Auf Polynomgleichungen basierende Public-Key-Kryptosysteme

Diplomarbeit

Alexander May
Johann Wolfgang Goethe - Universität
Frankfurt am Main
Fachbereich Informatik

Betreuer: Prof. Dr. C.P. Schnorr

4. Juni 1999

„It's the strangest cipher I ever encountered," said Mr. Keen „ the strangest I ever heard of – I have seen hundreds of ciphers, hundreds secret codes of the State Department, secret military codes, elaborate Oriental ciphers, symbols used in commercial transactions, symbols used by criminals and every species of malefactor. And every one of them can be solved with time and patience and a little knowledge of the subject. But this" – he sat looking at it with eyes half closed – „this is *too* simple."

[from David Kahn, *„The Codebreakers"*, 1996]

Danksagung

Ich bedanke mich bei Prof. C.P. Schnorr und Jean-Pierre Seifert für Hinweise zum Aufschreiben der Diplomarbeit und Verbesserungen. Joseph H. Silverman und Phong Nguyen danke ich für Anregungen zu den neu vorgestellten Gitterbasenattacken. Ich danke Henrik Koy, der mir stabile Gitterreduktionsalgorithmen für die Attacken auf hohe Sicherheitsdimensionen zur Verfügung stellte. Für nicht-wissenschaftliche Unterstützung bedanke ich mich bei meiner Familie und Petra Lorenz.

Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbständig verfaßt und keine anderen Hilfsmittel als die angegebenen Quellen verwendet zu haben.

Alexander May — Frankfurt, den 4. Juni 1999

Zusammenfassung

Wir betrachten in dieser Diplomarbeit die Sicherheit des ringbasierten Public Key Kryptosystems NTRU, das 1996 von J. Hoffstein, J. Pipher und J.H. Silverman [24] vorgeschlagen wurde. Dieses Kryptosystem bietet schnelle Kodierung und Dekodierung in Laufzeit $O(n^2)$ bei kleinem Sicherheitsparameter n. Die Sicherheit des Systems beruht auf einem Polynomfaktorisierungsproblem (PFP) im Polynomring $\mathbb{Z}_q[X]/(X^n - 1)$.

Das PFP wurde von Coppersmith und Shamir [14] auf ein Kürzestes Vektor Problem im Gitter L^{cs} reduziert. Die neuen Ergebnisse dieser Arbeit bauen auf dem Gitter L^{cs} auf. Wir betrachten die Nachteile von L^{cs} und konstruieren verbesserte Gitterbasen zum Angriff auf das NTRU-Kryptosystem. Dabei nutzen wir Strukturen des Polynomrings $\mathbb{Z}_q[X]/(X^n - 1)$ und der geheimen Schlüssel aus. Durch die neuen Gitterbasen wird der Quotient aus der Länge des zweitkürzesten und der Länge des kürzesten Gittervektors vergrößert. Da wir Approximationsalgorithmen zum Finden eines kürzesten Vektors verwenden, beschleunigt dies die Attacken. Wir präsentieren verschiedene Methoden, wie man die Dimension der Gitterbasen verkleinern kann.

Durch die verbesserten Gitterattacken erhalten wir eine Cryptanalyse des NTRU-Systems in der vorgeschlagenen mittleren Sicherheitsstufe. Beträgt die Zeit zum Brechen eines Public-Keys unter Verwendung der Coppersmith/Shamir-Basis 1 Monat, so verringert sich die Laufzeit durch einen kombinierten Einsatz der neuen Gitterbasen auf ca. 5 Stunden auf einem Rechner und bei Parallelisierung auf ca. 1:20 Stunde auf 4 Rechnern. Wir erwarten, daß die neuen Methoden NTRU in hoher Sicherheitsstufe $n = 167$ brechen, obwohl für dieses n bisher nur „schwache“ Schlüssel gebrochen wurden.

Trotz signifikanter Verbesserungen deuten die experimentellen Ergebnisse auf ein exponentielles Laufzeitverhalten bei steigendem Sicherheitsparameter n hin. Der Laufzeitexponent kann allerdings gesenkt werden, so daß man n größer wählen muß, um Sicherheit gegenüber den neuen Attacken zu erzielen. Auch wenn das NTRU-Kryptosystem nicht vollständig gebrochen wird, verliert es seinen größten Vorteil gegenüber anderen Public Key Kryptosystemen: Die effiziente Kodierung und Dekodierung bei kleinem Sicherheitsparameter n.

Inhaltsverzeichnis

1 Einleitung **5**

2 Einführung in die Theorie der Gitter und Polynomringe **11**
- 2.1 Gitter und Komplexität von Gitterproblemen 11
- 2.2 Gitterbasenreduktion 13
- 2.3 Polynomringe 17

3 Das NTRU-Cryptosystem **21**
- 3.1 Notationen 21
- 3.2 Schlüsselerzeugung und Parameterwahl 23
- 3.3 Zugrundeliegendes Problem 24
- 3.4 Ver- und Entschlüsselung einer Nachricht m 24
- 3.5 Korrektheit 26
- 3.6 Verbessern der Nachrichtenexpansion 26
  - 3.6.1 Turbo-NTRU 27

4 Angriffe auf NTRU **29**
- 4.1 Brute-Force Attacke 29
- 4.2 Odlyzko's Meet-In-The-Middle Attacke 30
- 4.3 Multiple-Transmission-Attacke 33
- 4.4 Adaptive Chosen Ciphertext Attacken 34
  - 4.4.1 Ein Analogon von Bleichenbachers Attacke für NTRU . . 34
  - 4.4.2 Korrelationsattacke über induzierten Dekodierfehler . . . 35

5 Gitterreduktions-Angriff auf den Public-Key **37**
- 5.1 Das Coppersmith-Shamir Gitter L^{cs} 37
- 5.2 Optimale Wahl von λ 40
- 5.3 Ergebnisse des Angriffs 41
- 5.4 Alternative Schlüssel f' 45

6 Neue verbesserte Angriffsgitter **46**
- 6.1 Herleitung des Run-Gitters L^r 47
  - 6.1.1 Die zyklische Struktur von L^{cs} 47
  - 6.1.2 Die Gitterbasis von L^r 49
  - 6.1.3 Randomisierung der Basis für Run-Gitter 53
  - 6.1.4 Zero-Forced Gitter 55
- 6.2 Verallgemeinerte Run-Gitter: Das Gitter L^{part} 56

6.3 Der kürzeste Vektor in $L^r(\theta)$ in Supremumsnorm 57
6.4 Dimensionsreduzierende Gitter 59
6.4.1 Das Gitter L_g^{red} 59
6.4.2 Das Gitter L_f^{red} 59
6.5 Herleitung des Balance-Gitters L^b 60
6.5.1 Balancierte Polynome 60
6.5.2 Das balancierte Gitter L^b 62
6.5.3 Ein weiteres balanciertes Gitter $\hat{L}^b$ 62
6.6 Kombinierbarkeit der Gittermethoden 64
6.7 Experimentelle Resultate 65
6.7.1 Mittlere Sicherheitsstufe $n = 107$ 65
6.7.2 Hohe Sicherheitsstufe $n = 167$ 66
6.8 Zwei neue Ansätze für NTRU-167 67
6.8.1 Verwendung mehrerer Schlüsselgleichungen 67
6.8.2 Brechen der Schlüsselsymmetrie: Das Gitter L^{sym} 70

Kapitel 1

Einleitung

Die Geburtsstunde der modernen Public-Key Kryptographie datiert auf das Jahr 1976, als Diffie und Hellman ihren berühmt gewordenen Artikel ,,New directions in Cryptography“ [15] veröffentlichten. Dort beschrieben sie ein Schlüsselaustauschprotokoll, mit Hilfe dessen zwei Partner auf einem öffentlichen, nicht abhörgeschützten Kanal einen gemeinsamen geheimen Schlüssel vereinbaren können. Ein Angreifer, der die Kommunikation der beiden belauscht, soll dabei keine Information über den vereinbarten Schlüssel gewinnen. Die Sicherheit des Protokolls basiert auf dem Diffie-Hellman-Problem: Gegeben eine zyklische Gruppe G und ein öffentlicher Generator $g \in G$. Berechne aus dem Tupel (g, g^a, g^b) den Wert g^{ab}. Kann man den diskreten Logarithmus in G effizient berechnen, so kann man auch das Diffie-Hellman-Problem lösen. Daß die beiden Probleme polynomialzeitäquivalent sind, wurde erst kürzlich von Maurer [32] gezeigt. Bis heute kennt man keine Polynomialzeitalgorithmen, die das Diffie-Hellman-Problem lösen können.
1978 wurde der wohl heutzutage bekannteste Public-Key Verschlüsselungsalgorithmus RSA von Rivest, Shamir und Adleman [39] vorgeschlagen. Die Sicherheit von RSA basiert auf der Schwierigkeit, eine aus zwei großen Primzahlen zusammengesetzte Zahl zu faktorisieren. Man hat allerdings momentan weder komplexitätstheoretische untere Schranken, wie „schwer“ Faktorisieren wirklich ist, noch weiß man, ob das Brechen von RSA wirklich äquivalent zum Faktorisierungsproblem ist. Das Vertrauen in RSA ist vielmehr darin begründet, daß das Faktorisierungsproblem schon seit sehr langer Zeit untersucht wurde und bis dato kein effizienter Faktorisierungsalgorithmus bekannt ist. Die Fortschritte, die in den letzten 10 Jahren mit der Entwicklung des Quadratischen Siebs [38] und des Zahlkörpersiebs [28] gemacht wurden, sind aber bedeutend.
Seitdem wurden einige Public-Key Protokolle vorgeschlagen, deren Sicherheit auf verschiedenen Problemen beruhen. Die bedeutendsten sind

Merkle-Hellman Rucksack '78 [21] Dieses und verwandte Systeme beruhen auf der Schwierigkeit des Subset Sum Problems, das NP-vollständig ist. Trotzdem wurden diese speziellen Instanzen von Subset Sum alle als unsicher nachgewiesen (bis zunächst auf das Chor-Rivest System).

McEliece '78 [33] Das McEliece Kryptosystem beruht auf algebraischer Kodierungstheorie und wird als sicher betrachtet. Seine Sicherheit beruht auf dem Problem des Dekodierens von linearen Codes, das ebenso NP-vollständig ist. Daß dieses System eher von akademischem anstatt praktischem Interesse ist, liegt an den großen Schlüssellängen – quadratisch im Sicherheitsparameter – und der Nachrichtenexpansion von 2:1. D.h. der kodierte Text ist doppelt solang wie der zugrundeliegende Klartext.

Chor-Rivest '84 [10] Dieses System ist ebenfalls Subset Sum-basiert. 1994 schlugen Schnorr, Euchner [58] eine Attacke vor, die das System für kleine Sicherheitsparameter bricht. Eine vollständige Cryptanalyse für alle vorgeschlagenen Sicherheitsstufen des Chor-Rivest Systems wurde '98 von Vaudenay [61] vorgestellt.

ElGamal '85 [17] Das ElGamal Kryptosystem basiert auf der Schwierigkeit des diskreten Logarithmus Problems in endlichen Körpern. Sein Nachteil liegt wie bei McEliece in einer Nachrichtenexpansionsrate von 2:1.

Elliptische Kurven '87 (Koblitz et al [25]) Kryptosysteme über elliptischen Kurven sind Modifikationen anderer, gruppenbasierter Systeme – wie z.B. RSA und ElGamal, die über der additiven Gruppenstruktur elliptischer Kurven anstatt über der multiplikativen Gruppe endlicher Körper arbeiten. Diese Systeme benötigen kleinere Sicherheitsparameter, da man z.B. für das Diskrete Logarithmus-Analogon in den Gruppen der Punkte auf elliptischen Kurven im allgemeinen noch keine subexponentiellen Algorithmen kennt, im Gegensatz zu den multiplikativen Gruppen über endlichen Körpern (Index-Calculus, Zahlkörpersiebvariante von Coppersmith).

Es ist wichtig anzumerken, daß ein *deterministisches* Public-Key Kryptosystem mit öffentlichem Schlüssel k niemals absolute Sicherheit gegen Angriffe im informationstheoretischen Sinne liefern kann. Ein Angreifer, der einen verschlüsselten Text y erhält, kann jeden möglichen Klartext mit Hilfe des öffentlich bekannten Kodieralgorithmus e_k verschlüsseln, bis er das eindeutig bestimmte x findet mit $y = e_k(x)$. Dieser Klartext x ist die Entschlüsselung von y. Deshalb können wir nur die komplexitätstheoretische Sicherheit von deterministischen Public-Key Kryptosystemen studieren. Dazu benutzen wir den Begriff der *Trapdoor Einweg Funktion*, den wir hier informal vorstellen wollen.

Die Verschlüsselungsfunktion e_k sollte einfach zu berechnen sein, wohingegen die inverse Dekodierungsfunktion – für nichtlegitimierte Empfänger – schwierig zu berechnen sein muß. Diese Eigenschaft, daß eine Funktion einfach zu berechnen aber schwierig zu invertieren ist, wird oft als *Einweg*-Eigenschaft bezeichnet. Wir benötigen zur Konstruktion eines Public-Key Kryptosystems also eine injektive (um die Entschlüsselung eindeutig zu machen) Einweg-Funktion e_k. Dies genügt aber noch nicht; der designierte Empfänger sollte in der Lage sein, die erhaltenen Nachrichten effizient dekodieren zu können. Deshalb muß er eine geheime Zusatzinformation – genannt *Trapdoor* (Hintertür) – besitzen, die es ihm erlaubt, eine Inversion der Kodierung einfach durchzuführen. Die Verschlüsselungsfunktion e_k sollte also eine *Trapdoor Einweg Funktion* sein. Obwohl es einige Funktionen gibt, von denen man annimmt, daß sie die Einweg-Eigenschaft besitzen, konnte dies bisher für keine Funktion bewiesen werden.

Seit zwei Jahrzehnten wurde in Veröffentlichungen immer wieder diskutiert (Brassard [8], Goldreich, Goldwasser [18]), ob es möglich ist, die Sicherheit von Public-Key Kryptosystemen auf der Komplexitätsannahme P $\neq$ NP aufzubauen. Ein Problem dabei ist, daß die Komplexitätstheorie sich mit der worst-case Komplexität algorithmischer Fragestellungen beschäftigt, wohingegen man für die Public Key Kryptographie Probleme benötigt, die im average-case schwierig sind. Es nutzt nichts, wenn Instanzen des Public Keys nur im worst-case schwierig sind, die potentielle Trapdoor Einwegfunktion muß für alle gewählten Schlüssel k schwierig zu invertieren sein.

Man betrachte dazu die Sicherheit von RSA und die Schwierigkeit des Faktorisierungsproblems. Erstens wissen wir nicht, ob Faktorisieren in P möglich ist. Zweitens: Wenn wir annehmen, daß Faktorisieren nicht in P ist, dann wissen wir noch nichts über den Spezialfall der Faktorisierung eines Produkts zweier großer Primzahlen $p \cdot q$, wie sie in RSA verwendet werden. Drittens: Selbst wenn die Faktorisierung von $p \cdot q$ im worst-case schwierig ist, wissen wir noch nichts über die Komplexität im average-case. Viertens wissen wir ebenfalls nicht, ob die Dekodierung in RSA ohne den privaten, geheimen Schlüssel äquivalent zum Berechnen der Eulerfunktion $\varphi(pq) = (p-1)(q-1)$ ist (obwohl das Berechnen von $\varphi(n)$ aus n äquivalent zum Faktorisieren ist). D.h. obwohl das Vertrauen in das RSA-Kryptosystem groß ist, existiert eine beachtliche Lücke zwischen der Annahme, daß Faktorisieren im worst-case schwierig ist, und einem Sicherheitsbeweis für das System.

1996 fand Ajtai [3] eine bemerkenswerte Reduktion des worst-case auf den average-case für eine Version des Kürzesten-Vektor-Problems (englisch SVP: Shortest Vector Problem). Eine solche Verbindung kennt man bisher für kein anderes NP-hartes Problem. Ajtai [2] bewies dann 1997, aufbauend auf Arbeiten von Schnorr [56] und Adleman [1], daß SVP unter randomisierten Reduktionen NP-hart ist. Nach dieser Arbeit gab es eine Reihe von Bemühungen, diese „worst-case auf average-case Reduktion“ zur Konstruktion sicherer Public Key Kryptosysteme unter gitterbasierten Annahmen zu konstruieren. Ajtai und Dwork [4] schlugen ein PKKS (Public Key Kryptosystem) vor, das im average-case beweisbar sicher ist unter der Annahme der worst-case Schwierigkeit einer speziellen Version des SVP, nämlich das Finden des kürzesten Vektors in einem Gitter mit n^c-eindeutig kürzestem Vektor für eine genügend große Konstante c.

Im selben Jahr veröffentlichten Goldwasser, Goldreich und Halevi [19] ein weiteres gitterbasiertes PKKS; das GGH-Kryptosystem. Die Sicherheit des Systems beruht auf der Schwierigkeit, aus einer randomisierten Basis eines Gitters die minimale Basis - in bezug auf die Euklidische Länge der Basisvektoren - zu rekonstruieren. Das GGH-System wurde 1999 von P. Nguyen [36] gebrochen.

Das in dieser Diplomarbeit behandelte ringbasierte NTRU-Kryptosystem – erstmals vorgestellt 1996 von J. Hoffstein, J. Silverman und J.H. Hoffstein [24, 23] – beruht auf der Schwierigkeit eines bestimmten Polynomfaktorisierungsproblems (PFP) im Polynomring $\mathbb{Z}_q[X]/(X^n-1)$, wobei n der Sicherheitsparameter des Systems ist. Dieses Faktorisierungsproblem läßt sich, wie 1997 in Coppersmith und Shamir [14] gezeigt, auf das SVP in einem $(2n \times 2n)$-Gitter L^{cs} reduzieren. Obwohl das SVP im worst-case NP-hart ist, wissen wir nicht, ob das Auffinden eines kürzesten Vektors im speziellen Gitter L^{cs} NP-hart ist. Man beachte, daß es Gitter gibt, deren kürzester Vektor mit Hilfe des L^3-Algorithmus in Polynomialzeit gefunden werden kann. Nehmen wir an, das SVP im Gitter L^{cs} wäre NP-hart im worst-case, dann wissen wir noch nichts über seine average-case Komplexität. Weiterhin ist die Dekodierung im NTRU-Kryptosystem sicher nicht äquivalent zum PFP. Das System benutzt probabilistische Dekodierung – wie zuerst von Goldwasser, Micali [20] vorgeschlagen – mittels eines zufälligen Randomisierungspolynoms ϕ. Das Erraten von ϕ erlaubt zwar die Dekodierung einer Nachricht, es hilft aber nicht bei der Lösung des PFPs.

Das NTRU-Kryptosystem bietet, wie zuvor diskutiert, genügend Ansatzmöglichkeiten für Angriffe. Bekannte allgemeine Attacken auf das System werden in Kapitel 4 aufgeführt, nachdem NTRU in Kapitel 3 vorgestellt wurde. Das Kryptosystem wäre aber für kryptanalytische Bemühungen nicht interessant, wenn es nicht neuartige Eigenschaften hätte, die es von derzeit existierenden Systemen unterscheidet. Die wohl bemerkenswerteste Verbesserung ist die Schnelligkeit des NTRU-Kryptosystems. Kodierung und Dekodierung sind in Zeit $O(n^2)$ mit kleinem Sicherheitsparameter n und kleinen Konstanten in der O-Notation möglich. Dies entspricht der Laufzeit der verwendeten Polynommultiplikation im Polynomring $\mathbb{Z}_q[X]/(X^n - 1)$. Deshalb kann man asymptotisch sogar $O(n \log n)$ mittels der schnellen Fouriertransformation erreichen.

Die Effizienz ist ein Hauptnachteil der bestehenden PKKS, die für relevante Daten zu langsam sind. Z.B. kodiert RSA in Zeit $O(n^2)$ unter Verwendung kurzer Exponenten und dekodiert in $O(n^3)$. Um in der Praxis größere Datenmengen hinreichend schnell zu ver- und entschlüsseln, verschickt man deshalb zumeist einen geheimen Sessionkey mit Hilfe eines PKKS und verwendet diesen für ein symmetrisches Secret-Key Verfahren, mit dem die restliche Kommunikation zwischen beiden Partnern abläuft. Die bestehenden PKKS mit schneller Kodierung und Dekodierung in Zeit $O(n^2)$ wie McElice und GGH besitzen unpraktikabel große Schlüsselpaare der Länge $O(n^2)$. Bei NTRU haben öffentlichen Schlüssel die Länge $O(n \log q)$ und geheime Schlüssel die Länge $O(n)$. Damit ist das NTRU-PKKS für kleine n auch für SmartCard Anwendungen geeignet.

Die Kodier- und Dekodiereffizienz liegt hauptsächlich an dem zugrundeliegenden Polynomring $\mathbb{Z}_q[X]/(X^n - 1)$, auf dem die verwendeten arithmetischen Operationen schnell durchgeführt werden können. NTRU ist ein *ringbasiertes* PKKS, da es die beiden Ringoperationen Addition und Multiplikation verwendet. Damit unterscheidet es sich von den meisten gebräuchlichen System, die *gruppenbasiert* sind und nur Gruppenoperationen auf den Parametern zulassen. Der Vorteil am PKKS-Design ist die reichhaltigere arithmetische Struktur der Ringe, die effizientere Kodierung ermöglicht. Der Nachteil ist, daß über Ringstrukturen bisher weniger bekannt ist. Die Gruppentheorie ist ausgefeilter und Sicherheitsbeweise in Gruppen deshalb leichter möglich. Trotzdem könnte die Zukunft der modernen Public Key Kryptographie in *ringbasierten* PKKS liegen.

Das NTRU-Kryptosystem hat zwei offensichtliche Nachteile. Der erste ist seine Nachrichtenexpansionrate, der zweite seine Anfälligkeit gegenüber Gitterreduktionsattacken. Um das letzte Problem zu beseitigen, schlugen Hoffstein und Silverman [22] eine nichtkommutative Variante des NTRU-PKKS vor, die auf dem nichtkommutativen Polynomring der dihedralen Gruppe $\mathbb{Z}_q[X,Y]/(X^n-1,Y^2-1,XYXY-1)$ operiert. Dieses System wurde allerdings von Coppersmith [13] vollständig gebrochen. Die erfolgreichsten bekannten Angriffe auf NTRU mittels Gitterbasenreduktion (Einführung dazu in Kapitel 2) basierten auf der zuvor erwähnten Arbeit von Coppersmith und Shamir [14], deren Resultate wir in Kapitel 5 vorstellen und erweitern. Mit Hilfe dieser Gitter und der Verbesserung der Gitterbasenreduktionsalgorithmen von Schnorr et al [57, 58] wurde das NTRU-Kryptosystem zum ersten Mal in mittlerer Sicherheitsstufe gebrochen.

Die neuen Ergebnisse dieser Diplomarbeit, die verbesserte Gitterbasen zum Angriff auf das NTRU-PKKS vorschlägt, werden in Kapitel 6 vorgestellt. Dabei knüpfen wir an das Gitter L^{cs} von Coppersmith/Shamir an. Im wesentlichen liefern die neuen Gitter die folgenden Resultate:

Das Run-Gitter L^r: Wir zeigen, daß der kürzeste Vektor in L^{cs} aufgrund einer zyklischen Eigenschaft des Polynomringsrings $\mathbb{Z}[X]/(X^n-1)$ nicht eindeutig ist. Damit beträgt der Quotient (Gap) $c = \frac{\lambda_2}{\lambda_1}$ aus zweitkürzestem und kürzestem Vektor 1. Da wir zur Gitterbasenreduktion aber Approximationsalgorithmen verwenden, sollte dieser Gap so groß wie möglich sein. Das Run-Gitter L^r bricht die Symmetrie durch Fixieren eines Runs in den Vektoren und macht den kürzesten Vektor im Gitter eindeutig, vergrößert damit also den Gap c. Zusätzlich geben wir einen Algorithmus an, der die Gitterdimension um die Länge des Runs verringert.
Die Erfolgswahrscheinlichkeit des Angriffs hängt bei dieser Attacke von dem Bruchteil ϵ der Schlüssel mit Runlänge r ab, also von den Münzwürfen des Schlüsselerzeugers, der seine Schlüssel uniform aus der Schlüsselmenge auswählen muß. Durch Randomisierung der fixierten Koeffizienten erreichen wir eine Erfolgswahrscheinlichkeit von ϵ des Angreifers, die ausschließlich von seinen eigenen Münzwürfen abhängt. Der Angriff kann durch diese Methode ebenfalls parallelisiert werden.

Die dimensionsreduzierenden Gitter L^{red}: Wir zeigen, daß der gesuchte Targetvektor τ im Gitter L^r nicht nur eindeutig ist, sondern daß er auch der einzige mit Supremums-Norm 1 ist. Unter der heuristischen Annahme, daß der Targetvektor ebenfalls der kürzeste Vektor in der Euklidischen Norm ist, können wir Gitter mit kleinerer Dimension konstruieren und die Gitterbasenreduktions-Algorithmen damit beschleunigen.

Das balancierte Gitter L^b: Wir wissen von den geheimen Schlüsselpolynomen $f(X), g(X) \in \mathbb{Z}_q[X]/(X^n-1)$, daß sie nach NTRU-Spezifikation die Gleichungen $f(1) = 1$ und $g(1) = 0$ erfüllen. Dies wird mit Hilfe des Gitters L^b ausgenutzt, in dem nur solche Vektoren enthalten sind, deren korrespondierende Polynome den obigen Gleichungen genügen. Alternativ wird eine Gitterbasis vorgeschlagen, mit der zwar auch andere Vektoren linearkombiniert werden können, allerdings besitzen diese große Euklidische Norm.

Das Schlüsselgitter L^{key}: Hier werden zusätzliche, selbsterzeugte Schlüsselgleichungen dazu verwendet, die Anzahl kleiner Vektoren im Gitter zu verringern. Dabei wird die Gitterdimension nicht erhöht; lediglich die Länge der Basisvektoren steigt an. Die Hoffnung ist, durch weitere Schlüsselgleichungen den Gap c zu erhöhen.

Das symmetriebrechende Gitter L^{sym}: Bei diesem Ansatz wird versucht, die Symmetrie der Einsen und Minuseinsen im geheimen Schlüssel aufzubrechen. Da die Anzahl dieser Koeffizienten im Polynom ein öffentlicher Parameter ist, können ins Gitter integrierte Zähler diese Eigenschaft ausnutzen. Zusätzlich wird ein Trick analog zur CJLOSS-Basis [12] für Subset Sum Probleme verwendet, um die Norm des Zielvektors weiter zu verringern.

Mit Hilfe der neuen Methoden erhalten wir eine Cryptanalyse des NTRU-Kryptosystems in mittlerer Sicherheitsstufe $n = 107$. Benötigt man mit dem Coppersmith/Shamir Gitter L^{cs} noch ca. 1 Monat zum Brechen von NTRU, so verringert sich diese Zeit mit Hilfe der neuen Gitter auf ungefähr fünf Stunden (ca. 1 Stunde bei Parallelisierung). Die verbesserten Angriffsgitter könnten auch zum Brechen des NTRU-167 Challenge in hoher Sicherheitsstufe genügen, wenn man genügend Rechenzeit aufwendet. Instanzen mit Sicherheitsparameter $n = 167$ und „schwachen" Schlüsseln werden in Kapitel 6 gebrochen.

Man muß beachten, daß die neuen Attacken – auch wenn sie die bisherigen Methoden signifikant verbessern – keine Polynomialzeitalgorithmen zur Cryptanalyse von NTRU liefern. Die Laufzeiten bei experimentellen Resultaten deuten auf ein Exponentialzeitverhalten zum Brechen von NTRU-Schlüsseln bei steigendem Sicherheitsparameter hin. Der Laufzeitexponent kann aber bedeutend reduziert werden, weshalb man in der Praxis das n signifikant größer wählen muß, um Sicherheit gegenüber den neuen Attacken zu erzielen. D.h. obwohl das NTRU-Kryptosystem nicht vollständig gebrochen wird, verliert es seinen größten Vorteil gegenüber anderen Systemen: Die effiziente Kodierung und Dekodierung bei *kleinem* Sicherheitsparameter n.

Kapitel 2

Einführung in die Theorie der Gitter und Polynomringe

Wir stellen in Abschnitt 2.1 die wichtigsten Begriffe aus der Gittertheorie vor. Die Probleme Kürzester Gittervektor, Nächster Gittervektor und Minimale Basis werden definiert und ihre Komplexität betrachtet. Dabei werden wir sehen, daß die Sprachversionen dieser Probleme NP-hart sind. Sogar Approximationsvarianten werden bis auf einen Faktor als NP-hart nachgewiesen. Damit können wir unter der Komplexitätsannahme NP$\neq$P nicht erwarten, diese Probleme in Polynomialzeit exakt zu lösen.

Wir erhalten aber approximative Lösungen unter Verwendung der in Abschnitt 2.2 vorgestellten Gitterbasenreduktionsalgorithmen. Wir beschreiben den bekannten L^3-Reduktionsalgorithmus von Lenstra, Lenstra und Lovasz [29] und die Erweiterung zum BKZ-Algorithmus von Schnorr [54, 55].

In Abschnitt 2.3 betrachten wir Polynomringe. Wir definieren die Multiplikation, Division und Inversenbildung in diesen Ringen und betrachten im speziellen Eigenschaften des Polynomrings $\mathbb{Z}_q[X]/(X^n - 1)$, der im NTRU-Kryptosystem verwendet wird.

2.1 Gitter und Komplexität von Gitterproblemen

Wir stellen in diesem Abschnitt die benötigten Grundlagen und Definitionen aus der Theorie der Gitter und Reduktion von Gitterbasen vor. Das Kapitel beschränkt sich auf die im weiteren verwendeten Begriffe, für eine tiefergehende und ausführliche Einführung in die Grundlagen der Gittertheorie verweisen wir auf das Vorlesungsskript von C.P. Schnorr [52, 53]. Ein kompakte Zusammenfassung über die aktuellen Fortschritte bezüglich der Komplexität von Gitterproblemen findet sich bei J.-Y. Cai [9].

Ein Gitter ist eine diskrete, additive Untergruppe des $\mathbb{R}^m$. Eine alternative und anschaulichere Beschreibung eines Gitters mit Hilfe von Basisvektoren liefert die folgende Definition.

Definition 2.1.1 (Gitter) *Seien $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ linear unabhängige Vek-*

toren. Wir nennen die additive Untergruppe

$$L(b_1, b_2, \ldots, b_n) = \sum_{i=1}^{n} b_i \mathbb{Z} = \left\{ \sum_{i=1}^{n} t_i b_i \,|\, t_1, t_2, \ldots, t_n \in \mathbb{Z} \right\}$$

des $\mathbb{R}^m$ *ein Gitter mit der Basis* $b_1, b_2, \ldots, b_n$ *(das L steht für Lattice). Die Dimension oder der Rang des Gitters ist* $\dim(L) = n$.

Die Basis eines Gitters ist nicht eindeutig. Zwei verschiedene Basen können durch Multiplikation mit einer ganzzahligen Matrix mit Determinante ± 1 ineinander umgewandelt werden. Eine solche Matrix nennen wir unimodular. Mit der Cramer'schen Regel zeigt man, daß eine ganzzahlige Matrix genau dann ein ganzzahliges Inverses besitzt, wenn sie unimodular ist. Damit bilden die unimodularen Matrizen zusammen mit der Matrixmultiplikation eine Gruppe.

Definition 2.1.2 (Gitterdeterminante) *Sei* $b_1, b_2, \ldots, b_n$ *eine Basis des Gitters L. Dann ist die* Gitterdeterminante *definiert als die Determinante der Basismatrix* $B = [b_1, b_2, \ldots, b_n]$ *und wird mit* $\det(L)$ *bezeichnet.*

Man beachte: Da Basistransformationen unimodular sind, ist die Determinante eines Gitters L unabhängig von der Basisdarstellung. Diese Größe ist eine *Invariante* von L.
Wir definieren nun ein Parallelepiped, das von den Basisvektoren des Gitters aufgespannt wird; die Grundmasche.

Definition 2.1.3 (Grundmasche) *Sei* $b_1, b_2, \ldots b_n$ *eine Basis des Gitters L. Das folgende Parallelepiped heißt Grundmasche zur Basis* $b_1, b_2, \ldots, b_n$

$$\left\{ \sum_{i=1}^{n} t_i b_i \,|\, 0 \le t_i < 1 \right\}.$$

Betrachten wir das Standard-Skalarprodukt $\langle \cdot, \cdot \rangle : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ mit $\langle (x_1, \ldots, x_m), (y_1, \ldots, y_m) \rangle = \sum_{i=1}^{m} x_i y_i$. Dann bezeichne $\|x\| = \sqrt{\langle x, x \rangle}$ die Euklidische Länge bzw. ℓ_2-Norm des Vektors x. Die ℓ_p-Norm des Vektors x ist allgemein als $\sqrt[p]{\sum_{i=1}^{m} x_i^p}$ definiert. Das folgende Lemma liefert für das Standard-Skalarprodukt eine anschauliche Interpretation der Gitterdeterminante.

Lemma 2.1.4 *Für jedes Gitter* $L = L(b_1, \ldots b_n) \subset \mathbb{R}^m$ *und das Standardskalarprodukt gilt*

$$\det(L) = vol_n \left(\left\{ \sum_{i=1}^{n} t_i b_i \,|\, 0 \le t_i < 1 \right\} \right)$$

D.h. das n-dimensionale Volumen der Grundmasche ist gleich der Determinanten der Basismatrix.

Beweis: Wir beschränken uns hier auf den Fall vollständiger Gitter L, d.h. $Rang(L) = m$. Für die Basismatrix $B = [b_1, b_2, \ldots, b_n]$ gilt:

$$\det(L) = |\det(B)| = (\det(B^T B))^{\frac{1}{2}} = (\det[\langle b_i, b_j \rangle]_{1 \le i,j, \le n})^{\frac{1}{2}}. \quad \square$$

Definition 2.1.5 (Untergitter) *Eine Teilmenge $\bar{L} \subset L$ eines Gitters L heißt* Untergitter, *falls $\bar{L}$ ein Gitter ist.*

Sei $b_1, \ldots, b_n \in I\!R^m$ eine Gitterbasis. Dann betrachten wir informal folgende drei Gitterprobleme für die Euklidische Norm

- Kürzester Gittervektor: Finde kurzen Gittervektor ungleich dem Nullvektor.
- Nächster Gittervektor: Finde zu gegebenem $y \in I\!R^m$ einen möglichst nahen Gittervektor
- Minimale Basislänge: Finde eine Basis bestehend aus kurzen Gittervektoren.

Betrachten wir die Sprachversionen dieser algorithmischen Probleme und ihre algorithmische Komplexität.

Satz 2.1.6 (Shortest Vector Problem SVP (Ajtai '97 [2])) *Das Problem* Kürzester-Gittervektor

$$\ell_2 - SVP = \left\{ (b_1, b_2, \ldots, b_n, k) \;\middle|\; \begin{array}{l} b_1, b_2, \ldots, b_n \in I\!R^m, k \in I\!R \\ \exists v \in L(b_1, b_2, \ldots, b_n) \setminus \{0\} : \|v\| \leq k \end{array} \right\}$$

ist NP-hart unter randomisierten Reduktionen.

Der Beweis erfolgt durch randomisierte Reduktion vom eingeschränkten Subset-Sum Problem auf das SVP. D. Micciancio [35] zeigte, daß es NP-hart ist, den kürzesten Vektor bis auf eine Konstante kleiner $\sqrt{2}$ zu approximieren. Die Reduktion benutzt die NP-Hardness einer Variante des CVP.

Satz 2.1.7 (Closest Vektor Problem CVP (van Emde Boas '81 [60])) *Das Problem* Nächster Gittervektor

$$\ell_2 - CVP = \left\{ (b_1, b_2, \ldots, b_n, k, y) \;\middle|\; \begin{array}{l} y, b_1, b_2, \ldots, b_n \in I\!R^m, k \in I\!R \\ \exists v \in L(b_1, b_2, \ldots, b_n) \setminus \{0\} : \|v - y\| \leq k \end{array} \right\}$$

ist NP-hart.

I. Dinur, G. Safra und S. Safra [16] zeigten, daß die Approximation von CVP bis auf einen Faktor $2^{\log^{1-\epsilon} n}$ für $\epsilon = o(1)$ NP-hart ist.
Aufbauend auf diesem Resultat bewiesen Blömer und Seifert [7], daß die Probleme, eine Menge kürzester linear unabhängiger Vektoren und eine kürzeste Basis mit Verlustfaktor $n^{c/\log\log n}$ zu approximieren, NP-hart sind.

2.2 Gitterbasenreduktion

Im letzten Abschnitt haben wir gesehen, daß es für die Probleme SVP, CVP und Kürzeste-Basis unter der Annahme P$\neq$NP keine Polynomialzeit-Algorithmen gibt und daß auch Approximationsvarianten bis auf einen Faktor NP-hart sind. Hier wollen wir nun betrachten, was algorithmisch möglich ist. Um einen Maßstab für die Reduziertheit einer Basis zu erhalten, führen wir eine weitere Gitterinvariante ein; die sukzessiven Minima.

Definition 2.2.1 (sukzessive Minima) *Sei $\|\cdot\|$ eine beliebige ℓ_p-Norm. Zu einem Gitter $L \subset \mathbb{R}^m$ vom Rang n sind die sukzessiven Minima $\lambda_1, \lambda_2, \ldots, \lambda_n$ wie folgt definiert*

$$\lambda_i(L) = \inf \left\{ r > 0 \;\middle|\; \begin{array}{l} \textit{Es gibt } i \textit{ linear unabhängige} \\ \textit{Vektoren } v_1, v_2, \ldots, v_i \in L \\ \textit{mit } \|v_j\| \le r \textit{ für } j = 1, \ldots, i \end{array} \right\} \quad \textit{für } i = 1, \ldots, n$$

Für jede Gitterbasis $b_1, b_2, \ldots, b_n$ mit $\|b_1\| \le \|b_2\| \le \ldots \le \|b_n\|$ gilt für $i = 1, 2, \ldots, n$

$$\|b_i\| \ge \lambda_i.$$

Im allgemeinen bilden die kürzesten n linear unabhängigen Vektoren keine Basis, wie man es intuitiv erwarten könnte. Betrachte als Gegenbeispiel

$$L = \mathbb{Z}^n + \mathbb{Z}(\frac{1}{2}, \ldots, \frac{1}{2}).$$

Eine Basis von L bildet der Vektor $h = (\frac{1}{2}, \ldots, \frac{1}{2})$ zusammen mit den Einheitsvektoren e_i, $2 \le i \le n$, da $e_1 = 2h - \sum_{i=2}^n e_i$. Dagegen ist $\{e_1, e_2, \ldots, e_n\}$ keine Basis von L, weil h nicht zu $\mathbb{Z}^n$ gehört. Es gilt damit $\lambda_1(L) = \ldots = \lambda_n(L) = 1$. Definieren wir die Basislänge als

$$bl(L) = \min_{\text{alle Basen } b_1, \ldots, b_n \text{ für } L} \max_i \|b_i\|,$$

dann beträgt die minimale Basislänge $\sqrt{n}/2$ für $n > 4$.
Wir werden die Größen $\frac{\|b_i\|}{\lambda_i}$ als Maßstab für die Reduziertheit einer Gitterbasis verwenden. Sei L ein n-dimensionales Gitter im $\mathbb{R}^m$ mit Basis $\{b_1, b_2, \ldots, b_n\}$. Da die Translationen der Grundmasche eine „Kachelung" des $\mathbb{R}^m$ erzeugen, ist das Volumen $\det(L)$ ein gutes Maß für die Größe von L. Der erste Satz von Minkowski zeigt eine Beziehung zwischen kürzestem Gittervektor und der Gitterdeterminante.

Satz 2.2.2 (Minkowski 1896) *Es gilt*

$$\lambda_1(L) \le \gamma_n (\det(L))^{\frac{1}{n}},$$

wobei γ_n die Hermitekonstante vom Rang n ist.

Beweis: Betrachte das Gitter $L' = 2L$, eine Streckung von L um den Faktor 2 in alle Richtungen. Damit ist $\det(L') = 2^n \det(L)$. Betrachte weiterhin einen Ball vom Radius r um jeden Gitterpunkt von L. Sei ω_n das Volumen des n-dimensionalen Einheitsballs B_n mit Radius 1, dann ist $\omega_n r^n$ das Volumen des Balls $B_n(r)$ mit Radius r. Gilt nun $\omega_n r^n = \det(L')$, dann müssen zwei verschiedene Bälle überlappen. Damit gibt es zwei Gitterpunkt $l \neq l'$ in L mit $2l + x = 2l' + y$ für Punkte $x, y \in B_n(r)$. Wir erhalten aufgrund der Konvexität des Balls $B_n(r)$: $l - l' = (y - x)/2 \in B_n(r)$ und $l - l'$ ist der gesuchte von Null verschiedene Gitterpunkt von L. Es ist bekannt, daß $\omega_n = \pi^{n/2}/\Gamma(\frac{n}{2} + 1)$. Damit gilt

$$\lambda_1(L) \le l - l' \le r = \frac{2}{\sqrt{\pi}} \Gamma(\frac{n}{2} + 1)^{1/n} (\det(L))^{1/n} = \theta(\sqrt{n}) (\det(L))^{1/n}$$

und die Behauptung folgt. $\square$

Der Satz von Minkowski garantiert zwar die Existenz eines Gittervektors der Länge höchstens $\theta(\sqrt{n})(\det(L))^{1/n}$, er ist aber nicht-konstruktiv und liefert somit keinen Algorithmus, um einen solchen Vektor zu finden.

Der berühmte L^3- oder auch Lovasz-Algorithmus findet eine Approximation des kürzesten Vektors. In [29] werden die folgenden Eigenschaften gezeigt.

Satz 2.2.3 (Lenstra, Lenstra, Lovasz '82) *Gegeben eine Basis $\{b_1, \dots, b_n\}$ des Gitters L. Dann findet der L^3-Basisreduktions-Algorithmus eine neue Basis $\{b_1', \dots, b_n'\}$ mit*

1. $\|b_1'\| \leq 2^{\frac{n-1}{2}} \lambda_1(L)$
2. $\|b_1'\| \leq 2^{\frac{n-1}{4}} \sqrt[n]{\det(L)}$
3. $\|b_1'\| \cdot \dots \cdot \|b_n'\| \leq 2^{\frac{1}{2}\binom{n}{2}} \det(L)$

Wir wollen das Prinzip des L^3 nun skizzieren. Gegeben eine Basis $\{b_1, \dots, b_n\}$, betrachte das Orthogonalsystem $\{\hat{b}_1, \dots, \hat{b}_n\}$, das von der Gram-Schmidt Orthogonalisierung erzeugt wird:

$$\begin{aligned} \hat{b}_1 &= b_1 \\ \hat{b}_i &:= \pi_i(b_i) = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \hat{b}_j \quad \text{für } i = 2, 3, \dots, n \end{aligned}$$

Dabei sind die Gram-Schmidt-Koeffizienten $\mu_{i,j}$ erklärt durch

$$\mu_{i,j} = \frac{\langle b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle}$$

Wir bezeichnen den von den Vektoren $b_1, b_2, \dots, b_{i-1}$ aufgespannten linearen Raum mit $span(b_1, b_2, \dots, b_{i-1}) = \sum_{j=1}^{i-1} b_j \mathbb{R}$. Der Orthogonalvektor $\hat{b}_i = \pi_i(b_i)$ ist die Projektion des Vektors b_i auf das orthogonale Komplement $span(b_1, b_2, \dots, b_{i-1})^\perp = \{y \in \mathbb{R}^m \mid \langle y, b_j \rangle = 0 \text{ für } j = 1, 2, \dots, i-1\}$.

Der erste Teil des L^3-Algorithmus besteht nun aus einer ganzzahligen Adaption der Gram-Schmidt Orthogonalisierung; der sogenannten Längenreduktion.

Algorithmus Längenreduktion

EINGABE: Gitterbasis $b_1, b_2, \dots, b_n \in \mathbb{R}^m$

```
FOR i = 2, 3, ..., n DO
        FOR j = i - 1, i - 2, ..., 1 DO
                b_i := b_i - ⌈μ_{i,j}⌋ · b_j
        END
END
```

AUSGABE: Längenreduzierte Basis $b_1, b_2, \dots, b_n$

Nach Beendigung des Algorithmus gilt $|\mu_{i,k}| \leq 1/2$ für $k < i$. Geometrisch sind die durchgeführten Schritten offensichtlich. Was die Stärke des L^3-Algorithmus ausmacht, ist die Lovasz-Bedingung

$$\delta \cdot \|\hat{b}_{i-1}\|^2 \leq \|\hat{b}_i\|^2 + \mu_{i,i-1}^2 \cdot \|\hat{b}_{i-1}\|^2 \quad \text{für } i = 2, 3, \ldots, n$$

für den Reduktionsparameter δ, $\frac{1}{4} < \delta < 1$. Mit der orthogonalen Projektion

$$\pi_i : I\!R^m \to span(b_1, b_2, \ldots, b_{i-1})^{\perp}$$

kann man die Lovasz-Bedingung auch wie folgt schreiben

$$\delta \cdot \|\pi_i(b_i)\|^2 \leq \|\pi_i(b_{i+1})\|^2 \quad \text{für } i = 1, 2, \ldots, n-1.$$

Wir erklären diese Bedingung anhand Abbildung 2.1. Betrachte den linearen

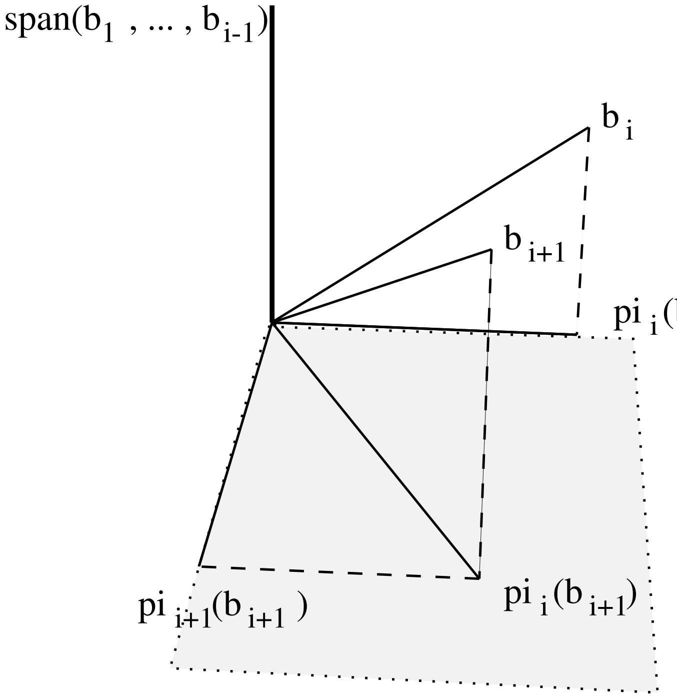

Abbildung 2.1: Geometrische Darstellung des Lovasz-Austauschs

Raum $span(b_1, \ldots, b_{i+1})$, dieser entspricht dem linearen Erzeugnis von $\{b_1, \ldots, b_{i-1}\} \cup \{\pi_i(b_i), \pi_i(b_{i+1})\}$. In $span(\pi_i(b_i), \pi_i(b_{i+1}))$ liegt das 2-dimensionale Gitter $L(\pi_i(b_i), \pi_i(b_{i+1}))$, die orthogonale Projektion von $L(b_1, \ldots, b_{i+1})$ auf $span(b_1, \ldots, b_{i-1})^{\perp}$. Wir können nun den Gaußschen Austauschschritt aus der 2-dimensionalen Gitterreduktionstheorie (siehe Schnorr [52], Kapitel 4) auf $\pi_i(b_i)$ und $\pi_i(b_{i+1})$ anwenden, falls $\|\pi_i(b_{i+1})\| < \|\pi_i(b_i)\|$. Dies entspräche einem Reduktionsparameter $\delta = 1$. Nun müssen wir aber garantieren, daß der Algorithmus schnell terminiert. Deshalb tauschen wir b_i und b_{i+1} nur falls $\|\pi_i(b_{i+1})\| < 1/\sqrt{\delta} \cdot \|\pi_i(b_i)\|$. Damit erzielen wir bei jedem Austauschschritt einen signifikanten Gewinn. Mit dem Vektortausch $b_i \leftrightarrow b_{i+1}$ vertauschen sich auch die Vektoren $\pi_i(b_i)$ und $\pi_i(b_{i+1})$, wie man in der Abbildung leicht sehen kann, und damit gilt die Lovasz-Bedingung nach dem Austausch.

Der L^3-Algorithmus besteht nun aus den beiden alternierend durchgeführten Schritten

- Längenreduktion der Basis
- Falls es ein i gibt, das die Lovasz-Bedingung verletzt, vertausche b_i mit b_{i+1}.

Satz 2.2.4 (Lenstra, Lenstra, Lovasz '82) *Sei $L \subset \mathbb{Z}^m$ ein Gitter mit Basis $b_1, b_2, \ldots, b_n$ und sei $B \in \mathbb{R}$ eine obere Schranke für die Quadratnorm der Basisvektoren, $\|b_i\|^2 \leq B$ für $1 \leq i \leq n$. Dann ist die Anzahl der Bitoperationen des L^3-Algorithmus $O(n^5 m \log B)$, und die Zahlen auf denen die Operationen ausgeführt werden besitzen eine binäre Länge von $O(n \log B)$.*

Es ist wichtig anzumerken, daß sowohl die gegebene worst-case Abschätzung der Laufzeit als auch die worst-case Approximationsfaktoren aus Satz 2.2.3 pessimistisch sind. In der Praxis werden deutlich bessere Resultate erreicht.

C.P. Schnorr [54, 55] verallgemeinerte den Begriff der starken HKZ- (Hermite, Korkine, Zolotareff 1850, 1877) und der L^3-reduzierten Basen zu den BKZ- oder β-reduzierten Basen. Alle bekannten Reduktionsalgorithmen zu HKZ-Basen haben exponentielle Laufzeit in der Gitterdimension. Für kleine Blockweiten β liefert der BKZ-Algorithmus (Block-Korkine-Zolotareff) dagegen mit dem L^3-Algorithmus vergleichbare Laufzeiten bei besseren Reduktionsergebnissen. Es ist derzeit offen, ob die BKZ-Algorithmen polynomielle Laufzeit haben.

Definition 2.2.5 (Schnorr '94 [57]) *Sei $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ eine Basis, $\beta \in \{2, 3, \ldots, n\}$ und δ mit $\frac{1}{4} < \delta < 1$ gegeben. Die Basis $b_1, b_2, \ldots, b_n$ heißt (β, δ)-reduziert, wenn:*

1. *$|\mu_{i,j}| \leq \frac{1}{2}$ für $1 \leq j < i \leq n$ (Längenreduziertheit)*
2. *$\delta \|\pi_i(b_i)\|^2 \leq \lambda_1(\pi_i(L(b_i, b_{i+1}, \ldots, b_{i+\beta-1})))^2$ für $i = 1, 2, \ldots, n$*

Die L^3-Reduktion ist ein Spezialfall der Blockreduktion (BKZ) mit $\beta = 2$. In diesem Fall entspricht Bedingung *2.* der Lovasz-Bedingung.

2.3 Polynomringe

Für eine umfassende Einführung in die Theorie der Polynome und Polynomringe verweisen wir auf die Lehrbücher von Bach/Shallit [5] und Cohen [11].

Wir nennen eine Menge R mit zwei Verknüpfungen $+ : R \times R \to R, (a, b) \mapsto a + b$ und $\cdot : R \times R \to R, (a, b) \mapsto a \cdot b$ einen *Ring*, wenn

- $(R, +)$ eine abelsche Gruppe ist,

- die Multiplikation assoziativ ist, d.h.

$$a(bc) = (ab)c \quad \forall a, b, c \in R$$

- und die Distributivgesetze gelten, d.h.

$$a(b+c) = ab + ac\,,\ (b+c)a = ba + ca \quad \forall a, b, c \in R.$$

Wir wollen uns im folgenden auf *kommutative Ringe mit Eins* beschränken und verlangen zu den bisherigen Ringaxiomen außerdem, daß

- die Multiplikation kommutativ ist und

- ein Element $1 \in R$ existiert, so daß

$$1 \cdot a = a \cdot 1 = a \quad \forall a \in R.$$

Der Polynomring $\mathbb{Z}[X]$ bestehe nun aus allen Polynomen der Form

$$f(X) = f_0 + f_1 X + \ldots + f_{n-1} X^{n-1} + f_n X^n \qquad f_i \in \mathbb{Z}, 0 \leq i \leq n$$

mit endlich vielen von Null verschiedenen Koeffizienten f_i. Sei X^n die größte Potenz von X im Polynom $f(X)$ mit einem von Null verschiedenen Koeffizienten f_n. Dann ist n der Grad von $f(X)$ und f_n heißt führender Koeffizient. Wir werden $f(X)$ oft in der Koeffizientenvektordarstellung $f(X) = (f_0, f_1, \ldots, f_n)$ schreiben. Die ℓ_p-Norm des Polynoms $f(X)$ wird als die ℓ_p-Norm seines Koeffizientenvektors definiert. Wenn klar ist, daß $f(X)$ ein Polynom ist, so schreiben wir auch einfach f.

Die Multiplikation von Polynomen erfolgt mit dem Standardkonvolutionsprodukt

$$(\sum_{i=0}^{m} a_i X^i)(\sum_{j=0}^{n} b_j X^j) = \sum_{k=0}^{n+m} c_k X^k \quad \text{mit } c_k = \sum_{i=0}^{k} a_i b_{k-i}.$$

Dabei gilt $a_i = 0$ für $i > m$ und $b_j = 0$ für $j > n$. Diese Methode benötigt $(m+1)(n+1)$ Multiplikationen und mn Additionen. Asymptotisch schnelle Multiplikationsalgorithmen wie die Fast Fourier Transformation werden in dieser Arbeit nicht betrachtet, da der Grad der verwendeten Polynome so klein ist, daß die Konvolution effizienter ist.

Betrachten wir Polynome $f(X) \in \mathbb{Z}_p[X]$, p prim, – d.h. die Koeffizienten sind aus dem Primkörper $\mathbb{Z}_p$ – so können wir auch eine Division für Polynome definieren, die sogenannte *Euklidische Division.*

Satz 2.3.1 (Euklidische Division) *Seien $a(X), b(X) \neq 0$ Polynome in $\mathbb{Z}_p[X]$. Dann gibt es Polynome $q(X)$ und $r(X)$ in $\mathbb{Z}_p[X]$ mit*

$$a(X) = q(X)b(X) + r(X) \quad \textit{mit } grad(r(X)) < grad(b(X))$$

Beweis: Für $grad(b(X)) > grad(a(X))$ gilt der Satz für $q(X) = 0$. Sei also $grad(a(X)) = n \geq grad(b(X)) = m$. Das führende Glied des Quotientenpolynoms $q(X)$ ist $a_n b_m^{-1} X^{n-m}$, denn es gilt

$$grad(a(X) - a_n b_m^{-1} X^{n-m} b(X)) < grad(a(X)).$$

Analog können vom linksstehenden Polynom weitere Vielfache von $b(X)$ abgezogen werden, bis der Grad des Restpolynom $r(X)$ kleiner als $grad(b(X))$ ist. □

Man beachte, daß der Beweis konstruktiv ist. Er liefert einen Algorithmus, um die Euklidische Division zweier Polynome zu berechnen. Damit können wir den gcd zweier Polynome $a(X), b(X)$ über dem Körper $\mathbb{Z}_p$ bestimmen.

1. Falls $b(X) = 0$, gib $a(X)$ aus und terminiere.

2. Sei $a(X) = q(X)b(X) + r(X)$ mit $grad(r(X)) < grad(b(X))$ die Euklidische Division von $a(X)$ mit $b(X)$. Setze $a(X) := b(X)$, $b(X) := r(X)$ und gehe zu Schritt 1.

Dieser Algorithmus kann leicht derart modifiziert werden (Erweiterter Euklidischer Algorithmus, z.B. [11]), daß er auf Eingabe $a(X)$, $b(X)$ Polynome $u(X)$, $v(X)$, $c(X)$ ausgibt mit

$$a(X)u(X) + b(X)v(X) = c(X) = \gcd(a(X), b(X)).$$

Betrachten wir nun den Restklassenring $R_q = \mathbb{Z}_q[X]/(X^n - 1)$, $q = p^k$, auf dem die Arithmetik im NTRU-Cryptosystem basiert, das im folgenden Kapitel vorgestellt wird. Für Kodierung und Dekodierung werden ausschließlich Polynommultiplikation und -addition verwendet. Dies erfolgt analog zu den vorgestellten Methoden. Es ist lediglich darauf zu achten, daß $X^n \equiv 1, X^{n+1} \equiv X$, usw.

Für die Schlüsselerzeugung müssen Inverse eines Polynoms $f(X)$ in R_q berechnet werden. Nun ist q eine Primzahlpotenz $q = p^r$, p prim. Wir berechnen zunächst mit Hilfe des Erweiterten Euklidischen Algorithmus über $R_p = \mathbb{Z}_p[X]$ Polynome $u(X), v(X), c(X)$ mit

$$f(X)u(X) + (X^n - 1)v(X) = c(X) = \gcd(f(X), X^n - 1).$$

Lemma 2.3.2 (Existenz von Inversen) *Ein Inverses von $f(X)$ in $R_p = \mathbb{Z}_p[X]/(X^n - 1)$, p prim, existiert genau dann, wenn $c(X) = \gcd((f(X), X^n - 1) \equiv 1 \pmod{p}$.*

Beweis: Wir betrachten alle Gleichungen in $\mathbb{Z}_p$. Sei $f^{-1}(X)$ Inverses von $f(X)$ in R_p. Damit ist $f(X)f^{-1}(X) \equiv 1 \pmod{X^n - 1}$, d.h. $f(X)f^{-1}(X) + (X^n - 1)v(X) = 1$ für ein $v(X) \in \mathbb{Z}[X]$. Es folgt $\gcd(f(X), X^n - 1) = 1$.
Existieren andererseits Polynome $u(X), v(X)$ mit $f(X)u(X) + (X^n - 1)v(X) = 1$, so gilt $f(X)u(X) \equiv 1 \pmod{X^n - 1}$. Das gesuchte Inverse von $f(X)$ ist das Polynom $u(X)$. □

Damit berechnet der Erweiterte Euklidische Algorithmus das Inverse $u(X) = f^{-1}(X)$ im Ring $R_p = \mathbb{Z}_p[X]/(X^n - 1)$, falls es existiert. Die Menge der invertierbaren Polynome

$$\{f(X) \in R \mid \exists u(X) : f(X)u(X) \equiv 1 \text{ in } R\}$$

eines Polynomrings R heißt auch Einheitenmenge und wird mit R^* bezeichnet. Sie bildet zusammen mit der Polynommultiplikation eine Gruppe.
Wir benutzen nun das folgende Lemma von Silverman [46], das $u(X)$ in ein Inverses von $f(X)$ im Ring R_{p^k} liftet.

Lemma 2.3.3 (Silverman '98 [46]) *Sei p prim und f(X) in R_p. Ist f in R_p^*, dann ist f auch eine Einheit in R_{p^k} für $k \geq 1$.*

Beweis: Da $f(X) \in R_p^*$, gibt es ein Polynom $u(X)$ mit $f(X)u(X) \equiv 1 \pmod p$. Wir konstruieren nun ein Inverses von $f(X)$ modulo größerer Exponenten von p induktiv. Setze $u_0(X) = u(X)$ und

$$u_{i+1}(X) = 2u_i(X) - f(X)u_i^2(X).$$

Dann gilt $f(X)u_i(X) \equiv 1 \pmod{p^{2^i}}$. Dies gilt für $i = 0$ nach Konstruktion. Angenommen es gelte für i, dann gilt $f(X)u_i(X) = 1 + p^{2^i}h(X)$ für ein $h \in \mathbb{Z}[X]$. Es folgt für den Induktionsschritt

$$\begin{aligned} f(X)u_{i+1}(X) &= f(X)(2u_i(X) - f(X)u_i^2(X)) = 1 - (1 - f(X)u_i(X))^2 \\ &= 1 - p^{2^{i+1}}h^2(X) \equiv 1 \pmod{p^{2^{i+1}}} \quad \square \end{aligned}$$

Wir geben nun eine notwendige Bedingung für die Invertierbarkeit eines Polynoms $f(X) \in R_q$.

Lemma 2.3.4 *Sei $f(X) \in R_q = \mathbb{Z}[X]/(X^n - 1)$ mit $f(1) \equiv 0 \pmod q$. Dann ist $f \in R_q \setminus R_q^*$. Insbesondere ist die Anzahl der Nullteiler $|R_q \setminus R_q^*|$ mindestens q^{n-1}.*

Beweis: Wegen $f(1) \equiv 0 \pmod q$ teilt $(X - 1)$ das Polynom $f(X)$ in $\mathbb{Z}_q[X]/(X^n - 1)$ (Notation: $(X - 1)|f(X)$). Es gilt aber $(X^n - 1) = (X - 1)(X^{n-1} + X^{n-2} + \ldots + X + 1)$. Deshalb gilt $(X - 1)|\gcd(f(X), X^n - 1)$ und damit ist nach Lemma 2.3.2 $f(X)$ nicht invertierbar.
Sei $f(X) = (f_0, f_1, \ldots, f_{n-2}) \in R_q$ in Koeffizientenschreibweise. Es gibt q^{n-1} viele Polynome mit $Grad(f) \leq n - 2$. Wähle

$$f_{n-1} \equiv -\sum_{i=0}^{n-2} f_i \pmod q.$$

Damit gilt $f(1) \equiv 0 \pmod q$ und $|R_q \setminus R_q^*| \geq q^{n-1}$. $\square$

Notwendige Bedingung für die Invertierbarkeit von $f(X)$ ist somit $f(1) \not\equiv 0 \pmod q$.

Kapitel 3

Das NTRU-Cryptosystem

Wir stellen in diesem Kapitel das NTRU-Cryptosystem [23] vor. Zunächst definieren wir den Polynomring, auf dem die Arithmetik in NTRU basiert. Die geheimen Schlüssel f und g werden aus einer Teilmenge des Rings gewählt, deren Polynome kleine Koeffizienten besitzen. Die Parameterwahl und die Erzeugung des öffentlichen Schlüssels h aus den geheimen Schlüsseln f und g wird in Abschnitt 3.2 erklärt.

In 3.3 betrachten wir das Polynomfaktorisierungsproblem (PFP), auf dem die Sicherheit des NTRU-Kryptosystems basiert. Dazu definieren wir *Faktorisierungen* des öffentlichen Schlüssels h im Polynomring. Dies sind Polynome f' und g', die der Identität $f' * h \equiv g'$ im Ring genügen. Wir werden sehen, daß die Anzahl der Faktorisierungen f', g' exponentiell im Sicherheitsparameter n ist. Von diesen Faktorisierungen lösen aber nur solche das PFP, deren Koeffizientenvektoren hinreichend kleine ℓ_2-Norm haben. Deshalb eignet sich das PFP sehr gut zur Reduktion auf ein Kürzestes Gittervektor Problem.

In Abschnitt 3.4 und 3.5 wird das Ver- und Entschlüsselungsprotokoll vorgestellt und die Korrektheit der Dekodierung gezeigt.

Abschnitt 3.6 befaßt sich mit der großen Nachrichtenexpansionsrate des NTRU-Cryptosystem. Für mittlere Sicherheit ist der verschlüsselte Text ca. 4-mal so groß wie der zugrundeliegende Klartext. Es wird das sogenannte Zwei-Wege-NTRU vorgeschlagen, das die Nachrichtenexpansion auf 2:1 senkt. Man weiß nicht, ob das Brechen von Zwei-Wege-NTRU äquivalent zum Brechen von NTRU ist. Falls NTRU gebrochen wird, dann ist Zwei-Wege-NTRU ebenfalls gebrochen. Eine Variante von Zwei-Wege-NTRU, genannt Turbo-NTRU, kann dazu verwendet werden, die Nachrichtenexpansion weiter zu verringern.

3.1 Notationen

$\mathbb{Z}_q[X]/(X^n-1)$ sei der Ring der Polynome $\mathbb{Z}_q[X]$, wobei $X^n=1$, d.h. wir betrachten den Polynomring modulo dem Polynom X^n-1. Die Koeffizienten der Polynome sind aus dem Ring $\mathbb{Z}_q$=$[-\lfloor\frac{q-1}{2}\rfloor, \lceil\frac{q-1}{2}\rceil]$. Dabei bezeichne $\lfloor x\rfloor$ die größte ganze Zahl $\leq x$, analog ist $\lceil x\rceil$ die kleinste ganze Zahl $\geq x$. Wir verwenden dieses um Null zentrierte Intervall $[-\lfloor\frac{q-1}{2}\rfloor, \lceil\frac{q-1}{2}\rceil]$ als Darstellung der Repräsentanten aus $\mathbb{Z}_q$ anstatt des üblichen Intervalls $[0, q-1]$, um zu gewährleisten, daß das Produkt zweier Polynome mit betragsmäßig kleinen Koeffizienten wieder ein Polynom ist, das kleine Koeffizienten hat. Der Modul q ist nicht notwendigerweise prim, d.h. $\mathbb{Z}_q$ ist im allgemeinen kein Körper. Wir definieren nun eine kanonische Repräsentantendarstellung der Polynome aus $\mathbb{Z}_q[X]/(X^n - 1)$.

Definition 3.1.1 (Koeffizientendarstellung, Norm) *Sei $f \in \mathbb{Z}_q[X]/(X^n-1)$, d.h. $f = \sum_{i=0}^{n-1} f_i X^i$ mit $f_i \in \mathbb{Z}_q$, dann stellen wir f als Koeffizientenvektor $f = (f_0, f_1, \dots, f_{n-1})$ dar. Die ℓ_p-Norm von f ist definiert als die ℓ_p-Norm des Koeffizientenvektors.*

Als Produkt $f * g$ zweier Polynome $f, g \in R_q = \mathbb{Z}_q[X]/(X^n - 1)$ des Polynomrings bezeichnen wir das folgende Standard-Konvolutions-Produkt:

Definition 3.1.2 (Konvolution) *Sei $h = f * g$, dann gilt für den k-ten Koeffizienten h_k von h:*

$$h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{n-1} f_i g_{n+k-i} = \sum_{\substack{i+j \equiv k \\ \pmod{n}}} f_i g_j \pmod{q}$$

Das Produkt zweier Polynome kennzeichnen wir mit einem Stern $*$, die Multiplikation von einem Skalar λ mit einem Polynom f mit einem Punkt: $\lambda \cdot f$ oder einfach λf.

Anmerkung : Die Berechnung des Produkts zweier Polynome erfolgt mit der Rechenvorschrift aus Definition 3.1.2 in Zeit $O(n^2)$. Asymptotisch kann die Multiplikation mit Hilfe der schnellen Fourier-Transformation auf $O(n \log n)$ gedrückt werden.
Wir werden zudem gewisse Polynome im Cryptosystem so wählen, daß die Polynommultiplikation auf Additionen in $\mathbb{Z}_q$ reduziert werden kann.

Definition 3.1.3 (Schlüsselmenge) *Mit $D(k, l) \subset \mathbb{Z}_q[X]/(X^n - 1)$ bezeichnen wir die Menge aller Polynome aus $\mathbb{Z}_q[X]/\ (X^n - 1)$, bei denen k Koeffizienten 1, l Koeffizienten -1 und $n - (k + l)$ Koeffizienten gleich 0 sind. Wählt man ein zufälliges Polynom aus $D(k, l)$ mit uniformer Wahrscheinlichkeitsverteilung, so schreiben wir $f \in_R D(k, l)$.*

Weiterhin benötigen wir zur Schlüsselerzeugung und zur Dekodierung Inverse in den Ringen R_q und R_p.

Definition 3.1.4 (Einheiten) *Es sei $R_q^* \subset R_q$ die multiplikative Einheitengruppe des Rings R_q. D.h. für alle $f \in R_q$ gibt es ein f_q^{-1}, so daß $f * f_q^{-1} \equiv 1$ in R_q. Wenn klar ist, in welchem Ring wir rechnen, schreiben wir auch kurz f^{-1} anstatt f_q^{-1}.*

3.2 Schlüsselerzeugung und Parameterwahl

Um das intuitive Verständnis des Lesers für die Idee der NTRU-Verschlüsselung zu verstärken, geben wir zunächst die Parameter und den zugehörigen Wert für die mittlere Sicherheitsstufe des Cryptosystems an, wie er von den NTRU-Autoren vorgeschlagen wird:

- öffentliche Parameter
  - Sicherheitsparameter $n = 107$
    Er bestimmt den maximalen Grad der Polynome im Ring. Alle verkommenden Polynome sind aus $\mathbb{Z}[X]/(X^n - 1)$.
  - zwei teilerfremde Module $p = 3$, $q = 64$
    Sie bestimmen die beiden Ringe $\mathbb{Z}_p[X]/(X^n - 1)$ und $\mathbb{Z}_q[X]/(X^n - 1)$, in denen die Arithmetik operiert.
  - zwei Polynomringe $R_q = \mathbb{Z}_q[X]/(X^n - 1)$ und $R_p = \mathbb{Z}_p[X]/(X^n - 1)$
- geheime Schlüssel
  - Secret-Key $f \in_R D(15, 14) \subset R_q$
    Wir fordern zusätzlich, daß f ein Inverses f_p^{-1} in $\mathbb{Z}_p[X]/(X^n - 1)$ und ein Inverses f_q^{-1} in $\mathbb{Z}_q[X]/(X^n - 1)$ besitzt. Diese Inversen werden mit Hilfe des Erweiterten Euklidischen Algorithmus für Polynome bestimmt. Sollte eine der Inversenberechnungen scheitern, wird ein neues f gewählt.
  - Schlüsselerzeugungs-Polynom $f_q^{-1} \in R_q$
    Das Polynom f_q^{-1} wird lediglich zur Berechnung des Public-Keys h benutzt. Es muß nicht gespeichert werden.
  - Dekodier-Polynom $f_p^{-1} \in R_p$
    Dieses Polynom wird zum Dekodieren benutzt. Es sollte daher gespeichert werden, um eine Neuberechnung aus f zu vermeiden.
  - das Polynom $g \in_R D(12, 12)$
    g verdeckt f_q^{-1} (und damit auch f) im öffentlichen Schlüssel. Man beachte, daß g ebenso wie f ein Polynom mit kleinen Koeffizienten ist.
- öffentlicher Schlüssel
  - Public-Key $h \in R_q$
    Berechnung von h aus den geheimen Schlüsseln:
    $$h \equiv f_q^{-1} * g \pmod{q}$$

Die vorgeschlagene Größe der Parameter in allen Sicherheitsstufen ist in Tabelle 3.1 aufgeführt.

Die Parameter wurden so gewählt, daß die Komplexität von Meet-in-the-Middle Attacken (s. Abschnitt 4.2) auf den Public-Key eine Komplexität von respektive 2^{50}, 2^{83} und 2^{285} aufweisen. Nachdem die Autoren von den neuen Gitterangriffen aus dieser Arbeit erfuhren (siehe auch [30, 31]), setzten sie die Systemparameter höher [48].

Sicher-heit	n	q	p	D_f	D_g	D_ϕ	geheim (bit)	öffentl. (bit)
mittel	107	64	3	$(15,14)$	$(12,12)$	$(5,5)$	340	642
hoch	167	128	3	$(61,60)$	$(20,20)$	$(18,18)$	530	1169
höchst	503	256	3	$(216,215)$	$(72,72)$	$(55,55)$	1595	4024

Tabelle 3.1: vorgeschlagene Systemparameter von NTRU

3.3 Zugrundeliegendes Problem

Wie wir sehen werden, besitzen die Elemente des Ring $R_q = \mathbb{Z}_q[X]/(X^n - 1)$ keine eindeutige Faktorisierung. Wir definieren die Faktorisierung eines Polynoms h wie folgt.

Definition 3.3.1 (Faktorisierung) *Sei $h \in R_q$. Dann heißen die Polynome f' und g' eine* Faktorisierung *von h in R_q, wenn sie die Identität $f' * h \equiv g'$ in R_q erfüllen.*

Die Sicherheit von NTRU basiert auf der folgenden Komplexitätsannahme.

Annahme 3.3.2 (Polynomfaktorisierungs-Problem (PFP)) *Gegeben ein Polynom $h = f^{-1} * g$ in $R_q = \mathbb{Z}_q[X]/(X^n - 1)$, wobei die Koeffizienten der geheimen Schlüssel f und g klein gegenüber q sind. Für hinreichend große n ist es schwer, eines der beiden Polynome f oder g aus h zu rekonstruieren oder eine Faktorisierung f', g' von h mit $\|(f',g')\| \leq \|(f,g)\|$ und $f' \in R_q^*$ zu finden.*

Falls das *Polynomfaktorisierungs-Problem* leicht zu berechnen ist, so ist das Cryptosystem unsicher. Man beachte, daß der Umkehrschluß nicht gültig ist; aus der Schwierigkeit des PFP's folgt nicht die Sicherheit von NTRU. Es gibt auch andere Wege, NTRU-kodierte Texte zu entschlüsseln, ohne den Public-Key h geeignet zu faktorisieren (z.B. durch Raten von ϕ, siehe Abschnitt 3.4).

Man kennt keine Aussagen über die Schwierigkeit des *Polynomfaktorisierungs-Problem* in der Komplexitätstheorie. Die Hoffnung, daß es schwierig sein könnte, beruht vielmehr auf einem heuristischen Argument:
Jedes Polynom $f' \in R_q$, das modulo q teilerfremd zu $X^n - 1$ ist, hat ein Inverses in R_q und „teilt“ somit h in R_q, d.h. definiert eine Faktorisierung f', g' mit $f' * h \equiv g'$. Das Problem ist nun, unter $|R_q^*|$ vielen möglichen Faktorisierungen solche zu finden, die den Bedingungen von Annahme 3.3.2 genügen. Man beachte, daß $|R_q^*|$ von der Größenordnung q^n ist (für eine Schranke siehe [46]). Bis heute kennt man keine Polynomialzeit-Algorithmen, die dieses Problem lösen. Da die geheimen Polynome f und g aber beide kleine ℓ_2-Norm besitzen, eignen sich Gitterreduktionsmethoden sehr gut zum Angriff auf den Public-Key.

3.4 Ver- und Entschlüsselung einer Nachricht m

Angenommen Alice möchte eine NTRU-verschlüsselte Nachricht an Bob schikken. Dann benutzt sie das folgende Protokoll:
Gegeben:

- öffentliche Systemparameter n, q, p, R_q, R_p
- Public-Key h (von Bob)
- Nachricht $m \in R_p$

Verschlüsselungsprotokoll:

1. Wähle ein Randomisierungspolynom $\phi \in_R D(k,k)$
   Der Parameter k ist abhängig von der Sicherheitsstufe. Für mittlere Sicherheit schlagen die NTRU-Autoren $k = 5$ vor.
2. Berechne :
   $$e \equiv p\phi * h + m \pmod q \tag{3.1}$$
3. Schicke den Ciphertext $e \in R_q$ an Bob.

Anmerkung: Da der Plaintext m in R_p und der Ciphertext e in R_q ist, haben wir eine Nachrichtenexpansionsrate $\frac{|Ciphertext|}{|Plaintext|} = \frac{\log q}{\log p}$.

Erhält Bob den Ciphertext e von Alice, entschlüsselt er ihn mit seinen privaten Schlüsseln f und f_p^{-1} wie folgt:
Gegeben:

- öffentliche Systemparameter n, q, p, R, R_p
- Ciphertext $e \in R$
- Secret-Key f und f_p^{-1} (von Bob)

Entschlüsselungsprotokoll:

1. Berechne:
   $$\begin{aligned} a &\equiv f * e \pmod q \\ &\equiv p\phi * f * h + f * m \pmod q \\ &\equiv p\phi * f * f_q^{-1} * g + f * m \pmod q \\ &\equiv p\phi * g + f * m \pmod q \end{aligned}$$
   Für geeignete Wahl der Parameter kann man mit hoher Wahrscheinlichkeit sicherstellen, daß die Koeffizienten von $p\phi * g + f * m$ im Intervall $[-\lfloor\frac{q-1}{2}\rfloor, \lceil\frac{q-1}{2}\rceil]$. liegen. Man beachte, daß dazu sowohl f als auch g kleine ℓ_2-Norm besitzen müssen. D.h. wir erhalten:
   $$a \;=\; p\phi * g + f * m \quad \textbf{in } \mathbb{Z}[X]/(X^n - 1) \textbf{ exakt} \quad \text{(anstatt modulo } q\text{)}$$
2. Berechne:
   $$\begin{aligned} f_p^{-1} * a &\equiv f_p^{-1} * (p\phi * g + f * m) \pmod p \\ &\equiv f_p^{-1} * f * m \pmod p \\ &\equiv m \pmod p \end{aligned}$$

3.5 Korrektheit

Wir müssen nun zeigen, daß der Entschlüsselungsprozeß mit hoher Wahrscheinlichkeit korrekt arbeitet. Dazu führen wir zunächst einige Notationen ein:

Definition 3.5.1 *Sei $f = (f_0, \ldots, f_{n-1}) \in R_q$, dann ist die* nullzentrierte ℓ_∞-Norm*:*

$$\| f \|_\infty^\perp = \max_{0 \le i \le n-1} \{f_i\} - \min_{0 \le i \le n-1} \{f_i\}$$

Definition 3.5.2 *Sei $f = (f_0, \ldots, f_{n-1}) \in R$, dann ist die* nullzentrierte ℓ_2-Norm:

$$\| f \|_2^\perp = \Big(\sum_{i=0}^{n-1} (f_i - \bar{f})^2 \Big)^{1/2}, \quad \textit{wobei } \bar{f} = \frac{1}{n} \sum_{i=0}^{n-1} f_i$$

Anmerkung: Die nullzentrierte ℓ_∞-Norm und die nullzentrierte ℓ_2-Norm sind nur Halbnormen, da sie nicht nur 0 für den Nullvektor $(0, \ldots, 0)$, sondern auch für jeden konstanten Vektor $(c, \ldots, c)$, $(c \in \mathbb{R})$ liefern. Geometrisch entspricht die zentrierte ℓ_2-Norm der ℓ_2-Norm der orthogonalen Projektion des Vektors f auf den Vektor $(1, \ldots, 1)$.

Lemma 3.5.3 (Normabschätzung HPS '98 [24]) *Für jedes $\epsilon > 0$ gibt es Konstanten $\gamma_1, \gamma_2 > 0$ in Abhängigkeit von ϵ und n, so daß zwei zufällig mit uniformer Verteilung gewählte Polynome $f, g \in R = \mathbb{Z}[X]/(X^n - 1)$ mit Wahrscheinlichkeit größer als $1 - \epsilon$ die folgende Ungleichung erfüllen:*

$$\gamma_1 \| f \|_2^\perp \| g \|_2^\perp \le \| f * g \|_\infty^\perp \le \gamma_2 \| f \|_2^\perp \| g \|_2^\perp \tag{3.2}$$

Lemma 3.5.4 (Entschlüsselungskriterium HPS [24]) *Damit der Entschlüsselungsprozeß korrekt arbeitet, muß $a = p\phi * g + f * m$ in $\mathbb{Z}[X]/(X^n - 1)$ sein, d.h. wir benötigen*

$$\|p\phi * g + f * m\|_\infty^\perp < q,$$

um das korrekte a zu ermitteln (eventuell durch Addieren von $(t, \ldots, t)$,$t = \pm 1, \pm 2, \ldots$).

Dieses Kriterium wird mit vernachlässigbar kleiner Fehlerwahrscheinlichkeit erfüllt sein für

$$\|p\phi * g\|_\infty^\perp \le q/4 \quad \text{und} \quad \|f * m\|_\infty^\perp \le q/4,$$

so daß die Autoren mit Lemma 3.5.3 die folgende Parameterwahl vorschlagen

$$\|\phi\|_2^\perp \|g\|_2^\perp \approx q/(4p\gamma_2) \quad \|f\|_2^\perp \|m\|_2^\perp \approx q/(4\gamma_2).$$

3.6 Verbessern der Nachrichtenexpansion

Ein Hauptnachteil des NTRU-Cryptosystems ist die Nachrichtenexpansionsrate von $\frac{\log q}{\log p} = \log_p q$ zu 1 (Dies ist 3,79:1 für die mittlere Sicherheitsstufe). Um dies zu verbessern, haben die Autoren von NTRU das folgende System vorgeschlagen.

Zwei-Wege-NTRU

Verschlüsselung:

1. Wähle ein Polynom ψ uniform aus $R_p[X]/(X^n - 1)$. Dieses tritt an die Stelle der Nachricht m.

2. Berechne : $e \equiv p\phi * h + \psi \pmod q$

3. Berechne : $E \equiv \psi * e + m \pmod q$, wobei $m \in \mathbb{Z}_q[X]/(X^n - 1)$

4. Verschicke den Ciphertext (e, E).

Entschlüsselung:

1. Berechne ψ mittels NTRU-Dekodierung

2. Berechne : $m \equiv \psi * e - E \pmod q$

Bemerkung: Für den Nachrichtenraum gilt nun $L_m = Z_q^n$. Daher beträgt die Nachrichtenexpansion $\frac{Länge(e)+Länge(E)}{Länge(m)} = 2$.

Offenbar folgt aus der Korrektheit der NTRU-Verschlüsselung die Korrektheit von Zwei-Wege-NTRU. Ebenfalls impliziert die Sicherheit von Zwei-Wege-NTRU die Sicherheit des ursprünglichen Systems. Die Rückrichtung folgt nicht offensichtlich, da der Angreifer zusätzliche Information aus dem Tupel (e, E) gewinnen könnte.

Zwei-Wege-NTRU bietet einen weiteren Vorteil. Es verhindert durch die Einführung der zusätzlichen Zufallskomponente ψ eine Multiple-Transmission-Attacke (siehe Abschnitt 4.3).

3.6.1 Turbo-NTRU

Die NTRU-Autoren schlagen auf der Basis von Zwei-Wege-NTRU eine Erweiterung zu „Turbo-NTRU“ vor, das für die Übertragung von k aufeinanderfolgenden Nachrichtenblöcken nur eine Nachrichtenexpansion von $\frac{k+1}{k}$ besitzt. Für das Schema benötigen wir eine einfach zu berechnende, nichtlineare Funktion B. Hoffstein und Silverman schlagen vor :

$$\begin{aligned} &B : R_p[X]/(X^n - 1) \to R_2[X]/(X^n - 1) \\ &B(m) = m^2 \pmod 2 \end{aligned}$$

Seien nun $m_1, m_2, m_3, \ldots, m_k$ die zu übertragenden Nachrichtenblöcke.

Turbo-NTRU
Verschlüsselung:

$$\begin{aligned} e &= p\phi * h + \psi \pmod q \\ e_1 &= \psi * h + m_1 \pmod q \\ e_2 &= B(m_1) * h + m_2 \pmod q \\ e_3 &= B(m_2) * h + m_3 \pmod q \\ &\vdots \quad \vdots \end{aligned}$$

Ausgabe: Ciphertext $(e, e_1, e_2, \ldots, e_k)$
Entschlüsselung:

- Berechne ψ mittels NTRU-Dekodierung.
- Berechne : $m_1 = e_1 - \psi * h \pmod q$
- Für $2 \leq i \leq k$ berechne: $m_i = e_i - B(m_{i-1}) * h \pmod q$

Die Autoren erwähnen, daß Turbo-NTRU einen großen Nachteil besitzt: Falls während der Übertragung des i-ten Blocks ein Übertragungsfehler auftritt, können die nachfolgenden Blöcke nicht mehr entziffert werden. Der $(i+1)$-te Block hängt vom i-ten Block durch das Polynom $B(m_i)$ ab.

Was in der Originalarbeit unerwähnt bleibt, ist die Tatsache, daß Turbo-NTRU kein Public-Key-Verschlüsselungsprotokoll ist. Was de facto kodiert wird, ist ein geheimer Secret-Key ψ, dessen Kenntnis die Gleichung $e_1 = \psi * h + m_1 \pmod q$ mit den Unbekannten ψ und m_1 eindeutig löst. Der nächste Secret-Key ist dann $B(m_1)$. Warum eine Anwendung der nicht-linearen Funktion B notwendig ist, wird von Hoffstein und Silverman nicht erläutert. Ebenso könnte man m_1 verwenden oder bei Abwandlung auch stets denselben Secret-Key ψ. Deshalb schlagen wir ein vereinfachtes Übertragungsprotokoll vor:

vereinfachtes Turbo-NTRU
Verschlüsselung:

$$
\begin{aligned}
e_0 &= p\phi * h + \psi \pmod q \\
e_1 &= \psi * e_0 + m_1 \pmod q \\
e_2 &= \psi * e_1 + m_2 \pmod q \\
e_3 &= \psi * e_2 + m_3 \pmod q \\
\vdots &\quad \vdots
\end{aligned}
$$

Ausgabe: Ciphertext $(e_0, e_1, e_2, \ldots, e_k)$
Entschlüsselung:

- Berechne ψ mittels NTRU-Dekodierung.
- Für $1 \leq i \leq k$ berechne: $m_i = e_i - \psi * e_{i-1} \pmod q$

Bemerkung 1: Vorausgesetzt es tritt kein Übertragungsfehler beim 0-ten Block auf, dann hängt die Korrektheit der Dekodierung des i-ten Blocks ausschließlich vom $(i-1)$-ten Block ab. Tritt ein Übertragungsfehler im $(i-1)$-ten Block auf, so können der $(i+1)$-te und nachfolgende Blöcke trotzdem korrekt dekodiert werden.
Bemerkung 2: Das vereinfachte Turbo-NTRU ist schneller, da es die Berechnung von B nicht benötigt.

Kapitel 4

Angriffe auf NTRU

In diesem Kapitel stellen wir bekannte Attacken auf das NTRU-Kryptosystem vor, die nicht gitterbasiert sind. In Abschnitt 4.1 berechnen wir die Größe des Schlüsselraums der geheimen Schlüssel. Die Brute-Force Attacke zählt Schlüssel aus diesem Raum auf, bis sie die geheimen Schlüssel zur Lösung des Polynomfaktorisierungsproblems (PFP) findet.

Die Meet-In-The-Middle Attacke von Odlyzko aus Abschnitt 4.2 splittet dagegen das geheime Polynom f in zwei Teilpolynome f_1 und f_2 auf und sucht Lösungen des PFPs für diese Teilpolynome. Die Größe des Schlüsselraums der Teilpolynome f_1 und f_2 beträgt nur die Wurzel der Größe des Schlüsselraums für f. Damit ist die Laufzeit des Meet-In-The-Middle Algorithmus von der Größenordnung der Wurzel der Laufzeit der Brute-Force Methode.

In Abschnitt 4.3 stellen wir einen Angriff auf eine verschlüsselte Nachricht m vor, die mehrfach mit NTRU kodiert und verschickt wird. Diese Attacke rekonstruiert nicht die geheimen Schlüssel aus dem öffentlichen Schlüssel.

Sogenannte „Adaptive Chosen Ciphertext Attacken“ werden in Abschnitt 4.4 betrachtet. Diese Attacken arbeiten nur in einem speziellen Sender-Empfänger Szenario. Die Idee ist, manipulierte Ciphertexte an den Empfänger zu verschicken. Der Sender erfährt, ob die Nachrichten vom Empfänger als gültig akzeptiert oder verworfen werden. Daraus erhält er Informationen über die Nachricht m (Abschnitt 4.4.1) oder das Randomisierungspolynom ϕ (Abschnitt 4.4.2).

4.1 Brute-Force Attacke

Ein Angreifer kann den geheimen Schlüssel f rekonstruieren, indem er alle möglichen $f' \in D(d_f+1, d_f)$ aufzählt und testet, ob $f' * h \pmod q$ ein Polynom g' aus $D(d_g, d_g)$ ist. Äquivalent dazu kann er alle möglichen Schlüssel g' aus $D(d_g, d_g)$ aufzählen und überprüfen, ob $g' * h^{-1} \pmod q$ ein Polynom $f' \in D(d_f+1, d_f)$ ist.
Hierbei kann man für h^{-1} modulo q ein Pseudoinverses von h in R_q benutzen. Wegen $g \in D(d_g, d_g)$ gilt $g(1) = 0$, d.h. $g = (X-1) \cdot \mathit{Restpolynom}$ und somit $(X-1) | ggT(g, X^n - 1)$. Damit ist $g \in R_q \setminus R_q^*$ und somit $h \in R_q \setminus R_q^*$ wegen $h \equiv f^{-1} * g$.
Da für die vorgeschlagenen Sicherheitsparameter stets $d_g < d_f$ gilt und deshalb

die Größe $|D_g|$ des Sampleraums kleiner als $|D_f|$ ist, erhalten wir:

$$Time(Brute\!-\!Force) = |D_g| = \binom{n}{d_g\,;\,d_g} = \binom{n}{d_g}\cdot\binom{n-d}{d_g} = \frac{n!}{(d_g!)^2\cdot(n-2d_g)!}$$

(Wir wählen zunächst d_g aus n Positionen für die Einsen in g und danach d_g aus den verbleibenden $n - d_g$ Positionen für die Minuseinsen).

4.2 Odlyzko's Meet-In-The-Middle Attacke

Dieser von Andrew Odlyzko [41] vorgeschlagene Angriff reduziert die Laufzeit auf die Wurzel der Brute-Force Methode; d.h. hat der Sample-Raum des privaten Schlüssels f die Größe 2^n, so beträgt der Sicherheitslevel des Schlüssels $O(2^{\frac{n}{2}})$. Obwohl die Laufzeit exponentiell im Sicherheitsparameter n ist, und der Angriff somit für größere n nicht mehr relevant ist, muß man für den vorgeschlagenen mittleren Sicherheitslevel auch diese Attacke betrachten. Der Meet-In-The-Middle Angriff ist der einzig bekannte auf das NTRU-Cryptosystem, der keine weiteren Annahmen (Verschicken derselben Nachricht mehrere Male, Senden der Nachricht in einem digitalen Umschlag, etc.) macht und dessen Laufzeit beweisbar besser ist als eine Brute-Force Attacke. Die später vorgestellten Gitterreduktionsattacken sind in der Praxis allerdings wesentlich effizienter als die Odlyzko-Attacke. Das Interessante an dieser Meet-In-The-Middle Attacke ist die Art und Weise, wie die Laufzeitschranke erreicht wird.

Der Angriff, so wie er von Odylzko und Silverman beschrieben wird, bezieht sich auf eine frühere Version von NTRU. Nachfolgend werden die entsprechenden Anpassungen für die in Kapitel 3 gegebene NTRU-Spezifikation gemacht.

Idee des Algorithmus Meet-In-The-Middle-Attacke:
Zerlege das Polynom $f \in D(d, d)$ in zwei Polynome $f_1, f_2 \in D(\frac{d}{2}, \frac{d}{2})$. Wir wissen:

$$\begin{aligned} f * h &\equiv g \pmod q \\ \Leftrightarrow (f_1 + f_2) * h &\equiv g \pmod q \\ \Leftrightarrow f_1 * h &\equiv -f_2 * h + g \pmod q \end{aligned}$$

Da g ein Polynom mit kleinen Koeffizienten ist, gilt $f_1 * h \approx -f_2 * h \pmod q$. Im folgenden nehmen wir stets an, daß n und d gerade sind und $f \in D(d, d)$ (d.h f besitzt genauso viele Einsen wie Minuseinsen), um Gaußklammern zu vermeiden. Die notwendigen Änderungen für ungerade Parameter sind offensichtlich.

Algorithmus Meet-In-The-Middle Attacke:

EINGABE: Sicherheitsparameter n,
Public-Key $h \equiv f^{-1} * g \pmod q$

1. (Parameterwahl für Besetzungsrate)
   Wähle k und l so, daß gilt:

$(\frac{q}{2^l})^k \approx 100\binom{n/2}{d/2\,;\,d/2}$
// Bedeutung wird bei der Laufzeitanalyse klar.

2. (Aufsplitten von f)
Wähle zufällig eine Indexmenge $I \subset \{1,\ldots,n\}, |I| = \frac{n}{2}$.
// Zerlege $f = f_1 + f_2$, wobei $(f_1)_i = \begin{cases} f_i & \text{für } i \in I \\ 0 & \text{sonst} \end{cases}$
// bzw. $(f_2)_i = \begin{cases} f_i & \text{für } i \in \{1,\ldots,n\} \setminus I \\ 0 & \text{sonst} \end{cases}$

3. (Enumeration von f_1)
Zähle alle $\binom{n/2}{d/2\,;\,d/2}$ f_1-Vektoren auf.
Berechne die ersten k Koeffizienten von $f_1 h \pmod q$. Seien dies $(a_1,\ldots,a_k)$. Splitte das Intervall $[0, q-1]$ in Unterintervalle der Länge 2^l (mit $2^l|q$). Wir bezeichnen die Menge der Intervalle mit I:
$I_j = [2^l(j-1), 2^l j - 1]$ mit $1 \leq j \leq q/2^l$
Ein Topf sei ein k-Tupel von Intervallen aus I. Packe f_1 in den Topf
$(I_1,\ldots,I_k)$ mit $a_i \in I_i$ für $1 \leq i \leq k$
// In diesem Algorithmus ist ausnahmsweise $\mathbb{Z}_q = [0, q-1]$,
// um die Notation zu vereinfachen.

4. (Enumeration von f_2)
Zähle alle $\binom{n/2}{d/2\,;\,d/2}$ f_2-Vektoren auf.
Berechne die ersten k Koeffizienten von $-f_2 h \pmod q$.
Verteile die Vektoren f_2 analog zu Schritt 3 in die Töpfe $(J_1,\ldots,J_k)$, wobei :
$J_i = [2^l(j-1)+1, 2^l j - 2]$ mit $1 \leq j \leq q/2^l$.
// Beachte: Die Töpfe sind an beiden Intervallgrenzen um 1 kleiner,
// damit man durch Addition von g noch im Intervall
// $[2^l(j-1), 2^l j - 1]$ ist.
// Landet ein Koeffizient zwischen den Töpfen J_i und J_{i+1}, so
// sortiere ihn in beide Töpfe ein.

5. (Topfvergleich)
Falls die beiden Töpfe
$(I_{j_1},\ldots,I_{j_k})$ und $(J_{j_1},\ldots,J_{j_1})$
nichtleer sind, so berechne die entsprechenden Produkte $(f_1+f_2)h$.
Liefert diese Berechnung ein Polynom $g \in D(d_g, d_g)$, dann stoppe.

AUSGABE: $f = f_1 + f_2$, g

Analyse des Algorithmus:
Korrektheit: Falls wir in Schritt 2 eine Zerlegung des Originalschlüssels in zwei Polynome $f_1, f_2 \in D(\frac{d}{2}, \frac{d}{2})$ finden, so werden diese Polynome in korrespondierende Töpfe $(I_{j_1},\ldots,I_{j_k})$ und $(J_{j_1},\ldots,J_{j_k})$ in den Schritten 3 und 4 einsortiert und im 5. Schritt durch Topfvergleich gefunden.
Laufzeit: Entscheidend für die Laufzeit ist die Besetzungsrate der Töpfe. Diese gibt an, welcher Bruchteil der Töpfe einen Schlüssel f_1 bzw. f_2 besitzen. Man beachte, daß die Anzahl der Töpfe gleich $|I^k| = (q/2^l)^k$ beträgt, da wir $[0, q-1]$ in $(q/2^l)$ Intervalle der Länge 2^l aufteilen und I^k ein k-Tupel dieser Intervalle

repräsentiert. Durch Schritt 1 sichern wir:

$$\text{Besetzungsrate } = \frac{\binom{n/2}{d/2\,;\,d/2}}{|I^k|} = \frac{\binom{n/2}{d/2\,;\,d/2}}{|J^k|} = \frac{\binom{n/2}{d/2\,;\,d/2}}{(q/2^l)^k} \approx 1\%$$

D.h. circa 99 % der f_1's werden in Töpfen sein, deren korrespondierender f_2-Topf leer ist. Diese f_1 brauchen nicht betrachtet zu werden. Die restlichen f_1 besitzen einen nichtleeren f_2-Topf, aber fast immer wird dieser Topf nur einen einzigen Vektor f_2 enthalten. Für dieses Paar (f_1, f_2) prüfen wir, ob $(f_1 + f_2)h \pmod q$ kleine Koeffizienten hat. Wir ignorieren hier den Effekt, daß manche f_2 in zwei Töpfe einsortiert werden. Die Intervallgröße 2^l sollte so gewählt werden, daß dieser Effekt vernachlässigbar ist.

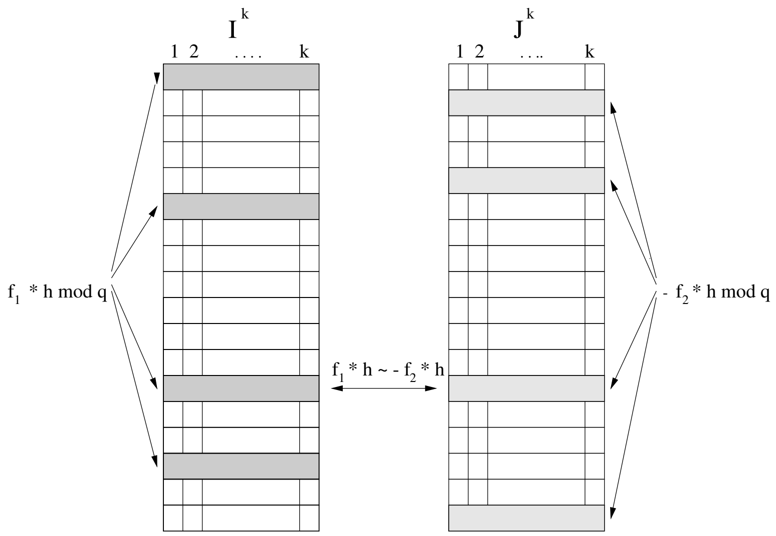

Abbildung 4.1: Meet-In-The-Middle Algorithmus

Wir erhalten die Laufzeiten:

1. Schritt: $O(1)$
2. Schritt: Wir müssen $O(\sqrt{d})$-mal zufällig eine Indexmenge I wählen, bis wir ein I gefunden haben, das den Vektor f wie gewünscht splittet. Da wir f nicht kennen, können wir nicht prüfen, ob I eine gewünschte Aufteilung der Polynome liefert. D.h. wir müssen den Algorithmus $O(\sqrt{d})$-mal ausführen.
3. Schritt: Es sind $\binom{n/2}{d/2\,;\,d/2}$ Polynome in Zeit $O(k \cdot n)$ (vereinfachte Polynommultiplikation) in die Töpfe I^k einzusortieren.
4. Schritt: Wie Schritt 3.

5. Schritt: Wir checken für jeden der $(q/2^l)^k$ f_1-Töpfe, ob der korrespondierende f_2-Topf nichtleer ist. Falls dies der Fall ist, berechnen wir $(f_1+f_2)h \pmod q$ in Zeit $O(n^2)$.

Gesamtlaufzeit:

$$\begin{aligned}
& \sqrt{d}\cdot O\left(2\cdot k\cdot n\cdot \binom{n/2}{d/2\,;\,d/2} + n^2\cdot(\tfrac{q}{2^l})^k\right) \\
= \; & \sqrt{d}\cdot O\left(2\cdot k\cdot n\cdot \binom{n/2}{d/2\,;\,d/2} + n^2\cdot 100\binom{n/2}{d/2\,;\,d/2}\right) \\
= \; & O\left(\sqrt{d}\cdot n^2\binom{n/2}{d/2\,;\,d/2}\right)
\end{aligned}$$

Gesamtspeicherbedarf:
$|I^k|+|J^k| = 2(\frac{q}{2^l})^k \approx 200\binom{n/2}{d/2\,;\,d/2}$
Die nachfolgende Tabelle gibt eine Übersicht über Rechenzeit und Speicherplatz einer Meet-In-The-Middle Attacke auf die vorgeschlagenen NTRU-Sicherheitsstufen.

N	d	q	k	l	Besetzungsrate	Zeit	Platz
107	12	64	15	10	$10^{-2.91}$	2^{50}	2^{49}
167	20	128	14	4	$10^{-1.93}$	2^{83}	2^{82}
503	72	256	25	4	$10^{-1.65}$	2^{285}	2^{280}

4.3 Multiple-Transmission-Attacke

Angenommen Alice verschickt dieselbe Nachricht m an Bob k-mal unter Verwendung verschiedener ϕ's, d.h. sie sendet:

$$\begin{aligned}
e_1 &= p\phi_1 * h + m \pmod q \\
e_2 &= p\phi_2 * h + m \pmod q \\
&\vdots \\
e_k &= p\phi_k * h + m \pmod q
\end{aligned}$$

Ein Angreifer Eve kann diese Nachrichten mithören und die folgenden Werte für $1 \le i \le k$ berechnen :

$$\begin{aligned}
p_q^{-1}(e_i - e_1) * h^{-1} &= p_q^{-1}(p\phi_i * h + m - p\phi_1 * h - m) * h^{-1} \pmod q \\
&= \phi_i - \phi_1 \pmod q
\end{aligned}$$

Die Differenz $(e_i - e_1)$ entfernt den Nachrichtenanteil m. Durch Multiplikation mit den Inversen von p und h modulo q erhalten wir die Koeffizienten $\phi_i - \phi_1 \pmod q$ und *exakt*, da $\phi \in D(d_\phi, d_\phi)$. Die Differenz enthält nur Koeffizienten aus $[-2,2]$ gemäß folgender Tabelle:

ϕ_i/ϕ_1	-1	0	1
-1	0	-1	-2
0	1	0	-1
1	2	1	0

Koeffizienten von $\phi_i - \phi_1$

D.h. falls für den k-ten Koeffizienten $(\phi_i)_k = -1$ und $(\phi_1)_k = 1$, so erhält der Angreifer $(\phi_1)_k$, da diese Kombination die einzige ist, bei der die Differenz -2 ist. Analog lernt Eve für die Kombination $(\phi_i)_k = 1$ und $(\phi_1)_k = -1$ den Koeffizienten $(\phi_1)_k = -1$. Beide Kombinationen treten bei zufällig gewählten ϕ's mit Wahrscheinlichkeit $(\frac{d_\phi}{n})^2$ auf.

Die Multiple-Transmission Attacke kann durch den Einsatz von Zwei-Wege NTRU aus Abschnitt 3.6 verhindert werden, da dort an Stelle von m ein weiteres Randomisierungspolynom ψ verwendet wird.

4.4 Adaptive Chosen Ciphertext Attacken

Es werden zwei Adaptive Chosen Ciphertext-Attacken vorgestellt [44]. Die erste Attacke ist eine Variante von Bleichenbacher's Attacke auf den RSA Public Key Cryptography Standard $\sharp 1$ und setzt voraus, daß der Plaintext in einen speziellen, digitalen Umschlag gesteckt wird, bevor er kodiert wird. Dies wird in der Praxis öfters gemacht, damit gültige Plaintexte vom Empfänger schnell identifiziert werden können. Der zweite Angriff (Korrelationsattacke) versucht Dekodierfehler des NTRU-Cryptosystems zu erzeugen, um Informationen über die Zufallskomponente ϕ zu bekommen. Beide Attacken arbeiten nur in folgendem Sender-Empfänger-Szenario:

- Das Cryptosystem besitzt nicht *Plaintext-Awareness* (eingeführt und definiert von Bellare, Rogaway [6]); d.h. informal der Sender kann gültige Ciphertexte erstellen, ohne den zugrundeliegenden Plaintext zu kennen.
- Der Empfänger akzeptiert Nachrichten ohne Kenntnis der Sender-Identität.
- Der Sender kann polynomiell viele Nachrichten an den Sender schicken, die dieser entweder als gültige Nachricht akzeptiert oder verwirft.
- Der Sender erfährt, ob seine Nachricht als gültig akzeptiert wurde oder nicht.

4.4.1 Ein Analogon von Bleichenbachers Attacke für NTRU

Der RSA Public Key Cryptosystem Standard $\sharp 1$ benutzt den digitalen Umschlag:

00	02	PS	00	D

Dabei ist PS ein Pseudo-Zufallsstring und D bezeichnet die eigentlichen Daten.

Wir definieren nun einen analogen digitalen Umschlag für NTRU. Ein NTRU-Plaintext-Block soll also stets die folgende Form besitzen:

$$0 + 0 \cdot X + 0 \cdot X^2 + 2 \cdot X^3 + PS(X) + 0 \cdot X^k + 0 \cdot X^{k+1} + D(X)$$

Hier ist $PS(X)$ ein Pseudozufalls-Paddingpolynom und das Polynom $D(X)$ enthält die eigentlichen Daten.

Ein Angreifer fängt einen Ciphertext des Senders ab und wählt ein zufälliges, kleines Polynom ψ mit wenigen Koeffizienten ± 1, dem Rest 0. Er sendet die Nachricht

$$E \equiv \psi * e \pmod q$$

und erfährt vom Empfänger, ob E als gültige Nachricht akzeptiert wurde.

Wenn E mittels des NTRU-Systems dekodiert wird, so erhält der Empfänger die Nachricht $\psi * m$ modulo p. Dieser Plaintext wird aber nur als gültig akzeptiert, falls der digitale Umschlag unversehrt ist. Dazu müssen die 4 kleinsten Koeffizienten gleich 0002 modulo p sein und dem Datenstring $D(X)$ zwei Nullen vorangehen. Daß diese sechs vorgegebenen Stellen die angegebenen Werte annehmen, tritt bei uniform gewähltem ψ mit Wahrscheinlichkeit p^{-6} ein (bei NTRU: 3^{-6}). Jedesmal, wenn E als gültiger Ciphertext akzeptiert wird, lernt der Angreifer 6 Linearkombinationen zu den Koeffizienten des Plaintexts m:

$$\begin{aligned} \sum_{\substack{i+j \equiv l \\ (\mathrm{mod}\ n)}} \psi_i \cdot m_j &\equiv 0 \pmod p \qquad \text{für } l = 0, 1, 2, k, k+1 \\ \sum_{\substack{i+j \equiv l \\ (\mathrm{mod}\ n)}} \psi_i \cdot m_j &\equiv 2 \pmod p \qquad \text{für } l = 3 \end{aligned}$$

Damit hat der Angreifer nach $O(\frac{n}{6} \cdot p^6)$ Schritten genügend Linearkombinationen, um die Nachricht m zu rekonstruieren.

4.4.2 Korrelationsattacke über induzierten Dekodierfehler

Die NTRU-Parameter sind so gewählt, daß die Wahrscheinlichkeit eines Dekodierfehlers vernachlässigbar klein ist.

Hier wird eine Attacke vorgestellt, bei der ein abgefangener Ciphertext so modifiziert wird, daß ein Dekodierfehler viel wahrscheinlicher wird. Aus der Information, ob ein solcher Fehler auftritt, gewinnt der Angreifer Information über das Zufallspolynom ϕ.

Der Angreifer Eve nimmt einen abgefangenen Ciphertext e und wählt uniform ein ψ mit Koeffizienten aus $\{-1, 0, 1\}$ aus einer zu $D(d_\phi, d_\phi)$ ähnlichen Verteilung. Sie überträgt die Nachricht

$$E = e + p\psi * h \pmod q$$

an den designierten Empfänger und erfährt, ob E akzeptiert wird, d.h. ob ein Dekodierfehler auftritt oder nicht.
Der Sender berechnet

$$\begin{aligned} E * f &\equiv p\phi * h * f + m * f + \psi * h * f \pmod q \\ &\equiv p(\phi + \psi) * g + m * f \pmod q \end{aligned}$$

Beobachtung: Falls $\|p(\phi + \psi) * g + m * f\|_\infty^\perp > q$, dann tritt ein Dekodierfehler auf. Dies geschieht mit größerer Wahrscheinlichkeit, falls 1-Koeffizienten oder (-1)-Koeffizienten von ϕ und ψ aufaddiert werden.

Der Angreifer Eve wiederholt den obigen Angriff hinreichend oft und speichert die Werte $\psi_1, \psi_2, \ldots, \psi_\nu$, für die ein Dekodierfehler auftrat. Der Durchschnitt

$$\Psi = \frac{1}{\nu} \sum_{i=1}^{\nu} \psi_i \quad \in \mathbb{R}[X]/(X^n - 1)$$

wird wegen obiger Beobachtung mit dem Polynom ϕ korreliert sein (daher der Name Korrelationsattacke).

Eve berechnet nun ein Polynom ϕ' mit derselben Verteilung wie das originale $\phi \in D(d_\phi, d_\phi)$. Dazu ersetzt sie die größten d Koeffizienten aus Ψ mit 1 und analog die d kleinsten Koeffizienten mit -1, den Rest setzt sie Null. Sie kann nun überprüfen, ob $\phi' = \phi$, indem sie

$$e - p\phi' * h \pmod q$$

berechnet. Für $\phi' = \phi$ liefert dies die Nachricht $m \in D_m$, d.h. mit Koeffizienten aus $\{-1, 0, 1\}$. Gilt $\phi' \neq \phi \pmod q$, dann erhält man mit vernachlässigbar kleiner Fehlerwahrscheinlichkeit kein Polynom aus D_m und damit keine gültige Nachricht m. Falls der Test fehlschlägt, starten wir eine an ϕ' zentrierte Suche nach ϕ. Dabei wird die Information von Ψ ausgenutzt; je größer der Koeffizient in Ψ, desto wahrscheinlicher ist er ein 1-Koeffizient und analog für die (-1)-Koeffizienten.

Die Attacke kann noch verbessert werden, indem der Angreifer die ψ's adaptiv auswählt, je nachdem ob die vorhergehenden Werte Dekodierfehler lieferten oder nicht.

Silverman [44] hat diese Attacke implementiert und berichtet, daß bei den Parametern für mittlere Sicherheit 10.000 Iterationen einige hundert Dekodierfehler erzeugten. Diese genügten im allgemeinen, um mindestens 8 der 10 von Null verschiedenen Koeffizienten von ϕ korrekt in ϕ' zu plazieren.

Kapitel 5

Gitterreduktions-Angriff auf den Public-Key

Wir betrachten in diesem Kapitel Angriffe auf den öffentlichen Schlüssel h mittels Gitterbasenreduktion. Coppersmith und Shamir [14] reduzierten das Polynomfaktorisierungsproblem (PFP) auf das Kürzeste Vektor Problem in einem Gitter L^{cs}. Eine Gitterbasis von L^{cs} wird in Abschnitt 5.1 vorgestellt. Wir zeigen, daß beliebige Faktorisierungen f', g' von h in R_q – d.h. Polynome f' und g', die der Identität $f' * h \equiv g'$ in R_q genügen – als Koeffizientenvektoren im Gitter L^{cs} sind. Man beachte, daß die Koeffizienten der Polynome f, g und h aus $\mathbb{Z}_q = [-\lfloor \frac{q-1}{2} \rfloor, \lceil \frac{q-1}{2} \rceil]$ sind, wohingegen die Koeffizienten der Gittervektoren bei ganzzahligen Gittern aus $\mathbb{Z}$ sind. Wir zeigen, daß die Einbettung des $\mathbb{Z}_q$ in $\mathbb{Z}$ durch sogenannte q-Vektoren im Gitter erfolgt.

Im Gitter L^{cs} wird ein Parameter λ eingeführt, um die ℓ_2-Normen der Koeffizientenvektoren der geheimen Polynome f und g zu balancieren. Dieser Parameter wird in Abschnitt 5.2 optimiert.

In Abschnitt 5.3 stellen wir experimentelle Resultate unter Verwendung der Gitterbasis von L^{cs} vor. Dabei gelang es zum ersten Mal, einen NTRU-Schlüssel in mittlerer Sicherheitsstufe $n = 107$ zu brechen, d.h. die geheime Faktorisierung f, g zu rekonstruieren. Wir benötigten hierfür eine Laufzeit von 1 Monat.

Abschnitt 5.4 zeigt, daß approximative Lösungen des PFPs bis auf einen Faktor 2.5 genügen, um einen Großteil der benötigten Gleichungen zur Entzifferung einer NTRU-kodierten Nachricht m zu erhalten.

5.1 Das Coppersmith-Shamir Gitter L^{cs}

Wir versuchen hier das in Kapitel 3 vorgestellte Polynomfaktorisierungs-Problem (PFP) mit Hilfe der Gitterreduktion zu lösen und den Secret-Key f oder ein zum Entschlüsseln gleichwertiges f' aus dem Public-Key h zu bestimmen. Dazu betrachten wir das Gitter L^{cs}, das von Coppersmith und Shamir [14] vorgeschlagen wurde. L^{cs} wird von den Zeilenvektoren $b_1, b_2, \ldots, b_{2n}$ der folgenden $2n \times 2n$-Matrix generiert.

$$
\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \hline b_{n+1} \\ b_{n+2} \\ \vdots \\ b_{2n} \end{pmatrix} = \left(\begin{array}{cccc|cccc} \lambda & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & \lambda & \dots & 0 & h_1 & h_2 & \dots & h_0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda & h_{n-1} & h_0 & \dots & h_{n-2} \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right)
$$

Definition 5.1.1 (Koeffizientenspiegelung) *Der Automorphismus*

$$
\begin{aligned} \sigma : \mathbb{Z}_q[X]/(X^n-1) &\rightarrow \mathbb{Z}_q[X]/(X^n-1) \\ \sigma(g(X)) &= g(X^{-1}) \end{aligned}
$$

heißt Koeffizientenspiegelung *auf dem Ring* $R_q = \mathbb{Z}[X]/(X^n-1)$.

Man beachte: Im Ring R_q gilt $X^n = 1 \Rightarrow X^{-i} = X^n \cdot X^{-i} = X^{n-i}$. Daher liefert die Koeffizientenspiegelung des Polynoms $g = (g_0, g_1, g_2, \dots, g_{n-2}, g_{n-1})$ gerade das Polynom $\sigma(g) = (g_0, g_{n-1}, g_{n-2}, \dots, g_2, g_1)$.

Lemma 5.1.2 (CS '97) *Sei f', g' eine beliebige Faktorisierung von h in R_q, d.h. es gilt die Identität $f' * h \equiv g' \pmod q$. Dann enthält das Gitter L^{cs} den Vektor $(\lambda\sigma(f'), g')$*

Beweis: Sei $f' * h \equiv g' \pmod q$. Dann erhalten wir für jeden der n Koeffizienten von g eine Gleichung der Form

$$
\sum_{\substack{i+j\equiv k \\ (\mathrm{mod}\ n)}} f'_i \cdot h_j \equiv g'_k \pmod q \qquad \text{für } 0 \leq k < n
$$

Schreiben wir dies in Vektor-Matrix-Form, so gilt:

$$
(f'_0, f'_{n-1}, f'_{n-2}, \dots, f'_1) \cdot \begin{pmatrix} h_0 & h_1 & \dots & h_{n-1} \\ h_1 & h_2 & \dots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_0 & \dots & h_{n-2} \end{pmatrix} \equiv \begin{pmatrix} g'_0 \\ g'_1 \\ g'_2 \\ \vdots \\ g'_{n-1} \end{pmatrix} \qquad \text{über } \mathbb{Z}_q
$$

Sei H die obige zirkuläre $(n \times n)$-Matrix der h_i's. Dann werden die Koeffizienten von g' als Linearkombination der Zeilen von H dargestellt, wobei der Vektor $\sigma(f')$ die Koeffizienten der Linearkombination bestimmt.

Diese Struktur findet sich in der Gitterbasis für L^{cs} wieder. Die mit λ gewichtete Einheitsmatrix zählt die Anzahl der einzelnen H-Zeilen in der Linearkombination. Die ersten n Koeffizienten eines Gittervektors enthalten also den Vektor $\lambda\sigma(f')$. Durch diese Linearkombination wird in den hinteren n Koeffizienten der Vektor g' gebildet. Die komponentenweise Reduktion modulo q wird durch Aufaddieren bzw. Subtrahieren der Basisvektoren $b_{n+1}, \ldots, b_{2n}$ erreicht. $\square$

Da das Gitter L^{cs} beliebige Faktorisierungen f', g' von h enthält, ist insbesondere die geheime Faktorisierung f, g im Gitter enthalten.

Bezeichnung 5.1.3 (Targetvektor) *Der Vektor $\tau = (\lambda\sigma(f), g) \in L^{cs}$, der zur geheimen Faktorisierung f, g des öffentlichen Schlüssels h in R_q korrespondiert, heiße* Targetvektor.

Dieser Targetvektor τ hat nach Konstruktion des öffentlichen Schlüssels $h \equiv f^{-1} * g \pmod q$ kleine ℓ_2-Norm. Es ist bisher nicht bewiesen, daß τ der kleinste Vektor in L^{cs} ist. Coppersmith und Shamir zeigten aber, daß kleineren Gittervektoren $(\lambda\sigma(f'), g')$ Faktorisierungen f', g' entsprechen, die ebenso zum Entschlüsseln im NTRU-Cryptosystem verwendet werden können, wenn $f' \in R_p^*$ ist (siehe Abschnitt 5.4).

Bezeichnung 5.1.4 (q-Vektor) *Wir bezeichnen einen Vektor als* q-Vektor, *falls er genau einen q-Eintrag und sonst Nullen hat. Zur Unterscheidung der q-Vektoren bezeichnen wir den q-Vektor mit q im i-ten Koeffizienten auch als q_i-Vektor.*

Man beachte, daß die Koeffizienten des Public-Keys h und der geheimen Schlüssel f und g aus $\mathbb{Z}_q = [-\lfloor \frac{q-1}{2} \rfloor, \lceil \frac{q-1}{2} \rceil]$ sind, wohingegen die Gittervektoren $v \in L^{cs}$ aus dem $\mathbb{R}^{2n}$ sind. Betrachte den Fall $\lambda = 1$, d.h. $v \in \mathbb{Z}^{2n}$ für alle Gittervektoren $v \in L^{cs}$. Die Einbettung des $\mathbb{Z}_q$ in $\mathbb{Z}$ erfolgt für die hinteren n Koeffizienten von v komponentenweise mit Hilfe der Basisvektoren $b_{n+1}, \ldots, b_{2n}$ wie im Beweis von Lemma 5.1.2 gezeigt wurde. Diese Basisvektoren entsprechen den q_i-Vektoren $(n + 1 \leq i \leq n)$. Die q_i-Vektoren sind für $1 \leq i \leq n$ aber ebenfalls in L^{cs}. Berechne $q \cdot b_i$ $(1 \leq i \leq n)$; dieser Vektor besitzt einen q-Eintrag im i-ten Koeffizienten und Vielfache von q in den Koeffizienten $n + 1, \ldots, 2n$. Diese letzten n Koeffizienten können durch die Basisvektoren $b_{n+1}, \ldots, b_{2n}$ genullt werden. Damit sind die q_i-Vektoren für $1 \leq i \leq 2n$ im Gitter und wir erhalten eine Einbettung des $\mathbb{Z}_q^{2n}$ in den $\mathbb{Z}^{2n}$. Da wir die Repräsentanten von $\mathbb{Z}_q$ um Null zentriert haben, ist stets $\|v \pmod q\| \leq \|v\|$.

Falls $\lambda = \frac{s}{t} \in \mathbb{Q}$ rational ist, dann können wir die Gitterbasis von L^{cs} mit dem Faktor t multiplizieren. Dies entspricht einer Streckung von L^{cs} um t und liefert $L^{cs} \subset \mathbb{Z}^{2n}$. Wir können uns also für $\lambda \in \mathbb{Q}$ auf ganzzahlige Gitter beschränken. Die Einbettung des $\mathbb{Z}_q$ erfolgt analog zum Fall $\lambda = 1$.
Bemerkungen zum Gitter L^{cs}:

1. Die Basis des Gitters L^{cs} befindet sich in oberer Dreiecksform. Die Gitterdeterminate ist das Produkt der Diagonaleinträge:
$$\det(L^{cs}) = \prod_{i=1}^{2n} b_{ii} = \lambda^n q^n$$
2. Man beachte, daß der Targetvektor $\tau = (\lambda\sigma(f), g)$ beide geheime Schlüssel f und g enthält, obwohl man für das Polynomfaktorisierungsproblem nur

einen der beiden benötigt. Der andere folgt aus der Identität $h \equiv f^{-1} * g \pmod q$. Setzt man in der Gitterbasis von L^{cs} den Parameter $\lambda = 0$, so erhält man den Targetvektor $(0, g)$. In L^{cs} sind dann aber viele parasitäre Vektoren $(0, g')$ mit kleinerer Norm als $(0, g)$. Diese erfüllen eine Identität der Form $f' * h \equiv g'$, wobei f' im allgemeinen große Koeffizienten hat. Die Faktorisierungen f', g' lösen für $\lambda = 0$ daher nicht das PFP.

3. Das Kürzeste Vektor Problem (SVP) im Gitter L^{cs} löst im allgemeinen nicht das PFP, da für einen kürzesten Gittervektor $(\lambda\sigma(f'), g')$ nicht sichergestellt wird, daß das korrespondierende Polynom f' in R_q^* ist. In der Praxis spielt dies keine große Rolle, da die Einheitengruppe R_q^* von der Größenordnung q^n ist (für eine Schranke siehe [46]) und vermutet wird, daß der Targetvektor ein kürzester Vektor ist.

5.2 Optimale Wahl von λ

Um die Effizienz einer Gitterattacke auf NTRU mittels L^{cs} zu maximieren, müssen wir den Parameter λ in der Gitterbasis optimieren.

Lemma 5.2.1 (Minkowskischranke) *Die Länge $\lambda_1(L^{cs})$ des kürzesten Vektors in L^{cs} in der ℓ_2-Norm läßt sich nach oben beschränken durch:*

$$\lambda_1(L^{cs}) \leq \gamma_{2n}\sqrt{\lambda q},$$

wobei γ_{2n} die Hermite-Konstante zur Dimension $2n$ ist.

Beweis:
Wir setzen in den Satz von Minkowski (siehe Satz 2.2.2) ein:

$$\begin{aligned} \lambda_1(L^{cs}) &\leq \gamma_{2n} \det(L^{cs})^{\frac{1}{\dim(L^{cs})}} \\ &= \gamma_{2n}(\lambda^n q^n)^{\frac{1}{2n}} \\ &= \gamma_{2n}\sqrt{\lambda q} \quad \Box \end{aligned}$$

Wir wollen nun den Parameter λ unter Verwendung der folgenden beiden Annahmen optimieren:

1. Der Targetvektor $\tau = (\lambda\sigma(f), g)$ ist der kürzeste Vektor in L^{cs} in der ℓ_2-Norm, d.h. $\lambda_1(L^{cs}) = ||\tau||$.
(Wie wir in Kapitel 6 sehen werden, ist unter dieser Annahme der kürzeste Vektor nicht eindeutig. Alle zyklischen Shifts des Targetvektors sind ebenfalls im Gitter enthalten und besitzen gleiche ℓ_2-Norm.)

2. Für den nächstlängeren Vektor s mit $||s|| = \min_{v \in L^{cs}}\{||v|| \mid ||v|| > ||\tau||\}$ gilt: $||s|| = \delta(n)\gamma_{2n}\sqrt{\lambda q}$ für eine Gapfunktion $\delta(n)$.

Unter diesen Annahmen soll der Gap $\frac{||s||}{||\tau||}$ maximiert werden. Je größer dieser Gap ist, desto leichter hat es der Angreifer, den Targetvektor τ mittels Gitterbasenreduktion zu bestimmen.

Lemma 5.2.2 (Wahl von λ) *Der Quotient $\frac{||s||}{||\tau||}$ wird unter obigen Annahmen und für festes n durch die Wahl $\lambda = \frac{||g||}{||f||}$ maximiert.*

Beweis: Betrachte die Quadratnormfunktion $qnorm(\lambda)$ des Quotienten:

$$\begin{aligned}
qnorm(\lambda) &:= \left(\frac{||s||}{||\tau||}\right)^2 \\
&= \frac{\delta^2(n)\gamma_{2n}^2 \lambda q}{\lambda^2 \,||f||^2 + ||g||^2} \\
&= c \cdot \frac{\lambda}{\lambda^2 \,||f||^2 + ||g||^2} \\
&= c \cdot \left(\lambda \,||f||^2 + \lambda^{-1} \,||g||^2\right)^{-1}
\end{aligned}$$

Ableiten nach λ liefert

$$qnorm'(\lambda) = -c \cdot \left(\lambda \,||f||^2 + \lambda^{-1} \,||g||^2\right)^{-2} \cdot \left(||f||^2 - \lambda^{-2} \,||g||^2\right)$$

Wir setzen $qnorm'(\lambda) = 0$, indem wir den Zähler auf Null setzen:

$$\begin{aligned}
& ||f||^2 - \lambda^{-2} \,||g||^2 = 0 \\
\Rightarrow \quad & \lambda^{-2} = \frac{||f||^2}{||g||^2} \\
\Rightarrow \quad & \lambda = \frac{||g||}{||f||} \qquad \square
\end{aligned}$$

Man beachte, daß die Normen $||f||$ und $||g||$ öffentliche Systemparameter sind. Das Ergebnis sollte nicht überraschen: Betrachten wir die Norm der ersten n Koordinaten des Targetvektors τ:

$$\begin{aligned}
||(\tau_1, \ldots, \tau_n)|| &= \lambda \cdot ||f|| \\
&= ||g|| \\
&= ||(\tau_{n+1}, \ldots, \tau_{2n})|| \,.
\end{aligned}$$

Damit balanciert λ den Targetvektor so, daß der f-Teil und der g-Teil gleiche Euklidische Norm haben.

5.3 Ergebnisse des Angriffs

Die NTRU-Autoren haben selbst ihr System mittels der Gitterreduktionsalgorithmen NTL Version 1.7 von Victor Shoup angegriffen. Dabei gelang es ihnen aber nie, Dimensionen > 100 zu brechen. Der L^3 geriet hier vermutlich in Endlos-Schleifen. Diese treten auf, wenn aufgrund von Rundungsfehlern in der Float-Arithmetik der Algorithmus zur Erfüllung der Lovasz-Bedingung stets die beiden selben Vektoren hin und her tauscht.

Bei den hier vorgestellten Angriffen wurde der δ-Parameter auf 0.95 heruntergesetzt, um Endlos-Schleifen aufgrund mangelnder Bit-Genauigkeit der verschwendeten Float-Arithmetik zu vermeiden und um eine schnellere L^3-Routine zu haben. Der Durchbruch in Dimensionen $n > 100$ gelang aber erst unter dem Einsatz von Pruning-Techniken bei der Blockreduktion, wie sie von Schnorr et al [54, 55, 57, 58] vorgestellt wurden. Damit konnte erstmals NTRU mit der vorgegebenen Parameterwahl $n = 107$ für mittlere Sicherheit gebrochen werden. Die NTRU-Autoren beschrieben, daß sie bei ihren Versuchen auch den Pruning-Parameter variierten, dies aber keinen Vorteil brachte. Deshalb verwendeten sie stets die Blockreduktion mit vollständiger Enumeration. Daß NTRU mit der geschnittenen Blockreduktion gebrochen wurde, hat zwei Gründe:

- Die NTRU-Autoren attackierten ihr System nur in Dimensionen $n \leq 100$. In diesen Gitterdimensionen $2n \leq 200$ kann die vollständige Enumeration auch bei geringeren Blockweiten $\beta < 25$ den Targetvektor finden. Da die BKZ-Reduktion für diese Blockweiten nach empirischen Daten nur etwa um den Faktor 10 langsamer ist als der L^3, kommt die Gitterreduktion hier noch schnell zum Erfolg. Benötigt man allerdings Blockweiten $\beta > 30$, um den Targetvektor zu finden, so steigt die Rechenzeit des Algorithmus bei der Aufzählung kleiner Gittervektoren sehr schnell an (man beachte, daß die Enumeration eines kürzesten Vektors wie in Definition 2.2.5 Punkt *2.* mit Laufzeit $\beta^{O(\beta)}$ exponentiell in der Blockweite ist). Für diese Blockweiten muß eine geschnittene Aufzählung verwendet werden, um vernünftige Laufzeiten zu erzielen.

- Wie die Experimente zeigen, ist die Laufzeit sehr stark abhängig von der Wahl des Pruning-Parameters p. Für $p < 5$ läuft die Enumeration sehr schnell, die in der Blockreduktion benötigte Blockweite zum Auffinden des Targets τ wächst aber viel zu schnell an. Bei Parameterwahl $p \geq 10$ benötigt die Enumeration für Blockweiten $\beta \geq 30$ dagegen mehr als 100 Stunden. Die optimale Parameterwahl von p liegt - abhängig von n - im Intervall $[5, 9]$ bei der mittleren Sicherheitsstufe.

Wir geben nun einige Reduktionsergebnisse an, die wir mit der Parameterwahl $\delta = 0.95$, $p \in \{0, 5, 7, 10\}$ erhalten haben ($p = 0$ bedeutet vollständige Enumeration). Zur Reduktion wurde der folgende Algorithmus verwendet.

Algorithmus ATTACK-NTRU:

EINGABE: Sicherheitsparameter n,
Gitterbasis B von L^{cs}

$\beta = 3$;

Lösungstest $= L^3(B)$;
while (not(Lösungstest)) {
  Loesungstest $=$ BKZ-Reduktion(B, β);
  $\beta = \beta + 1$;
}

AUSGABE: Zeilenindex i der reduzierten Gitterbasis B von L^{cs}.

Die i-te Zeile enthält einen Vektor τ' mit
$\| \tau' \| < 1.5 \| \tau \|$, wobei $\| \tau \|$ die Norm des geheimen
Vektors $(\lambda\sigma(f), g)$ ist.

In der folgenden Tabelle sind alle Zeiten aufgeführt, die zum Brechen des Public-Keys h mittels Gitterreduktion benötigt wurden. Dabei wurde jeweils der Targetvektor τ gefunden, also der originale Secret-Key f rekonstruiert (bis auf die mit * gekennzeichnete Ausnahme, die nachfolgend erläutert wird). In der ersten Spalte sind die Reduktionszeiten aufgeführt, die von Hoffstein, Pipher und Silverman [24] ermittelt wurden. Die Laufzeitangabe erfolgt in hh:mm:ss.

N	HPS	$p = 0$		$p = 5$		$p = 7$		$p = 10$	
	Zeit	Zeit	β	Zeit	β	Zeit	β	Zeit	β
75	00:26:44	00:14:29	10	*00:12:36	14	$\underline{\text{00:15:12}}$	11	00:23:47	17
80	00:56:46	00:25:04	14	$\underline{\text{00:24:30}}$	20	00:34:21	20	00:34:36	17
85	01:26:08	$\underline{\text{00:47:26}}$	17	01:11:01	34	01:09:00	23	01:10:56	22
90	03:08:18	03:53:28	25	$\underline{\text{01:05:10}}$	33	01:07:25	21	02:23:33	23
92	04:28:22	06:26:49	27	04:04:08	30	04:12:55	30	$\underline{\text{03:02:42}}$	24
94	17:18:41	$\underline{\text{03:05:11}}$	23	07:10:29	54	06:26:09	30	04:13:03	24
96	22:14:05	16:51:09	26	$\underline{\text{10:17:12}}$	58	15:26:44	37	14:37:43	26
98	103:53:54	90:38:30	27	$\underline{\text{06:41:19}}$	52	60:43:53	40	31:38:56	28
100	50:55:07	–		26:53:53	63	$\underline{\text{24:58:12}}$	34	158:54:53	30
102	–	–		95:26:38	69	$\underline{\text{70:19:49}}$	39	–	
104	–	–		>20Tage		$\underline{\text{129:37:37}}$	41	–	
107	–	–		–		$\underline{\text{663:08:52}}$	57	–	

Tabelle 5.1: Gitterreduktionsattacken bei verschiedenen Pruning-Parametern

Anmerkungen zu den Testergebnissen:

- Die best-case Zeiten sind unterstrichen.
- Alle Ergebnisse wurden auf einer HP-Unix-Workstation Modell 9000/780/C160 mit 160 MHz erzielt, bis auf die Spalte $p = 5$. Diese Ergebnisse wurden auf einer HP-Unix-Workstation Modell 9000/780/C180 mit 180 MHz errechnet.
- Bis auf sehr wenige Ausnahmen waren alle durchgeführten Reduktionen schneller als bei Hoffstein, Pipher und Silverman [24].
- Die ungeschnittene Blockreduktion ($p = 0$) liefert nur bei kleinen Dimensionen gute Ergebnisse.
- Pruning $p = 5$ läuft sehr schnell, benötigt aber für Dimensionen > 100 Blockweiten, die größer als 60 sind.
- Pruning $p = 7$ erweist sich für große Dimensionen als am schnellsten.
- Pruning $p = 10$ liefert für $n < 100$ gute Werte; die Laufzeiten wachsen allerdings sehr schnell bei Blockweite $\beta \geq 30$.

- Die Zeit zum Brechen eines Public-Keys h für die vorgegebene mittlere Sicherheitsstufe von NTRU mit $n = 107$ ist ≈ 28 Tage.

- Die Gitterbasis wurde mit $\lambda = 1$ verwendet. Obwohl gemäß der Analyse in Abschnitt 5.2 die optimale Wahl $\lambda = \sqrt{\frac{29}{24}} \approx 0.9097$ ist, hat unsere Wahl von λ einige Rechenzeitvorteile bei der Reduktion mit ganzzahligen Gittern.

- Bei der mit * gekennzeichneten Reduktion wurde ein Nullteiler $(1, \ldots, 1, 0, \ldots, 0)$ in R_q gefunden (man beachte $h(1) \equiv 0$ modulo q, siehe dazu auch den Abschnitt 6.5.1 über balancierte Polynome). Dieser Vektor kann nicht zum Dekodieren in NTRU verwendet werden, da $f = (1, \ldots, 1)$ das Polynom $X^n - 1$ teilt. Deshalb ist $f \notin R_p^*$.

Eine graphische Darstellung des Zeitverhaltens der durchgeführten Reduktionen findet sich in Abbildung 5.1. Dabei tragen wir den Logarithmus der Rechenzeit in Sekunden gegen die Sicherheitsdimension n auf:

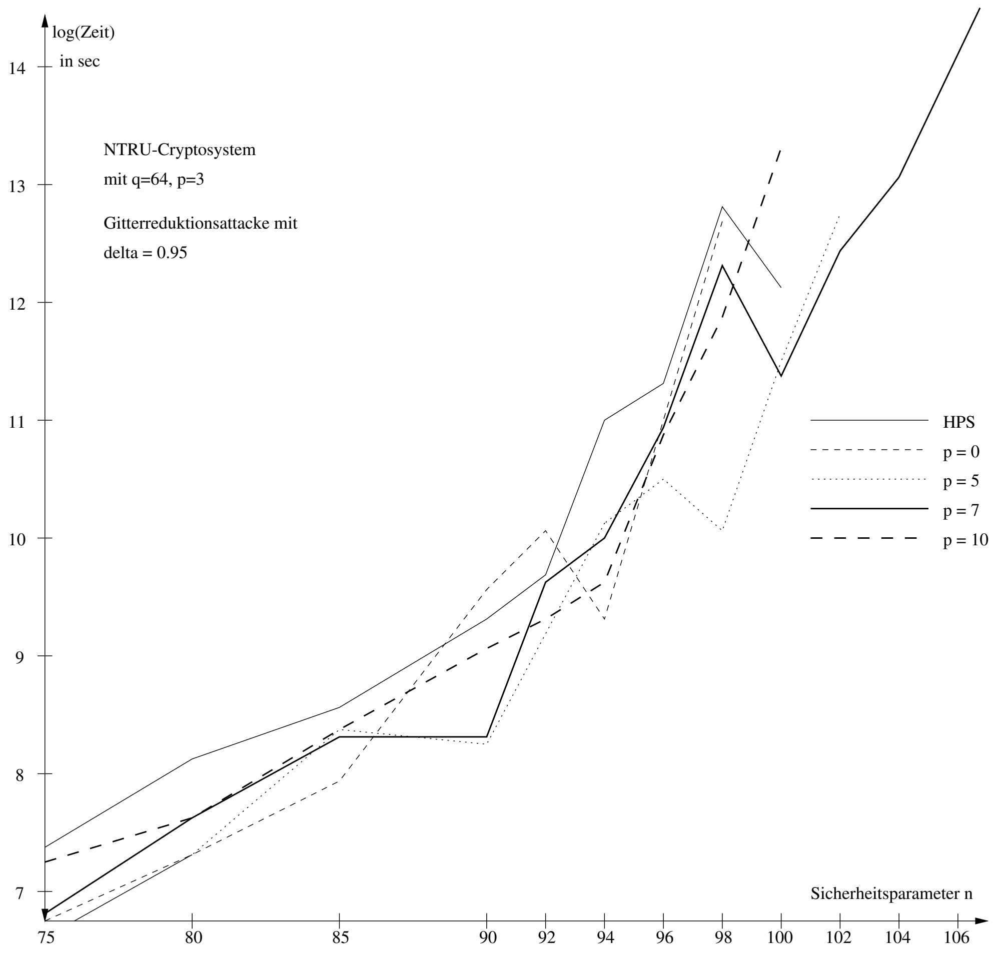

Abbildung 5.1: zeitlicher Verlauf der Reduktion bei variablem n

5.4 Alternative Schlüssel f'

Coppersmith und Shamir [14] zeigen, daß ein Vektor $\tau' = (\lambda\sigma(f'), g')$ mit $\|\tau'\| \leq \|\tau\|$ anstelle des Secret-Keys f verwendet werden kann, falls $f' \in R_p^*$ ist. Man beachte, daß wir beim Dekodieren ein Inverses ${f'_p}^{-1}$ von f' in R_p benötigen, wenn wir im Entschlüsselungsprotokoll den geheimen Schlüssel f durch ein alternatives f' ersetzen. Sie beschreiben weiterhin, daß ein Cryptanalyst einen Teil der benötigten Gleichungen zum Entziffern der verschlüsselten Nachricht m erhält, wenn er einen Schlüssel τ' besitzt mit $\|\tau'\| \leq 2.5\,\|\tau\|$.

Lemma 5.4.1 (CS '97 [14]) *Sei $\|\tau'\| = \|(\lambda\sigma(f'), g')\| \leq 2.5\,\|\tau\|$ und f' in R_p invertierbar. Dann ist die erwartete Anzahl der korrekten Koeffizienten des Polynoms a' mit:*

$$a' \equiv p\phi * g' + f' * m \pmod q$$

gerade $0.68n$.

Die korrekten Koeffizienten liefern Gleichungen der Form $a'_k \equiv \sum_{i+j\equiv k} f'_i m_j \pmod p$. Wenn wir zwei Vektoren f'_1 und f'_2 finden, die jeweils $0.68n$ lineare Relationen liefern, dann können wir das resultierende lineare Gleichungssystem lösen, um $m \pmod p$ zu bestimmen.

Lemma 5.4.1 konnte bei den Attacken nie angewendet werden. Die gefundenen Vektoren waren entweder zu groß oder das gesuchte Target τ wurde gefunden. Vektoren mit kleinerer ℓ_2-Norm als $||\tau||$ traten nie auf. Dieser Gap zwischen zu großen Vektoren und dem Targetvektor wird aber in Abschnitt 6 ausgenutzt werden, um neue Gitter zu konstruieren, die die Laufzeit der Gitterreduktion heruntersetzen.

Vermutung 5.4.2 (Gap-Hypothese) *Aufgrund der experimentellen Evidenzen wird vermutet, daß der Targetvektor τ der kürzeste Vektor in L^{cs} in der Euklidischen Norm ist. Der nächstlängere Vektor in L^{cs} war bei den durchgeführten Gitterreduktionen stets ein q-Vektor der Länge q (mit Ausnahme des Nullteilers $v = (1, \ldots, 1, 0, \ldots, 0)$). Der Gap beträgt somit experimentell $\frac{q}{\|\tau\|} = \sqrt{64^2/53} \approx 8.79$, falls der Vektor v eliminiert werden kann.*

Kapitel 6

Neue verbesserte Angriffsgitter

In diesem Kapitel stellen wir eine Reihe von Gittern vor, die Strukturen von NTRU-Parametern ausnutzen und die Gitterdimension des Standard-Gitters L^{cs} von Coppersmith und Shamir [14] verringern.

Zur Konstruktion neuer Gitterbasen weisen wir zunächst auf eine zyklische Eigenschaft des Rings $R = \mathbb{Z}[X]/(X^n - 1)$ hin; multipliziert man ein Polynom $f = (f_0, \ldots, f_{n-1}) \in R$ mit X, so werden die Polynomkoeffizienten um eine Stelle zyklisch nach rechts geshiftet. Diese Eigenschaft bewirkt, daß der kürzeste Vektor in L^{cs} nicht eindeutig ist; seine geshifteten Varianten sind ebenfalls im Gitter enthalten.

Wir nutzen das zyklische Shiften, um einen sogenannten Nullerrun in den geheimen Polynomen an einer bestimmten Stelle in der Gitterbasis zu fixieren. Damit können wir im neu konstruierten Gitter L^r den kürzesten Vektor eindeutig machen und den Gap $c = \frac{\lambda_2(L^r)}{\lambda_1(L^r)}$ vergrößern. Man beachte, daß die verwendeten Gitterreduktionsalgorithmen wie der L^3-Algorithmus von Lenstra, Lenstra und Lovasz [29] und der BKZ-Algorithmus von Schnorr et al [54, 55, 57, 58] den kürzesten Vektor in einem Gitter L bis auf einen Faktor approximieren. Wenn wir also den kürzesten Vektor bis auf einen Faktor $< c$ approximieren, so erhalten wir die exakte Lösung. Heuristisch gesprochen; je größer der Gap c, desto größer ist die Wahrscheinlichkeit, daß der Gitterreduktionsalgorithmus statt einer Approximation tatsächlich den kürzesten Vektor liefert. Um den Gap c zu vergrößern, müssen wir λ_1 verringern oder λ_2 vergrößern. Letzteres erreichen wir durch das Einführen der Run-Gitter L^r. Mittels L^3-Algorithmus verringern wir die Dimension der Gitter L^r um die Anzahl r der fixierten Koeffizienten und beschleunigen damit die Gitterbasenreduktion. Wir zeigen eine natürliche Randomisierung der Gitterbasen, indem wir die fixierten Koeffizienten aus einer zufällig gewählten Indexmenge auswählen.

J.H. Silverman [50] schlug eine weitere Variante der Run-Gitter L^r vor, die sogenannten Zero-Forced Gitter. Diese werden anschließend vorgestellt und analysiert.

Eine Verallgemeinerung der Basen für Run-Gitter wird in 6.2 vorgeschlagen, die anstatt Nullen beliebige Teilschlüssel ausnutzt und die Norm des Targets verringert.

In Abschnitt 6.3 zeigen wir, daß der gesuchte Targetvektor τ unter einer Verteilungsannahme für den öffentlichen Schlüssel der kürzeste Vektor im Gitter L^r in der Supremumsnorm ist. In 6.4 stellen wir dimensionsreduzierende Gitterbasen L_g^{red} und L_f^{red} vor, die den Gap c ausnutzen, um die Gitterdimension des Gitters L^{cs} von $2n$ auf $n(1+\alpha)$ zu drücken. Diese Attacken beruhen auf der heuristischen Gap-Hypothese aus Vermutung 5.4.2.

In 6.5 stellen wir eine weitere Gitterbasis L^b vor. Sie nutzt die Eigenschaft, daß der geheime Schlüssel g so gewählt ist, daß er gleiche Anzahl von (1)-Koeffizienten und (-1)-Koeffizienten besitzt.

Die Stärke der Gittermethoden liegt in ihrer Kombinierbarkeit (siehe 6.6). In Abschnitt 6.7 zeigen wir einige experimentelle Resultate, wie NTRU für die mittlere Sicherheitsstufe gebrochen wird. NTRU-Schlüssel mittlerer Sicherheit $n = 107$ werden in ca. 5 Stunden gebrochen (ca. 1 Stunde bei Parallelisierung). Es werden ebenso mehrere „schwache" Schlüssel – d.h. Schlüssel mit vielen fixierten Koeffizienten – mit hoher Sicherheitsstufe $n = 167$ gebrochen.

In Abschnitt 6.8 präsentieren wir zwei neue Ansätze für Gitterreduktionsattacken auf NTRU-167 (d.h. NTRU mit $n = 167$). Die erste Idee ist das Verwenden weiterer, selbsterzeugter Schlüsselgleichungen, um die Anzahl kleiner Vektoren im Gitter zu verringern. Der zweite Ansatz bricht die Symmetrie der Einsen und Minuseinsen im geheimen Schlüssel f, um Koeffizientenzähler im Gitter zu integrieren. Man beachte, daß die Anzahl der (1)-Koeffizienten und (-1)-Koeffizienten in f bekannt ist, diese Information von der Gitterbasis für L^{cs} aber nicht ausgenutzt wird. Zusätzlich werden Ideen der CJLOSS-Gitterbasis [12] für Subset Sum Probleme adaptiert, um die Norm des Targetvektors weiter zu verringern.

Notation 6.0.3 (Gitter L_g, L_f) *Bei einigen Gittern werden Eigenschaften der geheimen Schlüssel f und g ausgenutzt. Die jeweiligen Gitter L werden zur Unterscheidung mit dem entsprechenden Schlüssel indiziert, d.h. L_f respektive L_g.*

Bemerkung 6.0.4 (Balancierung mit λ) *Wir verzichten in diesem Abschnitt auf das Mitführen des Balancierungsgewichts λ wie es im Gitter L^{cs} in Kapitel 5 beschrieben wird, um die Notationen zu vereinfachen. Man beachte, daß der f-Teil – das sind diejenigen Spalten der Basis, in denen f linearkombiniert wird – der neu konstruierten Gitter analog zur Analyse in Abschnitt 5.2 stets mit λ gewichtet werden muß, um die Gitterattacken zu optimieren. Alternativ kann der g-Teil mit λ^{-1} skaliert werden.*

6.1 Herleitung des Run-Gitters L^r

6.1.1 Die zyklische Struktur von L^{cs}

Wie in Lemma 5.1.2 gezeigt wurde, korrespondieren Polynome in R_q zu Vektoren im Gitter L^{cs}. Da man die Ringpolynome durch Multiplikation mit X zyklisch shiften kann, überträgt sich diese Eigenschaft auch auf die Gittervektoren.

Definition 6.1.1 (Linksshift) *Sei* $f = (f_0, f_1, f_2, \ldots, f_{n-1})$*. Dann bezeichnen wir:*

$$f^l = (f_l, f_{l+1}, f_{l+2}, \ldots, f_{l-1})$$

als Linksshift von f um l*.*

Man beachte, daß das Polynom $f^l = X^{n-l} \cdot f$ in $\mathbb{Z}_q[X]/(X^n - 1)$.

Lemma 6.1.2 (Shift-Lemma) *Das Gitter L^{cs} enthält neben dem Targetvektor $\tau = (\sigma(f), g)$ noch die geshifteten Versionen $\tau^l = (\sigma(f)^l, g^l)$ für $0 < l < n$.*

Beweis : Multipliziere die Faktorisierung $f * h \equiv q$ von beiden Seiten mit X^{n-l} für $1 \leq l \leq n-1$:

$$f^l * h \equiv g^l \pmod q \quad \text{für } 1 \leq l \leq n-1$$

Damit erhalten wir als Koeffizientengleichungen für alle l:

$$(g^l)_k \equiv \sum_{\substack{i+j \equiv k \\ (\mathrm{mod}\ n)}} (f^l)_i \cdot h_j \Leftrightarrow g_{k+l} \equiv \sum_{\substack{i+j \equiv k \\ (\mathrm{mod}\ n)}} f_{i+l} \cdot h_j \pmod q$$

Betrachten wir dies in Matrix-Vektor-Form, so erhalten wir:

$$\begin{pmatrix} h_0 & h_1 & \ldots & h_{n-1} \\ h_1 & h_2 & \ldots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_0 & \ldots & h_{n-2} \end{pmatrix} \cdot \begin{pmatrix} f_l \\ f_{l-1} \\ f_{l-2} \\ \vdots \\ f_{l+1} \end{pmatrix} \equiv \begin{pmatrix} g_l \\ g_{l+1} \\ g_{l+2} \\ \vdots \\ g_{l-1} \end{pmatrix} \quad \text{über } \mathbb{Z}_q$$

Analog zum Beweis von Lemma 5.1.2 gilt nun, daß der Vektor $\tau^l = (\sigma(f)^l, g^l)$ im Gitter enthalten ist. $\square$.

Man beachte, daß Lemma 6.1.2 nicht nur für den Targetvektor τ gilt, sondern daß für beliebige Vektoren die n zyklischen Shifts im Gitter enthalten sind. Insbesondere gilt dies für die n zyklischen Shifts des kürzesten Gittervektors – falls dieser verschieden vom Targetvektor sein sollte. Damit ist der kürzeste Vektor in L^{cs} nicht eindeutig; die ersten sukzessiven Minima des Gitters sind alle gleich groß. Wir wollen nun die n zyklischen Shifts des kürzesten Gittervektors v mit $\|v\| = \lambda_1(L^{cs})$ untersuchen.

Definition 6.1.3 (periodisch) *Sei $f = (f_0, f_1, f_2, \ldots, f_{n-1}) \in R_q$. Dann bezeichnen wir f als* periodisch mit Periode t*, falls $f = f^t$ und t teilt n. Falls f als kleinste Periode $t = n$ besitzt, so sagen wir f ist* nichtperiodisch*.*

Lemma 6.1.4 (Verschiedenheit der Shifts) *Sei f, g die geheime Faktorisierung von h in R_q mit $f \in D(d_f+1, d_f)$, $g \in D(d_g, d_g)$ und $f*h \equiv q \pmod q$. Dann gilt:*
Sei $\|(\sigma(f'), g')\|_p = \lambda_1(L^{cs})$ die kleinste Faktorisierung f', g' von h in R_q für eine Norm ℓ_p. Falls $n > (2d_f + 1) + 2d_g$, n prim, so sind die $(\sigma(f')^l, (g')^l)$, $0 \leq l < n$ paarweise verschieden.

Da n prim ist – wie es für NTRU-Parameter stets der Fall ist, können f', g' keine nichttrivialen Perioden $t > 1$ besitzen. Angenommen f' oder g' habe Periode $t = 1$, ist also von der Form $(b, b, \ldots, b)$, $(1 \leq |b| \leq \lceil q/2 \rceil)$. Dann gilt für die ℓ_p-Norm, $p < \infty$:

$$\|(\sigma(f'), g')\|_p \geq b \cdot \sqrt[p]{n} > \sqrt[p]{(2d_f + 1) + 2d_g}.$$

Damit kann $(\sigma(f'), g')$ nicht der kleinste Vektor in L^{cs} sein, da wir nach Konstruktion des öffentlichen Schlüssels h wissen, daß der Targetvektor τ eine ℓ_p-Norm von $\sqrt[p]{(2d_f + 1) + 2d_g}$ besitzt. In ℓ_∞-Norm ist das Target ein kürzester Vektor der Länge 1 und somit ein nichtperiodischer Vektor im Gitter. Damit sind die n zyklisch geshifteten Vektoren paarweise verschieden. $\square$

Man könnte vermuten, daß die zyklischen Shifts des kürzesten Gittervektors nicht nur paarweise verschieden sondern auch linear unabhängig sind. Damit würde $\lambda_1(L^{cs}) = \lambda_2(L^{cs}) = \ldots = \lambda_n(L^{cs})$ gelten. Dies ist aber im allgemeinen nicht der Fall wie man sich leicht am Beispiel $v = (f_1, f_0, g_0, g_1) = (1, -1, 1, -1)$ klarmachen kann. Shiftet man um 1, so erhält man den Vektor $-v$. Die Anzahl der linear unabhängigen, zyklisch geshifteten Vektoren entspricht dem Rang der Matrix M, die von diesen n Vektoren gebildet wird (M besteht aus zwei $(n \times n)$-Toeplitzmatrizen). Wie Coppersmith und Shamir [14] zeigten, kann jeder Vektor in L^{cs}, der höchstens so lang ist wie der Targetvektor und dessen f-Polynome ein Inverses in R_p besitzt, gleichermaßen zur Dekodierung in NTRU genutzt werden (siehe Abschnitt 5.4 über alternative Schlüssel). Damit erhalten wir das folgende Korollar.

Korollar 6.1.5 (Gleichheit der sukzessiven Minima) *Sei M die von den zyklischen Shifts $(\sigma(f')^l, (g')^l)$, $(0 \leq l < n)$ des kürzesten Vektors gebildete $(n \times 2n)$-Matrix, $rang(M)$ der Rang von M und $f' \in R_p^*$. Dann kann das Brechen von NTRU darauf reduziert werden, einen der n Vektoren mit Länge der ersten $rang(M)$ sukzessiven Minima $\lambda_1(L^{cs}) = \ldots = \lambda_{rang(M)}(L^{cs})$ zu berechnen.*

D.h. für einen Cryptanalysten genügt es, einen der n kürzesten Vektoren in L^{cs} mittels Gitterreduktion zu bestimmen, um NTRU zu brechen. Der kürzeste Vektor ist nicht eindeutig, wie in Lemma 6.1.2 gezeigt wurde. Unser Ziel ist nun die Konstruktion eines Gitter L^r mit einem eindeutig kürzesten Vektor v. Dies vergrößert den Gap $c = \frac{\lambda_2(L^r)}{\lambda_1(L^r)}$ in der ℓ_2-Norm und macht Gitterreduktionsattacken auf NTRU effizienter.

6.1.2 Die Gitterbasis von L^r

Eine Basis der Run-Gitter erhält man, wenn man in der Gitterbasis für L^{cs} die Spaltenvektoren $n+1$ bis $n+r$ mit einer geeigneten Konstante θ multipliziert. L_g^r wird von den Basiszeilenvektoren $b_1, b_2, \ldots, b_{2n}$ der folgenden $(2n \times 2n)$-Matrix aufgespannt.

$$\left(\begin{array}{cccc|ccccccc}
1 & 0 & \dots & 0 & \theta \cdot h_0 & \dots & \theta \cdot h_{r-1} & h_r & \dots & h_{n-1} \\
0 & 1 & \dots & 0 & \theta \cdot h_1 & \dots & \theta \cdot h_r & h_{r+1} & \dots & h_0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & & \\
0 & 0 & \dots & 1 & \theta \cdot h_{n-1} & \dots & \theta \cdot h_{r-2} & h_{r-1} & \dots & h_{n-2} \\
\hline
0 & 0 & \dots & 0 & \theta \cdot q & \dots & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & \theta \cdot q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & q
\end{array}\right)$$

Für $\theta = q + 1$ sichern wir, daß der kürzeste Vektor $v = (v_1, \dots, v_{2n})$ im Gitter $L_g^r(\theta)$ Nulleinträge in den Koordinaten $v_{n+1}, \dots, v_{n+r}$ hat. Andernfalls hat v mindestens ℓ_2-Norm θ, aber die q-Vektoren $b_{n+r+1}, \dots, b_{2n}$ haben Norm q. Damit kann v nicht der kürzeste Vektor in $L_g^r(\theta)$ sein.

Bezeichnung 6.1.6 (fixiert) *Eine in der Gitterbasis von $L^r(\theta)$ mit θ multiplizierte Spalte nennen wir* fixiert. *Die Koeffizienten eines Vektors $v \in L^r(\theta)$ in den fixierten Spalten bezeichnen wir als* fixierte Koeffizienten.

Wir wollen nun den Targetvektor eindeutig machen.

Definition 6.1.7 (Nullerrun) *Sei $g \in D_g(d, d)$. Dann bezeichnen wir $(g_i, g_{i+1}, \dots, g_j)$ mit $g_i, g_{i+1}, \dots, g_j = 0$ als* Nullerrun in g der Länge $j - i + 1$. *Die Indizes von g werden dabei modulo n betrachtet. Der längste Nullerrun in g hat Länge r, falls*

$$r = \max_{i,j \bmod n}\{j - i + 1 \mid g_i, g_{i+1}, \dots, g_j = 0\}.$$

Wir nennen einen längsten Nullerrun in g eindeutig, *falls es nur einen längsten Nullerrun in g gibt.*

Damit erhalten wir das folgende Lemma.

Lemma 6.1.8 (eindeutiger Targetvektor) *Der Koeffizientenvektor g enthalte einen eindeutig längsten Nullerrun der Länge r, der an Position g_l beginnt. Dann enthält $L_g^r(\theta)$ einen eindeutig bestimmten Targetvektor*

$$\tau^l = ((f_0, f_{n-1}, f_{n-2}, \dots, f_1)^l | \ (g_0, g_1, \dots, g_{n-1})^l)$$

mit ℓ_2-Norm $\sqrt{\|f\|^2 + \|g\|^2}$. Alle anderen Shifts τ^m, $(m \neq l)$ haben mindestens Norm $\sqrt{\|f\|^2 + \|g\|^2 - 1 + \theta^2}$.

Beweis: Nach Voraussetzung enthält g einen eindeutig längsten Nullerrun der Länge r. Damit enthält $L_g^r(\theta)$ gemäß Lemma 6.1.2 den geshifteten Targetvektor $\tau^l = (\sigma(f)^l, g^l)$, wobei g^l mit dem Nullerrun beginnt. Der Multiplikator θ ändert die Norm von τ^l nicht.
Da der längste Nullerrun eindeutig ist, besitzen alle anderen geshifteten Targetvektoren τ^m mindestens einen Eins- oder Minuseins-Eintrag in den Koeffizienten $n+1, \ldots, n+r$. Dieser Eintrag wird mit dem Faktor θ skaliert. □

Wählt man θ größer als q, so werden schon nach der L^3-Reduktion fast alle Koeffizienten $n+1$ bis $n+r$ der Gittervektoren auf 0 fixiert. Dies reduziert den Suchraum für kurze Vektoren. Je größer man r wählen kann, desto schneller wird die Reduktion Erfolg haben, d.h. geheime Schlüssel g mit langem Nullerrun sind besonders geeignet für diese Art von Gitterreduktionsattacken.

Wir zeigen nun, daß das Fixieren von r Spalten tatsächlich auch zu einer Dimensionreduzierung von $2n$ auf $2n-r$ führt. D.h. wir können r Basisvektoren entfernen und damit den Gitterangriff weiter beschleunigen.

Definition 6.1.9 (Untergitter $\bar{L}^r$) *Wir definieren das Untergitter von $\bar{L}^r \subset L^r(\theta)$ vermöge*

$$\bar{L}^r = \{v \in L^r(\theta) \mid v_{n+1}, \ldots, v_{n+r} = 0\}.$$

Algorithmus DIMENSIONSREDUZIERUNG

EINGABE: $(2n \times 2n)$-Gitterbasis von $L^r(\theta)$ mit $\theta > 2^{\frac{2n-r-1}{2}} q$

1. LLL-Reduktion von $L^r(\theta)$
2. Entferne alle Basisvektoren von $L^r(\theta)$, deren Norm größer als $2^{\frac{2n-r-1}{2}} q$ ist.
3. Entferne die Nullkoeffizienten aus den r fixierten Spalten der restlichen Basisvektoren.

AUSGABE: L^3-reduzierte $(2n-r \times 2n-r)$-Gitterbasis von $\bar{L}^r$

Satz 6.1.10 (Basis des Untergitters $\bar{L}^r$) *Der geheime Schlüssel besitze einen Nullerrun der Länge mindestens r. Dann hat das Untergitter $\bar{L}^r$ von Gitter $L^r(\theta)$ mit $\theta > 2^{\frac{2n-r-1}{2}} q$ Dimension $2n-r$. Eine L^3-reduzierte Basis von $\bar{L}^r$ kann mittels Algorithmus DIMENSIONSREDUZIERUNG in Zeit $O(n^6 \log B)$ berechnet werden. Dabei ist B eine obere Schranke für die Quadratnorm der Basisvektoren $b_1, b_2, \ldots, b_{2n}$ von $L^r(\theta)$: $\|b_i\|^2 < B$ für alle i.*

Beweis: Da $\bar{L}^r$ aus den Basisvektoren von $L^r(\theta)$ berechnet wird, ist $\bar{L}^r$ nach Definition 2.1.5 ein Untergitter von $L^r(\theta)$. Gemäß Definition 6.1.9 sind die Koeffizienten $n+1, \ldots, n+r$ auf Null fixiert, d.h. jeder Gittervektor $v \in \bar{L}^r$ ist aus $\mathbb{Z}^n \times 0^r \times \mathbb{Z}^{n-r}$. Damit hat $\bar{L}^r$ Dimension $2n-r$. Nun gilt für jedes Gitter L (siehe Cohen [11], Theorem 2.6.2, Seite 85):
Sei $b'_1, b'_2, \ldots, b'_m$ eine L^3-reduzierte Basis von L. Für linear unabhängige $v_1, \ldots, v_t \in L$ ist :

$$\|b'_j\| \leq 2^{\frac{m-1}{2}} \max_t \{\|v_1\|, \ldots, \|v_t\|\} \quad \text{für } 1 \leq j \leq t. \tag{6.1}$$

Im Untergitter $\bar{L}^r \subset L^r(\theta)$ sind die q_i-Vektoren $(0,\ldots,0,q,0,\ldots,q)$, die an der i-ten Stelle einen q-Eintrag haben für $i \in [1,n] \cup [n+r+1, 2n]$ enthalten. Die q_i-Vektoren mit $i \in [n+r+1, 2n]$ entsprechen den Basisvektoren b_i. Einen q_i-Vektor mit $i \in [1,n]$ erzeugt man durch $q \cdot b_i$. Dadurch entstehen im g-Teil des Vektors Vielfache von q bzw. $\theta \cdot q$, die durch die Basisvektoren $b_{n+1},\ldots,b_{2n}$ genullt werden. Diese $2n-r$ q_i-Vektoren sind linear unabhängig.

$$\Rightarrow \|\bar{b}_j\| \leq 2^{\frac{2n-r-1}{2}} \max_i \|q_i\| = 2^{\frac{2n-r-1}{2}} q \quad \text{für } 1 \leq j \leq 2n-r$$

Damit haben wir eine obere Schranke für die Norm der L^3-reduzierten Basisvektoren $\bar{b}_1,\ldots,\bar{b}_{2n-r}$ von $\bar{L}^r$.
Betrachte nun einen Basisvektor b nach Schritt 1 in Algorithmus DIMENSIONSREDUZIERUNG, d.h. nach L^3-Reduktion des Gitters $L^r(\theta)$. Falls b einen von Null verschiedenen Eintrag in den Koeffizienten $n+1,\ldots,n+r$ besitzt, so wird dieser Koeffizient mit θ skaliert. Damit kann b kein Basisvektor von $\bar{L}^r$ sein, da $\|b\| \geq \theta > 2^{\frac{n-1}{2}} q$. Die Wahl von θ sichert also das Nullen der fixierten Koeffizienten.
Der laufzeitbestimmende Schritt ist die L^3-Reduktion. Mit Satz 2.2.4 folgt die Behauptung. $\square$

In der Praxis genügt ein kleineres θ als die in Satz 6.1.10 angegebene Schranke, da die in Gleichung (6.1) verwendete Abschätzung für die sukzessiven Minima pessimistisch ist. Tatsächlich erreicht man wesentlich bessere Approximationen. Deshalb sagen wir im folgenden oft etwas lax, daß ein Gewicht θ groß genug gewählt werden muß. Um die Größe der Gewichte theoretisch exakt zu bestimmen, kann stets die obige Methode angewendet werden.

Bemerkung 6.1.11 (das Gitter L^r_f) *Man beachte, daß man anstatt Nullerruns in g auch Nullerruns in f fixieren kann, indem man analog die entsprechenden Spalten im f-Teil der Basis des Gitters L^{cs} multipliziert. Das entstehende Gitter bezeichnen wir mit $L^r_f(\theta)$. Der Grund, warum wir hier die Analyse für den g-Teil durchführen liegt darin, daß g für alle Parameterwahlen von NTRU einige Nullkoeffizienten mehr besitzt.*
Wenn wir die beiden Gitter $L^r_g(\theta)$ und $L^r_f(\theta)$ kombinieren wollen, so müssen wir die korrekte Position eines Nullerruns in f raten, da wir den Targetvektor τ^l gemäß Lemma 6.1.8 fixieren.

Um den Parameter r optimal zu wählen, ist es notwendig abzuschätzen, welcher Bruchteil der Schlüssel g einen Nullerrun der Länge r hat, wenn g zufällig und uniform aus der Schlüsselmenge $D(d,d)$ gewählt wird. Zur Erinnerung: g enthält d (1)-Koeffizienten, d (-1)-Koeffizienten und die anderen Einträge sind 0.

Lemma 6.1.12 (Ws-Lemma für Runs) *Sei*
$\Pr(r) = \Pr_{g \in_R D(d,d)}[g$ *enthält einen Run der Länge* $r]$. *Dann gilt:*

$$\Pr(r) \quad \leq \quad \frac{n \sum_{i=0}^{2} \binom{2}{i} \binom{n-(r+2)}{d-i\,;\,d-(2-i)}}{\binom{n}{d\,;\,d}}$$

$$\Pr(r) \quad \geq \quad \frac{n \sum_{i=0}^{2} \binom{2}{i} \binom{n-(r+2)}{d-i\,;\,d-(2-i)} - \frac{n(n-2(r+2)+1)}{2} \sum_{i=0}^{4} \binom{4}{i} \binom{n-2(r+2)}{d-i\,;\,d-(4-i)}}{\binom{n}{d\,;\,d}}$$

Beweis: Der Schlüsselraum von g hat Größe $|D(d,d)| = \binom{n}{d\,;\,d}$: Es gibt n Möglichkeiten für die Startposition eines Nullerruns. Ein Run der Länge r wird eingeschlossen von zwei (-1), einer (-1) und einer (+1) (bzw. (+1) und (-1)) oder zwei (+1). Im ersten Fall ist die Anzahl der günstigen Schlüssel $\binom{n-(r+2)}{d\,;\,d-2}$, da wir d (+1) und $d-2$ (-1) auf die verbleibenden $n-r-2$ Koeffizienten verteilen müssen. Die anderen Fällen sind analog. Man beachte, daß solche g's doppelt gezählt werden, die zwei Nullerruns der Länge r besitzen. Daher erhalten wir eine obere Schranke.
Die untere Schranke erhält man, indem man diejenigen Schlüssel eliminiert, die zwei Nullerruns der Länge r enthalten. Die Anzahl der Positionen an denen die beiden Nullerruns auftreten können ist $n(n-2(r+2)+1)/2$. Damit folgt die zweite Abschätzung. □

Die Differenz beider Abschätzungen konvergiert für $r \to n-2d$ schnell gegen Null. Sie beträgt z.B. 0.6% für $r = 15$ und $< 0.1\%$ für $n = 21$. D.h. für die relevanten Parameterwahlen von r, die wir in den Experimenten (siehe Abschnitt 6.7) betrachten, können wir Schlüssel mit mehreren Runs der Länge r vernachlässigen. Damit erhalten wir die Approximationen:

$$\Pr(r) \approx \frac{n \sum_{i=0}^{2} \binom{2}{i} \binom{n-(r+2)}{d-i\,;\,d-(2-i)}}{\binom{n}{d\,;\,d}}$$

$$\Pr_{g \in_R D(d,d)}[g \text{ enthält einen Run der Länge} \geq r] \quad \approx \quad \sum_{j=r}^{n-2d} \Pr(j) \qquad (6.2)$$

Das Problem ist, daß wir die Länge r eines längsten Nullerruns in g nicht a priori kennen. Mit Gleichung (6.2) können wir ein Intervall berechnen, in dem r mit hoher Wahrscheinlichkeit liegt und die Algorithmen für die relevanten Werte starten.

6.1.3 Randomisierung der Basis für Run-Gitter

Ein Ansatz, die Attacke mittels Run-Gittern uneffektiv zu machen, ist eine Zusatzoption bei der Schlüsselerzeugung. Geheime Schlüssel, deren längster Nullerrun eine vorgegebene Schranke r überschreitet, werden verworfen. Dies verkleinert bei geeigneter Wahl der Schranke den Schlüsselraum nur geringfügig.

Man beachte aber, daß die Effizienz der Run-Gitter nicht von den Nullerruns in den geheimen Schlüsseln abhängt, sondern von der großen Anzahl der Nullen in f und g und der zyklischen Eigenschaft. Man kann also anstatt r aufeinanderfolgende Spalten zu fixieren auch r zufällig gewählte Spalten fixieren. Diese Form des Angriffs kann der Schlüsselerzeuger nicht verhindern, da er die vom Angreifer gewählten Spalten nicht kennt.

Zum Erzeugen des Run-Gitters $L^r(\theta)$ sollte man daher in der Praxis eine zufällige Indexmenge $I = \{i_1, \ldots, i_r\}$, $I \subset \{n+1, \ldots, 2n\}$ wählen (bzw. $I \subset \{1, \ldots, n\}$ für das Fixieren im f-Teil) und die gemäß der Indexmenge festgelegten Spalten mit θ multiplizieren.

Die in Abschnitt 6.1.2 vorgestellten Lemmata und Sätze übertragen sich in natürlicher Weise auf diese Randomisierung der Run-Gitter. Die Wahrscheinlichkeitsabschätzung aus Lemma 6.1.12 vereinfacht sich. Offenbar gilt für ein Polynom $g \in D(d, d)$:

$$\Pr_I(g_{i_1} = \ldots = g_{i_r} = 0) = \frac{\binom{n-r}{d\,;\,d}}{\binom{n}{d\,;\,d}}.$$

Betrachten wir nun die n zyklischen Shifts von g, so erhalten wir

$$\begin{aligned} \Pr_I(r) &= \Pr_I(g_{i_1+k} = \ldots = g_{i_r+k} = 0 \text{ für ein } 0 \le k < n) \\ &= 1 - \Pr_I(\text{ für keinen } n \text{ der Shifts gilt } g_{i_1+k} = \ldots = g_{i_r+k} = 0) \\ &\approx 1 - \left(1 - \frac{\binom{n-r}{d\,;\,d}}{\binom{n}{d\,;\,d}}\right)^n . \end{aligned}$$

Wir erhalten nur eine Approximation, da die betrachteten Wahrscheinlichkeiten der n zyklischen Shifts in der vorletzten Zeile nicht unabhängig sind.

Bemerkung 6.1.13 (Runs vs. Indexmengen) *Man beachte, daß man dieselbe Wahrscheinlichkeitsanalyse auch für fixierte Nullerruns durchführen kann. Man kann aber erwarten, daß die Abhängigkeit der Wahrscheinlichkeiten der n Shifts dann größer ist. Deshalb sollte man beim Fixieren mittels zufällig gewählter Indexmengen größere r verwenden können als beim Fixieren von Runs.*

Wir wollen Bemerkung 6.1.13 durch Experimente stützen. Dazu haben wir 100.000 Schlüssel $g \in_R D(12, 12) \subset \mathbb{Z}_q[X]/(X^{107} - 1)$ mit den Parametern der mittleren Sicherheitsstufe zufällig gezogen. Zusätzlich wurde eine zufällige Indexmenge $I = [i_1, i_2, \ldots, i_{83}] \subset_R [0, 106]$ ausgewählt. In Tabelle 6.1 stellt Spalte run(%) den Prozentsatz der Polynome g dar, die einen längsten Nullerrun der Länge r besitzen. In Spalte rand(%) ist der Prozentsatz der Polynome aufgetragen, bei denen $g_{i_1+k} = \ldots = g_{i_r+k} = 0$ für mindestens einen Linksshift um k $(0 \le k < n)$ gilt, aber nicht $r+1$ Koeffizienten $g_{i_1+k}, \ldots, g_{i_{r+1}+k}$ auf Null fixiert werden können.

Die relativen Häufigkeiten der maximalen Runs nehmen ihr Maximum für $r = 12$ an, die relativen Häufigkeiten der maximalen Indexmengen bei $r = 17$. Interessant ist der letzte Spalteneintrag, da wir bei den Experimenten für NTRU-107 (NTRU mit Sicherheitsparameter $n = 107$) in Abschnitt 6.7 $r = 21$ Koeffizienten fixieren. Dies gelingt unter Verwendung einer zufälligen Indexmenge mit Wahrscheinlichkeit ca. $\frac{1}{4}$. Dagegen besitzen nur etwa 7% der Schlüssel $g \in D(12, 12)$ einen Nullerrun der Länge mindestens 21. Für Angriffe sollte also stets die randomisierte Variante der Run-Gitter verwendet werden.

r	8	9	10	11	12	13	14	15
run(%)	1.14	3.98	7.70	10.87	12.42	12.19	10.87	9.20
rand(%)	0.00	0.01	0.13	0.69	2.52	4.45	7.60	10.34
r	16	17	18	19	20	21	22	≥ 21
run(%)	7.57	5.92	4.51	3.48	2.72	1.97	1.42	7.32
rand(%)	11.54	11.61	10.67	9.19	7.88	5.99	4.63	23.38

Tabelle 6.1: Prozentsätze der maximalen Runs und maximalen Indexmengen

Die Wahrscheinlichkeit aus Lemma 6.1.12 hängt von der zufälligen, uniformen Wahl von g aus $D(d,d)$ ab; d.h. von den Münzwürfen des Schlüsselerzeugers. Bei der randomisierten Variante hängt die Wahrscheinlichkeit $\Pr_I(r)$ nur von der Indexmenge I ab.

Satz 6.1.14 (Erfolgswahrscheinlichkeit, Parallelisierung) *Bei den randomisierten Rungittern ist die Erfolgswahrscheinlichkeit* $\Pr_I(r)$ *nur von den Münzwürfen des Angreifers abhängig. Im Erwartungswert muß ein Angreifer* $\Pr_I(r)^{-1}$ *verschiedene Rungitter* L^r *betrachten, bis er Erfolg hat. Dies kann – im Gegensatz zu den bisher bekannten Gitterattacken auf NTRU –* parallelisiert *werden.*

6.1.4 Zero-Forced Gitter

J.H. Silverman [50] schlug eine Verallgemeinerung der Run-Gitter vor, die sogenannten Zero-Forced Gitter. Die Idee der Zero-Forced Gitter ist es, die potentiellen Nullkoeffizienten in einem geheimen Schlüssel gar nicht zu betrachten; diese Koeffizienten werden mittels linearer Algebra aus den Schlüsselgleichungen eliminiert.

Angenommen g haben einen längsten Nullerrun der Länge r. Da alle zyklischen Shifts von g im Gitter L^{cs} enthalten sind, können wir oBdA annehmen, daß der Nullerrun bei g_0 beginne. Wir erhalten das Gleichungssytem

$$\begin{pmatrix} h_0 & h_1 & \dots & h_{n-1} \\ h_1 & h_2 & \dots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_0 & \dots & h_{n-2} \end{pmatrix} \cdot \begin{pmatrix} f_0 \\ f_{n-1} \\ f_{n-2} \\ \vdots \\ f_1 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ \vdots \\ 0 \\ g_r \\ \vdots \\ g_{n-1} \end{pmatrix} \quad \text{über } \mathbb{Z}_q$$

Wir können nun einfach r lineare Kongruenzen für $f_1, \dots, f_r$ in Abhängigkeit von $f_0, f_{n-1}, \dots, f_{r+1}$ lösen und in die verbleibenden $n-r$ Kongruenzen zurückeinsetzen. Damit erhalten wir ein neues System von Kongruenzen

$$\begin{pmatrix} h'_{0,0} & h'_{0,1} & \dots & h'_{0,n-r-1} \\ h'_{1,0} & h'_{1,1} & \dots & h'_{1,n-r-1} \\ \vdots & \vdots & \ddots & \vdots \\ h'_{n-r-1,0} & h'_{n-r-1,1} & \dots & h'_{n-r-1,n-r-1} \end{pmatrix} \cdot \begin{pmatrix} f_0 \\ f_{n-1} \\ f_{n-2} \\ \vdots \\ f_{r+1} \end{pmatrix} \equiv \begin{pmatrix} g_r \\ g_{r+1} \\ \vdots \\ g_{n-1} \end{pmatrix} \quad \text{über } \mathbb{Z}_q$$

wobei die $h'_{i,j}$ bekannt und die Koeffizienten von f, g unbekannt sind. Daraus konstruieren wir das Zero-Forced Gitter L^{zf}, das von den Zeilen der folgenden $(2(n-r) \times 2(n-r))$-Matrix generiert wird.

$$\left(\begin{array}{cccc|cccc}
1 & 0 & \dots & 0 & h'_{0,0} & h'_{1,0} & \dots & h'_{n-r-1,0} \\
0 & 1 & \dots & 0 & h'_{0,1} & h'_{1,1} & \dots & h'_{n-r-1,1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 1 & h'_{0,n-r-1} & h'_{1,n-r-1} & \dots & h'_{n-r-1,n-r-1} \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q
\end{array}\right)$$

Der Targetvektor in L^{zf} ist $(f_0, f_{n-1}, \dots, f_{r+1}, g_r, g_{r+1}, \dots, g_{n-1})$. Die fehlenden Koeffizienten $f_1, \dots, f_r$ können wir über die zuvor erwähnten Gleichungen in Anhängigkeit von $f_0, f_{n-1}, \dots, f_{r+1}$ berechnen.

Der Vorteil dieser Variante der Run-Gitter ist, daß sie die Gitterdimension anstatt auf $2n-r$ auf $2n-2r$ verringert.

6.2 Verallgemeinerte Run-Gitter: Das Gitter L^{part}

Man beachte, daß für manche Parameterwahlen mehr Einsen im geheimen Schlüssel f sind als Nullen. Deshalb ist es vorteilhaft, anstatt Nullen beliebige geratene Teilstücke eines geheimen Schlüssels zu fixieren. Angenommen wir raten, daß im Schlüssel f ein Teilvektor mit den Koeffizienten $v_1, \dots, v_r$ enthalten ist. Dann konstruieren wir die folgende $(2n+1 \times 2n+1)$-Gitterbasis für das Gitter L^{part}.

$$\left(\begin{array}{cccccccc|ccccc}
\theta & 0 & \dots & 0 & 0 & \dots & 0 & & h_0 & h_1 & \dots & h_{n-1} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \vdots & & & \vdots & \vdots \\
0 & 0 & \dots & \theta & 0 & \dots & 0 & & h_{r-1} & h_r & \dots & h_{r-2} & 0 \\
0 & 0 & \dots & 0 & 1 & \dots & 0 & & h_r & h_{r+1} & \dots & h_{r-1} & 0 \\
\vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & & \vdots & & & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 1 & & h_{n-1} & h_0 & \dots & h_{n-2} & 0 \\
\hline
0 & 0 & \dots & 0 & 0 & \dots & 0 & & q & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & & 0 & q & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & & 0 & 0 & \dots & q & 0 \\
\hline
\theta v_1 & \theta v_2 & \dots & \theta v_r & 0 & \dots & 0 & & 0 & 0 & \dots & 0 & 1
\end{array}\right)$$

Man beachte, daß die Run-Gitterbasis von $L^r(\theta)$ der Spezialfall von $L^{part}(\theta)$ ist, bei der das Vektorteilstück $(v_1, \dots, v_r)$ aus dem r-stelligen Nullvektor besteht; dadurch können der Basisvektor b_{2n+1} und die zusätzliche Spalte entfernt werden. Wählt man nun θ groß genug, so beginnen alle kurzen Vektoren mit r Nullen. Jeder Gittervektor mit Norm kleiner θ besteht dann entweder nur aus einer Linearkombination der Basisvektoren $b_{r+1}, \ldots, b_{2n}$ oder er enthält den zusätzlichen Vektor b_{2n+1}. Kommt nämlich der Basisvektor b_i, $1 \leq i \leq r$ in der Linearkombination vor, dann erzeugt er im i-ten Koeffizienten als Eintrag ein Vielfaches von θ. Dieses kann aber nur durch den Basisvektor b_{2n+1} genullt werden.

Analog zu Satz 6.1.10 kann gezeigt werden, daß man mit der Wahl $\theta > 2^{\frac{2n-r}{2}} q$ und anschließender Anwendung des Algorithmus DIMENSIONSREDUZIERUNG ein Gitter der Dimension $2n - r + 1$ erhält. Dazu benötigen wir die zusätzlich eingeführte letzte Spalte in der Basismatrix, die sicherstellt, das die Basisvektoren des konstruierten Untergitter $\bar{L}^{part}$ linear unabhängig sind und der Vektor b_{2n+1} in die Basis von $\bar{L}^{part}$ eingeht. Durch die zusätzliche Spalte können wir bei der Gitterbasenreduktion einen effizienten Lösungstest verwenden. Ein potentieller Targetvektor muß im letzten Koeffizienten einen ± 1-Eintrag besitzen, da b_{2n+1} genau einmal in eine Linearkombination des Targets ein.

Die verallgemeinerte Variante der Run-Gitter hat noch einen weiteren Vorteil:

- Die Quadratnorm $\|\tau\|^2$ des Targets verringert sich um die Quadratnorm $\|(v_1, \ldots, v_r)\|^2$ des geratenen Teilvektors v.

Wenn wir annehmen, daß τ der kürzeste Vektor in L^{cs} ist, vergrößern wir damit auch den Gap $c = \frac{\lambda_2}{\lambda_1}$.

6.3 Der kürzeste Vektor in $L^r(\theta)$ in Supremumsnorm

Um zu zeigen, daß der Gap $c = \frac{\lambda_2(L^r(\theta))}{\lambda_1(L^r(\theta))}$ in der ℓ_∞-Norm größer als 1 ist, müssen wir beweisen, daß der Targetvektor τ^l im Gitter L_f^r analog zu Lemma 6.1.8 der einzige Vektor mit Supremumsnorm 1 ist. Wir müssen Nullen im f-Teil fixieren, da der öffentliche Schlüssel h nach Konstruktion die Gleichung $h(1) \equiv 0 \pmod{q}$ erfüllt (siehe dazu auch Abschnitt 6.5.1 über balancierte Polynome). Daher enthält das Gitter L^{cs} den Vektor $\tau' = (f', g')$ mit $f' = (1, \ldots, 1)$ und $g' = (0, \ldots, 0)$. Die ℓ_∞-Norm dieses Vektors wird durch Wahl von $L_f^r(\theta)$ auf θ vergrößert .

Wir treffen die folgende Annahme über die Verteilung der Koeffizienten des Public-Keys h:

Annahme 6.3.1 (Verteilung der h_i) *Sei $h = (h_0, h_1, \ldots, h_{n-1}) = f_q^{-1} * g$ mit $f \in_R D(d_f + 1, d_f)$, $g \in_R D(d_g, d_g)$. Dann sind die h_i unabhängig und uniform verteilt in $\mathbb{Z}_q$.*

Dabei können wir die Identität $h(1) \equiv 0$ bei der Unabhängigkeit der h_i vernachlässigen, da sie im Gitter $L_f^r(\theta)$ mit einem Vektor der ℓ_∞-Norm θ korrespondiert.
Die Annahme besagt mit anderen Worten:

> Falls PFP (Polynomfaktorisierungsproblem) schwer ist, dann können wir den öffentlichen Schlüssel h nicht von einem zufälligen Element aus R_q unterscheiden.

Wir wissen, falls PFP leicht ist, dann können wir h von einem zufälligen Element aus R_q unterscheiden. Dazu lösen wir PFP und betrachten die Norm der Faktorisierung. Die Kontraposition ist: h nicht unterscheidbar $\Rightarrow$ PFP ist schwer. Unsere Annahme ist die Umkehrung dieser Implikation. Wir nehmen also an, daß PFP und das Unterscheidungsproblem gleiche Komplexität besitzen.

Satz 6.3.2 (ℓ_∞-Eindeutigkeitssatz) *Unter Annahme 6.3.1 gilt:*

$$\lim_{n\to\infty} \Pr_h[\exists f', g' \mid f' * h \equiv g' \pmod q), \|f'\|_\infty, \|g'\|_\infty = 1, (\sigma(f'), g') \neq \tau^l] = 0$$

falls $q > 9$.
D.h. der Targetvektor τ ist der einzige Vektor mit $\|\tau\|_\infty = 1$ für hinreichend große n.

Beweis : Sei r die Länge des längsten Nullerruns in f. Betrachte das Gitter $L_f^r(2)$. Damit ein Vektor $v \in L_f^r(2)$ die ℓ_∞-Norm 1 besitzt, müssen sich die Einträge in den mit 2 multiplizierten Spalten $1, \ldots, r$ zur Null linearkombinieren. Der längste Nullerrun in f mit Länge r beginne mit f_{i-r+1}. Dann erhalten wir in Matrix-Vektor-Form:

$$\begin{pmatrix} h_0 & h_1 & \ldots & h_{n-1} \\ h_1 & h_2 & \ldots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_0 & \ldots & h_{n-2} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_{i-r} \\ f_{i-r-1} \\ \vdots \\ f_{i+1} \end{pmatrix} \equiv \begin{pmatrix} g_i \\ \vdots \\ g_{i+r-1} \\ g_{i+r} \\ g_{i+r+1} \\ \vdots \\ g_{i-1} \end{pmatrix} \quad \text{über } \mathbb{Z}_q$$

Wir betrachten nun die Wahrscheinlichkeit, daß h andere Faktorisierungen f', g' mit Supremumsnorm 1 besitzt.

$$\Pr_h\left[\exists f', g' \;\middle|\; \begin{array}{l} f' * h \equiv g', \\ \|f'\|_\infty, \|g'\|_\infty = 1, \\ (\sigma(f'), g') \neq \tau^l \end{array}\right] = \Pr_h\left[\exists f' \;\middle|\; \begin{array}{l} f' * h \equiv g', \\ \|f'\|_\infty = 1, \\ f' \neq (\sigma(f))^l, g' \text{ fest} \end{array}\right] \cdot |\{g \in \{\pm 1, 0\}^n\}|$$

Gemäß Annahme 6.3.1 gilt $h_j \in_R \mathbb{Z}_q$. Damit ist:

$$\Pr_h\left[\exists f', g' \;\middle|\; \begin{array}{l} f' * h \equiv g', \\ \|f'\|_\infty, \|g'\|_\infty = 1, \\ (\sigma(f'), g') \neq \tau^l \end{array}\right] \leq \frac{3^{n-r}}{q^n} \cdot 3^n = \frac{3^{2n-r}}{q^n}$$

Die Behauptung folgt. $\square$

Es ist offen, ob es ein ähnliches Result für die ℓ_2-Norm gibt.

6.4 Dimensionsreduzierende Gitter

6.4.1 Das Gitter L_g^{red}

Die experimentellen Beobachtungen suggerieren, daß es keine anderen Vektoren in L^{cs} gibt, die ähnlich kurz in der ℓ_2-Norm sind wie das Target τ – außer den n geshifteten Versionen. Vermutung 5.4.2 besagt, daß der Gap zum nächstgrößeren Vektor $\frac{q}{\|\tau\|}$ beträgt. Damit ist die Methode von Coppersmith und Shamir, alternative Schlüssel zu verwenden (siehe Lemma 5.4.1), in der Praxis nicht anwendbar. Falls es andererseits nur wenige kurze Gittervektoren gibt, genügt es, wenn die Koeffizientengleichungen $g_k \equiv \sum_{i+j\equiv k} f_i h_j$ nicht für alle Koeffizienten g_k $(0 \le k < n)$ erfüllt sind, sondern für einen Großteil der Koeffizienten. Wenn es keine anderen Vektoren mit kleinen Einträgen in den betrachteten Koeffizienten gibt, dann bleibt der Targetvektor der kürzeste Vektor. Diese Idee wird in dem Gitter $L_g^{red}(\alpha)$ $(0 < \alpha \le 1)$ umgesetzt, das von den Zeilenvektoren der folgenden Matrix generiert wird.

$$\left(\begin{array}{cccc|cccc}
1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{\lceil \alpha n\rceil -1} \\
0 & 1 & \dots & 0 & h_1 & h_2 & \dots & h_{\lceil \alpha n\rceil} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 1 & h_{n-1} & h_0 & \dots & h_{\lceil \alpha n\rceil -2} \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q
\end{array}\right)$$

Der Vorteil von $L_g^{red}(\alpha)$ gegenüber L^{cs} ist, daß die Gitterdimension von $2n$ auf $(1+\alpha)n$ gedrückt wird.
Nun kann man α nicht beliebig klein wählen, da es dann zu viele kleine, parasitäre Vektoren in L_g^{red} gibt, die in den nicht betrachteten Koeffizienten große Einträge besitzen. Um den Parameter α zu optimieren, müßte man mehr über die Dichte kleiner Vektoren in L^{cs} wissen. Es gibt leider nur wenige allgemeine Aussagen über die Anzahl kleiner Vektoren in Gittern; die Resultate von Lagarias, Odlyzko [27] sind für diese speziellen Gitter zu schwach.
Das vorgestellte Gitter ist somit nur eine Angriffsheuristik. In der Praxis liefert es für $\alpha \in [0.5; 0.8]$ signifikante Laufzeitgewinne. So verringert es die Rechenzeit zum Brechen eines Schlüssels in mittlerer Sicherheitsstufe $n = 107$ von 1 Monat (mit L^{cs}) auf ca. 3 Tage. Dies entspricht einem Gewinn um den Faktor 10.

6.4.2 Das Gitter L_f^{red}

Anolog zur Methode aus dem vorangegangenen Abschnitt 6.4.1, können wir anstatt einen Bruchteil der g-Koeffizienten zu betrachten ebenso nur einen Bruchteil der f-Koeffizienten bei der Gitterreduktion berücksichtigen. Diese Idee wird durch das Gitter $L_f^{red}(\beta)$ $(0 < \beta \le 1)$ realisiert, das von den Zeilenvektoren der folgenden Matrix generiert wird.

$$\left(\begin{array}{cccc|cccc}
1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\
0 & 1 & \dots & 0 & h_1 & h_2 & \dots & h_0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \dots & \dots & \vdots \\
0 & 0 & \dots & 1 & h_{\lceil \beta n \rceil -1} & \dots & \dots & h_{\lceil \beta n \rceil -2} \\
0 & 0 & \dots & 0 & h_{\lceil \beta n \rceil} & \dots & \dots & h_{\lceil \beta n \rceil -1} \\
\vdots & \vdots & & \vdots & \vdots & & & \vdots \\
0 & 0 & \dots & 0 & h_{n-1} & h_0 & \dots & h_{n-2} \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q
\end{array}\right)$$

Obwohl L_f^{red} von $2n$ Vektoren generiert wird, ist die Gitterdimension kleiner, da die Gittervektoren linear abhängig sind. Der L^3-Algorithmus entdeckt diese linearen Abhängigkeiten und entfernt die entstehenden Nullvektoren. Deshalb verringert sich die Dimension auf $n(1+\beta)$.
Da der Targetvektor das Polynom g enthält, können wir alle Koeffizienten von f berechnen, indem wir das Gleichungssystem $f * h \equiv g$ lösen.

6.5 Herleitung des Balance-Gitters L^b

6.5.1 Balancierte Polynome

P. Nguyen [37] schlug als Erweiterung der bisherigen, verbesserten Gitterattacken [30] vor, die Eigenschaften $f(1) = 1$ und $g(1) = 0$ der geheimen Schlüssel auszunutzen. Obwohl seine Methode des Schneidens von Gittern vom theoretischen Standpunkt interessant ist, liefert sie keinen praktischen Algorithmus, um solche geschnittenen Gitter in der Praxis zu konstruieren.
Wir stellen hier ein Gitter L^b vor, daß die inpraktikable Methode des Schneidens vermeidet und $g(1) = 0$ ausnutzt. Man beachte, daß im NTRU System $g \in D(d,d)$, d.h. g enthält genauso viele (1)-Koeffizienten wie (-1)-Koeffizienten. Daher gilt $g(\gamma) = 0$ für $\gamma \in \mathbb{Z}$.

Definition 6.5.1 (balanciert) *Sei $g = (g_0, \dots, g_{n-1})$ im Polynomring $R = \mathbb{Z}[X]/(X^n - 1)$ mit ganzzahligen Koeffizienten. Dann nennen wir g* balanciert, *falls*

$$\sum_{i=0}^{n-1} g_i = 0 \text{ in } R.$$

Die Balanciertheitsbedingung für g ist äquivalent zu $g(1) = 0$ bzw. $(X-1)|g$. Wegen letzterer Bedingung ist jedes balancierte Polynom in $R \setminus R^*$, da $X^n - 1 = (X-1)(X^{n-2} + X^{n-3} + \dots 1)$. Damit gilt $(X-1)$ teilt $\gcd(g, X^n - 1)$, und g ist nicht invertierbar.

Lemma 6.5.2 (Ideal I_b) *Die Teilmenge $I_b = \{g \mid g \text{ balanciert}\} \subset R = \mathbb{Z}[X]/(X^n - 1)$ ist ein Ideal von R, d.h.*

1. *$(I_b, +)$ ist eine Untergruppe von $(R, +)$*
2. *Für alle $r \in R$ gilt: $rI_b \subseteq I_b$ und $I_b r \subseteq I_b$.*

Beweis: Zu (1): $(I_b, +)$ ist abgeschlossen bezüglich der Addition, da für zwei Polynome $u, v \in R$ gilt:

$$\sum_{i=0}^{n-1} u_i + v_i = \sum_{i=0}^{n-1} u_i + \sum_{j=0}^{n-1} v_j = 0.$$

Das neutrale Element in I_b ist $(0, \ldots, 0)$. Das Inverse zu u in I_b ist $-u$. Die Kommutativität folgt aus der Kommutativität von R.
Zu (2): Seien $r \in R$, $u \in I_b$ beliebig. Dann ist

$$(r * u)(1) = r(1) \cdot u(1) = r(1) \cdot 0 = 0$$

Damit gilt $rI_b \subseteq I_b$, $I_b r \subseteq I_b$ folgt aus der Kommutativität der Polynommultiplikation. $\square$

Korollar 6.5.3 *Das Polynom $h \equiv f_q^{-1} * g \in \mathbb{Z}[X]/(X^n - 1)$ ist balanciert, da $g \in I_b$.*

Man beachte, daß der öffentliche Schlüssel $h \in R_q = \mathbb{Z}_q[X]/(X^n - 1)$ im allgemeinen nicht balanciert ist, da die Balanciertheit bei der komponentenweisen Reduktion modulo q verloren geht. Betrachte dazu das balancierte Polynom $(q-1, -\frac{q-1}{2}, -\frac{q-1}{2})$. Reduktion modulo q liefert das nichtbalancierte Polynom $(-1, -\frac{q-1}{2}, -\frac{q-1}{2})$. Es gilt lediglich $\sum h_i = t \equiv 0$ modulo q. Wir subtrahieren daher t von h_0, um den öffentlichen Schlüssel zu rebalancieren.

Unser Ziel ist es, das Gitter L^{cs} so zu modifizieren, daß im g-Anteil – d.h. in den Koeffizienten $n+1, \ldots, 2n$ – nur balancierte Polynome linearkombiniert werden, da der gesuchte geheime Schlüssel g nach Konstruktion balanciert ist. D.h. wir beschränken den Suchraum im g-Teil auf das Ideal der balancierten Polynome. Damit eliminieren wir einige kleine Vektoren im Gitter, wie z.B. die unbalancierten q-Vektoren $(0, \ldots, 0, q, 0, \ldots, 0)$. Wir wollen zunächst eine Gitterbasis für balancierte Polynome konstruieren.

Satz 6.5.4 (Basissatz für I_b) *Das Gitter $L_{balance}$ werde von den Zeilen der folgenden $(n - 1 \times n)$-Matrix erzeugt.*

$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & -1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -1 \end{pmatrix}$$

Dann ist $(b_0, b_1, \ldots, b_{n-2})$ eine Basis für das Ideal I_b der balancierten Polynome in $R = \mathbb{Z}[X]/(X^n - 1)$.

Beweis: Sei $v \in L_{balance}$. Dann ist v balanciert, da jeder Basisvektor balanciert ist. Damit gilt $L_{balance} \subseteq I_b$.
Bleibt $I_b \subseteq L_{balance}$ zu zeigen, d.h. wir müssen zeigen, daß ein beliebiges balanciertes Polynom v im Gitter ist. Sei $v = (v_0, \ldots, v_{n-1})$, gesucht sind die Koeffizienten $\alpha_0, \ldots, \alpha_{n-2}$ in der Basisdarstellung $v = \sum_{i=0}^{n-2} \alpha_i b_i$ von v. Die Konstruktion der Koeffizienten erfolgt mit dem Algorithmus

$$\begin{aligned} \alpha_0 &= v_0 \\ \alpha_i &= \alpha_{i-1} + v_i \quad \text{für } 1 \leq i \leq n-2. \end{aligned}$$

Man beachte, daß der Vektoreintrag v_{n-1} bei der Berechnung nicht benötigt wird. Er wird durch $v_0, \ldots, v_{n-2}$ festgelegt, da $v_{n-1} = -\sum_{i=0}^{n-2} v_i$.
Damit ist $L_{balance} = I_b$. $\square$

6.5.2 Das balancierte Gitter L^b

Wenn wir den öffentlichen Schlüssel $h \in R_q$ wie zuvor beschrieben mit $t = \sum_i h_i$ rebalancieren, unterscheidet er sich von dem bei der Schlüsselberechnung erzeugten $h \in R = \mathbb{Z}[X]/(X^n - 1)$ (vor der Reduktion modulo q) durch ein balanciertes Polynom $v \in I_b$, für das gilt $v \equiv (0, \ldots, 0)$ in R_q. Wir können v mittels der Gitterbasis $q \cdot L_{balance}$ erzeugen. Damit erhalten wir das balancierte Gitter L^b, das von den Zeilen der folgenden $(2n-1 \times 2n)$-Matrix aufgespannt wird.

$$\left(\begin{array}{cccc|ccccc} 1 & 0 & \ldots & 0 & h_0 - t & h_1 & \ldots & h_{n-2} & h_{n-1} \\ 0 & 1 & \ldots & 0 & h_1 & h_2 & \ldots & h_{n-1} & h_0 - t \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & h_{n-1} & h_0 - t & \ldots & h_{n-3} & h_{n-2} \\ \hline 0 & 0 & \ldots & 0 & q & -q & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 & q & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & q & -q \end{array}\right)$$

Das Gitter L^b hat die folgenden Vorteile gegenüber dem Standardgitter L^{cs}.

1. Die Gitterdimension beträgt $2n - 1$.
2. L^b ist ein Untergitter von L^{cs}. Es enthält genau die Gittervektoren von L^{cs}, deren g-Teil balanciert ist und damit auch den gesuchten Targetvektor τ. Der Suchraum wird durch die Einschränkung auf balancierte Polynome verkleinert, z.B. werden die unbalancierten, kleinen q-Vektoren im g-Teil eliminiert. Es gibt allerdings noch q-Vektoren im unbalancierten f-Teil.

6.5.3 Ein weiteres balanciertes Gitter $\hat{L}^b$

In Abschnitt 6.5.2 wird die Gitterbasis $q \cdot L_{balance}$ aus Satz 6.5.4 zur Basiskonstruktion des balancierten Gitters verwendet. Ein balanciertes Polynom v

muß analog zum Algorithmus aus dem Beweis von Satz 6.5.4 linearkombiniert werden. Um unnötige Schritte bei der Gitterreduktion zu vermeiden, kann die folgende Alternative eines balancierten Gitters Laufzeitvorteile in der Praxis haben. Das Gitter $\hat{L}^b$ werde von den Zeilenvektoren der folgenden $(2n \times 2n+1)$-Matrix generiert.

$$\left(\begin{array}{cccc|ccccc}
1 & 0 & \dots & 0 & h_0 - t & h_1 & \dots & h_{n-1} & 0 \\
0 & 1 & \dots & 0 & h_1 & h_2 & \dots & h_0 - t & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 1 & h_{n-1} & h_0 - t & \dots & h_{n-2} & 0 \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 & \gamma \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 & \gamma \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q & \gamma
\end{array}\right)$$

Das Gewicht γ sollte groß gewählt werden. Das Gitter $\hat{L}^b$ enthält zwar auch unbalancierte Polynome im g-Teil, die Norm der korrespondierenden Gittervektoren ist aufgrund des Gewichts aber mindestens γ. Wählt man γ groß genug, so kann man analog zu Satz 6.1.10 nach der L^3-Reduktion einen Gittervektor mit Eintrag γ sowie die Nullkoeffizienten in der γ-Spalte der restlichen Basisvektoren entfernen. Ein weiterer Vorteil von $\hat{L}^b$ liegt in der Kombinierbarkeit mit den verallgemeinerten Run-Gittern, wie sie in Abschnitt 6.6 beschrieben wird.

Bemerkung 6.5.5 (Balancierung von f) *Eine Methode analog zum Gitter $\hat{L}^b$ kann dazu benutzt werden, die Schlüsselinformation im Schlüssel f auszunutzen, der einen (1)-Koeffizienten mehr besitzt als (-1)-Koeffizienten. D.h. entfernt man eine 1 aus f, so ist das Restpolynom balanciert. Um diese Eigensschaft auszunutzen, modifizieren wir das Gitter L^{part} und erhalten eine Gitterbasis für L^b_f:*

$$\left(\begin{array}{ccccccc|cccc}
\theta & 0 & \dots & 0 & 0 & \dots & 0 & h_0 & \dots & h_{n-1} & \delta \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & \vdots \\
0 & 0 & \dots & \theta & 0 & \dots & 0 & h_{r-1} & \dots & h_{r-2} & \delta \\
0 & 0 & \dots & 0 & 1 & \dots & 0 & h_r & \dots & h_{r-1} & \delta \\
\vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 1 & h_{n-1} & \dots & h_{n-2} & \delta \\
\hline
0 & 0 & \dots & 0 & 0 & \dots & 0 & q & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & q & 0 \\
\hline
\theta v_1 & \theta v_2 & \dots & \theta v_r & 0 & \dots & 0 & 0 & \dots & 0 & \delta
\end{array}\right)$$

In eine Linearkombination des Targetvektors geht der letzte Basisvektor b_{2n+1} einmal ein. Damit sind nur solche Vektoren $v \in L^b_f$ klein, für die $f(1) = 1$, da für $v = \sum_{i=1}^{2n+1} \alpha_i b_i$ der δ-Koeffizient $\sum_{i=1}^{n} \alpha_i$-mal aufaddiert wird. Die $\alpha_1, \dots, \alpha_n$ entsprechen aber den f_i und müssen sich zu 1 aufaddieren, damit der letzte Koeffizient durch b_{2n+1} genullt wird.

6.6 Kombinierbarkeit der Gittermethoden

Wir untersuchen nun die in den vorherigen Abschnitten vorgestellten Gitterbasen auf ihre Kombinierbarkeit und geben Hinweise, was bei der Verwendung mehrerer Methoden beachtet werden muß. Wir betrachten die paarweisen Gitterkombinationen der verallgemeinerten Run-Gitter L^{part}, der balancierenden Gitter L^b und der dimensionsreduzierenden Gitter L^{red}.

	L_f^{part}	L_g^{part}	L_f^b	L_g^b	L_f^{red}
L_g^{part}	😐[1]				
L_f^b	🙂[2]	🙂			
L_g^b	🙂	🙂[3]	🙂		
L_f^{red}	🙂[4]	🙂	🙂	🙂	
L_g^{red}	🙂	🙂[5]	🙂	🙁[6]	🙂[7]

1. Diese Gitterbasen sind nur bedingt kombinierbar (siehe dazu Bemerkung 6.1.11). Hat man bestimmte Spaltenvektoren im g-Teil der Matrix fixiert, so muß man die korrekte Position des zu fixierenden Teilvektors im f-Teil raten, da man sich auf eine bestimmte geshiftete Version τ^l des Targetvektors festlegt.

2. Es ist vorteilhaft, den zusätzlich benötigten Vektor $(0, \ldots, 0, \delta)$ zur Balancierung mit dem Vektor des geratenen Teilstücks aus L_f^{part} zu kombinieren, da beide Vektoren in eine Linearkombination des neuen Targets genau einmal eingehen.

3. Das balancierte Alternativgitter $\hat{L}_g^b$ und L_g^{part} sind ohne Einschränkungen kombinierbar. Dagegen kann man das balancierte Gitter L_g^b nicht verwenden, da die Skalierung in der Basismatrix von L_g^{part} die Rebalanciertheit des Schlüssels h und die q-Basis $q \cdot L_{balance}$ für balancierte Polynome zerstört.

4. Wählt man die Randomisierungsvariante von L_f^{part}, so sollte man darauf achten, daß die bzgl. der Indexmenge $I = \{i_1, \ldots, i_r\}$ fixierten Spalten im Intervall $[1, \alpha n]$ sind, damit die fixierten Koeffizienten auch wirklich verwendet werden.

5. Analog zu Punkt 4. sollte darauf geachtet werden, daß die bzgl. der Indexmenge $I = \{i_1, \ldots i_r\}$ fixierten Koeffizienten in der Spaltenmenge $\{n+1, \ldots, n(1+\beta)\}$ sind, damit alle Fixierungen aus I genutzt werden.

6. Diese Gitter sind nicht kombinierbar, da in den Basisvektoren von L_g^{red} nicht mehr der volle Public-Key verwendet wird. Deshalb kann man nicht balancieren.

7. Die Methoden sind ohne weiteres kombinierbar. Man muß aber beachten, daß der gesuchte Targetvektor weder den kompletten Schlüssel f noch g enthält, so daß fehlende Koeffizienten geraten werden müssen, z.B. mittels Meet-in-the-Middle Algorithmus 4.2. Für ein moderat gewähltes α oder

β (im Bereich von 0.8) ist das Raten auch mit der Brute-Force Methode kein Problem bei mittlerer und hoher Sicherheitsstufe von NTRU.

6.7 Experimentelle Resultate

6.7.1 Mittlere Sicherheitsstufe $n = 107$

Bei den durchgeführten Gitterreduktionen wurde Algorithmus ATTACK-NTRU aus Abschnitt 5.3 benutzt. Der Reduktionsparameter δ und der Parameter für Tiefeninsertionen γ wurden beide auf 0.95 gesetzt. Wir wählten eine Reihe von Pruningparametern, um die Laufzeiten zu optimieren. Für Sicherheitsparameter $n > 95$ ist $p \in [6, 8]$ optimal.

Die folgenden Resultate wurde auf einer HP-Unix Workstation Modell 9000 mit 200 MHz berechnet. Die NTRU Parameter wurden gemäß Tabelle 3.2 für mittlere Sicherheit gewählt. In der ersten Spalte sind die best-case Laufzeiten aufgeführt, die benötigt wurden, um die geheimen Schlüssel mit Hilfe des Coppersmith-Shamir Gitter L^{cs} zu ermitteln (siehe auch Tabelle 5.3). Die Zeiten bei Verwendung der beiden dimensionsreduzierenden Gitter $L_g^{red}(\alpha)$ und $L_g^{red}(\beta)$ unter Benutzung verschiedener Parameter α, β sind in Spalte 2 angegeben. Man beachte: Falls α und β beide kleiner als 1 sind, müssen die fehlenden Koeffizienten im geheimen Schlüssel f geraten werden – z.B. mit der Meet-in-the-Middle Attacke von Odlyzko (siehe Abschnitt 4.2). In der 3. Spalte sind die ermittelten Laufzeiten aufgeführt, die unter Verwendung des Run-Gitter $L_g^r(q)$ aus Abschnitt 6.1.2 erzielt wurden. Dabei wurden solange geheime Schlüssel gewählt, bis der Nullerrun in g mindestens $\frac{21}{107}n$ betrug. Die Wahrscheinlichkeit eines solchen Nullerruns kann mit Gleichung (6.2) approximiert werden. Sie beträgt ca. 7% für $n = 107$ und bei Randomisierung nach Abschnitt 6.1.3 ca. 23%. D.h. die Laufzeiten der Run-Gitter müssen in der Praxis mit einem Faktor von ca. $(0.23)^{-1} \approx 4$ multipliziert werden. Die letzte Spalte stellt die Laufzeiten dar, wenn man sowohl Nullerruns in g als auch in f fixiert. Sie entspricht also einer Kombination der Gitter L_g^r und L_f^r. Die Schlüssel wurden so gewählt, daß die Summe der Runs mindestens $1.8 \cdot \frac{21}{107}n$ betrug. Die letzten beiden Spalten verwenden ebenso das dimensionsreduzierende Gitter $L_g^{red}(0.7)$. Die Laufzeiten sind angegeben in hh:mm:ss.

	L^{cs}	L_g^{red} & L_f^{red}			$L_g^r(q)$ & L_g^{red}		$L_{g\&f}^r(q)$ & L_g^{red}		
N	Zeit	Zeit	α	β	Zeit	r	Zeit	r_g	r_f
75	000:15:12	00:06:47	0.5	1.0	00:02:56	17	00:02:39	14	16
80	000:24:30	00:18:23	0.8	0.8	00:06:39	20	00:04:41	14	15
85	000:47:26	00:32:36	0.5	1.0	00:06:32	17	00:06:16	17	16
90	001:05:10	00:37:04	0.5	1.0	00:07:44	18	00:07:40	17	17
92	003:02:42	01:16:03	0.8	1.0	00:10:08	22	00:08:37	33	11
94	003:05:11	02:40:36	0.8	1.0	00:13:39	22	00:13:12	24	10
96	010:17:12	04:01:36	0.5	1.0	00:19:27	21	00:15:49	27	12
98	006:41:19	03:55:53	0.8	1.0	00:25:07	23	00:21:36	15	10
100	024:58:12	02:15:14	0.77	1.0	00:28:50	20	00:23:46	16	20
102	070:19:49	09:16:14	0.8	1.0	00:42:40	25	00:17:31	23	18
104	129:37:37	-			00:24:21	24	00:28:20	26	12
107	663:08:52	66:40:25	0.8	0.8	01:28:53	21	00:40:51	12	27

Eine graphische Darstellung der Laufzeiten gegen den Sicherheitsparameter n ist in Abbildung 6.1 zu sehen.

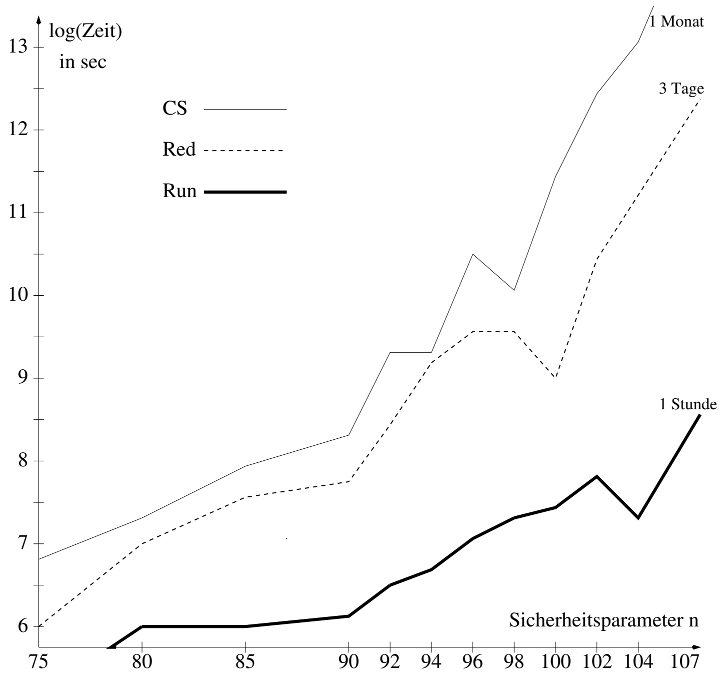

Abbildung 6.1: Laufzeitvergleich von L^{cs}, L_g^{red} & L_f^{red} und $L_g^r(q)$ & L_g^{red}

6.7.2 Hohe Sicherheitsstufe $n = 167$

Es wurden ebenso einige „schwache" Schlüssel – d.h. Schlüssel mit sehr großen Nullerruns in f und g – in hoher Sicherheitsstufe $n = 167$ gebrochen. Ob-

wohl die Wahrscheinlichkeit, daß solche Schlüssel in der Praxis auftauchen, vernachlässigbar klein ist, demonstrieren die Angriffe doch die Stärke der neuen Gittermethoden und zeigen, daß auch $n = 167$ wohl noch nicht genügt, um das NTRU Cryptosystem sicher gegen die neuen Gitterangriffe zu machen.
Wir verwendeten die Gittermethoden $L_g^r(500)$, $L_f^r(500)$ und $L_g^{red}(0.7)$. Es wurden also Spalten im f- und im g-Teil fixiert. Die auftretenden Gitterdimensionen sind deutlich größer als 200. Bei den Algorithmen mit Gram-Schmidt Orthogonalisierung kam es bei diesen Dimensionen zu Stabilitätsproblemen aufgrund numerischer Unkonditioniertheit in der Float-Arithmetik. Es wurden daher die Reduktionsalgorithmen von Henrik Koy [26] verwendet, bei denen die Gram-Schmidt Koeffizienten mittels Householdertransformation berechnet werden.

run$(f+g)$	173	155	135	116	97	82
Zeit	02:00:06	01:58:51	02:56:22	03:18:28	05:24:16	15:35:44

Tabelle 6.2: Laufzeiten für hohen Sicherheitslevel $n = 167$

6.8 Zwei neue Ansätze für NTRU-167

Die zuvor präsentierten Methoden verbessern die Laufzeiten beim Angriff auf NTRU-Schlüssel in hoher Sicherheitsstufe $n = 167$ signifikant. Die Anzahl der in Abschnitt 6.7.2 fixierten Koeffizienten kann aber in der Praxis nur mit vernachlässigbar kleiner Wahrscheinlichkeit geraten werden. Für realistische r scheinen die Laufzeiten einen Monat zu überschreiten.

Wir wollen deshalb hier zwei neue Ansätze zeigen, mit denen Schlüssel mit $n = 167$ möglicherweise auch schnell gebrochen werden können. Im ersten Ansatz werden alternative öffentliche Schlüssel h' aus dem Original h erzeugt, die eine Zerlegung in das geheime f und ein g' besitzen. Die neue Schlüsselgleichung wird im Gitter L^{keys} *zusätzlich* verwendet unter Erhaltung der Gitterdimension.

Der zweite Ansatz versucht, die Symmetrie von (1)- und (-1)-Koeffizienten im geheimen Schlüssel f aufzubrechen, um Zähler für die Anzahl dieser Koeffizienten zu integrieren. Damit konstruieren wir zum ersten Mal ein Gitter, das die öffentlich bekannte Anzahl der Einsen und Minuseinsen ausnutzt. Durch das Aufbrechen der Schlüsselsymmetrie können wir die Norm des Targetvektors weiter verringern, indem wir die Gittermethode der CJLOSS-Basis [12] für Subset Sum Probleme adaptieren. Wie Experimente zeigen, liefert das L^{sym}-Gitter den vielversprechendsten Ansatz für Angriffe auf NTRU-Challenges in hohen Sicherheitsdimensionen.

6.8.1 Verwendung mehrerer Schlüsselgleichungen

In diesem Abschnitt wollen wir aus der bekannten Schlüsselgleichung $f * h \equiv g$ neue öffentliche Schlüssel h' konstruieren, die der Gleichung $f * h' \equiv g'$ genügen. D.h. das Polynom h' besitzt eine Zerlegung in das ursprüngliche f und ein g', dessen ℓ_2-Norm von der Konstruktion abhängt. Dies ist leicht zu realisieren;

multipliziere beide Seiten der Gleichung mit einem selbstgewählten Polynom ψ

$$f * (h * \psi) \equiv g * \psi \pmod q,$$

und setze $h' \equiv h*\psi$ und $g' \equiv g*\psi$. Nun kann die neu gewonnene Beziehung zum Erweitern des Gitters L^{cs} verwendet werden, indem wir die folgende Gitterbasis konstruieren.

$$\left(\begin{array}{c|c|c} I & H & H' \\ \hline 0 & qI & 0 \\ \hline 0 & 0 & qI \end{array}\right)$$

Hierbei steht I für die n-dimensionale Einheitsmatrix und H bzw. H' für die zirkulären Matrizen der Schlüssel h respektive h'. Der Vorteil der neuen Matrix ist eine Gitterdeterminante von q^{2n}. Nach einer Heuristik die auf Gauß zurückgeht verringert dies die Anzahl der kleinen Vektoren im Gitter. Die Heuristik ist eine Umkehrung des ersten Satzes von Minkowski $\lambda_1(L) \leq \gamma_n(\det(L))^{1/n}$, die besagt, daß in zufälligen Gittern $\lambda_2(L) \geq c \cdot \gamma_n(\det(L))^{1/n}$ für eine Konstante c. Vergrößern wir nun die Gitterdeterminante, so muß $\lambda_2(L)$ größer werden, um die Gauß-Heuristik zu erfüllen. Eine Verringerung der Dichte kleiner Vektoren hilft den Reduktionsalgorithmen, kürzeste Vektoren zu finden.

Das Problem dieses Ansatzes ist die Dimensionserhöhung auf $3n$ bzw. $(2+m)n$ für m zusätzlich konstruierte Schlüsselgleichungen. Dieser Nachteil macht die Methode für relevante Dimensionen unpraktikabel. Weiterhin führen wir neue n kurze q-Vektoren in das Gitter ein, die das Auffinden des Targets erschweren. Man beachte: Die Quadratnorm des Targets erhöht sich um die Quadratnorm von g'. Ist die neue Norm größer als q, dann ist das Target kein kürzester Vektor und nicht mehr unter den $3n$ kürzesten linear unabhängigen Vektoren.

Unser Ziel ist eine Verwendung zusätzlicher Schlüsselgleichungen unter *Erhaltung der Gitterdimension* und *Vergrößern der Norm der zur modulo q-Reduktion benötigten Vektoren*. Dazu betrachten die Schlüsselgleichung einmal in $\mathbb{Z}[X]/\,(X^n-1)$

$$\begin{aligned} f*h+q(\alpha_0,\alpha_1,\ldots,a_{n-1}) &= g \quad \text{in } \mathbb{Z}[X]/(X^n-1) \\ \Rightarrow f*(h*\psi)+q(\alpha_0,\alpha_1,\ldots,a_{n-1})*\psi &= g*\psi \quad \text{in } \mathbb{Z}[X]/(X^n-1) \end{aligned}$$

Unbekannt in dieser Gleichung sind die Polynome f, g und α. Die Koeffizienten von f und α gehen aber in eine Linearkombination des Targets τ in L^{cs} ein

$$\tau = \sum_{i=0}^{n-1} f_{n-i} b_i + \sum_{i=n}^{2n-1} \alpha_{i-n} b_i = (\sigma(f), g).$$

Ziel ist die Konstruktion eines Gitters L^{keys} mit der $2n \times 3n$-Basismatrix

$$\left(\begin{array}{c|c|c} I & H & H*\Psi \\ \hline 0 & qI & Q^{neu} \end{array}\right),$$

wobei $h*\psi$ in $\mathbb{Z}[X]/(X^n-1)$ berechnet wird. Wir können nun die benötigten neuen q-Reduktionen in Abhängigkeit des α-Vektors berechnen.

$$q^{neu} = q(\alpha_0,\ldots,\alpha_{n-1})*\psi$$

Betrachte den j-ten Koeffizienten von q^{neu} für $0 \leq j < n$:

$$\begin{aligned}(q^{neu})_j &= q \cdot \sum_{\substack{i+k \equiv j \\ \bmod n}} \alpha_i \cdot \psi_k \\ &= q \cdot \alpha_i \cdot \psi_{n-i+j} \quad \text{für } 0 \leq i < n\end{aligned}$$

D.h. die j-te Spalte der Matrix $[Q^{neu}]_{0 \leq i,j < n}$ hat die Einträge

$$[Q^{neu}]_{i,j} = q \cdot \psi_{n-i+j},$$

da der Basisvektor b_{n+i+1}, der die i-te Zeile von $[Q^{neu}]$ enthält, α_i-mal in die Linearkombination des Targets eingeht. Es gilt mit Vektor-Matrix-Multiplikation $\alpha \cdot [Q^{neu}] = q^{neu}$.

Man mache sich die obige Konstruktion an einem Beispiel klar. Wähle den Vektor $\psi = (1, -1, 0, \ldots, 0)$. Dann erhalten wir

$$q^{neu} = q \cdot \alpha * \psi = q(\alpha_0 - \alpha_{n-1}, \alpha_1 - \alpha_0, \ldots, \alpha_{n-1} - \alpha_{n-2})$$

Dies liefert die Matrix

$$[Q^{neu}] = \begin{pmatrix} q & -q & 0 & \ldots & 0 & 0 \\ 0 & q & -q & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & q & -q \\ -q & 0 & 0 & \ldots & 0 & q \end{pmatrix}$$

und die folgende Identität gilt

$$(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \cdot [Q^{neu}] = q^{neu}.$$

Beachte, daß $[Q^{neu}] = q \cdot (L_{balance} \cup (-1, 0, \ldots, 0, 1))$. D.h. wir konstruieren auf natürliche Weise ein Erzeugendensystem für das Ideal der balancierten Polynome qI_b, da $h * \psi \in \mathbb{Z}[X]/(X^n - 1)$ balanciert ist.

Der neue Targetvektor ist $\tau^{neu} = (\sigma(f), g, g * \psi)$. Wir müssen nun die Normen von f- und g-Teil des Targets balancieren, d.h. wir benötigen Konstanten c_g, c_f mit $c_g\|g\| \approx \|g * \psi\|$ und $c_f\|f\| \approx \|(g, g * \psi)\|$. Angenommen wir kennen die Konstante c_g, dann können wir c_f berechnen.

$$\begin{aligned} c_f^2\|f\|^2 &\approx \|(g, g * \psi)\|^2 = \|g\|^2 + \|g * \psi\|^2 \approx (c_g^2 + 1)\|g\|^2 \\ \Rightarrow c_f &\approx \sqrt{c_g^2 + 1}\frac{\|g\|}{\|f\|}\end{aligned}$$

Um die Konstante c_g approximativ zu bestimmen, brauchen wir eine Abschätzung der Norm $\|g*\psi\|$. Dazu adaptieren wir eine Approximation von Coppersmith/Shamir [14].

$$\begin{aligned}\|g * \psi\|^2 &= \sum_{k=0}^{n-1} (g * \psi)_k^2 \\ &= \sum_{k=0}^{n-1} (\sum_{i=0}^{n-1} g_i \psi_{k-i})(\sum_{j=0}^{n-1} g_j \psi_{k-j}),\end{aligned}$$

wobei die Indizes wieder modulo n betrachtet werden. Für jedes Produkt $g_i\psi_{k-i}$ $g_j\psi_{k-j}$ in der Summe ist die Differenz $j-i$ der g-Indizes gleich der Differenz $(k-i)-(k-j)$ der ψ-Indizes. Sei $d=i-j$ diese gemeinsame Differenz und setze $l=k-j$, dann können wir die Summe umschreiben

$$\begin{aligned}
\|g*\psi\|^2 &= \sum_{d=0}^{n-1}(\sum_{i=0}^{n-1} g_i g_{i+d})(\sum_{l=0}^{n-1}\psi_l\psi_{l+d}) \\
&= (\sum_{i=0}^{n-1} g_i^2)(\sum_{l=0}^{n-1}\psi_l^2) + \sum_{d=1}^{n-1}(\sum_{i=0}^{n-1} g_i g_{i+d})(\sum_{l=0}^{n-1}\psi_l\psi_{l+d})
\end{aligned}$$

Angenommen g und ψ verhalten sich wie zufällige Vektoren. Für alle $n-1$ Terme mit $d>0$ sollte der Autokorrelationskoeffizient $\sum_i g_i g_{i+d}$ um einen Faktor von $1/\sqrt{n}$ kleiner sein als die korrespondierende Summe $\sum_i g_i^2$. Dies gilt analog für die Autokorrelation $\sum_l \psi_l\psi_{l+d}$, so daß das Produkt etwa um einen Faktor $1/n$ kleiner sein sollte. Weiterhin tauchen die Terme in der Summe mit zufälligem Vorzeichen auf, so daß sich einige gegenseitig auslöschen werden. Damit können wir annehmen, daß die zweite Summe viel kleiner als der erste Term ist. Dies führt zu der Approximation

$$\|g*\psi\| \approx \|g\|\cdot\|\psi\|.$$

Daher erhalten wir $c_g \approx \|\psi\|$. Wir können nun die ersten n Spalten der Basis von L^{keys} mit c_f multiplizieren und die zweiten n Spalten mit c_g, um den Gitterangriff zu optimieren.

6.8.2 Brechen der Schlüsselsymmetrie: Das Gitter L^{sym}

In Abschnitt 6.5 über balancierte Gitter haben wir ausgenutzt, daß nach Konstruktion $f(1)=1$ gilt. Nun wissen wir aber noch mehr über den geheimen Schlüssel f, nämlich daß er aus der Verteilung $D(d_f+1,d_f)$ gewählt wird, wobei d_f ein öffentlicher Parameter ist. D.h. wir wissen, daß f aus d_f+1 (1)-Koeffizienten und d_f (-1)-Koeffizienten besteht. Diese Eigenschaft wollen wir ausnutzen. Dazu müssen wir einen „Zähler" im Gitter integrieren, der bestimmt, wie oft die f-Koeffizienten in die Linearkombination eingehen. Man beachte, daß die δ-Spalte aus Bemerkung 6.5.5 ein solcher Zähler ist. Das Problem ist allerdings, daß der Targetvektor d_f (-1)-Koeffizienten enthält, die den Zähler wieder um den Betrag δ zurücksetzen. Da wir aber a priori nicht wissen, welche Basisvektoren zu (1)-Koeffizienten und welche zu (-1)-Koeffizienten korrespondieren, können wir diese Methode nicht ohne weiteres verwenden.

Das Problem im Targetvektor ist die Symmetrie von Einsen und Minuseinsen. Diese Symmetrie können wir zumindest partiell mit dem Gitter L^{part} brechen, das beliebige $\{0,\pm1\}$-Koeffizienten im Targetvektor fixiert. Angenommen wir fixieren den Teilvektor $(v_1,v_2,\ldots,v_r)\in\{0,1\}^r$ mit k Einsen. Dann wissen wir, daß die restlichen Basisvektoren mit d_f+1-k Einskoeffizienten und d_f Minuseinskoeffizienten eingehen. Man beachte, daß für NTRU-167 das Schlüsselpolynom $f\in_R D(61,60)$ 61 Einsen enthält. Je mehr (1)-Koeffizienten wir im Teilvektor v integrieren können, desto mehr wird die Symmetrie aufgebrochen. Der neue Targetvektor τ^{neu} enthält, wie in Abschnitt 6.2 beschrieben, an der Stelle des Teilvektors v Nullkoeffizienten. Da er nun – abhängig von k – mehr Minuseinsen enthält als Einsen, können wir die Norm $\|\tau^{neu}\|$ weiter senken, indem wir die verbesserten Algorithmen zum Lösen von Subset Sum Problemen nach M. Coster, A. Joux, B. LaMacchia, A. Odlyzko, C.P. Schnorr und J. Stern [12] durch Einführen eines Parameters μ adaptieren. Das Gitter $L^{sym}(\theta)$ wird von den Zeilenvektoren der folgenden $(2n+1 \times 2n+3)$-Matrix aufgespannt, wobei $k = \sum_{i=1}^{r} v_i$, $k' = 1-k$.

$$\left(\begin{array}{ccccccc|cccccc}
\theta & 0 & \dots & 0 & 0 & \dots & 0 & h_0 & \dots & h_{n-1} & \theta & \theta & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & \theta & 0 & \dots & 0 & h_{r-1} & \dots & h_{r-2} & \theta & \theta & 0 \\
0 & 0 & \dots & 0 & 1 & \dots & 0 & h_r & \dots & h_{r-1} & \theta & 0 & \theta \\
\vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 1 & h_{n-1} & \dots & h_{n-2} & \theta & 0 & \theta \\
\hline
0 & 0 & \dots & 0 & 0 & \dots & 0 & q & \dots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & q & 0 & 0 & 0 \\
\hline
\theta v_1 & \theta v_2 & \dots & \theta v_r & \mu & \dots & \mu & 0 & \dots & 0 & \theta & k\theta & k'\theta
\end{array}\right)$$

Die beiden letzten Spalten repräsentieren die neuen Zähler. Wir müssen wieder analog zu Satz 6.1.10 das Gewicht θ groß genug wählen. Sei τ^l der Shift des Targetvektors, der mit dem fixierten Teilvektor $(v_1, \dots, v_r)$ beginnt. OBdA sei $l = 0$, dann erhalten wir

$$\tau^{neu} = (0, \dots, 0, f_{n-r-1} - \mu, f_{n-r-2} - \mu, \dots, f_1 - \mu, g_0, g_1, \dots, g_{n-1}, 0, 0, 0)$$

Wir wollen nun die Norm von τ^{neu} minimieren.

Lemma 6.8.1 (Wahl von μ) *Die Norm des Targetvektors τ^{neu} ist minimal für $\mu = -\frac{k-1}{n-r}$.*

Beweis: Die Funktion $qnorm(\mu)$ beschreibe die Quadratnorm der ersten n Koeffizienten von τ^{neu}.

$$\begin{aligned}
qnorm(\mu) &= (d_f + 1 - k)(1-\mu)^2 + d_f(-1-\mu)^2 + \\
&\quad (n - r - (2d_f + 1 - k))(-\mu)^2 \\
&= (d_f + 1 - k)\mu^2 - 2(d_f + 1 - k)\mu + (d_f + 1 - k) + \\
&\quad d_f\mu^2 + 2d_f\mu + d_f + (n - r - 2d_f - 1 + k)\mu^2 \\
&= (n-r)\mu^2 + 2(k-1)\mu + 2d_f + 1 - k
\end{aligned}$$

Wir leiten nach μ ab:

$$\Rightarrow norm'(\mu) = 2(n-r)\mu + 2(k-1)$$

D.h. wir erhalten das Minimum für

$$\begin{aligned}
2(n-r)\mu &= -2(k-1) \\
\Rightarrow \mu &= -\frac{k-1}{n-r} \quad \square
\end{aligned}$$

Es ist eine interessante Frage, ob es möglich ist, die Schlüsselsymmetrie weiter aufzubrechen.

Literaturverzeichnis

[1] L.M. Adleman, „Factoring and Lattice Reduction“, Manuskript, Universität Süd-Kalifornien, 1995

[2] M. Ajtai, „The Shortest Vector Problem is NP-Hard for Randomized Reductions“, Proc. 30th Symposium on Theory of Computing, Seiten 10-19, 1998

[3] M. Ajtai, „Generating hard instances of lattice problems“, Proc. 28th Symposium on Theory of Computing, Seiten 99-108, 1996

[4] M. Ajtai, C. Dwork, „A Public-Key Cryptosystem with Worstcase/-Averagecase-Equivalence“, Proc. 29th Symposium on Theory of Computing, Seiten 284-293, 1997

[5] E. Bach, J. Shallit, „Algorithmic Number Theory Volume 1: Efficient Algorithms“, MIT Press, 1996

[6] M. Bellare, P. Rogaway, „Optimal asymmetric encryption – How to encrypt with RSA“, Advances in Cryptology – Eurocrypt 94, Lecture Notes in Computer Science 950, 1994

[7] J. Blömer, J-P. Seifert, „On the complexity of computing short linearly independent vectors and short bases in a lattice“, Proc. 31st Symposium on Theory of Computing, Seiten 711-720, 1999

[8] G. Brassard, „Relativized Cryptography“, 20th Symposium on Foundations of Computer Science, Seiten 383-391, 1979

[9] J.Y. Cai, „Some Recent Progress on the Complexity of Lattice Problems“, Electronic Colloqium on Computational Complexity, Report No.6 1999

[10] B. Chor, R. Rivest, „A knapsack-type public key cryptosystem based on arithmetic in finite fields“, Advances in Cryptology – Crypto 84, Springer Verlag 1985; revidierte Version in IEEE Transactions on Information Theory IT 34, Seiten 901-909, 1988

[11] H. Cohen, „A Course in Computational Algebraic Number Theory“, Graduate Texts in Mathematics 138, 3. Auflage, Springer Verlag, 1996

[12] M. Coster, A. Joux, B. LaMacchia, A. Odlyzko, C.P. Schnorr und J. Stern, „Improved Low-Density Subset Sum Algorithms“, Computational Complexity, Band 2, Seiten 111-128, 1992

[13] D. Coppersmith, „Attacking Non-Commutative NTRU“, IBM Research Report RC 20819 (92227) 25APR97, 24. April 1997

[14] D. Coppersmith, A. Shamir, „Lattice Attacks on NTRU“, Proc. Eurocrypt '97, Lecture Notes in Mathematics 1233, Springer Verlag, Seiten 52-61, 1997

[15] W. Diffie, M.E. Hellman, „New directions in cryptography“, IEEE Trans. on Information Theory 22, Seiten 644-654, 1976

[16] I. Dinur, G. Kindler und S. Safra, „Approximating CVP to within almost-polynomial factors is NP-hard“, Proc. 39th Symposium on Foundations of Computer Science, Seiten 99-109, 1998

[17] T. ElGamal, „A public key cryptosystem and a signature scheme based on discrete logarithms“, IEEE Transactions on Information Theory IT-31, Seiten 473-481, 1985

[18] O. Goldreich, S. Goldwasser, „On the possibility of basing Cryptography on the assumption that P $\neq$ NP“, Manuskript, 1998

[19] O. Goldreich, S. Goldwasser und S. Halevi, „Public-Key Cryptosystems from Lattice Problems“, Advances in Cryptology – CRYPTO '97, Lecture Notes in Computer Science 1294, Springer Verlag, Seiten 112-131, 1997

[20] S. Goldwasser, A. Micali, „Probabilistic encryption“, J. Computer and Systems Science 28, Seiten 270-299, 1984

[21] M.E. Hellman, R.C. Merkle, „Hiding information and signatures in trapdoor knapsacks“, IEEE Transactions on Information Theory IT-24, Seiten 525-530, 1978

[22] J.Hoffstein, J.H.Silverman, „A non-commutative version of the NTRU public key cryptosystem“, Manuskript, 14. Februar 1997

[23] J. Hoffstein, J. Pipher, J.H. Silverman, „NTRU: A Ring-Based Public Key Cryptosystem“, Algorithmic Number Theory (ANTS III), Portland, Lecture Notes in Computer Science 1423, J.P. Buhler (ed.), Springer-Verlag, Seiten 267-288, 1998

[24] J. Hoffstein, J. Pipher und J.H. Silverman, „NTRU: A new high speed public key cryptosystem“, Manuskript, Rump Session Crypto '96, 30. August 1996

[25] N. Koblitz, „Elliptic curve cryptosystems“, Compositio Math. 58, Seiten 13-44, 1986

[26] H. Koy, „Attacken auf das GGH-Cryptosystem mittels verbesserter Gitterreduktion“, Diplomarbeit, J.W. Goethe Universität Frankfurt, 1999

[27] J.C. Lagarias, A.M. Odlyzko, „Solving low-density subset sum problems“, Proc. 24th Symposium on Foundations of Computer Science, Seiten 1-10, 1983

[28] A. Lenstra, H.W. Lenstra Jr., „The Development of the Number Field Sieve“, Lecture Notes in Mathematics 1554, Springer-Verlag, 1993

[29] A.K. Lenstra, H.W. Lenstra and L. Lovasz, „Factoring Polynomials with Integer Coefficients“, Mathematische Annalen 261, Seiten 513-534, 1982

[30] A. May, „Cryptanalysis of NTRU-107“, Manuskript, 1999

[31] A. May, „New Lattice Attacks On NTRU“, Manuskript, 1999

[32] U.M. Maurer, „Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms“, Advances in Cryptology – Crypto '94, Lecture Notes in Computer Science 839, Seiten 271-281, 1994

[33] R.J. McEliece, „A public-key cryptosystem based on algebraic coding theory“, JPL Pasadena, DSN Progress Reports 42-44, Seiten 114-116, 1978

[34] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, „Handbook of Applied Cryptography“, CRC Press, 1997

[35] D. Micciancio, „The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant“, Proc. 39th Symposium on Foundations of Computer Science, Seiten 92-98, 1998

[36] P. Nguyen, „Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97“, Advances in Cryptology – Crypto '99, 1999

[37] P. Nguyen, „Lattice Attacks on NTRU, Revisited“, unveröffentlichtes Paper aus privater Kommunikation, 1999

[38] C. Pomerance, „Analysis and comparison of some integer factoring algorithms“, Computational Methods in Number Theory, Mathematisch Centrum Amsterdam, Seiten 98-139, 1982

[39] R.L. Rivest, A. Shamir, L. Adleman, „A method for obtaining digital signatures and public key cryptosystems“, Communications of the ACM 21, Seiten 120-126, 1978

[40] J.H. Silverman, „Commutative NTRU: Pseudo-code implementation“, Technical Report #001, unter http://www.ntru.com/documentcenter.htm, 1997

[41] J.H. Silverman, A. Odlyzko, „A Meet-In-The-Middle Attack on an NTRU Private Key“, Technical Report #004, unter http://www.ntru.com/-documentcenter.htm, 1997

[42] J.H. Silverman, „Hard Problems and Backdoors for NTRU and Other PKCS's“, Technical Report #005, unter http://www.ntru.com/document-center.htm, 1997

[43] J.H. Silverman, J. Hoffstein „Implementation Notes for NTRU PKCS Multiple Transmissions“, Technical Report #006, unter http://www.ntru.-com/documentcenter.htm, 1998

[44] J.H. Silverman, „Plaintext Awareness and the NTRU PKCS“, Technical Report #007, unter http://www.ntru.com/documentcenter.htm, 1998

[45] J.H. Silverman, „Efficient Conversions from Mod q to Mod p“, Technical Report #008, unter http://www.ntru.com/documentcenter.htm, 1998

[46] J.H. Silverman, „Invertibility in Truncated Polynomial Rings“, Technical Report #009, unter http://www.ntru.com/documentcenter.htm, 1998

[47] J.H. Silverman, „High-Speed Multiplication of (Truncated) Polynomials“, Technical Report #010, unter http://www.ntru.com/documentcenter.htm, 1999

[48] J.H. Silverman, „Wraps, Gaps, and Lattice Constants“, Technical Report #011, unter http://www.ntru.com/documentcenter.htm, 1999

[49] J.H. Silverman, „Estimated Breaking Times for NTRU Lattices“, Technical Report #012, unter http://www.ntru.com/documentcenter.htm, 1999

[50] J.H. Silverman, „Dimension-Reduced Lattices, Zero-Forced Lattices, and the NTRU Public Key Cryptosystem“, Technical Report #013, unter http://www.ntru.com/documentcenter.htm, 1999

[51] J.H. Silverman, „Almost Inverses and Fast NTRU Key Creation“, Technical Report #014, unter http://www.ntru.com/documentcenter.htm, 1999

[52] C.P. Schnorr, „Ganzzahlige Optimierung und Gitterreduktion“, Skriptum, J.W. Goethe Universität Frankfurt, 1996

[53] C.P. Schnorr, „Gittertheorie und algorithmische Geometrie, Reduktion von Gitterbasen und Polynomidealen“, Skriptum, J.W. Goethe Universität Frankfurt, 1997

[54] C.P. Schnorr, „A hierarchy of polynomial time lattice basis reduction algorithms“, Theoretical Computer Science 53, Seiten 201-224, 1987

[55] C.P. Schnorr, „Block reduced lattice basis and successive minima“, Combinatorics, Probability and Computing 3, Seiten 507-522, 1994

[56] C.P. Schnorr, „Factoring Integers and Computing Discrete Logarithms via Diophantine Approximation“, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 13, Seiten 172-181, 1993

[57] C.P. Schnorr, M. Euchner, „Lattice basis reduction: Improved practical algorithms and solving subset sum problems“, Mathematical Programming 66, Seiten 181-199, 1994

[58] C.P. Schnorr, H.H. Hörner, „Attacking the Chor Rivest cryptosystem by improved lattice reduction“, Proc. Eurocrypt '95, Lecture Notes in Computer Science 921, Springer-Verlag, Seiten 1-12, 1995

[59] D. Stinson, „Cryptography: Theory and Practice“, CRC Press, 1995

[60] P. van Emde Boas, „Another NP-complete partition problem and the complexity of computing short vectors in lattices“, Technischer Report 81-04, Mathematisches Department, Universität Amsterdam, 1981

[61] S. Vaudenay, „Cryptanalysis of the Chor-Rivest Cryptosystem“, Advances in Cryptology – Crypto '98, Lecture Notes in Computer Science 1462, Seiten 243-256, 1998

[62] J. Wolfart, „Einführung in die Zahlentheorie und Algebra“, Vieweg Verlag, 1996