbei dafür, dass die berechnete Gitterbasis dabei möglichst nah am aktuellen Orthogonalsystem liegt. Die hier vorgestellte Variante des LLL-

Algorithmus 4.4.2 LLL

Eingabe: eine Gitterbasis $B = (b_1, b_2, \ldots, b_m) \in \mathbb{Z}^{n\times m}$,
ein Reduktionsparameter $\alpha \in \mathbb{R}$ mit $1/4 < \alpha \leq 1$
Ausgabe: die α-reduzierte Gitterbasis $B = (b_1, b_2, \ldots, b_m)$

```
1:  k = 2
2:  while k ≤ m do
3:     Berechne (μ_ij), B̂ und ‖b̂_1‖, ‖b̂_2‖, ..., ‖b̂_m‖
4:     k-Reduce(B, (μ_ij), k)
5:     if α‖b̂_{k-1}‖^2 > ‖b̂_k‖^2 + μ^2_{kk-1}‖b̂_{k-1}‖^2 then
6:        Vertausche b_k und b_{k-1}
7:        k = max(k−1, 2)
8:     else
9:        k = k + 1
10:    end if
11: end while
12: return B = (b_1, b_2, ..., b_m)
```

Algorithmus verwendet den k-Reduce Algorithmus. Eingabegrößen sind die zu reduzierende Gitterbasis B und der Reduktionsparameter α. Der Parameter k steht für die aktuell betrachtete Spalte, begonnen wird also mit der zweiten Spalte. Zunächst werden in jedem Durchlauf die Gram-Schmidt-Koeffizienten und die Gram-Schmidt-Orthogonaliserung $\hat{B}$ zur aktuellen Gitterbasis B berechnet, sowie die Vektornorm der Vektoren $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m$. Dies ist erforderlich, weil sich die Gitterbasis mit jedem Schleifendurchlauf potenziell ändert. Anschließend wird die k-te Spalte mithilfe des k-Reduce Algorithmus und unter Verwendung von $(\mu_{ij}) \in \mathbb{Q}^{m\times m}$ längenreduziert. Nun wird das LLL-Kriterium für die k-te und $(k-1)$-te Spalte von B anhand der zuvor berechneten Normen der Vektoren $\hat{b_k}$ und $\hat{b}_{k-1}$ überprüft. Ist es erfüllt, wird k inkrementiert und mit der nächsten Spalte fortgefahren. Anderenfalls werden die

Vektoren der k-ten und (k–1)-ten Spalte vertauscht. In diesem Fall wird der Wert von k um eins verringert, außer k ist 2, dann wird die Schleife erneut mit $k = 2$ durchlaufen, da es sich hierbei um die Anfangsposition handelt. Wenn der Algorithmus sich also nicht am Anfang der Basismatrix ($k = 2$) befindet, wird nach Vertauschen zweier Basisvektoren ein Schritt zurückgegangen, um so durch weitere Vertauschung einen Vektor gegebenenfalls noch weiter nach vorne verschieben zu können. Auf diese Weise erfolgt ein schwaches Sortieren der längenreduzierten Basisvektoren anhand des LLL-Kriteriums. Ergebnis ist eine längenreduzierte Gitterbasis, die unter Berücksichtung der Reihenfolge der Basisvektoren hergestellt wurde und dadruch eine stärkere Reduktion darstellt als die reine Längenreduzierung. Hierbei ist wichtig, dass die Längenreduktion mittels k-Reduce spaltenweise erfolgt. Eine vollständige Längenreduktion in jedem Durchlauf würde durch das anschließende Vertauschen von Gittervektoren in jedem Schritt obsolete Zwischenergebnisse erzeugen. Lenstra und Lovász beweisen die Korrektheit des Algorithmus, indem sie zeigen, dass er terminiert[45]. Daraus folgt auch, dass die durch den Algorithmus berechnete Gitterbasis in jedem Fall LLL-reduziert ist. Für $\alpha < 1$ arbeitet der Algorithmus mit polynomieller Laufzeit.

5 Gitterbasierte Verfahren

Unter den gitterbasierten Kryptosystemen lassen sich verschiedene Strömungen identifizieren. Die ersten Verfahren wurden auf Grundlage allgemeiner Gitter formuliert. Nachfolgende Ansätze versuchten, die gut belegte Schwierigkeit bekannter Problemstellungen in Gittern zu nutzen, aber dabei in einer effizienteren mathematischen Struktur zu arbeiten. Eine wichtige Gruppe bilden die Verfahren der NRTU Familie, von denen einige bereits durch das NIST[1] standardisiert wurden[31]. Andere Arbeiten zu gitterbasierten Verfahren stützen sich auf das (Ring-)SIS Problem oder das in Unterabschnitt 2.4.3 vorgestellte (Ring-)LWE Problem. Letzteres hat große Bedeutung für die gitterbasierte Kryptographie gewonnen, weil gezeigt werden konnte, dass es im durchschnittlichen Fall so schwer zu berechnen ist, wie verschiedene Gitterprobleme im Worst Case[50][65]. Als gitterbasierte Verfahren werden heute also weitestgehend Kryptosysteme verstanden, deren sicherheitsbestimmendes Problem auf ein möglichst schwer zu berechnendes Gitterproblem reduziert werden kann. In diesem Kapitel sollen gitterbasierte Ansätze zu Public-Key-Verfahren, Schlüsselaustauschverfahren und digitalen Signaturverfahren behandelt werden. Dabei soll je ein gitterbasiertes Kryptosystem aus jeder Kategorie näher beschrieben und in den Kontext der aktuellen Entwicklungen eingeordnet werden. Hierbei soll ein gewisses Spektrum verschiedener Ansätze abgedeckt werden, um die grundlegenden Vor- und Nachteile vergleichend zu skizzieren.

5.1 Public-Key-Verfahren

5.1.1 NTRUEncrypt

Als Bestandteil des NTRU Kryptosystems geht NTRUEncrypt auf die Arbeit von Hoffstein, Pipher und Silverman zurück[38]. Weiterentwicklungen erfolgten unter anderem durch Stehle und Steinfeld, die eine Version des NTRU Verfahrens mit Sicherheitsbeweis vorstellten[68]. Im Jahre 2016 veröffentlichten Bernstein et al. mit NTRU Prime die aktuell sicherste Version, indem eine potenzielle Sicherheitslücke der vorherigen Verfahren geschlossen wurde[12]. Die grundlegende Funktionsweise von NTRUEncrypt wird anhand des ursprünglichen Verfahrens von Hoffstein, Pipher und Silverman erläutert. Einige Unterschiede zu aktuellen Weiterentwicklungen werden in Unterabschnitt 5.1.2 diskutiert.

[1] https://www.nist.gov/

Grundlagen

Ausgangspunkt für das NTRU Verfahren bilden die Parameter $N, p, q \in \mathbb{N}$, wobei N den primären Sicherheitsparameter bildet. Die Parameter p und q müssen keine Primzahlen sein, aber es soll gelten $\gcd(p, q) = 1$ und $q \gg p$.

Operiert wird im Ring $R = \mathbb{Z}[X]/(X^N - 1)$. Ein Element $F \in R$ kann als Polynom oder Vektor

$$F = \sum_{i=0}^{N-1} F_i x^i = (F_0, F_1, \ldots, F_{N-1})^T \tag{5.1}$$

geschrieben werden. Für zwei Polynome $F, G \in R$ wird mit $H = F \odot G$ die Multiplikation in R durchgeführt. Dabei heißt H *zyklisches Konvolutionsprodukt* und kann anhand der folgenden Gleichung dargestellt werden.

$$H_k = \sum_{i+j \equiv k \bmod N} F_i G_j \quad (0 \leq k < N). \tag{5.2}$$

Mit $\mathcal{L}_m, \mathcal{L}_g, \mathcal{L}_\phi, \mathcal{L}_f$ werden vier unterschiedliche Mengen mit Elementen aus R definiert. Es gilt also $\operatorname{grad} \mathcal{L}_i \leq N - 1$.

Die erste Menge $\mathcal{L}_m$ bildet den Nachrichtenraum und ist die Menge aller Polynome m, deren Koeffizienten m_i ausschließlich im Intervall $[-p/2, p/2]$ liegen. Es gilt also

$$\mathcal{L}_m = \{m \in R : m_i \in \mathbb{Z}, -p/2 \leq m_i, \leq p/2\}. \tag{5.3}$$

Alle weiteren Mengen werden mithilfe des folgenden Schemas gebildet.

$$\begin{aligned} \mathcal{L}(d_1, d_2) = \{F \in R : \quad & F \text{ hat } d_1 \text{ Koeffizienten gleich } 1, \\ & d_2 \text{ Koeffizienten gleich } -1, \\ & \text{alle restlichen Koeffizienten gleich } 0\} \end{aligned} \quad . \tag{5.4}$$

Es soll gelten: $\mathcal{L}_g = \mathcal{L}(d_g, d_g)$, $\mathcal{L}_\phi = \mathcal{L}(d_\phi, d_\phi)$ und $\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1)$ mit $d_i \in \mathbb{N}$. Die Polynome der Menge $\mathcal{L}_f$ sollen in R invertierbar sein. Aus diesem Grund wird als zweiter Parameter $d_f - 1$ gewählt. Anderenfalls würden Polynome f entstehen, für die gilt $f(1) = 0$ und diese sind niemals invertierbar[38]. Die genaue Größe der Parameter d_g, d_ϕ und d_f ist sicherheitsrelevant und wird in [38] diskutiert.

Nachfolgend sollen Schlüsselerzeugung, Verschlüsselung und Entschlüsselung anhand des Paradigmas *Alice sendet eine Nachricht an Bob* dargestellt werden.

Schlüsselerzeugung

Zur Erstellung eines öffentlichen Schlüssels wählt Bob zwei Polynome $f \in \mathcal{L}_f$ und $g \in \mathcal{L}_g$. Das Polynom f muss dabei bezüglich $\odot$ Inverse modulo p und modulo q haben. Diese werden mit F_p bzw. F_q bezeichnet. Es gilt

$$F_p \odot f \equiv 1 \bmod p \tag{5.5a}$$

$$F_q \odot f \equiv 1 \bmod q. \tag{5.5b}$$

Bob berechnet nun den öffentlichen Schlüssel h.

$$h = F_q \odot g \bmod q \tag{5.6}$$

Der geheime Schlüssel von Bob ist f. Zusätzlich sollte F_p gespeichert werden.

Verschlüsselung

Alice wird nun die Verschlüsselung einer Nachricht m aus dem Nachrichtenraum $\mathcal{L}_m$ vornehmen, um die verschlüsselte Nachricht e an Bob zu schicken. Sie nutzt Bobs öffentlichen Schlüssel h und wählt ein Polynom $\phi \in \mathcal{L}_\phi$ zur Berechnung der Chiffre.

$$e = p\phi \odot h + m \bmod q \tag{5.7}$$

Diese wird an Bob gesendet.

Entschlüsselung

Bob möchte die verschlüsselte Nachricht e von Alice entschlüsseln. Er berechnet zunächst das Polynom a mithilfe seines geheimen Schlüssels f.

$$a \equiv f \odot e \bmod q \tag{5.8}$$

mit Koeffizienten $a_i \in [-q/2, q/2]$. Die Berechnung von m erfolgt dann mithilfe des gespeicherten Polynoms F_p.

$$m = F_p \odot a \bmod p. \tag{5.9}$$

Das Polynom a genügt dabei der folgenden Gleichung.

$$\begin{aligned} a \equiv f \odot e &\equiv f \odot p\phi + f \odot m && (\mathrm{mod} q) \\ &\equiv f \odot p\phi \odot F_q \odot g + f \odot m && (\mathrm{mod} q) \\ &\equiv p\phi \odot g + f \odot m && (\mathrm{mod} q) \end{aligned} \tag{5.10}$$

Dies ermöglicht die Berechnung von

$$a \bmod p = f \odot m. \tag{5.11}$$

Mit Gleichung 5.5a lässt sich also durch die Kommutativität von $\odot$ in R die Nachricht m durch Multiplikation mit F_p wiederherstellen.

$$f \odot m \odot F_p = m \tag{5.12}$$

5.1.2 Sicherheit und Effizienz

Auch wenn für das originale NTRU Verfahren kein formaler Sicherheitsbeweis existiert, wird es für hinreichend große Parameter für sicher gehalten. Aktuelle Empfehlungen für Parameter- und Schlüsselgrößen sind in Tabelle 5.1 zu finden. Die

Sicherheit [Bit]	p	q	n	Öffentl. Schlüssel [Bits]
128	3	2048	439	4839
192	3	2048	593	6523
256	3	2048	743	8173

Tabelle 5.1: Empfohlene Paramtergrößen für NTRU[40].

in Bits angegebene Sicherheit entspricht der jeweiligen Referenzschlüsselgröße eines symmetrischen Verfahrens bei gleichem Sicherheitslevel. Diese Schreibweise dient im Wesentlichen der Quantifizierung und somit der Vergleichbarkeit unterschiedlicher Verfahren hinsichtlich ihrer Sicherheit.

Eine grundlegende Angriffsmöglichkeit auf NTRUEncrypt besteht im Durchprobieren aller Polynome f, die für die Parameter N und d_f in Frage kommen. Eine effizientere Möglichkeit liegt in einer sogenannten *Meet-In-The-Middle* Attacke. Das Verfahren wird in einer optimierten Variante mit reduziertem Speicherbedarf bei Vredendaal beschrieben[70]. Dabei wird anstatt eines Polynomes der vollen Länge N, simultan nach zwei Polynome der Länge $N/2$ gesucht. Der Suchraum wird also in zwei kleinere Teile aufgeteilt, welche separat durchsucht werden, um den geheimen Schlüssel zu erlangen. Hierfür wird nur noch die Quadratwurzel an Schritten benötigt verglichen mit der Suche nach einem Polynom der vollen Länge. Das gesuchte Polynom f wird also aufgeteilt nach $f = f_1 + f_2$, wobei f_1 einen maximalen Grad von $(n-1)/2 - 1$ aufweist und f_2 nur Potenzen zwischen $(n-1)/2$ und $(n-1)$ enthält. Es gilt damit also

$$h = (f_1 + f_2)^{-1} g \tag{5.13}$$

Dies lässt sich unter den genannten Bedingungen für f_1 und f_2 in

$$f_1 h = g - f_2 h \tag{5.14}$$

umformen, wodurch eine simultane Suche nach Polynomen f_1 und f_2 möglich wird, die Gleichung 5.14 erfüllen, um so das Polynom $f = f_1 + f_2$ zu finden.

Eine weitere, noch effizientere Angriffsmöglichkeit besteht in der Anwendung der Gitterreduktion[25], bspw. mithilfe des in Unterabschnitt 4.4.3 gezeigten LLL-Algorithmus. Aus der Definition $h = gf^{-1}$ folgt, dass es einen ganzzahligen Vektor der

Länge $2N$ gibt, sodass

$$(k, f)\begin{pmatrix} qI & 0 \\ H & I \end{pmatrix} = (g, f) \tag{5.15}$$

gilt mit der *zyklischen* Matrix H. Diese wird aus h nach

$$H_{ij} = h_{i+j \bmod N} \tag{5.16}$$

generiert. Der Teilvektor k realisiert dabei die Reduktion aller Koeffizienten modulo q, während das Teilprodukt fH der Multiplikation fh im Ring $R = \mathbb{Z}[X]/(x^N - 1)$ entspricht. Die $2N \times 2N$-Matrix

$$\begin{pmatrix} qI & 0 \\ H & I \end{pmatrix} \tag{5.17}$$

bildet die Basis eines Gitters und wird als öffentliche NTRU Gitterbasis bezeichnet. Coppersmitth belegt auf Grundlage von Heuristiken, dass es in diesem Gitter keine kürzeren Vektoren gibt als (g, f). Es kann nun also versucht werden, mithilfe von Gitterreduktionsalgorithmen (g, f) als einen kürzesten Vektor zu approximieren (vgl. Unterabschnitt 4.3.1).

Howgrave-Graham stellte 2007 eine hybride Attacke vor, die beide Angriffsmöglichkeiten kombiniert, wodurch die exponentielle Komplexität im Vergleich zu den obigen Angriffen veringert wird[41].

Die hier vorgestellte Variante von NTRUEncrypt performt im Bereich der kostenintensiven Entschlüsselungsoperationen bei vergleichbarer Sicherheit deutlich schneller als RSA[20]. So erhöht sich beim RSA Verfahren der Aufwand für diese Operationen mit $O(n^3)$ in etwa kubisch zur Schlüsselgröße, während er bei NTRUEncrypt mit $O(n \log n)$ angegeben wird. Die NTRU Variante von Stehle und Steinfeld arbeitet signafikant weniger effizient als das Originalschema, verfügt jedoch über einen Sicherheitsbeweis[68]. Den Flaschenhals der NTRU Verfahren bildet die polynomielle Multiplikation. Für die aktuellste Variante, NTRU Prime, beschreiben Bernstein et al. einen Trade-Off der die Sicherheit des Verfahrens bei akzeptablem Effizienzverlust erhöhen soll[12].

5.1.3 Weitere Verfahren

Das im Jahre 1996 von Ajtai vorgestellte Kryptosystem gilt als erstes gitterbasiertes Public-Key-Verfahren[3]. Die Sicherheit des Verfahrens beruht auf der Worst Case Schwierigkeit einer speziellen Variante von SVP. Die kryptoanalytische Erforschung des Verfahrens zeigte jedoch eine heuristische Angrifssmöglichkeit auf, die unpraktikabel große Schlüssellängen verlangen würde[60]. Ein Jahr später folgte mit dem GGH Kryptosystem ein weiteres bedeutendes gitterbasiertes Verfahren, dass auf der

Schwierigkeit von CVP für möglichst ungünstige Gitterbasen beruhte[35]. Im Jahre 2005 stelte Oded Regev das LWE Problem vor, das durch die aufgezeigte Average Case/Worst Case-Beziehung zur Schwierigkeit bekannter Gitterprobleme die Grundlage für zahlreiche Überlegungen zu gitterbasierten Verfahren bildete[65][63]. Im Zentrum der jüngsten Entwicklungen auf dem Gebiet der gitterbasierten Public Key Kryptosysteme steht das Ring-LWE Problem. Entsprechende Ansätze versprechen eine bessere Effizienz als LWE-basierte Ansätze und verfügen zudem über starke beweisbare Sicherheitsgarantien[62]. Eine große Menge der bisher gewonnenen Erfahrungen und Erkenntnisse auf diesem Gebiet wurden durch die Arbeiten von Lyubashevsky, Peikert und Regev in konkrete Richtlinien und Tools für die Entwicklung realer Kryptosysteme umgesetzt[49][62]. Dabei zeigen sie Wege auf, gitterbasierte Methoden als Drop-in Ersatz für bestehende Verfahren wie RSA oder Diffie-Hellman zu entwickeln.

5.2 Schlüsselaustauschverfahren

5.2.1 Ding Schlüsselaustausch

Das hier vorgestellte Verfahren wurde von Ding et al. vorgestellt und hat den Vorsatz, ein in Einfachheit und Eleganz mit dem Diffie-Hellman-Verfahren vergleichbares Schema zum Schlüsselaustausch zwischen zwei Parteien zu liefern[27]. Ding et al. stellen dabei ein Verfahren vor, das auf dem allgemeinen Learning With Errors Problem (LWE) basiert und darauf aufbauend ein weiteres, das mit der Ringvariante von Learning With Errors (Ring-LWE) arbeitet. Im Folgenden soll die Ringvariante des Schlüsselaustauschverfahrens beschrieben werden.

Grundlagen

Grundlage des Verfahrens bildet der Ring $R = \mathbb{Z}_q[X]/(x^n + 1)$ mit n als einer Potenz von 2 und einer Primzahl q mit $q \equiv 1 \bmod 2n$. Weiterhin sei die l_∞-Norm eines Polynoms $F \in R$ gegeben durch

$$\|F\|_\infty = \max(F_i) < q \tag{5.18}$$

Ein zentraler Bestandteil des Verfahrens ist eine Verteilung χ über R, die durch β beschränkt ist. Das heißt die Wahrscheinlichkeit, dass für ein $x \leftarrow \chi$ gilt

$$\|x\|_\infty > \beta \tag{5.19}$$

soll vernachlässigbar klein sein. Durch den Parameter β wird also angegeben, wie klein die Koeffizienten eines Polynoms x sein sollen, das auf Grundlage der Verteilung χ aus R gewählt wird.

Analog der diskreten Exponentiation beim Diffie-Hellman Verfahren, wird die Information bei diesem Verfahren durch das gestörte innere Produkt gekapselt, wie es im Problem Ring-LWE beschrieben wird (vgl. Unterabschnitt 2.4.3). Der Unterschied liegt darin, dass es in LWE-basierten Verfahren notwendig wird, eine Trennung von Störung und Information vorzunehmen bzw. im Falle des hier vorgestellten Verfahrens, die Extraktion einer gemeinsamen Information aus zwei unterschiedlichen, fehlerbehafteten Polynomen vorzunehmen. Ding et al. definieren dafür eine sogenannte *robuste Extraktorfunktion* für ein Polynom $a \in R$ und ein Signalpolynom σ mit Koeffizienten $\sigma_i \in \{0, 1\}$.

$$E(a, \sigma) = \left(a + \sigma \frac{q-1}{2} \bmod q \right) \bmod 2 \tag{5.20}$$

Das Signalpolynom bildet eine Art Zusatzhinweis, um die gewünschte Information aus a zu extrahieren. Die genaue Funktionsweise ist in Abschnitt 5.2.1 zu sehen. Die Berechnung von σ erfolgt mithilfe einer Signalfunktion $\sigma(a)$ mit $a \in R$. Es sollen dabei zwei unterschiedliche Signalfunktionen $\sigma_0(a)$ und $\sigma_1(a)$ definiert werden.

$$\sigma_0(a) = \sum_{i=0}^{n-1} \sigma_0^*(a_i) x^i \tag{5.21a}$$

$$\sigma_1(a) = \sum_{i=0}^{n-1} \sigma_1^*(a_i) x^i \tag{5.21b}$$

mit

$$\sigma_0^*(a_i) = \begin{cases} 0 & \text{für } a_i \in [-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor] \\ 1 & \text{sonst} \end{cases} \tag{5.21c}$$

$$\sigma_1^*(a_i) = \begin{cases} 0 & \text{für } a_i \in [-\lfloor q/4 \rfloor + 1, \lfloor q/4 \rfloor + 1] \\ 1 & \text{sonst} \end{cases} \tag{5.21d}$$

Weiterhin wird eine sogenannte Hinweisfunktion $S(a)$ definiert, die zufällig eine der beiden Signalfunktionen auswählt und auf ein Polynom $a \in R$ anwendet.

$$S(a) = \sigma_b(a), \quad b \in \{0, 1\} \tag{5.22}$$

Dieses Vorgehen hat einen bestimmten Grund. Die Notwendigkeit eines Signalpolynoms bildet eine gewisse Schwachstelle, da damit auch einem Angreifer zusätzliches Wissen zur Verfügung gestellt wird. Ding et al. zeigen, dass beide Signalfunktionen äquivalent bei der Extraktion der gewünschten Information funktionieren, aber dadurch, dass ein Angreifer nicht weiß, welche Signalfunktion verwendet wurde, wird die Aussagekraft des Signalpolynoms für den Angreifer minimiert.

Funktionsweise

Alice und Bob wollen sich über einen unsicheren Kanal auf einen gemeinsamen Schlüssel einigen. Zunächst werden die öffentlichen Parameter q, n und β, sowie die Verteilung χ festgelegt. Anschließend wird ein zufälliges Element $m \in R_q$ generiert.

Alice wählt nun ein geheimes Polynom $s_A \leftarrow \chi$ und ein Fehlerpolynom $e_A \leftarrow \chi$ und berechnet das Polynom

$$p_A = ms_A + 2e_A \bmod q \tag{5.23}$$

unter Verwendung von m und sendet dieses an Bob.

Bob wählt seinerseits ein geheimes Polynom $s_B \leftarrow \chi$ und ein Fehlerpolynom $e_B \leftarrow \chi$ und berechnet das Polynom p_B.

$$p_B = ms_B + 2e_B \bmod q \tag{5.24}$$

Zusätzlich wählt Bob ein weiteres Fehlerpolynom $e'_B \leftarrow \chi$ und berechnet mittels des von Alice übermittelten Polynoms p_A ein Schlüsselpolynom K_B. Dieses dient später zur Berechnung des geheimen Schlüssels.

$$K_B = p_A s_B + 2e'_B \bmod q \tag{5.25}$$

Auf K_B wird nun die Hinweisfunktion angewendet, um ein Signalpolynom $\sigma \leftarrow S(K_B)$ zu erhalten. Bob sendet nun das Paar (p_B, σ) an Alice und berechnet den geheimen Schlüssel SK_B unter Anwendung der Extraktorfunktion auf das Schlüsselpolynom K_B und das Signalpolynom σ.

$$SK_B = E(K_B, \sigma) \tag{5.26}$$

Alice wählt nun ebenfalls ein zweites Fehlerpolynom $e'_A \leftarrow \chi$ und berechnet ihr Schlüsselpolynom K_A mit dem von Bob erhaltenem Polynom p_B.

$$K_A = s_A p_B + 2e'_A \bmod q \tag{5.27}$$

Damit kann Alice nun ihrerseits den geheimen Schlüssel SK_A berechnen, indem sie das zuvor empfangene Signalpolynom σ benutzt.

$$SK_A = E(K_A, \sigma) \tag{5.28}$$

Ding et al. zeigen, dass die jeweils ermittelten geheimen Schlüssel SK_A und SK_B mit überragender Wahrscheinlichkeit übereinstimmen, wenn die Bedingung

$$8n\beta^2 \leq \frac{q}{4} - 2 \tag{5.29}$$

erfüllt ist. Für eine detaillierte Betrachtung der zugrunde liegenden Wahrscheinlichkeiten wird auf die Originalpublikation verwiesen[27].

5.2.2 Sicherheit und Effizienz

Die Sicherheit von Ring-LWE-basierten Verfahren wie dem vorgestellten Ding Schlüsselaustausch beruht auf einem Satz, der von Lyubashevski, Peikert und Regev aufgestellt und bewiesen wurde[50]. Zur Erklärung der Norm $\|\cdot\|_\infty$ vgl. Gleichung 5.18.

Satz 5.2.1 *Gegeben sei ein Ring $R = \mathbb{Z}_q[X]/(x^n + 1)$ mit n als einer Potenz von 2 und einer Primzahl $q = q(n) \equiv 1 \bmod 2n$, sowie ein $\beta = \omega(\sqrt{n \log n})$ und eine Verteilung χ, die Elemente $F \in R$ ausgibt. Es soll dabei mit überragender Wahrscheinlichkeit gelten $\|F\|_\infty \leq \beta$.*
Existiert für genannte Bedingungen ein effizienter Quantenalgorithmus, der das Problem Ring-LWE$^{(m)}_{n,q,\chi}$ löst, dann gibt es auch einen effizienten Quantenalgorithmus der das worst-case $n^{2,5}(q/\beta)(nm/\log(nm))^{1/4}$-Approximate Shortest Vector *Problem löst für ideale Gitter über R, wobei m die Anzahl der Proben angibt, die zur Lösung des Problems zur Verfügung stehen.*

Beweis 5.2.1 *siehe [50].*

Dieser Satz stellt den Zusammenhang zum Shortest Vector Problem in idealen Gittern her, wodurch sich die Sicherheit Ring-LWE-basierter Verfahren mit der Worst Case Schwierigkeit von App-SVP mit spezifiziertem Approximationsfaktor angeben lässt (vgl. Unterabschnitt 4.3.1). Applebaum zeigt, dass die Schwierigkeit des Problems auch dann bestehen bleibt, wenn s aus der Fehlerverteilung χ gewählt wird[8]. Um Schwachstellen in Ring-LWE-basierten Verfahren zu finden, wird nach potenziell schwachen Instanzen des Problems gesucht, die effizient angegriffen werden können[23][22][29]. Die derzeit bekannten problematischen Instanzierungen wurden jedoch noch nicht für ein kryptographisches Verfahren vorgeschlagen. Wichtig hierbei ist insbesondere die gewählte Fehlerverteilung. Peikert diskutiert die Frage, wie nahe die unsicheren Instanzen an jenen liegen, die Worst Case Schwierigkeit haben und ob aus deren Existenz abgleitet werden kann, welche Ringe mehr oder weniger sicher in Bezug auf Ring-LWE sind[61]. Er zeigt hierbei eine Klasse von Instanzierungen auf, die nach bisherigem Kenntnisstand als unverletzlich gelten.

In Hinblick auf die Effizienz ermöglicht das Ring-LWE Problem eine Verbesserung gegenüber LWE-basierten Verfahren[50]. So beschreiben Ding et al. in der Vorbetrachtung für das hier gezeigte Schlüsselautauschverfahren auch eine LWE-basierte Variante des Verfahrens mit analoger Funktionsweise[27]. Dabei wird anstelle des öffentlichen Polynoms $m \in R$ eine Matrix $M \in \mathbb{Z}^{n \times n}$ verwendet und die Geheimnisse von Alice und Bob werden über

$$p_A = Ms_A + 2e_A \bmod q \tag{5.30a}$$

$$p_B = M^T s_B + 2e_B \bmod q \tag{5.30b}$$

gekapselt mit $p_i, s_i, e_i \in \mathbb{Z}_q^n$. Das gemeinsame Geheimnis wird dann mithilfe einer Extraktorfunktion und eines zusätzlichen Hinweises $\sigma \in \{0, 1\}$ aus

$$s_A^T p_B \approx s_B^T p_A \approx s_A^T M^T s_B \tag{5.31}$$

gewonnen. Für das LWE-basierte Verfahren werden also n linerare Gleichungen mit Fehlern benötigt, was für die Schlüsselgröße eine Anzahl von $O(n^2)$ Koeffizienten bedeutet. Für Ring-LWE ergibt sich lediglich eine Anzahl von $O(n)$ Koeffizienten, da nur ein Polynom m benötigt wird. Weiterhin ist eine effizientere Multiplikation unter Verwendung der Fast-Fourier-Transformation (FFT) möglich[21]. Ein Vergleich der Schlüsselgrößen und der Komplexitäten beider Varianten findet sich in Tabelle 5.2. Darin werden für beide Varianten die Größe des öffentlichen Parameters, die Kommunikationskomplexität und die benötigte Anzahl der Multiplikationen angegeben. Es wird deutlich, dass die Ring-LWE Variante sowohl in der Größe des öffentlichen Parameters als auch in Hinblick auf die benötigten Multiplikationen deutlich effizienter ist. Die leicht erhöhte Kommunikationskomplexität erklärt sich durch die Signalfunktion, die in der LWE Variante nur eine Zahl anstelle eines Vektors ausgibt.

Variante	Öffentl. Parameter	Kommunikation	Multiplikationen
LWE	$n^2 \log q$	$2n \log q + 1$	$2n^2$
Ring-LWE	$n \log q$	$2n \log q + n$	4

Tabelle 5.2: Effizienzgewinn des Ding-Schlüsselaustauschs durch Ring-LWE[27].

Ein rechnerisch sehr aufwendiger Aspekt LWE-basierter Verfahren bildet die Nachbildung einer diskreten Gaußschen Fehlerverteilung, weshalb hier ein Schwerpunkt für die Suche nach möglichen Optimierungen liegt[19].

5.2.3 Weitere Verfahren

Im Jahre 2014 wurde mit NTRU-KE ein gitterbasiertes Schlüsselaustauschverfahren beschrieben, dass auf dem NTRU Kryptosystem beruht[44]. Im selben Jahr wurde ein auf SIS basierendes Verfahren zum Schlüsselaustausch vorgestellt[71]. Es zeigte sich jedoch, dass beide Verfahren anfällig für sogenannte *Man-In-The-Middle* Angriffe sind[37][69]. Bos et al. stellten 2016 mit Frodo ein LWE basiertes Schlüsselaustauschverfahren vor[15]. Ähnlich wie bei den Public-Key-Kryptosystemen, nehmen auch viele der aktuellen Vorschläge Bezug auf die Arbeit von Peikert und versuchen sich damit an potenziell praxistauglichen Schlüsselaustauschverfahren basierend auf dem Ring-LWE Problem[62]. So schufen Bos et al. mit BCNS ein Protokoll, das im Wesentlichen das Verfahren von Ding mit den von Peikert vorgeschlagenen Verbesserungen implementiert. Dieses konnte als zusätzliche Cipher Suite in das Transport

Layer Security Protokoll (TLS) von OpenSSL integriert werden[14]. Eine Weiterentwicklung des BCNS Protokolls mit verbesserter Performance und Sicherheit bildet das 2016 von Alkim et al. vorgestellte Schlüsselaustauschverfahren New Hope[7]. Ein weiterer Ansatz für ein gitterbasiertes Verfahren names spKEX beruht auf dem *Learning with Roundings* Problem[13]. Dabei handelt es sich um eine Variante von LWE, bei der anstelle eines additiven Fehlerterms, ein Fehler durch Anwendung einer speziellen Rundungsfunktion erzeugt wird[11]. Dadurch bietet sich die Möglichkeit einer verbesserten Effizienz, da auf die aufwendige Erzeugung von Fehlern auf Grundlage einer Gaußschen Fehlerverteilung verzichtet werden kann.

5.3 Digitale Signaturverfahren

5.3.1 GGH

Das GGH Signaturverfahren geht auf die Arbeit von Goldreich, Goldwasser und Halevi zurück[35], die den Grundstein für die Entwicklung des NTRU Kryptosystems bildete. In der originalen Publikation wurde das Signaturerfahren nicht im Detail beschrieben, sondern nur das zugehörige Verschlüsselungsschema. Aus diesem Grund soll hier auf die Darstellung von Nguyen und Regev zurückgegriffen wird[59].

Schlüsselerzeugung

Als geheimer Schlüssel wird eine Matrix $R \in \mathbb{Z}^{n \times n}$ gewählt mit sehr kurzen Spaltenvektoren. Das heißt, die Einträge von R sollen von polynomieller Größe zum Sicherheitsparameter n sein. Die Matrix R bildet die Basis des Gitters $\Lambda = \Lambda(R)$. Weiterhin wird ein Schwellwert-Parameter $\tau \in \mathbb{R}$ festgelegt.

Den öffentlichen Schlüssel bildet das Paar (B, τ) mit einer Matrix $B \in \mathbb{Z}^{n \times n}$. Diese ergibt sich aus der *Hermite* Normalform (HNF) der Matrix R.

$$B = \mathrm{HNF}\,(R) \tag{5.32}$$

Die Hermite Normalform B der Gitterbasis R ist eine obere Dreieicksmatrix, die den folgenden Bedingungen genügt:

1. Jedes Element b_{ii} auf der Hauptdiagonale ist positiv.
2. Für jedes andere Element b_{ij} mit $i < j$ gilt $0 \leq b_{ij} \leq b_{ii}$.
3. Es existiert eine ganzzahlige unimodulare Matrix U, sodass

$$B = \mathrm{HNF}\,(R) = RU \tag{5.33}$$

gilt. Daraus folgt $\Lambda(B) = \Lambda(R)$, die Matrix B ist also auch eine Gitterbasis.

Die Verwendung der Hermite Normalform stellt eine Verbesserung des ursprünglichen Verfahrens dar, in der die Basis B durch Multiplikation der Matrix R mit einer hinreichenden Anzahl kleiner unimodularer Matrizen gewonnen wird. Der Vorschlag geht auf eine Arbeit von Micciancio zurück, in der dargelegt wird, dass eine Gitterbasis in Hermite Normalform dem Angreifer den geringsten Vorteil gibt[54]. Jedes Gitter verfügt über eine einzigartige Basis in Hermite Normalform, die sich effizient berechnen lässt[57].

Signierung

Die zu signierende Nachricht wird in den Nachrichtenraum $\mathbb{Z}^n$ gehasht. Das Resultat $m \in \mathbb{Z}$ ist ein Vektor, der nun die Grundlage für eine Instanz des Closest Vector Problems in Λ bildet. Als Signatur soll der Vektor $s \in \Lambda$ dienen, der eine Approximation Gittervektors mit dem geringsten Abstand zu m darstellt. Die Ermittlung erfolgt mithilfe eines CVP-Approximationsalgorithmus und des geheimen Schlüssels R. Hier soll der Algorithmus von Babai verwendet werden[10].

$$s = R\lceil R^{-1}m \rfloor \tag{5.34}$$

Die Schreibweise $\lceil \cdot \rfloor$ bezeichnet dabei das kaufmännische Runden. Die Signatur ist gültig, wenn gilt $\|s - m\| \leq \tau$. Je nach Größe des Schwellwertparameters τ sind Fehlschläge bei der Signierung möglich, sodass eine Prüfung erforderlich sein kann.

Verifizierung

Zur Verifizierung einer Signatur mithilfe des öffentlichen Schlüssels B und des Schwellwertparameters τ ist lediglich zu prüfen, ob der Signaturvektor s im Gitter $\Lambda(B)$ liegt und der Abstand zum Vektor m kleiner oder gleich τ ist. Eine Signatur s ist also gültig, wenn

$$s \in \Lambda(B) \wedge \|s - m\| \leq \tau \tag{5.35}$$

erfüllt ist.

5.3.2 Sicherheit und Effizienz

Die Sicherheit des GGH Verfahrens basiert direkt auf dem Closest Vector Problem (vgl. Unterabschnitt 4.3.2). Jedoch hat das Verfahren die Eigenschaft, mit jeder erzeugten Signatur ein gewisses Maß an Information über den geheimen Schlüssel preiszugeben[59]. Das heißt, dass für eine gegebene Nachricht viele verschiedene valide Signaturen möglich sind und diejenige, die durch den geheimen Schlüssel erzeugt wird, Rückschlüsse auf den geheimen Schlüssel zulässt. Nguyen formuliert mit dem sogenannten *Hidden Parallelepiped* Problem (HPP) eine besondere Angriffsmöglichkeit auf GGH und das verwandte Signaturverfahren NTRUSign[59]. Dabei handelt

es sich um ein Lernproblem auf Grundlage bekannter (*Nachricht, Signatur*) Paare. Es wird gezeigt, dass diese Paare Punkten auf einem unbekannten n-dimensionalen Parallelepiped zugeordnet werden können. Durch Finden oder Approximieren dieses Parallelepipeds kann es gelingen, den geheimen Schlüssel zu rekonstruieren. In der Arbeit von Nguyen werden Szenarien gezeigt, in denen dies mit einer polynomiellen Anzahl von (*Nachricht, Signatur*) Paaren und in polynomieller Zeit möglich ist.

Aus der kryptoanalytischen Untersuchung des GGH Verfahrens geht hervor, dass für ein annehmbares Maß an Sicherheit die Verwendung sehr großer Gitterdimensionen erforderlich wäre[59]. Die Schlüsselgröße wächst dabei quadratisch zur Gitterdimension, wodurch die Effizienz des Verfahrens nicht konkurrenzfähig im Hinblick auf modernere Verfahren ist.

5.3.3 Weitere Verfahren

NTRUSign ist als Signaturverfahren Bestandteil des NTRU Kryptosystems und wurde im Jahre 2003 beschrieben[39]. Es gilt als Inspiration für spätere Ansätze gitterbasierter Signaturverfahren, deren Sicherheit auf dem Ring-SIS Problem beruht[46][28]. Die Vorteile dieser Verfahren liegen in der gut belegten Sicherheitsgarantie, die sich auf die Worst Case Schwierigkeit von App-SVP in idealen Gittern stützt.

Buchmann diskutierte in einer 2009 erschienenen Arbeit die Möglichkeit, gitterbasierte Signaturverfahren auf Grundlage des Merkle-Signaturschemas zu entwickeln[18]. Ausgangspunkt hierfür bilden zwei im wesentlichen frei wählbare Komponenten: eine kollisionsresistene Hashfunktion H und ein Einmalsignaturschema (OTS). Diese werden mithilfe sogenannter *Merkle-Bäume* zu einem Signaturverfahren verbunden. Einmalsignaturverfahren zeichnen sich dadurch aus, dass ein Schlüsselpaar nur zur Signierung bzw. Verifzierung einer einzigen Nachricht verwendet wird. Um aus einem OTS nun ein vollwertiges Signaturverfahren zu konstruieren, muss eine möglichst große Anzahl von gültigen öffentlichen Schlüsseln des OTS fest und überprüfbar miteinander arrangiert werden. Die hierbei konstruierten Merkle-Bäume entstehen, indem die Hashwerte von 2^n öffentlichen Schlüssel des OTS als Blätter eines binären Baums der Tiefe n angeordnet werden. Die inneren Knoten werden nun ebenfalls mithilfe der kollisionsresistenten Hashfunktion H gebildet, indem diese auf die Konkatenation der Kindknoten angewendet wird. Ein innerer Knoten y wird also durch

$$y = H(\text{linker Kindknoten} \parallel \text{rechter Kindknoten}) \tag{5.36}$$

erzeugt. Somit sind die Werte aller inneren Knoten bis hin zur Wurzel durch die Werte der Blätter vorbestimmt. Ein Beispiel für einen Merkle-Baum mit $n = 2$ ist in Abbildung 5.1 zu sehen. Als öffentlicher Schlüssel e wird nur der Wert der Wurzel des

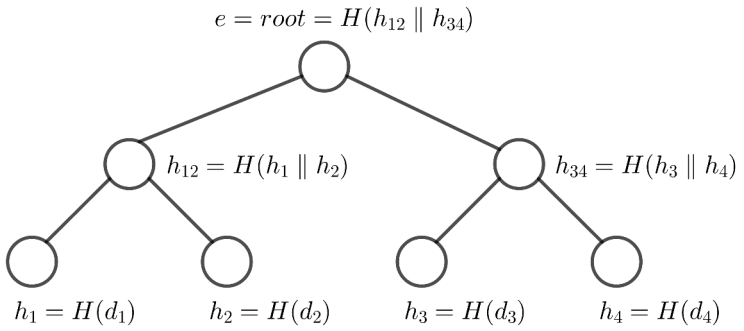


Abbildung 5.1: Merkle Baum mit $n = 2$ für $2^n = 4$ Einmalsignaturen.

Baumes benötigt. Den geheimen Schlüssel d bildet die Sammlung der Schlüsselpaare des OTS. Eine Signatur σ eines solchen Merkle-Signaturschemas ist ein Tupel

$$\sigma = (\varphi, \sigma_{OTS}, e_{OTS}, \{\text{AUTH}_i\}_\varphi) \tag{5.37}$$

mit φ, dem Index des Blattes, σ_{OTS}, der Einmalsignatur, e_{OTS}, dem Einmalverifizierungsschlüssel und dem sogenannten Authentifizierungspfad $\{\text{AUTH}_i\}_\varphi$. Dieser besteht aus der Menge aller n weiteren benötigten Knoten AUTH_i, die zur Berechnung der Wurzel aus e_{OTS} notwendig sind. Die Verfifizierung einer Signatur erfolgt also in zwei Schritten. Zunächst erfolgt die Verifizierung der Einmalsignatur σ_{OTS} mithilfe des Einmalverifizierungsschlüssels e_{OTS}. Danach erfolgt der Vergleich der auf Grundlage des Hashwertes von e_{OTS} und des Authentifizierungspfades $\{\text{AUTH}_i\}_\varphi$ berechneten Wurzel mit dem öffentlichen Schlüssel e. Die Darstellung eines Authentifizierungspfades in einem Merkle-Baum der Tiefe $n = 2$ ist in Abbildung 5.2 zu sehen. Mit $\varphi = 2$ wird die hierbei der Index des Blattes angegeben. Der Authentifizierungspfad hat also $n = 2$ Elemente und entspricht der Menge $\{\text{AUTH}_1, \text{AUTH}_2\}$.

Aus der Struktur des Merkle-Baumes folgt unmittelbar der größte Nachteil derartiger Verfahren: die Anzahl der signierbaren Nachrichten ist begrenzt. Der Vorteil dieser Verfahren besteht darin, dass die Sicherheit allein durch die Hashfunktion bestimmt wird. Buchmann konstruiert auf diese Weise Merkle-Signaturschemata mit der gitterbasierten Hashfunktion SWIFFT[51] und dem klassischen von Merkle vorgeschlagenem OTS[52], sowie einem von Lyubashevsky und Miccinaco beschriebenen gitterbasierten OTS[47].

Ähnlich wie im Bereich der Public-Key-Verfahren und Schlüsselaustauschverfahren, wird LWE auch als vielsversprechender Ansatz für gitterbasierte digitale Signaturverfahren diskutiert. Ein unter der Bezeichnung GPV bekanntes Verfahren wurde 2008 von Gentry, Peikert und Vaikuntanathan beschrieben[32]. Zu den aktuellsten

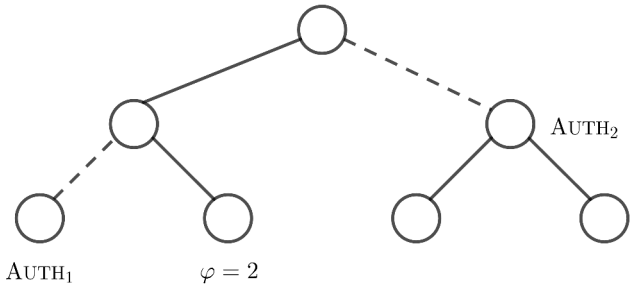


Abbildung 5.2: Authentifizierungspfad für $\varphi = 2$.

Arbeiten zählt ein durch Akleylek et al. im Jahre 2016 vorgstelltes, Ring-LWE basiertes Signaturschema, das aufgrund der effizienteren Struktur von Ring-LWE eine deutlich verbesserte Performance gegenüber GPV aufweist[5].

6 Fazit und Ausblick

Betrachtet man Gitter hinsichtlich ihrer Eignung für die Entwicklung quantensicherer asymmetrischer Kryptosysteme fällt zunächst ihre einfache mathematische Beschreibbarkeit und die gut belegte Schwierigkeit von Problemstellungen wie SVP, CVP und SIS auf, die nach bisherigem Kenntnisstand auch im Angesicht der Quanteninformatik Bestand hat. Ein Nachteil liegt darin, dass zur Beschreibung eines n-dimensionalen allgemeinen Gitters eine Matrix mit $O(n^2)$ Einträgen erforderlich ist, was sich ungünstig auf die Schlüsselgrößen potenzieller Verfahren auswirkt. Diesem Umstand kann durch Verwendung spezieller Gitter entgegengewirkt werden, die sich durch die Existenz innerer Gesetzmäßigkeiten effizienter beschreiben lassen. Von großer Bedeutung sind hier die idealen bzw. zyklischen Gitter, aber auch die NTRU Gitter. Weiterhin zu beachten ist der Umstand, dass die beschriebenen Gitterprobleme im durchschnittlichen Fall, also für Gitterbasen mit zufälliger Reduziertheit, deutlich weniger schwierig zu berechnen sind als im schlechtesten Fall mit Gitterbasen minimaler Reduziertheit. Somit ist die Generierung hinreichend schwieriger Probleminstanzen für ein Verfahren essentiell. Aus diesem Grund rücken zunehmend die Probleme der LWE Familie ins Zentrum des Interesses, da für diese gezeigt werden konnte, dass sie im durchschnittlichen Fall so schwer sind wie bestimmte Gitterprobleme im ungünstigsten Fall. Allerdings erfordern LWE basierte Verfahren die rechnerisch aufwendige Simulation Gaußscher Fehlerverteilungen. Darüber hinaus benötigen LWE-basierte Verfahren ebenfalls potenziell sehr große Schlüssel. Das Ring-LWE Problem beseitigt diesen Nachteil, da es auf Grund der Ringstruktur durch nur ein Polynom aus dem Ring R instanziert werden kann. Die Sicherheit geht hierbei auf die Schwierigkeit von Gitterproblemen in idealen Gittern zurück, welche nach derzeitigem Kenntnisstand als äquivalent zur Schwierigkeit derselben Problemstellungen in allgemeinen Gittern angenommen wird.

Im Vergleich zu den gezeigten klassischen kryptographischen Verfahren fällt auf, dass die ersten gitterbasierten Verfahren wie das GGH Verfahren eine vergleichbar einfache mathematische Struktur aufweisen, während modernere Verfahren wie NTRU und vor allem Ring-LWE basierte Verfahren wie der Ding Schlüsselaustausch einen deutlich komplexeren Aufbau aufweisen. Die Funktionsweise und Korrektheit von RSA, DSA und dem Diffie-Hellman Verfahren lässt sich relativ einfach rechnerisch zeigen. Im Bereich von Verfahren wie NTRU oder LWE-basierter Verfahren

besteht die Möglichkeit, dass es zu Fehlschlägen kommt, also dass der Fall

$$\textsc{Dec}(\textsc{Enc}(m)) \neq m \tag{6.1}$$

eintritt. Insofern besteht der Nachweis der Korrektheit der Verfahren in einer differenzierten Betrachtung von Wahrscheinlichkeiten, weshalb im Rahmen dieser Arbeit auf die entsprechenden Publikationen verwiesen werden musste. Es muss für diese Verfahren aufgezeigt werden, dass Fehlschläge nur mit vernachlässigbarer Wahrscheinlichkeit auftreten. Hierbei spielen insbesondere die Wahl der Parameter und die Eigenschaften der gewählten Fehlerverteilung eine Rolle. Die Verwendung von Störungen auf Grundlage diskreter Fehlerverteilungen bringt außerdem einen separat von den eigentlichen Kryptosystemen zu diskutierenden Aspekt mit sich, da die Nachbildung einer echten Gaußverteilung mit den Mitteln deterministischer Maschinen schwierig zu realisieren und rechnerisch aufwendig ist.

Generell besteht im Bereich der LWE- und SIS-basierten Verfahren eine Tendenz hin zu den Ringvarianten der Probleme, da für diese eine verbesserte Effizienz gezeigt werden konnte und ihre Sicherheit als äquivalent angenommen wird. Multiplikationsoperationen innerhalb der jeweiligen algebraischen Struktur gelten im Allgmeinen als Flaschenhals gitterbasierter Verfahren. Für allgemeiner Gitter handelt es sich dabei um Vektormultiplikationen mit zur Dimension quadratischem Aufwand oder Matrixmultiplikationen mit kubischem Aufwand. Verfahren, die in Quotientenringen arbeiten und ideale Gitter abbilden, benötigen die modulare Mutliplikation von Polynomen. Allerdings wurde gezeigt, dass sich hierfür die Fast-Fourier-Transformation (FFT) anwenden lässt, um den rechnerischen Aufwand zu verringern. Um (Ring)-LWE basierte Verfahren potenziell tauglich für den Embedded Bereich zu gestalten, werden alternative leichgewichtigere Fehlerverteilungen diskutiert[17]. Für das NTRU Verfahren existieren hinsichtlich der Schlüsselgrößen aktuelle Empfehlungen für verbreitete Sicherheitslevel. Diese sind in Tabelle 6.1 zu finden, wo sie mit den durch das ECRYPTII Projekt empfohlenen Schlüsselgrößen für das RSA Verfahren und für Elliptische-Kurven-Kryptosysteme (ECC) verglichen werden. Es zeigt sich, dass

Sicherheit	Schlüsselgröße in Bits		
in Bit	RSA	ECC	NTRU
128	3072	256	4829
192	7680	384	6523
256	15360	521	8173

Tabelle 6.1: Empfohlene Schlüsselgrößen RSA, ECC und NTRU[6][40].

NTRU vergleichbare Schlüsselgrößen zu RSA aufweist, die mit steigendem Sicherheitslevel geringer ansteigen und insofern eine günstige Progression aufweisen. Jedoch

liegen die Schlüsselgrößen ebenfalls eine Größenordnung über denen der Elliptische-Kurven-Kryptosysteme. Diese werden aber genau wie das RSA Verfahren mit dem Aufkommen von Quantencomputern relevanter Größe effizient angreifbar werden.

Der größte Vorteil gitterbasierter Verfahren liegt in der Sicherheit der zu Grunde liegenden Problemstellungen. Diese gilt als gut untersucht und hat nach derzeitigem Kenntnisstand auch im Angesicht der Quanteninformatik Bestand. Für die Approximation der gezeigten Gitterprobleme sind vor allem Gitterreduktionsalgorithmen interessant, wie der LLL-Algorithmus. Diese ermöglichen die effiziente Berechnung von Näherungslösungen zu verschiedenen Gitterproblemen mit in der Gitterdimension exponentiellen Approximationsfaktoren. Für polynomielle Approximationsfaktoren ist derzeit kein effizientes Verfahren bekannt. Das US-amerikanische National Institute of Standards and Technology (NIST) hat 2016 eine Ausschreibung gestartet, um kryptographische Verfahren für das Post-Quantenzeitalter zu evaluieren und die geeignetsten Anwärter zu standardisieren[1]. Insgesamt wurden 82 Vorschläge für Public-Key-Verfahren und digitale Signaturverfahren eingereicht, davon 28 gitterbasierte Verfahren. Nach der zweiten Runde sind insgesamt noch 26 Kandidaten im Rennen, darunter neun gitterbasierte Public-Key-Kryptosysteme und drei gitterbasierte Signaturverfahren. Von den in dieser Arbeit aufgeführten Verfahren zählen dazu NTRU, NTRU Prime, New Hope und das Frodo Schlüsselaustauschverfahren. Die gitterbasierten Verfahren bildeten bisher in jeder Runde den größten Anteil, gefolgt von Verfahren auf Grundlage fehlerkorrigierender Codes. Es besteht also eine gute Chance, dass die Kryptographie im Zeitalter der Quantencomputer von gitterbasierten Verfahren geprägt sein wird.

[1] www.nist.gov/pqcrypto

Literatur

[1] Miklós Ajtai. »Generating hard instances of lattice problems«. In: *Proceedings of the 28th annual ACM Symposium on Theory of Computing – STOC '96*. New York: ACM Press, 1996, S. 99–108.

[2] Miklós Ajtai. »The shortest vector problem in L2 is NP-hard for randomized reductions«. In: *Proceedings of the 30h annual ACM Symposium on Theory of Computing – STOC '98*. New York: ACM Press, 1998, S. 10–19.

[3] Miklós Ajtai und Cynthia Dwork. »A public-key cryptosystem with worst-case/average-case equivalence«. In: *Proceedings of the 29th annual ACM Symposium on Theory of Computing – STOC '97*. New York: ACM Press, 1997, S. 284–293.

[4] Miklós Ajtai, Ravi Kumar und D. Sivakumar. »A sieve algorithm for the shortest lattice vector problem«. In: *Proceedings of the 33rd annual ACM Symposium on Theory of Computing – STOC '01*. New York: ACM Press, 2001, S. 601–610.

[5] Sedat Akleylek u. a. »An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation«. In: *Progress in Cryptology – AFRICACRYPT 2016*. Hrsg. von David Pointcheval, Abderrahmane Nitaj und Tajjeeddine Rachidi. Cham: Springer International Publishing, 2016, S. 44–60.

[6] »Algorithms, Key Size and Protocols Report (2018)«. In: *H2020-ICT-2014 – Project 645421* D5.4 (Feb. 2018).

[7] Erdem Alkim u. a. »Post-quantum key exchange- a new hope«. In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016.

[8] Benny Applebaum u. a. »Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems«. In: *Advances in Cryptology – CRYPTO 2009*. Hrsg. von Shai Halevi. Berlin, Heidelberg: Springer Verlag, 2009, S. 595–618.

[9] Sanjeev Arora u. a. »The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations«. In: *Journal of Computer and System Sciences* 54.2 (Apr. 1997), S. 317–331.

[10] László Babai. »On Lovász' lattice reduction and the nearest lattice point problem«. In: *Combinatorica* 6 (1986), S. 1–13.

[11] Abhishek Banerjee, Chris Peikert und Alon Rosen. »Pseudorandom Functions and Lattices«. In: *Advances in Cryptology – EUROCRYPT 2012*. Hrsg. von David Pointcheval und Thomas Johansson. Springer Berlin Heidelberg, 2012, S. 719–737.

[12] Daniel J. Bernstein u. a. »NTRU Prime: Reducing Attack Surface at Low Cost«. In: *Selected Areas in Cryptography – SAC 2017*. Hrsg. von Carlisle Adams und Jan Camenisch. Cham: Springer International Publishing, 2018, S. 235–260.

[13] Sauvik Bhattacharya u. a. »spKEX: An optimized lattice-based key exchange«. In: *IACR Cryptology ePrint Archive 2017/709* (2017).

[14] Joppe W. Bos u. a. »Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem«. In: *2015 IEEE Symposium on Security and Privacy*. IEEE, Mai 2015, S. 553–570.

[15] Joppe Bos u. a. »Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE«. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security – CCS'16*. New York: ACM Press, 2016, S. 1006–1018.

[16] Johannes Buchmann. *Einführung in die Kryptographie*. 6. Aufl. Springer Berlin Heidelberg, 2016.

[17] Johannes Buchmann u. a. »High-Performance and Lightweight Lattice-Based Public-Key Encryption«. In: *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security – IoTPTS '16*. New York: ACM Press, 2016, S. 2–9.

[18] Johannes Buchmann u. a. »Post-quantum cryptography: lattice signatures«. In: *Computing* 85.1-2 (Juni 2009), S. 105–125.

[19] Wouter Castryck, Ilia Iliashenko und Frederik Vercauteren. »On error distributions in ring-based LWE«. In: *LMS Journal of Computation and Mathematics* 19.A (Aug. 2016), S. 130–145.

[20] Narasimham Challa und Jayaram Pradhan. »Performance Analysis of Public key Cryptographic Systems RSA and NTRU«. In: *IJCSNS International Journal of Computer Science and Network Security* 7.8 (2007), S. 87–96.

[21] Donald Donglong Chen u. a. »High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems«. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.1 (Jan. 2015), S. 157–166.

[22] Hao Chen, Kristin E. Lauter und Katherine E. Stange. »Vulnerable Galois RLWE Families and Improved Attacks«. In: *Cryptology ePrint Archive 2016/193* (2016).

[23] Hao Chen, Kristin Lauter und Katherine E. Stange. »Attacks on the Search RLWE Problem with Small Errors«. In: *SIAM J. Appl. Algebra Geometry* 1.1 (2017), S. 665–682.

[24] Don Coppersmith. »Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities«. In: *Journal of Cryptology* 10.4 (Sep. 1997), S. 233–260.

[25] Don Coppersmith und Adi Shamir. »Lattice Attacks on NTRU«. In: *Advances in Cryptology – EUROCRYPT '97*. Hrsg. von Walter Fumy. Springer Berlin Heidelberg, 1997, S. 52–61.

[26] Whitfield Diffie und Martin Hellman. »New directions in cryptography«. In: *IEEE Transactions on Information Theory* 22.6 (Nov. 1976), S. 644–654.

[27] Jintai Ding, Xiang Xie und Xiaodong Lin. »A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem«. In: *IACR Cryptology ePrint Archive* (2012).

[28] Léo Ducas u. a. »Lattice Signatures and Bimodal Gaussians«. In: *Advances in Cryptology – CRYPTO 2013*. Hrsg. von Ran Canetti und Juan A. Garay. Springer Berlin Heidelberg, 2013, S. 40–56.

[29] Yara Elias u. a. »Provably Weak Instances of Ring-LWE«. In: *Advances in Cryptology – CRYPTO 2015*. Hrsg. von Rosario Gennaro und Matthew Robshaw. Springer Berlin Heidelberg, 2015, S. 63–92.

[30] Peter van Emde-Boas. *Another NP-complete problem and the complexity of computing short vectors in a lattice*. Techn. Ber. Department of Mathematics, University of Amsterdam, 1981.

[31] Walter Fumy. »Quantencomputer und die Zukunft der Kryptographie«. In: *Datenschutz und Datensicherheit – DuD* 41.1 (Jan. 2017), S. 13–16.

[32] Craig Gentry, Chris Peikert und Vinod Vaikuntanathan. »Trapdoors for hard lattices and new cryptographic constructions«. In: *Proceedings of the 40th annual ACM Symposium on Theory of Computing – STOC '08*. New York: ACM Press, 2008, S. 197–206.

[33] O. Goldreich u. a. »Approximating shortest lattice vectors is not harder than approximating closest lattice vectors«. In: *Information Processing Letters* 71.2 (Juli 1999), S. 55–61.

[34] Oded Goldreich und Shafi Goldwasser. »On the Limits of Nonapproximability of Lattice Problems«. In: *Journal of Computer and System Sciences* 60.3 (Juni 2000), S. 540–563.

[35] Oded Goldreich, Shafi Goldwasser und Shai Halevi. »Public-Key Cryptosystems from Lattice Reduction Problems«. In: *Advances in Cryptology – CRYPTO '97*. Hrsg. von Burton S. Kaliski. Springer Berlin Heidelberg, 1997, S. 112–131.

[36] Markus Grassl u. a. »Applying Grover's algorithm to AES: Quantum Resource Estimates«. In: *Post-Quantum Cryptography*. Hrsg. von Tsuyoshi Takagi. Cham: Springer International Publishing, Dez. 2016, S. 29–43.

[37] Daya Sagar Gupta und G.P. Biswas. »Cryptanalysis of Wang et al.'s lattice-based key exchange protocol«. In: *Perspectives in Science* (2016).

[38] Jeffrey Hoffstein, Jill Pipher und Joseph H. Silverman. »NTRU: A ring-based public key cryptosystem«. In: *Algorithmic Number Theory*. Hrsg. von Joe P. Buhler. Springer Berlin Heidelberg, 1998, S. 267–288.

[39] Jeffrey Hoffstein u. a. »NTRUSign: Digital Signatures Using the NTRU Lattice«. In: *Topics in Cryptology – CT-RSA 2003*. Hrsg. von Marc Joye. Springer Berlin Heidelberg, 2003, S. 122–140.

[40] Jeff Hoffstein u. a. »Choosing Parameters for NTRUEncrypt«. In: *Topics in Cryptology – CT-RSA 2017*. Hrsg. von Helena Handschuh. Cham: Springer International Publishing, 2017, S. 3–18.

[41] Nick A. Howgrave-Graham. »A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU«. In: *Advances in Cryptology – CRYPTO 2007*. Hrsg. von Alfred Menezes. Springer Berlin Heidelberg, 2007, S. 150–169.

[42] Nick A. Howgrave-Graham und Nigel P. Smart. »Lattice Attacks on Digital Signature Schemes«. In: *Designs, Codes and Cryptography* 23.3 (2001), S. 283–290.

[43] Christian Karpfinger. *Algebra: Gruppen - Ringe - Körper*. 2. Aufl. Heidelberg: Spektrum Akad. Verl., 2010.

[44] Xinyu Lei und Xiaofeng Liao. »NTRU-KE: A Lattice-based Public Key Exchange Protocol«. In: *IACR Cryptology ePrint Archive 2013/718* (2013).

[45] Arjen K. Lenstra, Hendrik W. Lenstra und László Lovász. »Factoring polynomials with rational coefficients«. In: *Mathematische Annalen* 261.4 (Dez. 1982), S. 515–534.

[46] Vadim Lyubashevsky. »Lattice Signatures without Trapdoors«. In: *Advances in Cryptology – EUROCRYPT 2012*. Hrsg. von David Pointcheval und Thomas Johansson. Springer Berlin Heidelberg, 2012, S. 738–755.

[47] Vadim Lyubashevsky und Daniele Micciancio. »Asymptotically Efficient Lattice-Based Digital Signatures«. In: *Theory of Cryptography*. Hrsg. von Ran Canetti. Springer Berlin Heidelberg, 2008, S. 37–54.

[48] Vadim Lyubashevsky und Daniele Micciancio. »Generalized Compact Knapsacks Are Collision Resistant«. In: *Automata, Languages and Programming*. Hrsg. von Michele Bugliesi u. a. Springer Berlin Heidelberg, 2006, S. 144–155.

[49] Vadim Lyubashevsky, Chris Peikert und Oded Regev. »A Toolkit for Ring-LWE Cryptography«. In: *Advances in Cryptology – EUROCRYPT 2013*. Hrsg. von Thomas Johansson und Phong Q. Nguyen. Springer Berlin Heidelberg, 2013, S. 35–54.

[50] Vadim Lyubashevsky, Chris Peikert und Oded Regev. »On ideal lattices and learning with errors over rings«. In: *Lecture Notes in Computer Science*. 2010, S. 1–23.

[51] Vadim Lyubashevsky u. a. »SWIFFT: A Modest Proposal for FFT Hashing«. In: *Fast Software Encryption*. Hrsg. von Kaisa Nyberg. Springer Berlin Heidelberg, 2008, S. 54–72.

[52] Ralph C. Merkle. »A Certified Digital Signature«. In: *Advances in Cryptology – CRYPTO' 89 Proceedings*. Hrsg. von Gilles Brassard. Springer New York, 1989, S. 218–238.

[53] Daniele Micciancio. »Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions«. In: *Computational Complexity* 16.4 (Dez. 2007), S. 365–411.

[54] Daniele Micciancio. »Improving Lattice Based Cryptosystems Using the Hermite Normal Form«. In: *Cryptography and Lattice*. Hrsg. von Joseph H. Silverman. Springer Berlin Heidelberg, 2001, S. 126–145.

[55] Daniele Micciancio. »The Geometry of Lattice Cryptography«. In: *Foundations of Security Analysis and Design VI*. Hrsg. von Alessandro Aldini und Roberto Gorrieri. Springer Berlin Heidelberg, 2011, S. 185–210.

[56] Daniele Micciancio. »The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant«. In: *SIAM Journal on Computing* 30.6 (Jan. 2001), S. 2008–2035.

[57] Daniele Micciancio und Bogdan Warinschi. »A linear space algorithm for computing the hermite normal form«. In: *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation – ISSAC '01*. New York: ACM Press, 2001, S. 231–236.

[58] Phong Q. Nguyen. »Lattice Reduction Algorithms: Theory and Practice«. In: *Advances in Cryptology – EUROCRYPT 2011*. Hrsg. von Kenneth G. Paterson. Springer Berlin Heidelberg, 2011, S. 2–6.

[59] Phong Q. Nguyen und Oded Regev. »Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures«. In: *Journal of Cryptology* 22.2 (Apr. 2009), S. 139–160.

[60] Phong Nguyen und Jacques Stern. »Cryptanalysis of the Ajtai-Dwork cryptosystem«. In: *Advances in Cryptology – CRYPTO '98*. Hrsg. von Hugo Krawczyk. Springer Berlin Heidelberg, 1998, S. 223–242.

[61] Chris Peikert. »How (Not) to Instantiate Ring-LWE«. In: *Security and Cryptography for Network*. Hrsg. von Vassilis Zikas und Roberto De Prisco. Cham: Springer International Publishing, 2016, S. 411–430.

[62] Chris Peikert. »Lattice Cryptography for the Internet«. In: *Post-Quantum Cryptography*. Hrsg. von Michele Mosca. Cham: Springer International Publishing, 2014, S. 197–219.

[63] Chris Peikert. »Public-key cryptosystems from the worst-case shortest vector problem«. In: *Proceedings of the 41st annual ACM Symposium on Theory of Computing – STOC '09*. New York: ACM Press, 2009, S. 333.

[64] Thomas Plantard und Michael Schneider. »Creating a Challenge for Ideal Lattices«. In: *IACR Cryptology ePrint Archive 2013/039* (2013), S. 1–17.

[65] Oded Regev. »On lattices, learning with errors, random linear codes, and cryptography«. In: *Journal of the ACM* 56.6 (Sep. 2009), S. 1–40.

[66] Ronald L. Rivest, Adi Shamir und Leonard Adleman. »A method for obtaining digital signatures and public-key cryptosystems«. In: *Communications of the ACM* 21.2 (Feb. 1978), S. 120–126.

[67] Peter W. Shor. »Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer«. In: *SIAM Journal on Computing* 26.5 (1997), S. 1484–1509.

[68] Damien Stehlé und Ron Steinfeld. »Making NTRU as Secure as Worst-Case Problems over Ideal Lattices«. In: *Advances in Cryptology – EUROCRYPT 2011*. Hrsg. von Kenneth G. Paterson. Springer Berlin Heidelberg, 2011, S. 27–47.

[69] Maheswara Rao Valluri. »Cryptanalysis of Xinyu et al.'s NTRU-Lattice Based Key Exchange Protocol«. In: *CoRR* 1611.08686 (Nov. 2016).

[70] Christine van Vredendaal. »Reduced Memory Meet-in-the-Middle Attack against the NTRU Private Key«. In: *IACR Cryptology ePrint Archive 2016/177* (2016).

[71] ShanBiao Wang u. a. »Lattice-based key exchange on small integer solution problem«. In: *Science China Information Sciences* 57.11 (Nov. 2014), S. 1–12.

[72] Annette Werner. *Elliptische Kurven in der Kryptographie*. Springer Berlin Heidelberg, 2002.