

Potential und Grenzen der Anwendung von Gitterreduktionsalgorithmen in der Kryptographie



Technische Universität Darmstadt
Fachbereich Mathematik

Diplomarbeit
Raphael Overbeck
Betreuer: Prof. Dr. J. Buchmann

11. März 2004

Für meinen Großvater

Inhaltsverzeichnis

1	Vorwort	3
2	Sprache der Gitter	5
2.1	Darstellung von Gittern	5
2.2	Gitterprobleme - SVP und CVP	8
3	Reduktionsbegriffe	9
3.1	LLL-Reduktion	10
3.2	HKZ-Reduktion	13
3.3	BKZ-Reduktion	14
4	Gitterangriffe auf RSA	17
4.1	Angriffe bei zu kleinem öffentlichen Exponenten	17
4.2	Angriffe bei zu kleinem geheimen Exponenten	19
4.3	Angriffe bei teilweise bekanntem privaten Schlüssel	25
4.4	Faktorisierungsangriff	25
5	Gitterbasierte Kryptographie	27
5.1	Micciancios Kryptosystem	27
5.2	Angriffe auf Micciancios Kryptosystem	29
5.3	Eigenschaften des Orthogonalisationsdefekts	29
6	Das NTRU Kryptosystem	33
6.1	Gitterangriffe auf den privaten Schlüssel	35
6.2	Gitterangriffe auf den Schlüsseltext	37
6.3	Gitterangriffe auf alternative (gefälschte) Schlüssel	38
6.4	„Zero-Run Lattices“ und „Zero-Forced Lattices“	40
6.5	„Dimension-Reduced Lattices“	42
6.6	Entschlüsselungsfehler und ihre Folgen	43
6.7	Einige Überlegungen	46
7	Rucksackprobleme und Gitter	49
7.1	Lösungen bei geringer Dichte	49
7.2	Das Kryptosystem von Okamoto	51
7.3	Angriffe auf Okamotos Kryptosystem	53
A	Notationen und Konventionen	65
B	Grundlagen	67
B.1	Matrizen	67
B.2	Komplexität, \mathcal{NP} -Vollständigkeit	68
B.3	Public-Key Kryptosysteme und Sicherheitsbegriffe	71

1 Vorwort

Obwohl Gitter schon seit ca. 1900 erforscht wurden, wurden Gitter mit hoher Dimension erst durch die Arbeiten von A.K. Lenstra, H.W. Lenstra Jr. und L. Lóvasz [41] für die Informatik interessant. Ihre Bedeutung für die Kryptographie ergibt sich u.a. durch ihre Anwendung in zahlreichen Angriffen auf kryptographische Verfahren. Beispiele dafür sind die Kryptosysteme, die auf dem Rucksackproblem beruhen (Merkle-Hellman, [11], ect.), von denen die meisten inzwischen als gebrochen gelten ([1], [57]). Nur die Variante von Okamoto für Quantencomputer [58] konnte noch nicht angegriffen werden.

Nach dem Bekanntwerden von Gitterangriffen ([45], [14]) mußte die Wahl der Parameter von RSA und DSA überdacht werden. Die Algorithmen selbst sind nicht gebrochen worden.

Es gibt auch Versuche, Kryptosysteme zu entwickeln, welche auf den Gitterproblemen beruhen (z.B. GGH [25], Ajtai-Dwork[2]). Sie gelten jedoch als nicht praktikabel, da sie einen immensen Bedarf an Speicherplatz haben, wenn sie eine ausreichende Sicherheit bieten sollen.

Ziel der Arbeit ist es, eine Übersicht zu geben, wie „gut“ Gitterreduktionsalgorithmen sein müssen, um bessere Ergebnisse bei den Gitterangriffen auf Kryptosysteme zu erzielen.

Insbesondere im Hinblick auf zu erwartende Neuerungen durch Quantencomputer muß man die Sicherheit von vielen Kryptosystemen überdenken. So wird durch Quantencomputer z.B. ein Faktorisieren bzw. Bestimmen diskreter Logarithmen in Polynomialzeit möglich [67]. Dieses macht diverse populäre Kryptosysteme und deren Varianten, wie z.B. RSA, unsicher, sobald Quantencomputer verfügbar sind.

Da man bei \mathcal{NP} -schweren Problemen davon ausgeht, daß sie auch mit Quantencomputern nicht in Polynomialzeit lösbar sind [4], gewinnen Kryptosysteme, deren Sicherheit auf einem solch schweren Problem beruhen, an Bedeutung. Trotz einiger Schwierigkeiten gilt NTRU [29] als Kandidat für ein Public-Key-Kryptosystem der Zukunft. Es ist jedoch unklar, ob das NTRU-Problem wirklich \mathcal{NP} -schwer ist. Gleiches gilt für das SVP des NTRU-Gitters.

Die Gitterangriffe auf RSA basieren im wesentlichen auf den Ergebnissen von Coppersmith [12]. Ihre theoretischen Grenzen werden schon durch den LLL-Algorithmus erreicht. Anders verhält es sich bei der GGH-Variante von Micciancio [50] und NTRU. Hier versagt der LLL-Algorithmus wegen der ungenügenden Annäherung des kürzesten Gittervektors, so daß man auf BKZ-Varianten zurückgreifen muß.

Bei Micciancio hängt die Schwierigkeit des zu lösenden CVP von der Nachricht ab, während die Schwierigkeit des SVP des NTRU-Gitters von den gewählten Sicherheitsparametern und der damit verbundenen Wahrscheinlichkeit für Entschlüsselungsfehler abhängt. Da Micciancios Kryptosystem zu viel Speicherplatz benötigt, um bei entsprechender Sicherheit noch praktikabel zu sein [44], ist als einziges Kryptosystem noch das Okamoto-Kryptosystem interessant. Ein

Angriff auf eine Nachricht bedeutet hier, ein Rucksackproblem mit hoher Dichte zu lösen. Den privaten Schlüssel einer Okamoto-Instanz angreifen zu können, scheint bislang undenkbar.

Die Frage nach der Schwierigkeit des SVP im NTRU-Gitter, wenn man eine Schranke für die Wahrscheinlichkeit von Entschlüsselungsfehlern vorgibt, ist besonders interessant und dürfte in der nächsten Zeit zu lösen sein. Schwierig ist die Frage u.a. deshalb, da NTRU im Vergleich zu anderen Kryptosystemen über viele Sicherheitsparameter verfügt. Gegenüber einem Sicherheitsparameter bei RSA (in der ursprünglichen Version) hat NTRU sieben.

Ebenfalls interessant ist die Frage, ob man die Struktur der Rucksack-Instanzen des Okamoto-Kryptosystems auszunutzen kann um einzelne Schlüsseltexte anzugreifen. So gibt Okamoto die Anzahl der Gewichte vor, multiplikative Vielfache eines Gewichtes treten nicht auf und trotz hoher Dichte sind die Lösungen immer eindeutig.

Die Frage nach der notwendigen Güte von Gitterreduktionsalgorithmen für Angriffe auf RSA bzw. Micciancio scheinen erschöpfend behandelt worden zu sein. Obwohl noch nicht klar ist, ob das RSA-Gitter als zufällig anzusehen ist, geht man nach den Ergebnissen der bisher gemachten Experimente trotzdem davon aus.

Danksagung

Ich möchte zunächst Prof. Dr. J. Buchmann und C. Ludwig danken, die mir diese Arbeit ermöglicht haben. Mein Dank gilt ganz besonders meinen Eltern, die mich während meines gesamten Studiums und auf dem Weg dorthin in jeder Hinsicht unterstützt haben. Weiterhin möchte ich mich für die zahlreichen Ablenkungen, Aufheiterungen, Feten, Korrekturen und offenen Ohren bei Mo, Steffi, Claudia, Herrn Küpper, Clemens, Martina, Daniel, Julian, Tobi und Marcus bedanken.

2 Sprache der Gitter

Gitter als Punktfolgen des Vektorraums \mathbb{R}^n sind Gegenstand der Geometrie der Zahlen, die Minkowski um 1900 entwickelt hat. Die ganzzahligen Lösungen eines Gleichungssystems bilden ein typisches Gitter. Gitter bilden in diesem Sinne ein diskretes Analogon zu den linearen Räumen. In älteren Arbeiten wird oft noch die Sprache der quadratischen Formen anstelle der Linearen Algebra verwendet, um Gitter zu beschreiben.

2.1 Darstellung von Gittern

Definition 2.1.1 Zu beliebigen Vektoren $b_1, b_2, \dots, b_m \in \mathbb{R}^n$ bezeichne

$$L := L(b_1, b_2, \dots, b_m) := \sum_{i=1}^m b_i \mathbb{Z}$$

die von den Vektoren erzeugte additive Untergruppe des \mathbb{R}^n . Sind die Vektoren b_1, b_2, \dots, b_m linear unabhängig in \mathbb{R}^n , dann nennen wir $L(b_1, b_2, \dots, b_m)$ ein *Gitter der Dimension* (vom Rang) $\dim(L) := m$.

Ein Gitter heißt *volldimensional*, wenn es eine additive Untergruppe in \mathbb{R}^n , und vom Rang n ist.

Definition 2.1.2 Sei $b_1, b_2, \dots, b_m \in \mathbb{R}^n$ so, daß $L = L(b_1, b_2, \dots, b_m)$ ein Gitter ist, dann heißt b_1, b_2, \dots, b_m eine (geordnete) *Basis* von L und die Matrix $B = [b_1, b_2, \dots, b_m] \in \mathbb{R}^{n \times m}$ *Basismatrix* von L .

Wir schreiben in Kurzform: $L = L(B)$, wobei wir falls nicht anders definiert die Spalten von B , also B_i , als Gitterbasisvektoren betrachten wollen. Die Frage nach allen möglichen Basismatrizen für ein Gitter L beantwortet der folgende Satz:

Satz 2.1.3 Sei $B, B' \in \mathbb{R}^{n \times m}$. Es gilt $L(B) = L(B')$ genau dann, wenn es eine unimodulare Matrix $T \in \mathbb{Z}^{m \times m}$ gibt, so daß $B' = B \cdot T$.

Beweis. Sei $L(B) = L(B')$, dann ist klar, daß es ein $T \in \mathbb{Z}^{m \times m}$ gibt, so daß $B' = B \cdot T$. Da B' und B gleichen Zeilenrang haben, gilt $\det(T) \neq 0$. Wegen $B' \cdot T^{-1} = B$ und $L(B) = L(B')$ gilt, daß sowohl T als auch T^{-1} ganzzahlig sind, und somit auch ihre Determinanten. Daraus folgt, daß $|\det(T)| = 1$. Die Umkehrung ist offensichtlich. ■

Definition 2.1.4 Die *Determinante* $\det(L)$ des Gitters $L = L(B)$ ist definiert als

$$\det(L) := (\det(B^T B))^{\frac{1}{2}}.$$

Aus Satz 2.1.3 folgt, daß die Determinante eines Gitters L unabhängig von der Wahl der Gitterbasis ist. Ist $B \in \mathbb{R}^{n \times n}$ (also $L(B)$ volldimensional), so ist die Gitterdeterminante gleich dem Lebesgue-Maß (= euklidisches Volumen) der *Grundmasche* $M(B) = \{Bx \in \mathbb{R}^n : x \in \mathbb{R}^n, 0 \leq x_i < 1\}$ im \mathbb{R}^n .

Definition 2.1.5 Sei $\|\cdot\|$ eine beliebige Norm. Zu einem Gitter L vom Rang n sind die *sukzessiven Minima* λ_i , $i = 1, \dots, n$ bezüglich der Norm $\|\cdot\|$ definiert durch:

$$\lambda_i = \lambda_i(L) := \inf \left\{ r > 0 \left| \begin{array}{l} \text{Es gibt } i \text{ linear unabhängige} \\ \text{Vektoren } c_j \in L \text{ mit } \|c_j\| \leq r \\ \text{für } j = 1, \dots, i \end{array} \right. \right\}$$

Wir schreiben im folgenden $\lambda_{i,p}$, falls wir uns auf die L^p -Norm und λ_i , falls wir uns auf die euklidische Norm beziehen.

Bemerkung 2.1.6 Für zufällige Gitter der Dimension n im \mathbb{R}^n gilt nach der *Heuristik von Gauß* (siehe [42] bzw. [29]) die folgende Abschätzung für den Wert von λ_1 :

$$\sqrt{\frac{n}{2\pi e}} (\det(L))^{\frac{1}{n}} \leq \lambda_1 \leq \sqrt{\frac{n}{\pi e}} (\det(L))^{\frac{1}{n}}.$$

Diese Abschätzung ergibt sich, wenn man das Volumen der n -dimensionalen Kugel mit Radius r gegenüber dem Volumen der Grundmasche abschätzt. Soll ein Gitterpunkt in der entsprechenden Kugel liegen, so sollte das Volumen der Grundmasche kleiner gleich dem der Kugel mit Radius r sein.

Für den allgemeinen Fall liefert uns die *Hermite-Konstante*

$$\gamma_n := \sup \left\{ \frac{\lambda_1(L)^2}{(\det(L))^{\frac{2}{n}}} \left| L \subseteq \mathbb{R}^n \text{ vollständiges Gitter} \right. \right\}$$

eine scharfe Schranke: $\lambda_1(L) \leq \sqrt{\gamma_n} \cdot (\det(L))^{\frac{1}{n}}$. Nun wollen wir γ_n grob abschätzen (bessere Schranken sind in [62] zu finden):

Satz 2.1.7 (Minkowski 1896) Sei $L \in \mathbb{R}^n$ ein Gitter vom Rang m . Dann gilt:

$$\lambda_{1,\infty}(L) \leq (\det(L))^{\frac{1}{m}}$$

Beweis. Siehe [62]. ■

Die Schranke ist scharf, da $\lambda_{1,\infty}(\mathbb{Z}^n) = 1 = (\det(\mathbb{Z}^n))^{\frac{1}{n}}$. Weiterhin gilt für alle $x \in \mathbb{Z}^n$, daß $\|x\|_2 \leq \sqrt{n} \cdot \|x\|_\infty$, woraus $\lambda_{1,2} \leq \sqrt{n} (\det(L))^{\frac{1}{n}}$ folgt. Wir erhalten die Abschätzung $0 < \gamma_n \leq n$.

Satz 2.1.8 Sei L ein Gitter von Rang n , dann gilt

$$\det(L) \leq \prod_{i=1}^n \lambda_i(L) \leq (\gamma_n)^{\frac{n}{2}} \cdot \det(L)$$

Beweis. Siehe [62]. ■

Definition 2.1.9 (Gram-Schmidt-Orthogonalisierung) Zu der geordneten Gitterbasis b_1, b_2, \dots, b_m bezeichne

$$\pi_i : \mathbb{R}^n \longrightarrow \text{span}(b_1, b_2, \dots, b_{i-1})^\perp$$

die *orthogonale Projektion* derart, daß für alle $b \in \mathbb{R}^n$

$$b - \pi_i(b) \in \text{span}(b_1, b_2, \dots, b_{i-1}) .$$

Die Vektoren $\hat{b}_i := \pi_i(b_i)$ sind für $i = 1, 2, \dots, m$ paarweise orthogonal. Man berechnet sie durch das *Gram-Schmidt-Verfahren*

$$\hat{b}_1 := b_1, \hat{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \hat{b}_j \text{ für } i = 2, \dots, m ,$$

wobei $\mu_{i,j} := \frac{\langle b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle}$ die *Gram-Schmidt-Koeffizienten* sind, die im Falle $i = j$ als $\mu_{i,j} := 1$ und im Fall, daß $i < j$ als $\mu_{i,j} := 0$ definiert sind.

Diese Darstellung der Basisvektoren im Orthogonalsystem lautet in Matrixform: $[b_1, \dots, b_m] = [\hat{b}_1, \dots, \hat{b}_m] \cdot (\mu_{i,j})_{1 \leq i, j \leq m}^T$ und damit $\det(L) = \prod_{i=1}^m \|\hat{b}_i\|$.

Lemma 2.1.10 Für jede Basis b_1, \dots, b_n eines Gitters $L \in \mathbb{R}^n$ gilt:

$$\min_i \|\hat{b}_i\| \leq \lambda_1(L) .$$

Beweis. Siehe Lemma 5.3.11 aus [27]. ■

Wir wollen nun das zu einem Gitter L *duale* Gitter betrachten.

Definition 2.1.11 Zur Gitterbasis $b_1, \dots, b_m \in \mathbb{R}^n$ ist die *duale Gitterbasis* definiert als b_1^*, \dots, b_m^* mit

$$\langle b_i, b_{m-j+1}^* \rangle = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{sonst.} \end{cases}$$

Es ist klar, daß mit $U_m = (u_{i,j})_{1 \leq i, j \leq m}$, $u_{i,j} = 1$ falls $j = m - i + 1$, 0 sonst, die Beziehung $B \cdot B^* \cdot U_m = \text{Id}_m$ gilt. Weiter kann man sich überlegen, daß

$$\hat{b}_{m-i+1}^* = \frac{\hat{b}_i}{\|\hat{b}_i\|^2}$$

für alle $i = 1, \dots, m$ gilt, sowie $\|\hat{b}_i\| = 1/\|\hat{b}_{m-i+1}^*\|$, was man mit der Darstellung der Basismatrix mit Hilfe der Gram-Schmidt-Koeffizienten leicht sieht.

2.2 Gitterprobleme - SVP und CVP

In einem Gitter sind insbesondere kurze Gittervektoren und die Annäherung eines Punktes im \mathbb{R}^n durch einen Gittervektor interessant. In den meisten Anwendungen von Gittern wird ein Problem auf eines dieser beiden zurückgeführt.

Definition 2.2.1 (CVP) Das *nächster Vektor*-Problem lautet:

- Gegeben eine Basismatrix B eines Gitters $L \subset \mathbb{R}^n$ und $y \in \mathbb{R}^n$.
- Finde $x \in L$, so daß $\|x - y\| \leq \|x' - y\|$ für alle $x' \in L$ ist.

Dabei ist die Verwendete Norm meist die euklidische, falls nicht abweichend angegeben.

Ist ein CVP durch B, y gegeben, so sprechen wir von dem CVP für y in $L(B)$. Babai entwickelte eine Methode [3], um das CVP in Polynomialzeit zu lösen, wenn der gegebene Vektor y einen relativ kleinen Abstand zum nächsten Gittervektor hat. Im allgemeinen ist das Problem jedoch \mathcal{NP} -schwer [26].

Definition 2.2.2 (α -SVP) Das *kürzester Vektor*-Problem mit *Annäherungsfaktor* α lautet:

- Gegeben eine Basismatrix B eines Gitters L und $\alpha \in \mathbb{R}$.
- Finde $x \in L, x \neq 0$, so daß $\|x\| \leq \alpha \lambda_1(L)$ ist.

Dabei ist die Verwendete Norm meist die euklidische, falls nicht abweichend angegeben.

Nach [51] ist das α -SVP \mathcal{NP} -schwer für jedes $\alpha \leq \sqrt{2}$. Wir bezeichnen das 1-SVP kurz als das *kürzester Vektor*-Problem (SVP) und einen Lösungsvektor als *kürzesten Gittervektor*. O. Goldreich, D. Micciancio, S. Safra und J.P. Seifert haben in [26] gezeigt, daß man das SVP mit einem CVP-Orakel in Polynomialzeit lösen kann.

3 Reduktionsbegriffe

Von den verschiedenen Darstellungen eines Gitters sind besonders diejenigen interessant, die eine gute Ausgangsmöglichkeit zum Lösen des α -SVP bzw. CVP bieten. Deshalb wurden verschiedene Begriffe einer *reduzierten* Basis gebildet. Je stärker der Reduktionsbegriff, desto höher wird der Aufwand, zu einer gegebenen Basismatrix eine reduzierte Darstellung zu berechnen. Kriterien für eine reduzierte Basis sind meist kleine Gram-Schmidt-Koeffizienten, eine gute Annäherung des kürzesten Gittervektors und relativ orthogonale Basisvektoren.

Definition 3.0.3 Eine geordnete Basis b_1, b_2, \dots, b_m heißt *längenreduziert* falls

$$\forall_{1 \leq j < i \leq m} : |\mu_{i,j}| \leq \frac{1}{2}$$

Die Berechnung einer längenreduzierten Basis eines Gitters kann mit dem folgenden Algorithmus in Polynomialzeit erfolgen, wobei $\lceil \cdot \rceil$ das normale Runden zu der nächsten ganzen Zahl bedeutet:

Algorithmus 3.0.4 (*k*-LRED)

Eingabe: $b_1, \dots, b_m \in \mathbb{Z}^n$ (eine Gitterbasis),
 $(\mu_{i,j}) \in \mathbb{Q}^{m \times m}$ (Die Matrix der Gram-Schmidt-Koeffizienten)
 $k \in \mathbb{N}$ mit $1 \leq k \leq m$
Ausgabe: b_1, \dots, b_m (Gitterbasis mit $|\mu_{j,k}| \leq \frac{1}{2}$ für $j = 1, \dots, k-1$)
 $(\mu_{i,j})$.

- FOR ($j = k-1, \dots, 1$) DO {
 - IF ($|\mu_{k,j}| > \frac{1}{2}$) THEN {
 - * $b_k = b_k - \lceil \mu_{k,j} \rceil b_j$
 - * FOR ($i = 1, \dots, m$) DO {
 - $\mu_{k,i} = \mu_{k,i} - \lceil \mu_{k,j} \rceil \mu_{j,i}$
 - }
 - * }
 - }
- RETURN $b_1, \dots, b_k, \dots, b_m, (\mu_{i,j})$

Man kann sich leicht überlegen, daß sich die $\hat{b}_1, \dots, \hat{b}_m$ während der gesamten Laufzeit nicht verändern und somit die $\mu_{j,\cdot}$ für $j \neq k$ konstant bleiben. Die aktuelle Matrix der Gram-Schmidt-Koeffizienten ist nach jedem Durchlauf der äußeren FOR-Schleife korrekt, da

$$\begin{aligned} \mu_{k,i}^{\text{neu}} &= \frac{\langle \hat{b}_k^{\text{neu}}, \hat{b}_i^{\text{neu}} \rangle}{\langle \hat{b}_i^{\text{neu}}, \hat{b}_i^{\text{neu}} \rangle} = \frac{\langle \hat{b}_k^{\text{alt}} - \lceil \mu_{k,j}^{\text{alt}} \rceil \hat{b}_j^{\text{alt}}, \hat{b}_i^{\text{neu}} \rangle}{\langle \hat{b}_i^{\text{neu}}, \hat{b}_i^{\text{neu}} \rangle} \\ &= \frac{\langle \hat{b}_k^{\text{alt}}, \hat{b}_i^{\text{neu}} \rangle}{\langle \hat{b}_i^{\text{neu}}, \hat{b}_i^{\text{neu}} \rangle} - \frac{\langle \lceil \mu_{k,j}^{\text{alt}} \rceil \hat{b}_j^{\text{alt}}, \hat{b}_i^{\text{neu}} \rangle}{\langle \hat{b}_i^{\text{neu}}, \hat{b}_i^{\text{neu}} \rangle} = \mu_{k,i}^{\text{alt}} - \lceil \mu_{k,j}^{\text{alt}} \rceil \mu_{j,i}^{\text{alt}} \end{aligned}$$

für alle $i = 1, \dots, m$ gilt. Weiterhin ist

$$\mu_{k,i}^{\text{neu}} = \begin{cases} \mu_{k,i}^{\text{alt}} - \left\lceil \mu_{k,j}^{\text{alt}} \right\rceil \mu_{j,i}^{\text{alt}} & \text{falls } i < j \\ \mu_{k,i}^{\text{alt}} - 1 \cdot \left\lceil \mu_{k,i}^{\text{alt}} \right\rceil & \text{falls } i = j \\ \mu_{k,i}^{\text{alt}} & \text{sonst.} \end{cases}$$

Da deshalb nach jedem Durchlauf der äußeren Schleife $\|\mu_{k,j}\| \leq \frac{1}{2}$ gilt, und j daraufhin reduziert wird, gilt $|\mu_{k,i}| \leq \frac{1}{2}$ für alle $i = 1, \dots, k-1$ wenn der Algorithmus terminiert. Offensichtlich ist die Laufzeit $\mathcal{O}(mn)$.

Eine Längenreduzierung des Vektors b_k beeinflusst die Längenreduziertheit der restlichen Basisvektoren nicht. Führt man k -LRED für alle $k = 1, \dots, m$ durch, so erhält man eine längenreduzierte Basis.

Bemerkung 3.0.5 Für jede geordnete, längenreduzierte Basis b_1, \dots, b_m gilt:

$$\|b_i\|^2 \leq \|\hat{b}_i\|^2 + \frac{1}{4} \sum_{j=1}^{i-1} \|\hat{b}_j\|^2 \text{ für } i = 1, \dots, m.$$

3.1 LLL-Reduktion

Der Begriff der längenreduzierten Basis ist sehr schwach, da er z.B. keine Angabe einer Grenze für das Verhältnis $\lambda_1/\|b_1\|$ ermöglicht. Deswegen führen wir den stärkeren Begriff der *LLL-reduzierten* Basis ein. A.K. Lenstra, H.W. Lenstra und L. Lóvasz präsentieren in [41] einen Polynomzeitalgorithmus, der eine beliebige Basis in eine LLL-reduzierte transformiert. Der Algorithmus findet dabei einen Gittervektor, der nicht mehr als $2^{(n-1)/2}$ mal länger ist, als der kürzeste Gittervektor.

Definition 3.1.1 Eine geordnete Gitterbasis $b_1, \dots, b_m \in \mathbb{R}^n$ heißt LLL-reduziert mit δ , $\frac{1}{4} < \delta \leq 1$, wenn

1. $|\mu_{i,j}| \leq \frac{1}{2}$ für $1 \leq j < i \leq m$,
2. $\delta \|\hat{b}_{k-1}\|^2 \leq \|\hat{b}_k\|^2 + \mu_{k,k-1}^2 \|\hat{b}_{k-1}\|^2$ für $k = 2, \dots, m$.

Der Parameter δ kontrolliert die Güte der reduzierten Basis. Je kleiner δ um so schwächer ist die Reduktion. A.K. Lenstra, H.W. Lenstra und L. Lóvasz studieren die LLL-Reduktion in [41] speziell für den Parameter $\delta = \frac{3}{4}$.

Algorithmus 3.1.2 (LLL)

Eingabe: $b_1, \dots, b_m \in \mathbb{Z}^n$ (eine Gitterbasis),
 $(\mu_{i,j}) \in \mathbb{Q}^{m \times m}$ (Die Matrix der Gram-Schmidt-Koeffizienten)
 $\|\hat{b}_i\|^2$ für $i = 1, \dots, m$
 $\frac{1}{4} < \delta \leq 1$

Ausgabe: mit δ LLL-reduzierte Basis $r_1, \dots, r_m \in \mathbb{Z}^n$

- $k = 2$
- WHILE ($k \leq m$) DO {
 - Berechne $(\mu_{i,j})$ und $\|\hat{b}_1\|, \dots, \|\hat{b}_m\|$.
 - $b_1, \dots, b_m, (\mu_{i,j}) = k\text{-LRED}(b_1, \dots, b_m, (\mu_{i,j}), k)$
 - IF ($\delta\|\hat{b}_{k-1}\|^2 > \|\hat{b}_k\|^2 + \mu_{k,k-1}^2\|\hat{b}_{k-1}\|^2$) THEN {
 - * Vertausche b_k und b_{k-1}
 - * $k = \max\{k-1, 2\}$
 - * }
 - ELSE {
 - * $k = k + 1$ }
 - }
- RETURN b_1, \dots, b_m

Die Korrektheit ergibt sich daraus, daß die Teilbasis b_1, \dots, b_{k-1} stets LLL-reduziert ist. Es genügt also zu zeigen, daß der Algorithmus terminiert. Dazu wollen wir sehen, daß die Bedingung $\delta\|\hat{b}_{k-1}\|^2 > \|\hat{b}_k\|^2 + \mu_{k,k-1}^2\|\hat{b}_{k-1}\|^2$ höchstens $\log_{1/\delta} \left(\prod_{i=1}^{m-1} \det(L(b_1, \dots, b_i))^2 \right)$ mal erfüllt ist, wobei wir uns auf die Eingabewerte beziehen. Bei einem Austausch von b_k und b_{k-1} werden die Werte $\det(L(b_1, \dots, b_i))$ für $i \neq k-1$ nicht verändert. Weiterhin gilt nach einem Austausch, daß

$$\det(L(b_1^{\text{alt}}, \dots, b_{k-1}^{\text{alt}})) = \prod_{i=1}^{k-1} \|\hat{b}_i^{\text{alt}}\| \leq \delta \prod_{i=1}^{k-1} \|\hat{b}_i^{\text{neu}}\| = \det(L(b_1^{\text{neu}}, \dots, b_{k-1}^{\text{neu}})) ,$$

da $\delta\|\hat{b}_{k-1}^{\text{alt}}\|^2 > \|\hat{b}_k^{\text{alt}}\|^2 + \mu_{k,k-1}^2\|\hat{b}_{k-1}^{\text{alt}}\|^2 \geq \|\hat{b}_{k-1}^{\text{neu}}\|^2$. Ein Austausch von b_k und b_{k-1} ist der einzige Schritt, in dem der Wert $\prod_{i=1}^{m-1} \det(L(b_1, \dots, b_i))^2$ verändert wird. Dieser Wert ist ganzzahlig, positiv und wird bei jedem Austausch echt kleiner, womit die Anzahl der Austausche A beschränkt ist. Es gilt weiter, daß

$$\prod_{i=1}^{m-1} \det(L(b_1^{\text{Start}}, \dots, b_i^{\text{Start}}))^2 \geq \frac{1}{\delta^A} \cdot \prod_{i=1}^{m-1} \det(L(b_1^{\text{Ende}}, \dots, b_i^{\text{Ende}}))^2 \geq \frac{1}{\delta^A} ,$$

woraus die Behauptung folgt. Man beachte, daß

$$\prod_{i=1}^{m-1} \det(L(b_1^{\text{Start}}, \dots, b_i^{\text{Start}}))^2 \leq \max_{i=1, \dots, m} \|b_i^{\text{Start}}\|^{2m(m-1)}$$

gilt, d.h. A polynomiell in der Eingabelänge beschränkt ist. Es bleibt zu bestimmen, wie aufwendig es ist, $(\mu_{i,j})$ und $\|\hat{b}_1\|, \dots, \|\hat{b}_m\|$ zu berechnen bzw. wieviele Schritte pro Austausch durchgeführt werden. Es ergibt sich, daß LLL eine Laufzeit von $\mathcal{O}\left(m^3 n \cdot \log_{1/\delta}(\max_i \|b_i\|)\right)$ hat. Einen detaillierten Beweis (mit einer besseren Abschätzung der Laufzeit) findet man in [41] bzw. [64].

Satz 3.1.3 Jede mit δ LLL-reduzierte Basis b_1, \dots, b_m des Gitters L erfüllt

- (i) $(\delta - \frac{1}{4})^{j-1} \leq \|\hat{b}_j\|^2 \lambda_j(L)^{-2}$ für $j = 1, \dots, m$
- (ii) $\|b_j\|^2 \lambda_j(L)^{-2} \leq (\delta - \frac{1}{4})^{1-m}$ für $j = 1, \dots, m$
- (iii) $\|b_k\|^2 \leq (\delta - \frac{1}{4})^{1-j} \|\hat{b}_j\|^2$ für $k \leq j$
- (iv) $\|b_1\|^2 \leq (\delta - \frac{1}{4})^{\frac{1-m}{2}} (\det(L))^{\frac{2}{m}}$
- (v) $\prod_{i=1}^m \|b_i\|^2 \leq (\delta - \frac{1}{4})^{-\binom{m}{2}} (\det(L))^2$.

Beweis. Siehe [41] bzw. [62]. ■

Wählt man wie in [41] $\delta = \frac{3}{4}$, so ergibt sich $\|b_1\| \leq \lambda_1(L) 2^{\frac{m-1}{2}}$ beziehungsweise $\|b_1\| \leq 2^{\frac{m}{2}} (\det(L))^{\frac{1}{m}}$.

Lemma 3.1.4 Für eine LLL-reduzierte Basis b_1, b_2, \dots, b_m gilt

$$\|\hat{b}_i\|^2 \leq \left(\delta - \frac{1}{4}\right)^{j-i} \|\hat{b}_j\|^2,$$

für $1 \leq i \leq j \leq m$.

Beweis. Es gilt nach Definition:

$$\delta \|\hat{b}_i\|^2 \leq \|\hat{b}_{i+1}\|^2 + \mu_{i+1,i}^2 \|\hat{b}_i\|^2 \leq \|\hat{b}_{i+1}\|^2 + \frac{1}{4} \|\hat{b}_i\|^2$$

und somit $(\delta - \frac{1}{4}) \|\hat{b}_i\|^2 \leq \|\hat{b}_{i+1}\|^2$. Der Rest der Aussage folgt durch Induktion. ■

Mit diesem Lemma können wir auch eine Schranke für die Länge des zweiten Basisvektors einer LLL-reduzierten Basis angeben.

Folgerung 3.1.5 Für eine mit $\delta = 3/4$ LLL-reduzierte Basis b_1, b_2, \dots, b_m mit $\|b_1\| \geq 1$ gilt

$$\|b_2\| \leq 2^{\frac{m}{2}} (\det(L))^{\frac{1}{m-1}} .$$

Beweis. Zunächst können wir mit Lemma 3.1.4 den Wert $\|\hat{b}_2\|$ abschätzen:

$$\det(L) = \prod_{i=1}^m \|\hat{b}_i\| \geq \|\hat{b}_1\| \cdot \|\hat{b}_2\|^{m-1} \cdot 2^{-(m-1)^2/2} \geq \|\hat{b}_2\|^{m-1} \cdot 2^{-(m-1)^2/2},$$

da $b_1 = \hat{b}_1$ und $\|\hat{b}_{k-1}\|^2 \leq 2\|\hat{b}_k\|^2$. Weiterhin können wir schließen, daß

$$\|b_2\|^2 \leq \|\hat{b}_2\|^2 + \frac{1}{4} \|b_1\|^2 \leq 2^{m-1} \det(L)^{\frac{2}{m-1}} + 2^{m-2} \det(L)^{\frac{2}{m}} \leq 2^m \det(L)^{\frac{2}{m-1}},$$

da die Basis längenreduziert ist. ■

Das CVP $[b_1, \dots, b_m], y$ läßt sich in Polynomialzeit lösen, wenn y nicht zu weit vom nächsten Gittervektor entfernt und das Gitter volldimensional ist. Dies geschieht mit Hilfe des folgenden Algorithmus aus [3]:

Algorithmus 3.1.6 (ROUNDING-OFF)

Eingabe: $b_1, \dots, b_n \in \mathbb{R}^n$, eine mit $\delta = \frac{3}{4}$ LLL-reduzierte Basis,
 $y \in \mathbb{R}^n$.

Ausgabe: $x \in L(b_1, \dots, b_n)$.

- Berechne $\beta_1, \dots, \beta_n \in \mathbb{R}^n$, so daß $y = \sum_{i=1}^n \beta_i b_i$.
- FOR ($i = 1, \dots, n$) DO {
 - $\alpha_i = \lceil \beta_i \rceil$
 - }
- RETURN $w = \sum_{i=1}^n \alpha_i b_i$

Die Berechnung einer ausreichenden Annäherung der β_i kann in Polynomialzeit erfolgen, so daß ROUNDING-OFF in Polynomialzeit ist.

Satz 3.1.7 Ein CVP im \mathbb{R}^n sei durch $[b_1, \dots, b_n], y$ beschrieben (das Gitter $L(b_1, \dots, b_n)$ ist damit volldimensional). Dann kann man das CVP mit dem Algorithmus ROUNDING-OFF in Polynomialzeit lösen, falls

$$\min_{x \in L(b_1, \dots, b_n)} \|x - y\|_\infty < \frac{1}{2 \max_i \|b_i^*\|_1}.$$

Beweis. Sei x der (eindeutige) Vektor, der das CVP löst. Der Algorithmus gibt genau dann x zurück, wenn $\text{ROUNDING-OFF}(b_1, \dots, b_n, z) = 0$, wobei $z := y - x$. Ist dieses nicht der Fall, so gilt $|\beta_j| = |(B^{-1}z)_j| = |\langle b_{n-j+1}^*, z \rangle| \geq \frac{1}{2}$ für ein $j \in \{1, \dots, n\}$. Nach Voraussetzung gilt allerdings

$$|\langle b_i^*, z \rangle| = \left| \sum_{j=1}^n b_{ij}^* z_j \right| \leq \|x - y\|_\infty \sum_{j=1}^n |b_{ij}^*| < \frac{1}{2},$$

für alle $i = 1, \dots, n$, was ein Widerspruch ist. (Siehe auch [25]). ■

Analog kann man den Satz beweisen, wenn man die Ungleichung in dem Satz durch $\min_{x \in L(b_1, \dots, b_n)} \|x - y\|_2 < (2 \max_i \|b_i^*\|_2)^{-1}$ ersetzt. Selbst wenn die Voraussetzungen für Satz 3.1.7 nicht erfüllt sind, findet ROUNDING-OFF Gittervektoren deren Abstand zu dem Vektor y höchstens $1 + 2n(9/2)^{n/2}$ mal größer ist, als zu dem Vektor x , der das CVP löst. Babai präsentiert in [3] einen weiteren Algorithmus, der Gittervektoren findet, deren Abstand zu dem Vektor y höchstens $2^{n/2}$ mal größer ist, als $\|x - y\|$.

3.2 HKZ-Reduktion

In vielen Situationen ist es nicht ausreichend, den kürzesten Gittervektor nur mit dem Faktor anzunähern, der von dem LLL-Algorithmus erreicht wird. Auch in Hinsicht auf die Orthogonalität der Basisvektoren ist eine LLL-reduzierte Basis nicht ausreichend, deshalb wollen wir einen weiteren zentralen Reduziertheitsbegriff für Gitterbasen einführen. Dazu definieren wir zunächst $L_i := \pi_i(L)$.

Definition 3.2.1 Eine geordnete Basis b_1, b_2, \dots, b_m heißt *nach Hermite und Korkine-Zolotareff reduziert* (HKZ-reduziert), wenn sie längenreduziert ist und

$$\|\hat{b}_i\| = \lambda_1(L_i) \text{ für } i = 1, \dots, m.$$

HKZ-reduzierte Basen werden oft auch als Korkine-Zolotareff bzw. Hermite reduziert bezeichnet, da beide unabhängig voneinander äquivalente Definitionen gegeben haben. Aus der Definition folgt direkt, daß $\|\hat{b}_j\|^2 \leq \|b_i\|^2$ für $j \leq i$. Damit haben wir eine in gewissem Sinne gute Gitterbasis definiert. Die Berechnung kann z.B. mit der Hilfe des ENUM-Algorithmus aus [64] erfolgen. Derzeit sind jedoch keine Polynomialzeitalgorithmen bekannt.

Die Definition einer HKZ-reduzierten Basis sagt nicht, daß b_i der kürzeste Gittervektor ist, der von b_1, \dots, b_{i-1} linear unabhängig ist, was man an dem folgenden Beispiel sieht:

Beispiel 3.2.2 Wir betrachten das von den folgenden Vektoren aufgespannte (volldimensionale) Gitter im \mathbb{R}^4 :

$$[b_1, b_2, b_3, b_4] = \begin{pmatrix} 2 & 0 & 1 & -1 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Basis ist demnach längenreduziert. Sie ist HKZ-reduziert, da für $j = 1, \dots, 3$ die Ungleichung $\lambda_1(L(\pi_j(b_j), \dots, \pi_j(b_4))) \geq \min_{i>j} \|\hat{b}_i\| = 2$ gilt. Der Gittervektor $(0, 0, 1, 2)^\top = b_3 + b_4$ ist jedoch kürzer als b_3 bzw. b_4 .

Der Begriff der HKZ-reduziertheit ist trotzdem sehr stark, da er das Verhältnis von λ_i zu $\|b_i\|$ für alle $i = 1, \dots, m$ beschränkt.

Satz 3.2.3 Für jede HKZ-reduzierte Basis b_1, \dots, b_m des Gitters L gilt

$$\frac{4}{i+3} \leq \frac{\|b_i\|^2}{\lambda_i(L)^2} \leq \frac{i+3}{4}$$

Beweis. siehe [59] ■

Man vergleiche hierzu die Abschätzung, welche sich aus der LLL-reduziertheit einer Basis mit $\delta = 3/4$ ergibt:

$$2^{1-i} \leq \frac{\|b_i\|^2}{\lambda_i(L)^2} \leq 2^{n-1}$$

3.3 BKZ-Reduktion

Der Begriff der HKZ-Reduktion ist zwar stark, jedoch ist er in der Praxis bei hochdimensionalen Gittern meist nicht erreichbar, da hierfür keine Polynomialzeitalgorithmen bekannt sind. Die Laufzeit einer HKZ-reduktion (z.B.

durch ENUM) ist bei konstanter Dimension polynomiell in der Eingabelänge. Daher liegt es nahe, die Basis in Blöcke aufzuteilen und für diese eine HKZ-reduzierte Darstellung zu wählen. Das führt uns zu dem Begriff der *Block-Korkine-Zolotarev-reduzierten* Basen.

Definition 3.3.1 Eine geordnete Basis b_1, \dots, b_m heißt *k-reduziert*, wenn sie längenreduziert ist, und für $i = 1, \dots, m-k+1$ die Blöcke $\pi_i(b_i), \dots, \pi_i(b_{i+k-1})$ der Länge k HKZ-reduziert sind.

Wir wollen die Länge des kürzesten Vektors einer k -reduzierten Basis bzw. das Verhältnis der $\|b_i\|$ zu den λ_i für $i = 1, \dots, m$ abschätzen.

Satz 3.3.2 Für jede k -reduzierte Basis b_1, \dots, b_m eines Gitters L gilt:

$$\begin{aligned} (i) \quad & \frac{\|\hat{b}_i\|^2}{\lambda_i(L)} \leq \gamma_k^{2 \cdot \frac{m-i}{k-1}} \\ (ii) \quad & \frac{\|b_i\|^2}{\lambda_i(L)} \leq \gamma_k^{2 \cdot \frac{m-i}{k-1}} \cdot \frac{i+3}{4} \\ (iii) \quad & \frac{\lambda_i(L)^2}{\|b_i\|^2} \leq \gamma_k^{2 \cdot \frac{i-1}{k-1}} \cdot \frac{i+3}{4} \\ (iv) \quad & \|b_1\| \leq \gamma_k^{\frac{m-1}{k-1}} \cdot \lambda_1(L) \end{aligned}$$

Beweis. siehe [62] ■

Damit ergibt sich nun $\|b_1\|^2 \leq k^{\frac{m-1}{2(k-1)}} \lambda_1(L)^2 \leq k^{\frac{m}{k}} \lambda_1(L)^2$ als die vereinfachte Schranke für $\|b_1\|$ von k -reduzierte Basen.

Definition 3.3.3 Sei b_1, \dots, b_m eine Basis eines Gitters $L \subseteq \mathbb{R}^n$, $2 \leq \beta < m$ sowie $1/4 < \delta \leq 1$. Dann heißt diese Basis β, δ -reduziert mit, falls sie längenreduziert ist und

$$\forall_{1 \leq i \leq m-1} \sqrt{\delta} \cdot \|\hat{b}_i\| \leq \lambda_1(L_i(b_1, \dots, b_{\min\{i+\beta-1, m\}}))$$

Da $L_i(b_1, \dots, b_{\min\{i+\beta-1, m\}}) = L(\pi_i(b_1), \dots, \pi_i(b_{\min\{i+\beta-1, m\}}))$, kann man leicht sehen, daß eine mit $\beta, 1$ -reduzierte Basis auch k -reduziert ist. Eine Basis ist $2, \delta$ -reduziert mit $1/3 < \delta \leq 1$, genau dann, wenn sie mit δ LLL-reduziert ist (vergleiche [64] bzw. [60]), wobei jede $2, \delta$ -reduzierte Basis mit δ LLL-reduziert ist.

Eine Berechnung einer BKZ-reduzierten Basis kann mit dem BKZ-Algorithmus aus [64] erfolgen. Obwohl bislang nicht nachgewiesen wurde, daß BKZ in Polynomialzeit ist, verhält sich die Laufzeit in der Praxis relativ gut.

4 Gitterangriffe auf RSA

Wir erinnern uns an das RSA-Verfahren: Man wähle zwei große Primzahlen p, q , bilde $pq = n$ und wähle eine zufällige ganze Zahl $1 < e < \varphi(n) = (p-1)(q-1)$ mit $\text{ggT}(e, \varphi(n)) = 1$. Danach kann man mit dem erweiterten Euklidischen Algorithmus $d \in \mathbb{N}$ so berechnen, daß $ed \equiv 1 \pmod{\varphi(n)}$. Der öffentliche Schlüssel ist (n, e) und der private (d) . Die Verschlüsselung von $m \in \mathbb{Z}/n\mathbb{Z}$ erfolgt durch Berechnung von $c = m^e \pmod{n}$, die Entschlüsselung von c durch die Berechnung von $m = c^d \pmod{n}$. Die Entschlüsselung ist nach dem kleinen Satz von Fermat korrekt.

Obwohl es leicht zu sehen ist, daß derjenige, der n faktorisieren kann, das RSA-Verfahren brechen kann, ist der Beweis der Umkehrung bislang nicht gelungen. Damit ist RSA (zur Zeit) nicht beweisbar sicher.

Nach [7] ist es nicht klar, ob man sich bei der Auswahl des öffentlichen Exponenten e auf das Intervall $[1, \varphi(n)]$ beschränken sollte.

4.1 Angriffe bei zu kleinem öffentlichen Exponenten

D. Coppersmith präsentiert in [14] Methoden, um unter gewissen Voraussetzungen für Gleichungen der Form

$$f(x) \pmod{N} = 0 \text{ bzw. } g(x, y) = 0$$

mit Polynomen $f \in \mathbb{Z}[X], g \in \mathbb{Z}[X][Y]$ und $N \in \mathbb{Z}$ ganzzahlige Lösungen in Polynomzeit zu finden. Er nutzt dazu den LLL-Algorithmus aus [41]:

Satz 4.1.1 (Coppersmith 1) Sei $f(x)$ ein normiertes Polynom vom Grad nicht grösser als δ . Falls $X < 2^{-1/2}(\delta+1)^{-1/\delta} N^{2/\delta(\delta+1)}$, dann kann man alle $x_0 \in \mathbb{Z}$ mit $f(x_0) \equiv 0 \pmod{N}$ und $|x_0| \leq X$ in Polynomialzeit finden.

Beweis. In dem Beweis betrachtet Coppersmith die Menge der ganzzahligen Linearkombinationen der Polynome aus

$$B_x := \{x^i, 0 \leq i < \delta\} \cup \{f(x)/N\} .$$

Für alle Polynome b_x aus B_x gilt für alle $x \in \mathbb{Z}$ mit $f(x) \pmod{N} = 0$, daß $b_x(x) \in \mathbb{Z}$. Wir setzen $x = Xy$ und betrachten nun die Polynome aus

$$B_y := \{b_y(y) \in \mathbb{Q}[Y] : b_y(y/X) \in B_x\} .$$

Identifiziert man Polynome mit den δ -Tupeln, die den Koeffizienten entsprechen, kann man die Menge B_y als Basis eines Gitters betrachten:

$$B_y = \begin{pmatrix} 1 & 0 & \cdots & 0 & f_0/N \\ 0 & X & & 0 & f_1 X^1/N \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & & X^{\delta-1} & f_{\delta-1} X^{\delta-1}/N \\ 0 & 0 & \cdots & 0 & 1 \cdot X^\delta/N \end{pmatrix}$$

Man kann nun zeigen, daß LLL in $L(B_y)$ einen Vektor findet, der eine genügend kleine Norm hat, so daß man beweisen kann, daß für das ihm entsprechende Polynom $g(y)$ gilt:

$$\forall x_0 \leq X : f(x_0) \pmod{N} = 0 \Rightarrow g(x_0/X) < 1 .$$

Nun gilt aber $g(y/X) \in B_x \Rightarrow g(x_0/X) \in \mathbb{Z} \Rightarrow g(x_0/X) = 0$. Da die Nullstellen von g in Polynomialzeit zu berechnen sind, hat man alle Nullstellen von f modulo N gefunden, welche kleiner als X sind. Eine ausführlichere Darstellung findet man in [14]. ■

Satz 4.1.2 (Coppersmith 2) Mit denselben Voraussetzungen wie in 4.1.1 gilt für alle $\epsilon > 0$: Falls $X \leq \frac{1}{\sqrt{2}} N^{1/\delta - \epsilon}$ dann kann man alle $x_0 \in \mathbb{Z}$ mit $f(x_0) \equiv 0 \pmod{N}$ und $|x_0| \leq X$ in einer in $(\log N, \delta, 1/\epsilon)$ polynomialen Zeit finden.

Beweis. Der Beweis läuft analog zu dem von Satz 4.1.1. Man wählt ein $h \in \mathbb{N}$ und ersetzt B_x durch die Menge $\{ (f(x)/N)^j x^i : 0 \leq i < \delta, 0 \leq j < h \}$. Falls h (in Abhängigkeit von ϵ) groß genug gewählt wird, folgt die Behauptung. Siehe auch [12] bzw. [14]. ■

Aus diesen Theoremen ergeben sich verschiedene Möglichkeiten für Angriffe auf den RSA-Algorithmus. Die einfachste Form ist ein direkter Angriff bei zu kleinem öffentlichem Exponenten und zu kleiner Nachricht, z.B. $e = 3, m < N^{1/3}$. Coppersmiths Arbeit ermöglicht jedoch auch Angriffe bei zu schwachem „Padding“ oder bei Versand einer Nachricht an genügend Personen mit gleichem öffentlichem e und verschiedenen Moduln N [16]. Für die Darstellung eines Angriffs mit einer anderen Basismatrix wollen wir die verständlichere Version des Satzes 4.1.1 von Howgrave-Graham aus [32] benutzen:

Definition 4.1.3 Für ein Polynom $f(x, y) = \sum_{i,j} f_{i,j} x^i y^j \in \mathbb{R}[x, y]$ sei die Norm definiert als $\|f(x, y)\|^2 := \sum_{i,j} f_{i,j}^2$.

Satz 4.1.4 Sei $f(x) \in \mathbb{Z}[x]$ von Grad $(\delta - 1)$ und sei $X \in \mathbb{N}$, sowie $\|f(xX)\| < N/\sqrt{\delta}$. Falls $f(x_0) \equiv 0 \pmod{N}$ und $|x_0| < X$, dann gilt $f(x_0) = 0$ in \mathbb{Z} .

Beweis. Es gilt: $|f(x_0)| = |\sum f_i x_0^i| \leq \sum |f_i X^i| \leq \|f(xX)\| \sqrt{\delta} < N$. Daraus folgt die Behauptung. ■

Sei (n, e) ein öffentlicher RSA-Schlüssel, c ein mit diesem Schlüssel generierter Schlüsseltext und $f(x) = x^e - c \in \mathbb{Z}[X]$. Hat man für ein $h \in \mathbb{N}$ ein Polynom r mit genügend kleiner Norm und einer Nullstelle modulo n^{h-1} in m , dann kann man nach dem Satz den Schlüsseltext c auch ohne die Kenntnis des privaten Schlüssels d entschlüsseln. Ein solches Polynom wird als Linearkombination der folgenden Polynome erzeugt:

$$g_{u,v}(x) := n^{h-1-v} x^u f(x)^v ,$$

wobei r höchstens vom Grad he sein soll. Das Problem, ein solches Polynom zu finden, kann als SVP in dem von den $g_{u,v}$ aufgespannten Gitter gesehen werden. Man wählt für die i -te Zeile der Basismatrix B das Polynom $g_{u,v}(xX)$ mit $v = \lfloor (i-1)/e \rfloor$ und $u = i-1 - ev$. Falls $e = 3$ und $h = 2$, ergibt sich die Matrix B

$$\begin{array}{c|cccccc}
 i & 1 & x & x^2 & x^3 & x^4 & x^5 \\
 \hline
 1 & n & & & & & \\
 2 & & nX & & & & \\
 3 & & & nX^2 & & & \\
 4 & -c & & & X^3 & & \\
 5 & & -cX & & & X^4 & \\
 6 & & & -cX^2 & & & X^5
 \end{array}$$

Für den allgemeinen Fall gilt $\det(B) = X^{he(he-1)/2} \cdot n^{he(h-1)/2}$. Aus Satz 3.1.3 folgt, daß man mit dem LLL-Algorithmus für ein gegebenes h Nullstellen bis X finden, falls

$$X \leq \frac{1}{\sqrt{2}} n^{(h-1)/(he-1)} \cdot (he)^{-1/(he-1)}.$$

Mit wachsendem h nähert sich diese Grenze asymptotisch dem Wert $\frac{1}{\sqrt{2}} n^{1/e}$ an. Schon für kleine h ist man sehr nah an dieser Grenze, so daß man hoffen kann, daß die Matrix nicht allzu hochdimensional und ihre Einträge nicht allzu groß werden. Man kann daher alle Nachrichten mit $m < \frac{1}{\sqrt{2}} n^{1/e}$ mittels dieses Gitters und dem LLL-Algorithmus entschlüsseln.

Die Grenzen des Ansatzes von Coppersmith werden deutlich, wenn man sich die Abschätzungen für den kürzesten Gittervektor in einem zufälligen Gitter nach Gauss ansieht. Es ist nicht zu erwarten, daß es in dem gebildeten Gitter einen Vektor kürzer als $\det(L)^{1/he}$ gibt. Selbst wenn man die Abschätzung für X mit einem SVP-Orakel durchführt, dann erwartet man, daß man die Nullstellen von P nur bis

$$X \leq n^{(h-1)/(he-1)} (he)^{-1/(he-1)}$$

finden kann (vergleiche [16]). Wir haben hier die Annahme gemacht, daß die von uns verwendeten Gitter als zufällig anzusehen sind. Unter dieser Annahme gelten die Überlegungen analog für fast alle Angriffe auf RSA via Coppersmith.

4.2 Angriffe bei zu kleinem geheimen Exponenten

In diesem Abschnitt wollen wir den Ideen von Blömer und May [5], bzw. Boneh und Durfee [7] folgen. Sie beruhen auf den Ergebnissen von D. Coppersmith für den bivariaten Fall.

Für einen öffentlichen RSA-Schlüssel (n, e) mit $n = pq$ und zugehörigem privaten Schlüssel d gilt

$$ed = 1 \pmod{\frac{\varphi(n)}{2}} \iff \exists_{k \in \mathbb{Z}} : ed + k \left(\frac{n+1}{2} - \frac{p+q}{2} \right) = 1.$$

Wir betrachten nun die modulare Gleichung

$$0 \equiv k \left(\frac{n+1}{2} - \frac{p+q}{2} \right) - 1 \pmod{e}$$

und setzen $f(x, y) := x \left(\frac{n+1}{2} - y \right) - 1$. Auf Grund der Konstruktion ist klar, daß f eine Nullstelle in $(x_0, y_0) = (k, (p+q)/2)$ besitzt. Mit $d < n^{\hat{\delta}}$ und $e = n^\alpha$ für $\hat{\delta}, \alpha > 0$ gilt:

$$\begin{aligned} |x_0| &< \frac{2de}{\varphi(n)} \leq 3de/n < 3e^{1+(\hat{\delta}-1)/\alpha} \\ |y_0| &< 2\sqrt{n} = 2e^{1/2\alpha}. \end{aligned}$$

Ziel ist es nun, diese Nullstelle zu finden, damit man das nicht bekannte d berechnen kann. Wir nehmen zunächst $\alpha = 1$ an. Mit Hilfe eines Gitters soll ein Polynom gefunden werden, das die Bedingungen des Theorems von Howgrave-Graham aus [32] erfüllt.

Satz 4.2.1 (Howgrave-Graham) Sei $f(x, y)$ ein Polynom, die Summe von höchstens w Monomen, und es gelte $f(x_0, y_0) \equiv 0 \pmod{e^m}$ für ein $m \in \mathbb{N}$. Seien weiterhin $|x| < X$ und $|y| < Y$, dann gilt $f(x_0, y_0) = 0$ in \mathbb{Z} (also nicht nur in $\mathbb{Z}/e^m\mathbb{Z}$), falls $\sqrt[w]{e^m} \cdot \|f(xX, yY)\| < e^m$.

Beweis.

$$\begin{aligned} f(x_0, y_0) &= \left| \sum_{i,j} f_{i,j} x_0^i y_0^j \right| = \left| \sum_{i,j} f_{i,j} X^i Y^j \left(\frac{x_0}{X} \right)^i \left(\frac{y_0}{Y} \right)^j \right| \\ &\leq \sum_{i,j} |f_{i,j} X^i Y^j| \leq \sqrt[w]{e^m} \cdot \|f(xX, yY)\| < e^m \end{aligned}$$

Damit folgt die Behauptung, da $f(x_0, y_0) < e^m$ und $f(x_0, y_0) \equiv 0 \pmod{e^m}$. ■

Diesen Satz wollen wir für einen Angriff auf den privaten Schlüssel d ausnutzen. Mit Hilfe eines Gitters konstruieren wir (ohne die Kenntnis von p, q oder d) zwei Polynome mit ausreichend kleiner Norm, die in $(k, \frac{p+q}{2})$ eine Nullstelle modulo e^m haben. Falls für beide Polynome die Voraussetzungen von 4.2.1 erfüllt sind, dann wissen wir, daß beide eine gemeinsame Nullstelle in $(k, \frac{p+q}{2})$ haben. Diese berechnen wir auf folgende Weise, falls ihre Resultante $\neq 0$ ist: Sei $f(x_0, y_0) = 0$ und $g(x_0, y_0) = 0$ dann ist y_0 eine Nullstelle der Resultante $h(y) = \text{Res}_x(f, g)$, welche in polynomialer Zeit berechnet werden kann. Damit kennen wir $(p+q)/2$ und können n faktorisieren, bzw. d direkt berechnen.

Um nun Polynome mit Norm kleiner als $e^m/\sqrt[w]{e^m}$ mittels eines Gitters zu konstruieren, definiert man die Polynome

$$\begin{aligned} g_{m,i,k}(x, y) &:= x^i f^k e^{m-k} \text{ „x-shifts“ und} \\ h_{m,j,k}(x, y) &:= y^j f^k e^{m-k} \text{ „y-shifts“}. \end{aligned}$$

Man muß nun aus diesen geschickt eine Basis des Gitters wählen, damit der LLL-Algorithmus genügend kurze Gittervektoren findet (siehe Satz 3.1.3). Damit ergibt sich die Forderung $\det(L) < \frac{e^{mw}}{(2^w w)^w}$, an das zu konstruierende Gitter,

wenn man $\|b_1\| \leq 2^{\frac{\dim(L)}{2}} (\det(L))^{\frac{1}{\dim(L)}}$ zu Grunde legt (vergleiche [7]). Man beachte, daß die Dimension des Gitters gleich der möglichen Anzahl der Monome in Satz 4.2.1 ist. Die Basismatrix von Boneh und Durfee besteht aus den Polynomen

$$\begin{aligned} & \{g_{m,i,k}(xX, yY) \mid 0 \leq k \leq m, 0 \leq i \leq m-k\} \\ & \cup \{h_{m,j,k}(xX, yY) \mid 0 \leq k \leq m, 0 < j \leq t\}, \end{aligned}$$

wobei $m, t \in \mathbb{N}_0$ gewählt werden. Beschreibt man die Polynome als Zeilen einer Matrix, so ergibt sich für $m = 2$ und $t = 1$:

	1	x	xy	x^2	x^2y	x^2y^2	y	xy^2	x^2y^3
e^2	e^2								
xe^2		e^2X							
fe	–	–	eXY						
x^2e^2				e^2X^2					
xfe		–		–	eX^2Y				
f^2	–	–	–	–	–	X^2Y^2			
ye^2							e^2Y		
yfe			–				–	eXY^2	
yf^2			–		–	–	–	–	X^2Y^3

wobei die „–“ Symbole für Einträge $\neq 0$ stehen, die uns nicht interessieren.

Betrachtet man nur die Menge $\{g_{m,i,k}(xX, yY) \mid 0 \leq k \leq m, 0 \leq i \leq m-k\}$ als Basis B eines Gitters, so ist die Gitterdeterminante

$$\det_x = e^{m(m+1)(m+2)/3} \cdot X^{m(m+1)(m+2)/3} \cdot Y^{m(m+1)(m+2)/6},$$

welche die Dimension $w = \sum_{i=0}^m m+1-i = (m+1)(m+2)/2$ hat. Somit ergibt sich mit $X = e^{\hat{\delta}}$ und $Y = e^{\frac{1}{2}}$ (wir erinnern uns, daß wir $e = n^\alpha$ mit $\alpha = 1$ angenommen haben) die Ungleichung

$$\|b_1\| \leq \left(\delta - \frac{1}{4}\right)^{\frac{1-w}{4}} (\det_x)^{\frac{1}{w}} \leq e^m / \sqrt{w}$$

für die Norm des Vektors b_1 einer LLL-reduzierten Basis b_1, \dots, b_w des Gitters $L(B)$. Wir betrachten diese Ungleichung für feste gewählte Parameter m, t und wachsende n bzw. e . Da w nur von m abhängt, können wir für sehr große n zu einer asymptotischen Betrachtung der Ungleichung übergehen und alle konstanten Faktoren vernachlässigen. Wir haben damit $\|b_1\| \leq (\det_x)^{\frac{1}{w}} \leq e^{mw}$ und können schließen, daß

$$\begin{aligned} (\det_x)^{\frac{1}{w}} \leq e^{mw} & \iff \left(e^{m(m+1)(m+2)(5+4\hat{\delta})/12}\right)^{\frac{1}{w}} \leq e^{mw} \\ & \iff \left(e^{m(5+4\hat{\delta})/6}\right)^{\frac{1}{w}} \leq e^m \\ & \iff \lim_{e \rightarrow \infty} (5 + 4\hat{\delta}) < 6 \\ & \iff \hat{\delta} < 0.25, \end{aligned}$$

was Wieners Resultat aus [72] entspricht. Es reicht hier, die Abschätzung von $\|b_1\|$ zu betrachten, da für $\|b_2\|$ Folgerung 3.1.5 gilt, und damit

$$\|b_2\|^2 \leq 2^w \det(L)^{\frac{2}{w-1}} \leq e^m / \sqrt{w}.$$

Wie oben kann man die Ungleichung zu $(\det_x) \leq e^{m(w-1)}$ vereinfachen und sieht, daß die Veränderung bei großen m keine Rolle spielt. Damit hat man zwei geeignete (linear unabhängige) Polynome gefunden und kann n faktorisieren, falls deren Resultante $\neq 0$ ist. Dieser Angriff ist nur eine Heuristik, denn wir können nicht garantieren, daß letztere Bedingung erfüllt ist. In [5] wird u.a. diese Art von Basismatrizen analysiert. Die Autoren stellen fest, daß der LLL-Algorithmus in ihren Experimenten zu diesem Fall keine geeigneten Polynome, d.h. mit Resultante $\neq 0$ findet.

Wenn man wie Boneh und Durfee die oben beschriebenen „y-Shifts“ zur Basis hinzunimmt, findet der LLL-Algorithmus in den Experimenten von Blömer und May immer Polynome, deren Resultante nicht verschwindet. Die Basismatrix ist weiterhin in der Gestalt einer unteren Dreiecksmatrix. Wir erhalten ein Gitter mit Determinante

$$\det_y = \det_x \cdot e^{tm(m+1)/2} \cdot X^{tm(m+1)/2} \cdot Y^{t(m+1)(m+t+1)/2}$$

und Dimension $w = (m+1)(m+2)/2 + t(m+1)$, womit man die neue Grenze für $\hat{\delta}$ bestimmen kann, falls man $t \approx (m(1-2\hat{\delta})-1)/2$ wählt:

$$\hat{\delta} < \frac{7}{6} - \frac{1}{3}\sqrt{7} \approx 0.284$$

Um die Grenze auf 0.292 zu verbessern, treffen Boneh und Durfee eine andere Wahl für die „y-Shifts“, verletzen dabei aber Dreieckseigenschaft der Basismatrix. Über die Ergebnisse Ihrer Experimente gibt die folgende Tabelle aus [7] Aufschluß. An ihr kann man auch Beispiele für die Wahl von m und t ablesen.

n	d	$\hat{\delta}$	m	t	Gitterdimension	Dauer
1000 Bits	280 Bits	0.280	7	3	45	14 Std.
2000 Bits	550 Bits	0.275	7	3	45	65 Std.
4000 Bits	1060 Bits	0.265	5	2	25	14 Std.
10000 Bits	2550 Bits	0.255	3	1	11	90 min.

Blömer und May gehen in [5] einen anderen Weg. Sie wählen ebenfalls andere „y-Shifts“, eliminieren dann aber alle linear abhängigen Spalten, so daß die Dreieckseigenschaft erhalten bleibt. Die nicht in dem Gitter enthaltenen Koeffizienten kann man rekonstruieren und ihren Betrag nach oben abschätzen. Mit den veränderten Bedingungen erreicht man 0.290 als obere Schranke für $\hat{\delta}$. Der Vorteil ihres Ansatzes ist die geringere Gitterdimension ($w = (m+1)(t+1)$) und der Erhalt der Dreieckseigenschaft (Einfachheit der Beweise, Determinante leicht zu berechnen). Die Berechnungen sind nahezu analog zu dem oben dargestellten einfachen Fall. Ein Algorithmus profitiert hier von der kleinen Determinante (durch die kleinere Gitterdimension) und dem dadurch noch kleineren kürzesten Gittervektor.

Bei allen vorgestellten Varianten des Gitterangriffs handelt es sich um eine Heuristik, da wir nicht garantieren können, daß wir zwei geeignete Polynome finden, deren Resultante nicht verschwindet. Auch für relativ große Werte für α ist dieser Angriff noch durchführbar, je kleiner α , desto größer wird $\hat{\delta}$. Die Überlegungen sind analog zu denen für $\alpha = 1$, weswegen ich hier nicht weiter darauf eingehen will.

Wir wollen uns nun der Frage nach den Grenzen dieses Angriffs und dem Einfluß des gewählten Gitterbasisreduktionsalgorithmus zuwenden. Nach der Heuristik von Gauss gilt für zufällige Matrizen der Dimension n für den Wert von λ_1 :

$$\sqrt{\frac{n}{2\pi e}} (\det(L))^{\frac{1}{n}} \leq \lambda_1 \leq \sqrt{\frac{n}{\pi e}} (\det(L))^{\frac{1}{n}},$$

wobei e hier die Eulerzahl ist.

Definition 4.2.2 Wir bezeichnen mit $L_{\text{RSA-BD}}$ die Klasse der *RSA-BD-Gitter*. Für einen öffentlichen RSA-Schlüssel (n, e) mit $n = pq$ und $w, m \in \mathbb{Z}$ gelte $L \in L_{\text{RSA-BD}}^{(n,e)(w,m)}$ genau dann, wenn

- $L = L(B) \in \mathbb{R}^w$ volldimensional,
- eine lineare, injektive Polynomialzeit-Abbildung $\tau : L \rightarrow \mathbb{Q}[Z_1][Z_2]$ existiert und
- $X, Y \in \mathbb{Z}$ existieren, so daß

$$\forall_{1 \leq i \leq w} : \tau(B_{i \cdot}) \left(\frac{x}{X}, \frac{y}{Y} \right) \in \{g_{m,i,k}\} \cup \{h_{m,j,k}\}$$

gilt.

Wir definieren: $\bigcup_{(n,e)\text{RSA-Schlüssel}} \bigcup_{(w,m) \in \mathbb{N}^2} L_{\text{RSA-BD}}^{(n,e)(w,m)} = L_{\text{RSA-BD}}$.

Es sollte klar sein, daß die von uns in diesem Kapitel betrachteten Matrizen von May bzw. Boneh und Durfee zu $L_{\text{RSA-BD}}$ gehören. Zu einem gegebenen öffentlichen RSA-Schlüssel (n, e) wählen wir m, w und konstruieren ein Gitter L mit $X = 3e^{1-\frac{\hat{\delta}-1}{\alpha}}$ und $Y = 2e^{\frac{1}{2\alpha}}$, wobei $e = n^\alpha$ und $\hat{\delta}$ in Abhängigkeit von α gewählt wird. Wir können Satz 4.2.1 anwenden, falls wir einen Vektor $b \in L$ finden, für den $\|\tau(b)\| \leq e^m / \sqrt{w}$ gilt. Falls für den öffentlichen Schlüssel d die Ungleichung $d < n^{\hat{\delta}}$ mit genügend kleinem $\hat{\delta}$ erfüllt ist, existieren zwei solche Vektoren und wir können n faktorisieren.

Annahme 1 Für alle $(w, m) \in \mathbb{N}^2$ gibt es Konstanten $c, C > 0$, so daß es keine Möglichkeit gibt, für zufällige RSA-Schlüssel (n, e) ein $L \in L_{\text{RSA-BD}}^{(n,e)(w,m)}$ ohne Kenntnis des privaten Schlüssels in Polynomialzeit so auszuwählen, daß

$$\lambda_1(L) < c \cdot C^w \cdot \det(L)^{\frac{1}{w}}$$

mit nicht zu vernachlässigender Wahrscheinlichkeit gilt. Man beachte, daß w , m , c und C nicht von dem RSA-Schlüssel abhängen, die Gitterdeterminante jedoch sehr wohl.

Mit anderen Worten: Wir nehmen an, daß die hier betrachteten Gitter der Heuristik von Gauss genügen, d.h. daß man nicht zwischen einem RSA-BD-Gitter und einem zufälligen Gitter unterscheiden kann, wenn man nur das Verhältnis von λ_1 und der Gitterdeterminante kennt.

Folgerung 4.2.3 Sei O_{SVP} ein SVP-Orakel und O_{NS} ein Nullstellenorakel für bivariate Polynome und (n, e) ein öffentlicher RSA-Schlüssel mit ausreichend großem n . Für ein in Polynomialzeit aus $L_{\text{RSA-BD}}$ ausgewähltes Gitter $L = L(B)$ gilt mit Annahme 1: Die Wahrscheinlichkeit, daß man den geheimen Schlüssel d mit Hilfe der Matrix B , O_{SVP} und O_{NS} berechnen kann ist nicht signifikant größer als die, daß man den geheimen Schlüssel mit B , LLL und O_{NS} berechnen kann.

Beweis. Wir wählen zunächst ein solches B zu einem gegebenen RSA-Schlüssel aus. Wir müssen einen Vektor $b \in L(B)$ finden, so daß $\|b\| < e^m/\sqrt{w}$ gilt, damit wir Satz 4.2.1 anwenden können. Es gilt aber nach Annahme 1, daß $c \cdot C^w \cdot \det(B)^{\frac{1}{w}} < \|b\|$ bis auf eine zu vernachlässigende Anzahl von Fällen. Wenn wir uns an die Berechnung der Grenze für $\hat{\delta}$ im einfachen Fall erinnern, dann hatten wir den LLL-Faktor vernachlässigt. Dies wurde uns durch die Betrachtung des asymptotischen Falls ermöglicht. Analog zu den Überlegungen von oben ergibt sich folgende Ungleichung als Bedingung für einen erfolgreichen Angriff mittels O_{SVP} :

$$c \cdot C^w (\det(B))^{\frac{1}{w}} \leq \lambda_1 \leq e^m/\sqrt{w} \xrightarrow{\text{große } e, n} (\det(B)) \leq \lambda_1 \leq e^{wm}$$

Dies ist dieselbe Ungleichung, die wir bei der Anwendung von LLL erhalten (siehe oben). Das Finden des kürzesten Gittervektors führt daher nicht zu einer Verbesserung des Angriffs. ■

Der hier vorgestellte RSA-Angriff profitiert von dem kleinsten gefundenen Gittervektor. Je kleiner $\det(L)$, desto besser ist der Angriff, welcher genau eine LLL-Reduktion braucht. Damit übertragen sich Verbesserungen in der Laufzeit des Algorithmus direkt auf diesen RSA-Angriff. Eine bessere Annäherung des kürzesten Gittervektors bietet dagegen keine (oder nur geringe) Verbesserungsmöglichkeiten für die Grenze von $\hat{\delta}$ (vergleiche [16]). Es wäre interessant zu untersuchen, wie oft ein (perfektes) SVP-Orakel ein wirklich besseres Ergebnis liefert, d.h. wie oft bei konstanten p, q und variierenden e, d ein Angriff mit dem SVP-Orakel gelingt, während einer mit dem LLL-Algorithmus versagt. Damit verbunden ist die Frage, wie sich der kürzeste Gittervektor in den RSA-BD-Matrizen verhält, falls Annahme 1 entgegen den Resultaten der bisherigen Experimente nicht gilt.

4.3 Angriffe bei teilweise bekanntem privaten Schlüssel

In verschiedenen Szenarien ist es möglich, Informationen über den verwendeten geheimen Schlüssel d zu erlangen, ohne, daß man den RSA-Algorithmus selbst angreifen muß (engl. Side Channel Attacks). Dies ist z.B. bei SmartCard-Anwendungen der Fall. Eine gute Übersicht über Angriffe, die so gewonnene Informationen ausnutzen, geben die Artikel [6] und [8]. Sie basieren jedoch auch auf den Arbeiten von D. Coppersmith und haben daher unabhängig vom Reduktionsalgorithmus Grenzen in der Anwendung. Wir wollen hier exemplarisch nur einen Fall aus [6] betrachten, bei dem die letzten Stellen von d bekannt sind, denn die anderen laufen ähnlich. In dem selben Artikel wird ebenfalls ein Angriff vorgestellt, der keine Heuristik ist, der jedoch nur funktioniert, wenn $e < n^{0.5}$ ist.

Sei (n, e) ein öffentlicher RSA-Schlüssel, so daß für den privaten Schlüssel d gilt, daß $d \equiv d_0 \pmod{M}$, $M > n^{\frac{1}{6} + \frac{1}{3}\sqrt{1+6\alpha}}$ mit $\alpha = \log_n(e) < \frac{7}{8}$. Wir betrachten zunächst die Gleichung $ed - 1 = k\varphi(n)$ und schreiben $d = d_1M + d_0$. Wir erhalten damit die Gleichung

$$k(n - (p + q - 1)) - ed_0 + 1 = eMd_1.$$

Damit können wir $(k, (p + q - 1)) = (x_0, y_0)$ als Nullstellen des Polynoms

$$f_{eM}(x, y) = x(n - y) - ed_0 + 1$$

in $\mathbb{Z}/(eM)\mathbb{Z}$ auffassen. Weiterhin gilt $x_0 \leq X := n^\alpha$ sowie $y_0 \leq Y := 3n^{0.5}$, wenn wir davon ausgehen, daß p und q dieselbe Bitlänge haben, sowie $p < q$. Wir wollen jetzt ähnlich wie in dem vorigen Kapitel Satz 4.2.1 anwenden und definieren daher

$$\begin{aligned} g_{i,j} &= x^j (eM)^i f_{eM}^{m-i} && \text{für } i = 0, \dots, m; j = 0, \dots, i \\ h_{i,j} &= y^j (eM)^i f_{eM}^{m-i} && \text{für } i = 0, \dots, m; j = 1, \dots, t \end{aligned}$$

für gewählte $m, t \in \mathbb{N}$. Analog zu den vorherigen Angriffen definieren wir eine Gitterbasis aus den Koeffizienten der Polynome multipliziert mit den passenden Potenzen von X bzw. Y . Wir müssen nun ein Gitterelement mit Norm kleiner $(eM)^m / \sqrt{w}$ finden, wobei w die Dimension des Gitters ist. Man kann nachweisen, daß der LLL-Algorithmus für genügend große n ausreichend kurze Gittervektoren findet, um Satz 4.2.1 anwenden zu können. Falls die Resultante der gefundenen Polynome ungleich 0 ist, dann können wir n faktorisieren. Auch hier kann man analog zu Folgerung 4.2.3 schließen, daß keine Verbesserung der Grenzen für M bzw. α möglich ist, falls die hier verwendeten Gitter der Heuristik von Gauß genügen. Gleiches gilt für die weiteren Angriffe aus [6] und [8].

4.4 Faktorisierungsangriff

Wenn man den Modulus n eines RSA-Schlüssels faktorisieren könnte, wäre RSA gebrochen, auch wenn die Umkehrung noch nicht klar ist. C.P. Schnorr stellt

in [61] eine Methode vor, um eine Zahl n , welche das Produkt von mindestens zwei Primfaktoren ist, zu faktorisieren. Sein Ansatz arbeitet mit einer Matrix, die eine Dimension gleich der Mächtigkeit der Menge aller möglichen Primfaktoren kleiner als einer gewählten Grenze hat. Deshalb schnell die Dimension des Gitters in die Höhe, was den Ansatz für die Praxis unbrauchbar macht. Eine Alternative bietet derzeit nur der Algorithmus von Shor für Quantencomputer (siehe [67]), welcher bislang nicht praxisrelevant ist.

5 Gitterbasierte Kryptographie

Zu den Gitterbasierten Kryptosystemen gehört neben denen, die explizit mit Gittern operieren, auch das NTRU-Kryptosystem. Zu den ersteren gehören u.a. das Kryptosystem von Ajtai-Dwork [2] und das GGH-Kryptosystem [25], welches allerdings nicht mehr als sicher gilt, da es P.Nguyen gelungen ist, Angriffe auf alle praxisrelevanten Fälle durchzuführen [53].

5.1 Micciancios Kryptosystem

Micciancios Kryptosystem basiert auf dem GGH-Kryptosystem, versucht jedoch einige Unzulänglichkeiten dieses Systems zu umgehen. Der Grundgedanke ist es, eine Basis R eines Gitters $L \subseteq \mathbb{Z}^n$ als privaten Schlüssel zu nutzen, so daß das CVP mittels Babai [3] für relativ kleine Abstände ρ zum nächsten Vektor gelöst werden kann. Als öffentlicher Schlüssel wird eine alternative Gitterbasis B von L angegeben, welche ein schlechter Ausgangspunkt zum Lösen des CVP ist.

Definition 5.1.1 Der *Orthogonalisationsdefekt* einer Basis $B = [b_1, \dots, b_m]$ eines Gitters im \mathbb{R}^n ist definiert als

$$\text{odef}(B) := \det(B)^{-1} \prod_{i=1}^m \|b_i\| .$$

Mit dem *dualen Orthogonalisationsdefekt* sei der Orthogonalisationsdefekt der dualen Gitterbasis gemeint, d.h.

$$\text{odef}^*(B) := \text{odef}(B^*) .$$

Allgemein wird versucht, die Matrizen so zu wählen, daß sowohl primaler als auch dualer Orthogonalisationsdefekt der privaten Basis R möglichst klein und der öffentlichen Basis B möglichst groß sind. Micciancio wählt im Gegensatz zu GGH die Matrix $R = [r_1, \dots, r_n]$ als die LLL-reduzierte Darstellung einer aus $\{-n, \dots, n\}^{n \times n}$ zufällig geählten Matrix. Der öffentliche Schlüssel ergibt sich als die Matrix $B = \text{HNF}(R)$ (siehe Definition B.1.2). Damit wird der Speicheraufwand gegenüber dem GGH-Kryptosystem erheblich reduziert. Man verschafft einem Angreifer dadurch keinen Vorteil, da das Berechnen der HNF einer beliebigen Matrix in Polynomzeit erfolgen kann. Die HNF einer Gitterbasismatrix hängt zudem nur von dem Gitter L ab und nicht von der Ausgangsmatrix ab. Für die Ver- und Entschlüsselung brauchen wir den folgenden Satz:

Satz 5.1.2 Sei $L = L(b_1, \dots, b_n) \subseteq \mathbb{R}^n$ ein Gitter und $h(B) := \min_i \|\hat{b}_i\|$, dann kann man das CVP-Problem für einen Vektor der Form $y = Bx + m$ mit $x \in \mathbb{Z}^n$ und $\|m\| < \frac{1}{2}h(B)$, $m \in \mathbb{R}^n$ in polynomialer Zeit lösen.

Man beachte, daß $h(B)$ keine Gitterkonstante ist, sondern von der Wahl der Basis B abhängt.

Beweis. (Version aus [19]) Wir wissen nach Voraussetzung, daß es einen Gittervektor l gibt, so daß $\|y - l\| < \frac{1}{2}h(B)$. Daraus folgt, daß $\|\pi_n(y - l)\| < \frac{1}{2}h(B)$. Wir schreiben nun $y = \sum_{i=1}^n \alpha_i b_i$ und $l = \sum_{i=1}^n \beta_i \hat{b}_i$ mit $\alpha_i \in \mathbb{Z}$ und $\beta_i \in \mathbb{R}$, wobei wir letztere in Polynomialzeit ermitteln können. Betrachten wir nun die Projektion, so ergibt sich

$$\begin{aligned} \pi_n(y - l) &= \pi_n(y) - \pi_n(l) \\ &= \sum_{i=1}^n \alpha_i b_i - \sum_{j=1}^{n-1} \frac{\langle \sum_{i=1}^n \alpha_i b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle} \hat{b}_j - \beta_n \hat{b}_n \\ &= \sum_{i=1}^n \alpha_i \left(b_i - \sum_{j=1}^{n-1} \frac{\langle b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle} \hat{b}_j \right) - \beta_n \hat{b}_n \\ &= \alpha_n \hat{b}_n - \beta_n \hat{b}_n . \end{aligned}$$

Damit ergibt sich $\|\alpha_n - \beta_n\| \cdot \|\hat{b}_n\| < \frac{1}{2}h(B)$ woraus $\|\alpha_n - \beta_n\| < \frac{1}{2}$ folgt. Da wir β_n kennen, ist α_n eindeutig bestimmt und die restlichen α_i mit $i = n-1, \dots, 1$ können auf dieselbe Weise bestimmt werden. ■

Das Verschlüsseln einer Nachricht $m \in \mathcal{M} := \{m \in \mathbb{Z}^n : \|m\| < \frac{1}{2}h(R)\}$ erfolgt durch die zufällige Wahl eines $x \in \mathbb{Z}^n$ und der Berechnung des Schlüsseltextes $c = Bx + m$. Ein Schlüsseltext c wird entschlüsselt, indem man das CVP für c in $L(B)$ löst. Nach Lemma 2.1.10 gilt, daß $\|Bx - c\|_2 < \frac{1}{2}\lambda_1(L)$, deswegen erhält man als Lösung den ursprünglichen Vektor Bx und kann $m = c - Bx$ berechnen. Nach Satz 5.1.2 kann die Entschlüsselung in Polynomzeit erfolgen, falls R bekannt ist.

Es bleibt das Problem, bei der Verschlüsselung x zufällig aus \mathbb{Z}^n zu wählen. Dieses kann mit der Art der Darstellung von c behoben werden. Seien c und c' Schlüsseltexte desselben Klartextes, dann ist klar, daß $c - c' \in L$. Also kann man auf der Menge der Schlüsseltexte dadurch eine Äquivalenzrelation definieren, welche sich auf ganz \mathbb{R}^n fortsetzen läßt. Es bleibt jedoch die Frage, welchen Repräsentanten man wählen soll. Micciancio schlägt vor, diesen aus dem Parallelepiped

$$M(\hat{B}) = \left\{ \sum_{i=1}^n \hat{b}_i x_i \mid 0 \leq x_i < 1 \right\}$$

zu wählen. Dazu berechnet man rekursiv für eine Nachricht m :

$$\begin{aligned} m_n &= m \\ m_{j-1} &= m_j - \left\lfloor \frac{\langle m_j, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle} \right\rfloor \hat{b}_j \quad \text{für } j = n, \dots, 1 \\ c &= m \pmod{B} := m_0 . \end{aligned}$$

Man sagt, man reduziert m modulo B . Dieses reduziert den Speicheraufwand, der Angreifer bekommt keine zusätzliche Information und man muß keine Zufällige Auswahl mehr treffen, da $m \pmod{B}$ eindeutig ist. Letzteres ist klar, weil

$$\frac{\langle m \pmod{B}, \hat{b}_i \rangle}{\langle \hat{b}_i, \hat{b}_i \rangle} \in [0, 1[\quad \text{für alle } i = 1, \dots, n \text{ gilt.}$$

Damit ist das Micciancios Kryptosystem im Gegensatz zu GGH deterministisch.

5.2 Angriffe auf Micciancios Kryptosystem

Der folgende Angriff kann für alle bekannten GGH-Varianten durchgeführt werden. Zunächst ist zu bemerken, daß ein Angreifer nicht unbedingt das CVP lösen muß, sondern daß es ausreicht das zugehörige SVP zu lösen, um den Klartext zu ermitteln. Der Angreifer hat $B = [b_1, \dots, b_n]$, \mathcal{M} und c als Informationen gegeben. Wenn er nun

$$B' = \left[\begin{pmatrix} b_1 \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} b_n \\ 0 \end{pmatrix}, \begin{pmatrix} c \\ 1 \end{pmatrix} \right]$$

als Basismatrix eines Gitters nimmt, ist es zu erwarten, daß der kürzeste Gittervektor in dem zugehörigen Gitter die Form $(m, 1)^\top \in L(B')$ hat (vgl. [53]).

C.Ludwig nutzt dieses in [44] aus, um das Kryptosystem in niedriger Dimension mit BKZ anzugreifen. Der Autor kommt zu dem Schluß, daß man für eine ausreichende Sicherheit von Micciancio's Cryptosystem Dimensionsgrößen verwenden müßte, welche zu extrem großen Speicheraufwand führen würden. Damit ist das Kryptosystem in der Praxis unbrauchbar.

Das oben beschriebene Vorgehen wird „Embedding-Technik“ genannt, und ist in [53] bzw. [25] beschrieben. Ich will hier noch einmal auf die Grenzen der Methode eingehen. Falls $(m, 1)^\top$ tatsächlich der kürzeste Gittervektor in $L' := L(B')$ ist, dann ist es klar, daß ein Angreifer, falls er erfolgreich sein will, einen Vektor b'_1 finden muß, welcher der Ungleichung $\|b'_1\| < \lambda_1(L)$ genügt. Wir können mit der Hilfe von $h(R)$ das Verhältnis von $\lambda_1(L')$ und $\lambda_1(L)$ abschätzen. Dazu sei $\|m\|_2 \approx \frac{1}{\beta} h(R)$ mit $\beta > 2$. Dann genügt es, einen Vektor b'_1 zu finden, mit

$$\|b'_1\|_2^2 < \beta^2 \left(\lambda_1^2(L') - 1 \right) .$$

Damit das für den Angreifer zu lösende SVP möglichst schwer wird, ist bei der Wahl des Klartextraums zu beachten, daß die untere Schranke für β nicht von n abhängt.

5.3 Eigenschaften des Orthogonalisationsdefekts

Im Gegensatz zu anderen Kryptosystemen ist hier nicht nur die Länge des kürzesten Gittervektors interessant, sondern ebenfalls $\text{odef}(B)$, $\text{odef}^*(B)$ und $h(B)$. Bei der in [25] vorgestellten „Brute Force Attack“ auf GGH hängt die Größe der Menge, die man nach dem nächsten Gittervektor durchsuchen muß, exponentiell von $\text{odef}^*(B)$ ab (siehe auch Satz 3.1.7). Wir wollen nun die Möglichkeit eines Angriffs mittels einer Basis mit kleinem Orthogonalisationsdefekt betrachten.

Lemma 5.3.1 Für eine Basis b_1, \dots, b_n eines Gitters $L \in \mathbb{R}^n$ gilt:

$$h(B) \leq \lambda_1(L) \leq \min_i \|b_i\| \leq \text{odef}(B) \cdot h(B)$$

Beweis. Die erste Ungleichung ($h(B) = \min_i \|\hat{b}_i\| \leq \lambda_1(L)$) gilt nach Lemma 2.1.10. Es ist klar, daß $\|\hat{b}_i\| \leq \|b_i\|$ für alle $i = 1, \dots, n$ gilt, und somit können wir schließen, daß:

$$\begin{aligned} \forall_i : \|b_i\| &\leq \text{odef}(B) \|\hat{b}_i\| \\ \Rightarrow \min_i \|b_i\| &\leq \text{odef}(B) \min_i \|\hat{b}_i\|. \end{aligned}$$

Die mittlere Ungeleichung ist trivial. ■

Es ist also mindestens genau so schwer, einen Vektor zu finden, der nicht länger als $\alpha \cdot \lambda_1(L)$ ist, wie eine Basis zu finden, die einen Orthogonalisationsdefekt kleiner als α hat. Eine Basis mit minimalem Orthogonalisationsdefekt zu finden ist \mathcal{NP} -schwer, da man dazu den kürzesten Gittervektor finden muß. Gelingt es jedoch eine Basis mit sehr kleinem Orthogonalisationsdefekt zu finden, so ist vielleicht eine Entschlüsselung mit Satz 5.1.2 möglich (wenn $\|m\| < \frac{1}{2}h(B)$). P.Klein stellt in [36] einen Algorithmus vor, mit dem man eine Nachricht auch dann noch entschlüsseln kann, wenn $\|m\| \geq \frac{1}{2}h(B)$. Der Algorithmus hat eine Laufzeit von $\mathcal{O}\left(n^{\|x-y\|^2/h(B)^2}\right)$, wobei B, y das CVP und x der entsprechende Lösungsvektor sei. Für den hier betrachteten Spezialfall ergibt sich $\mathcal{O}\left(n^{h(R)/(4 \cdot h(B))}\right)$ bzw. nach Lemma 5.3.1 $\mathcal{O}\left(n^{(h(R) \cdot \text{odef}(B))/(4 \cdot \lambda_1(L))}\right)$ für die Laufzeit des Algorithmus.

Lemma 5.3.2 Für eine Basis b_1, \dots, b_n eines Gitters L gilt:

$$1 \leq h(B) \cdot \max_i \|b_i^*\| \leq \text{odef}^*(B)$$

Beweis. Da für alle $i = 1, \dots, n$ gilt, daß $\|\hat{b}_i\|^{-1} = \|\hat{b}_{n-i+1}^*\|$ ist, folgt:

$$\begin{aligned} \forall_i : \|\hat{b}_i\|^{-1} &\leq \|\hat{b}_{n-i+1}^*\| \leq \text{odef}^*(B) \|\hat{b}_i\|^{-1} \\ \Rightarrow \frac{1}{\min_i \|\hat{b}_i\|} &\leq \max_i \|b_i^*\| \leq \frac{\text{odef}^*(B)}{\min_i \|\hat{b}_i\|} \end{aligned}$$

Durch eine Multiplikation mit $h(B)$ folgt die Behauptung. ■

Demnach ist eine Entschlüsselung mittels Satz 5.1.2 eher möglich als eine unter Verwendung von Satz 3.1.7. Mit Lemma 3.1.4, Satz 3.2.3 und Satz 3.3.2 ergibt sich die folgende Tabelle für die Annäherungsfaktoren für den Orthogonalisationsdefekt:

	LLL	BKZ	HKZ
$\frac{\ b_1\ }{\lambda_1}$	$2^{(n-1)/2}$	$\gamma_k^{\frac{n-1}{k-1}}$	1
$\text{odef}(B)$	$2^{n(n-1)/4}$	$\gamma_k^{\frac{n-1}{k-1}} \cdot \frac{\prod_{i=1}^n \lambda_i}{\det(L)} \cdot \sqrt{\frac{(n+3)!}{6 \cdot 4^n}}$	$\frac{\prod_{i=1}^n \lambda_i}{\det(L)} \cdot \sqrt{\frac{(n+3)!}{6 \cdot 4^n}}$

Nach Satz 2.1.8 ist $\prod_{i=1}^n \lambda_i \leq \gamma_n^{\frac{n}{2}} \cdot \det(L)$. Falls n genügend groß ist, gilt

$$\gamma_n^{\frac{n}{2}} \cdot \sqrt{(n+3)! / (6 \cdot 4^n)} \leq n^{\frac{n}{2}} \cdot n^{\frac{n}{2}} \cdot 2^{-n} \leq 2^{n((\log_2 n) - 1)} < 2^{n(n-1)/4}.$$

Deshalb ist die Schranke für den Orthogonalisationsdefekt einer HKZ-reduzierten Basis besser, als die einer LLL-reduzierten Basis. Gleiches ist für eine BKZ-reduzierte Basis leicht zu sehen, falls $\gamma_k^{1/(k-1)} < 2^{1/4}$ gilt, also z.B. für $k = 20$.

Die Sicherheit des Cryptosystems von Micciancio beruht auf der Schwierigkeit, eine zum Entschlüsseln geeignete Basis B' zu finden, wenn nur die öffentliche Basis B bekannt ist. Kann man so eine Basis nicht finden, ist dennoch ein Angriff mittels der „Embedding-Technik“ möglich, welcher aber für jede Nachricht erneut durchgeführt werden muß. Es bleiben offene Fragen, die ich bislang nicht beantworten konnte:

- Wie verhalten sich $\text{odef}(B)$, $\text{odef}^*(B)$ und $h(B)$ in zufälligen Matrizen?
- Hängen $\text{odef}(B)$, $\text{odef}^*(B)$ und $h(B)$ zusammen, und wenn ja, wie?

Eine Anregung geben die folgenden Beispiele:

Beispiel 5.3.3 Wir betrachten die folgenden drei Basismatrizen des Gitters L :

$$A := \begin{pmatrix} \sqrt{5} & \sqrt{5} & 0 \\ 0 & 2 & 3 \\ 0 & 1 & 4 \end{pmatrix} \quad B := \begin{pmatrix} \sqrt{5} & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 3 & 4 \end{pmatrix} \quad C := \begin{pmatrix} \sqrt{5} & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 1 & 4 \end{pmatrix}$$

Es ergeben sich die Werte:

	odef	odef*	$h(\cdot)$
A	$\sqrt{10}$	$\sqrt{30}$	$\sqrt{5}$
B	$\sqrt{10}$	$\sqrt{10}$	$\sqrt{2.5}$
C	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{5}$

Damit ist kein direkter Zusammenhang zwischen odef , odef^* und $h(\cdot)$ einer Matrix erkennbar.

Beispiel 5.3.4 Wir betrachten die Gitterbasismatrizen mit der folgenden Struktur:

$$B := \begin{pmatrix} 1 & \alpha & \beta \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix}$$

Mit $\beta = \alpha^2$ ergibt sich $\text{odef}(B)/\text{odef}^*(B) \xrightarrow{\alpha \text{ groß}} \alpha^2$. Für $\alpha = \beta = 1/2$ ist B HKZ-reduziert und das Verhältnis $\text{odef}(B)/\text{odef}^*(B)$ ist $\sqrt{16/14}$. Auch hier ist damit kein direkter Zusammenhang zu vermuten.

6 Das NTRU Kryptosystem

Das NTRU Cryptosystem von J. Hoffstein, J. Pipher und J.H. Silverman aus [29] wird durch $(N, p, q) \in \mathbb{N}^3$ und die vier Mengen von Polynomen von Grad $N-1$: $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\Phi$ und \mathcal{L}_m beschrieben. Es operiert in dem Ring $R = \mathbb{Z}[X] / (X^N - 1)$. Wir stellen zunächst keine Forderungen an N, p oder q , allerdings nehmen wir an, daß $\gcd(p, q) = 1$ und q wesentlich grösser als p ist. In letzter Zeit wurde der Parameter $p \in \mathbb{N}$ durch $P \in R$ ersetzt [18]. Bei der Wahl von P ist dann zu beachten, daß P relativ prim zu q sein muß, d.h. daß $P \odot R + q \odot R = R$. Während für $p \in \mathbb{N}$ meist $p \in \{2, 3\}$ gewählt wird, ist $P = X + 2$ der häufigste Vorschlag für die Wahl von p als Polynom (siehe [30]). Ein Element $F \in R$ wird im folgenden als Polynom oder Vektor beschrieben:

$$F = \sum_{i=0}^{N-1} F_i X^i \text{ wird mit } (F_0, F_1, \dots, F_{N-1}) \in \mathbb{Z}^N \text{ identifiziert.}$$

Wir werden \odot für die Multiplikation in R schreiben:

$$F \odot G = H = (H_0, \dots, H_{N-1}) \text{ mit}$$

$$H_k = \sum_{i=0}^k F_i G_{k-i} + \sum_{i=k+1}^{N-1} F_i G_{N+k-i} = \sum_{i+j \equiv k \pmod N} F_i G_j$$

Dies ist das vom kanonischen Epimorphismus induzierte Produkt in dem Ring R . Man bildet das Produkt der Polynome F und G in $\mathbb{Z}[X]$ und ermittelt dann seine Restklasse modulo $(X^N - 1)$. Damit sind dann natürlich auch Nullteiler vorhanden. Zum Beispiel sind für $N = 4$ die Polynome $x+1$ und $X^3 - X^2 + X - 1$ Nullteiler. Wenn wir eine Multiplikation modulo q durchführen wollen, so ist damit die Reduktion der einzelnen Koeffizienten modulo q gemeint.

Bemerkung 6.0.5 Die Berechnung eines Produktes in R benötigt normalerweise N^2 Multiplikationen. Da aber die typischen NTRU-Polynome sehr kleine Koeffizienten haben ist dies relativ schnell, falls N nicht zu groß ist. Wenn N groß wird, dann ist es effektiver, das Produkt mittels der „Fast Fourier Transformation“ zu berechnen, welche nur $\mathcal{O}(N \log N)$ Berechnungen benötigt. Dieses Verfahren wird schon für in der Praxis übliche Parametersätze eingesetzt.

Neben der Norm $\|\cdot\|$ aus Definition 4.1.3 definieren wir den Begriff der *Koeffizientenweite* (engl. width):

Definition 6.0.6 Für $F \in R$ sei

$$\|F\|_w := \max_{0 \leq i \leq N} \{F_i\} - \min_{0 \leq i \leq N} \{F_i\}$$

die Koeffizientenweite des Polynoms.

Um Beispiele für die Mengen $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\Phi$ und \mathcal{L}_m zu geben, benutzen wir folgende Notation:

$$\mathcal{L}(d_1, d_2) = \left\{ F \in R \mid \begin{array}{l} \text{hat } d_1 \text{ Koeffizienten gleich } 1 \text{ und } d_2 \\ \text{Koeffizienten gleich } -1, 0 \text{ sonst} \end{array} \right\}$$

Beispiel 6.0.7 (aus [29]) Man wählt $d_f, d_g, d \in \mathbb{N}$ und setzt $\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1)$, $\mathcal{L}_g = \mathcal{L}(d_g, d_g)$ und $\mathcal{L}_\Phi = \mathcal{L}(d, d)$. Es ist zu beachten, daß \mathcal{L}_f speziell so gewählt wurde, daß es eine Menge invertierbarer Polynome ist. Als Nachrichtenraum wählen wir $\mathcal{L}_m = \{m \in R : |m_i| \leq (p-1)/2\}$. Wenn man nun konkrete Zahlen angeben will, so lauten die (veralteten) Vorschläge aus [29]:

N	p	q	d_f	d_g	d	Sicherheit
107	3	64	15	12	5	gering
167	3	128	61	20	18	mittel
503	3	256	216	72	55	hoch

In letzter Zeit wurden aber auch häufig Mengen der Form $\mathcal{L}_f = \mathcal{L}(d_f, 0)$, bzw. $\mathcal{L}_f = \{1 + p \odot F | F \in \mathcal{L}(d_f, 0)\}$, $\mathcal{L}_g = \mathcal{L}(d_g, 0)$ und $\mathcal{L}_\Phi = \mathcal{L}(d, 0)$ vorgeschlagen. Besonders die Wahl von \mathcal{L}_f soll die Entschlüsselung beschleunigen.

Um die Schlüssel zu erzeugen, wählt man nun $f \in \mathcal{L}_f$ und $g \in \mathcal{L}_g$ so, daß f modulo p und modulo q invertierbar ist. Wir berechnen $F_p \odot f \equiv 1 \pmod{p}$ und $F_q \odot f \equiv 1 \pmod{q}$. Dann ist der geheime Schlüssel f und der öffentliche h ergibt sich als

$$h = F_q \odot g \pmod{q}.$$

Die Verschlüsselung eines Klartextes $m \in \mathcal{L}_m$ erfolgt durch die Auswahl eines $\Phi \in \mathcal{L}_\Phi$ und der Berechnung des Schlüsseltextes

$$\hat{e} = p \odot \Phi \odot h + m \pmod{q}.$$

Zur Entschlüsselung eines Schlüsseltextes \hat{e} berechnet man zuerst

$$a = f \odot \hat{e} \pmod{q},$$

wobei die Koeffizienten von a aus einem passendem Intervall gewählt werden. Auf die Wahl des Intervalls wollen wir in Abschnitt 6.6 eingehen. Der Klartext kann nun durch die Berechnung von

$$F_p \odot a \pmod{p}$$

ermittelt werden. Die Entschlüsselung erfolgt in den meisten Fällen korrekt falls $\|p\Phi \odot g + f \odot m\|_w < q$, da man dann a die nichtmodulare Gleichung

$$a = p \odot \Phi \odot g + f \odot m$$

erfüllt, falls wir die Koeffizienten von a aus dem richtigen Intervall gewählt haben. Um dieses zu sehen, betrachte man die folgenden Kongruenzen:

$$\begin{aligned} a \equiv f \odot \hat{e} &\equiv f \odot p \odot \Phi \odot h + f \odot m \pmod{q} \\ &\equiv f \odot p \odot \Phi \odot F_q \odot g + f \odot m \pmod{q} \\ &\equiv p \odot \Phi \odot g + f \odot m \pmod{q}. \end{aligned}$$

In dem Beispiel 6.0.7 wäre ein passendes Intervall $[-q/2, q/2]$. Die Entschlüsselung erfolgt dann korrekt, falls $\|p\Phi \odot g + f \odot m\|_\infty < q/2$, was nach

[29] fast immer der Fall ist, wenn $\|f \odot m\|_\infty \leq q/4$ und $\|p \odot \Phi \odot g\|_\infty \leq q/4$. Im allgemeinen versucht man eine geringe Fehlerwahrscheinlichkeit durch eine geschickte Wahl der Mengen \mathcal{L}_f , \mathcal{L}_g , \mathcal{L}_Φ und \mathcal{L}_m zu erreichen. Wie geringe Fehlerwahrscheinlichkeiten und hohe Sicherheit gleichzeitig gewährleistet werden soll, ist noch unklar (vergleiche [69] und [34]).

Annahme 2 Für alle $\epsilon > 0$ gibt es Konstanten γ_1 und γ_2 , so daß für zwei zufällig und gleichverteilt gewählte Polynome $F, G \in R$ die Wahrscheinlichkeit, daß

$$\gamma_1 \|F\| \cdot \|G\| \leq \|F \odot G\|_\infty \leq \gamma_2 \|F\| \cdot \|G\|$$

gilt, größer als $1 - \epsilon$ ist.

Wir wollen mit dieser Annahme auf eine geeignete Wahl der Parameter \mathcal{L}_f , \mathcal{L}_g , \mathcal{L}_Φ und \mathcal{L}_m schließen. Daher wäre die Annahme aus der Sicht der Praxis nutzlos, wenn das Verhältnis γ_2/γ_1 für kleine ϵ sehr groß wäre. Es stellt sich jedoch heraus, daß auch für relativ große Werte N und sehr kleine ϵ die Werte γ_1 , γ_2 nicht extrem sind. J. Hoffstein, J. Pipher und J.H. Silverman behaupten in [29], daß sie dafür eine exponentiell große Anzahl von Beispielen untersucht, und die Skizze eines Beweises für die Korrektheit der Annahme haben.

Nach Annahme 2 sollte man die Parameter aus Beispiel 6.0.7 so wählen, daß

$$\|f\| \cdot \|m\| \approx q/4\gamma_2 \text{ und } \|\Phi\| \cdot \|g\| \approx q/4p\gamma_2$$

gilt. Für die Normen der Polynome erhalten wir $\|\Phi\| = \sqrt{2d}$, $\|g\| = \sqrt{2d_g}$ und $\|f\| = \sqrt{2d_f - 1 - N^{-1}}$. Für $N = 107$, 167 und 503 ergeben sich nach [29] die Werte $\gamma_2 = 0.35$, 0.27 bzw. 0.17 .

6.1 Gitterangriffe auf den privaten Schlüssel

Dieser Angriff geht auf D. Coppersmith und A. Shamir zurück [13]. Wir wollen zunächst davon ausgehen, daß \mathcal{L}_f von der Form $\mathcal{L}(d_1, d_2)$ ist. Wir betrachten die folgende $2N \times 2N$ -Matrix, welche aus vier $N \times N$ -Blöcken besteht:

$$B := \left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & \alpha & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array} \right),$$

wobei wir den Parameter α später erläutern wollen. Sei $L = L(B)$ das Gitter, welches durch die Zeilenvektoren dieser Matrix generiert wird. Wir werden es im folgenden *NTRU-Gitter* nennen. Die Gitterdeterminante ist $\det(L) = q^N \alpha^N$ und das Gitter enthält den Vektor $\tau = (\alpha f, g) \in \mathbb{R}^{2N}$, da der öffentliche Schlüssel $h \equiv g \odot F_q \pmod{q}$ ist.

Bemerkung 6.1.1 Das Gitter L enthält ebenfalls die Vektoren $\tau^{(k)}$, die durch eine zyklische Verschiebung der Einträge von λf und g in τ um eine gleiche Anzahl von Stellen entstehen:

$$\tau^{(k)} := (\lambda f_k, \lambda f_{k+1}, \dots, f_{k-1}, g_k, g_{k+1}, \dots, g_{k-1}).$$

Dieses wird klar, wenn man die zyklische Anordnung der Einträge von h in der Basismatrix des Gitters betrachtet.

Sei s die nach der Heuristik von Gauss erwartete untere Schranke für $\lambda_1(L)$:

$$s := \sqrt{\frac{\dim(L)}{2\pi e}} (\det(L))^{\frac{1}{\dim(L)}} \leq \lambda_1(L).$$

In unserem Fall hat das Gitter die Dimension $2N$ und eine Gitterdeterminante von $q^N \alpha^N$. Man würde also $\lambda_1(L) \approx \sqrt{(N\alpha q)/(\pi e)}$ erwarten. Da der Angreifer die Länge des Vektors τ kennt, wird er versuchen, das Verhältnis $s/\|\tau\|$ zu maximieren, damit er τ als kürzesten Gittervektor leicht finden kann. Wenn man dieses Verhältnis quadriert, kann man sehen, daß der Angreifer α so wählen sollte, daß

$$\frac{\alpha}{\alpha^2\|f\|^2 + \|g\|^2} = \left(\alpha\|f\|^2 + \frac{1}{\alpha}\|g\|^2 \right)^{-1}$$

maximiert wird. Dieses wird erreicht, wenn $\alpha = \|g\|/\|f\|$ gewählt wird, und damit ergibt sich

$$c_h^{-1} := \frac{s}{\|\tau\|} = \sqrt{\frac{Nq}{2\pi e \cdot \|f\| \cdot \|g\|}}.$$

Dabei sind zwar g und f dem Angreifer nicht bekannt, jedoch ihre Länge, da er den Parametersatz kennt, der zum Erstellen des Schlüsselpaars verwendet wurde. Wenn c_h nahe bei 1 liegt, dann ist die Länge des Zielvektors in etwa so lang, wie die Länge des kürzesten Vektors in einem zufälligen Gitter und Reduktionsalgorithmen werden den gesuchten Vektor nur schwer finden können. Mit kleiner werdendem c_h wird es einfacher sein, den Vektor τ zu finden, allerdings scheint der Aufwand weiterhin exponentiell zu sein. Bislang sind in Experimenten nur BKZ-Varianten in der Lage gewesen, einen erfolgreichen Angriff durchzuführen (vergleiche [29], [31]). Auf mehr theoretische Überlegungen wollen wir in Abschnitt 6.7 eingehen.

Bemerkung 6.1.2 Ist f in der Form $1 + p \odot F$ mit F zufällig, so ist der Vektor $(\alpha F, g - h)$ in dem Gitter

$$\left(\begin{array}{c|c} \alpha \cdot Id & p \odot h \\ \hline 0 & q \cdot Id \end{array} \right)$$

enthalten, da

$$\begin{aligned} f \odot h &\equiv g \pmod{q} \\ \Leftrightarrow (1 + p \odot F) \odot h &\equiv g \pmod{q} \\ \Leftrightarrow p \odot F \odot h &\equiv g - h \pmod{q}. \end{aligned}$$

Der gesuchte Vektor ist $(\alpha F, g)$ von dem nicht im Gitter enthaltenen Vektor $(0, -h)$ entfernt. Ein Angreifer wird versuchen, dieses CVP zu lösen. Die dazu nötige Gitterbasisreduktion hat dieselbe Laufzeit, wie die zuvor betrachtete Variante. Es ist also nicht nötig, diesen Fall separat zu betrachten, (vgl. [31]).

Um zu dem Beispiel 6.0.7 zurück zu kommen, so ergibt sich mit den Zahlen aus [29]:

N	p	q	d_f	d_g	d	γ_2	c_h	Sicherheit
107	3	64	15	12	5	0.35	0.257	gering
167	3	128	61	20	18	0.27	0.236	mittel
503	3	256	216	72	55	0.17	0.182	hoch

6.2 Gitterangriffe auf den Schlüsseltext

Nach [29] kann ein Gitterangriff analog zu dem Angriff auf den privaten Schlüssel auf eine einzelne Nachricht durchgeführt werden. Der gesuchte Vektor hat hierbei die Form $(\alpha m, \Phi)$ Wie oben sollte der Angreifer $\alpha = \|\Phi\|/\|m\|$ wählen, womit sich der Quotient

$$c_m^{-1} := \frac{s}{\|\tau\|} = \sqrt{\frac{Nq}{2\pi e \cdot \|\Phi\| \cdot \|m\|}}$$

ergibt. Wie auch oben gibt dieser Wert ein Maß der Angreifbarkeit einer einzelnen Nachricht wieder.

Bemerkung 6.2.1 Leider wird die entsprechende Matrix nicht vorgestellt. Ich schlage daher vor, die Matrix

$$B_{\hat{e}} = \left(\begin{array}{c|cc} 1 & 0 \cdots 0 & \hat{e}_0 \cdots \hat{e}_{N-1} \\ \hline 0 & & \\ \vdots & \alpha \cdot Id & p \odot h \\ \hline 0 & & \\ \vdots & 0 & q \cdot Id \\ \hline 0 & & \end{array} \right),$$

der Dimension $1 + 2N$ zu verwenden. Man kann leicht sehen, daß für den Schlüsseltext \hat{e} der Vektor $(1, \alpha \Phi, m)$ in dem Gitter $L(B_{\hat{e}})$ liegt. Wir erwarten dabei, daß $\|(1, 0, \hat{e})\| > \sqrt{N/(2\pi e)} \cdot \alpha q$ gilt. Dann wäre der Vektor in der ersten Zeile der Matrix lang genug, so daß sich die Überlegungen von oben übertragen. Die Erwartung scheint wegen der Struktur von \hat{e} berechtigt zu sein. Da $\det(B_{\hat{e}})$ gleich der Determinante der Matrix für den Angriff auf den privaten Schlüssel ist, sind die Überlegungen für die Wahl von α analog zu denen aus dem vorigen Abschnitt.

Um Angriffe auf f und m gleich schwer zu machen, ist es sinnvoll, die Parameter so zu wählen, daß $c_h \approx c_m$ bzw. $\|f\| \cdot \|g\| \approx \|\Phi\| \cdot \|m\|$ erreicht wird.

6.3 Gitterangriffe auf alternative (gefälschte) Schlüssel

Für den Angriff von D. Coppersmith und A. Shamir aus [13] führen wir die folgende Bezeichnungen ein:

Definition 6.3.1 Wir schreiben $\|x\|_{\perp}$ für die L^2 -Norm der orthogonalen Projektion von x entlang $\vec{1} := (1, 1, \dots, 1)^{\top}$. Mit $\bar{x} := \frac{1}{N} \sum_{i=0}^{N-1} x_i$ gilt:

$$\|x\|_{\perp}^2 = \left(\sum_{i=0}^{N-1} (x_i - \bar{x})^2 \right).$$

Den Wert $\|x\|_{\perp}$ kann man also als die Standardabweichung der Einträge von x , skaliert mit dem Faktor \sqrt{N} sehen. Diese Norm ist invariant unter der Addition von Vielfachen von $\vec{1}$, und damit $\|x \pmod{q}\|_{\perp} = \|x\|_{\perp}$.

Wenn wir nochmals die Entschlüsselung betrachten, dann berechnet der Empfänger der Nachricht

$$a = f \odot \hat{e} \pmod{q} .$$

Er wird versuchen, die Koeffizienten von a aus einem bestimmten Intervall, z.B. $[-q/2, q/2]$ zu wählen, so daß die Koeffizienten von a gleich denen von $b := p \odot \phi \odot g + f \odot m$ sind. Wir nennen den so modifizierten Vektor b . Ob es gelingt so einen Vektor zu konstruieren, können wir an der Standardabweichung $\sigma = \|b\|_{\perp} / \sqrt{N}$ messen. D. Coppersmith und A. Shamir zeigen, daß $\|x\|_{\perp} \cdot \|y\|_{\perp}$ eine gute Annäherung für $\|x \odot y\|_{\perp}$ ist, womit

$$\|b\|_{\perp}^2 \approx p^2 \|\Phi\|_{\perp}^2 \cdot \|g\|_{\perp}^2 + \|m\|_{\perp}^2 \cdot \|f\|_{\perp}^2$$

gilt. Somit ergibt sich, daß $q/2 \geq 5\sigma$ die Entschlüsselung der meisten Nachrichten ermöglicht.

Nehmen wir an, es gäbe einen alternativen Vektor f' , welchen ein Angreifer anstelle des korrekten geheimen Schlüssels f zur Entschlüsselung benutzen möchte. Der Angreifer kann ein g' mit

$$h = F'_q \odot g' \pmod{q}$$

berechnen, wobei $F'_q \odot f' \equiv 1 \pmod{q}$ sei. Dieses alternative Schlüsselpaar könnte nun zur Entschlüsselung von \hat{e} benutzt werden, wenn man aus $a' = f' \odot \hat{e} \pmod{q}$ die Koeffizienten von $b' = p \odot \phi \odot g' + f' \odot m \equiv a' \pmod{q}$ bestimmen könnte. Um zu bestimmen, wann dies der Fall ist, schätzen wir wie oben den Wert von $\|b'\|_{\perp}$ ab. Wenn man typische Werte für $\|m\|_{\perp}$ und $\|\Phi\|_{\perp}$ annimmt, erhält man mit $\alpha = \|m\|_{\perp} / (p \cdot \|\Phi\|_{\perp})$ die Abschätzung

$$\sigma' = \frac{\|b'\|_{\perp}}{N} \approx \frac{p^2 \|\Phi\|_{\perp}^2}{N} \left(\|g'\|_{\perp}^2 + \alpha \|f'\|_{\perp}^2 \right) .$$

Wenn also σ' bzw. $\|b'\|_{\perp}$ klein genug ist, dann können wir \hat{e} mit dem alternativen Schlüsselpaar entschlüsseln.

Um ein solches alternatives Schlüsselpaar zu erhalten betrachten wir zunächst nochmals die Matrix L aus 6.1. Dann gilt:

$$\exists_{x \in \mathbb{Z}^N} : L^\top \begin{pmatrix} f \\ x \end{pmatrix} = \begin{pmatrix} \alpha f \\ g \end{pmatrix} \text{ und } \exists_{x' \in \mathbb{Z}^N} : L^\top \begin{pmatrix} f' \\ x' \end{pmatrix} = \begin{pmatrix} \alpha f' \\ g' \end{pmatrix}$$

Da wir an der $\|\cdot\|_\perp$ -Norm interessiert sind, bilden D. Coppersmith und A. Shamir nun die Matrix

$$L' := L - \left(\begin{array}{c|c} \frac{\alpha}{N}1 & \beta 1 \\ \hline 0 & \frac{q}{N}1 \end{array} \right),$$

wobei ein Eintrag in der subtrahierten Matrix als $N \times N$ -Block, β als ein geeigneter Skalar und 1 als die Matrix mit allen Einträgen gleich 1 zu verstehen ist.

Bemerkung 6.3.2 L' hat nur $2N - 2$ unabhängige Vektoren, da

$$L' \begin{pmatrix} \vec{1} \\ 0 \end{pmatrix} = L' \begin{pmatrix} 0 \\ \vec{1} \end{pmatrix} = 0,$$

wenn man β geschickt wählt.

Ein gesuchter Gittervektor wäre jetzt ein Vektor der Form

$$L'^\top \begin{pmatrix} f' \\ y \end{pmatrix} = \begin{pmatrix} \alpha \left(f' - (\vec{f}' \cdot \vec{1}) \right) \\ g' - (\vec{g}' \cdot \vec{1}) \end{pmatrix}$$

mit $y \in \mathbb{Z}^N$. Seine quadrierte L^2 -Norm läßt sich wie folgt abschätzen:

$$\begin{aligned} \left\| L'^\top \begin{pmatrix} f' \\ y \end{pmatrix} \right\|^2 &= \alpha^2 \|f'\|_\perp^2 + \|g'\|_\perp^2 \\ &= \left(\frac{\|m\|_\perp^2 \cdot \|f'\|_\perp^2 + p^2 \|\Phi\|_\perp^2 \cdot \|g'\|_\perp^2}{p^2 \|\Phi\|_\perp^2} \right) \\ &\approx \frac{\|b'\|_\perp^2}{p^2 \|\Phi\|_\perp^2} \end{aligned}$$

Die L^2 -Norm des gesuchten Gittervektors steht folglich in direktem Zusammenhang mit der Eignung von f' als alternativem privatem Schlüssel. Als weitere Forderung haben wir die Invertierbarkeit von f' modulo p , aber das ist laut [13] ein schwaches Kriterium. Wir wollen nun wissen, wann ein alternativer Schlüssel geeignet ist. Dazu sei

$$n_{f'} := (p\|\Phi\|_\perp) \min_{y \in \mathbb{Z}^n} \left\| L'^\top \begin{pmatrix} f' \\ y \end{pmatrix} \right\| = \|b'\|$$

Wir haben gesehen, daß für den richtigen Schlüssel $n_f < q/10$ gelten sollte. Coppersmith und Shamir schließen, daß ein alternativer Vektor f' mit $n_{f'} \leq n_f$,

zwei alternative Vektoren mit $n_{f'} = q/4$ bzw. $n_{f'} = 2.5 \cdot n_f$ oder mehrere Vektoren mit $n_{f'} \approx 4 \cdot n_f$ zum Entschlüsseln reichen würden. Die Annäherung des Standard LLL-Algorithmus mit $n_{f'} \leq 2^{N/2} n_f$ reicht also bei weitem nicht aus. Ein weiteres Problem dieses Angriffs ist jedoch, daß diese spezielle Form von Basismatrix ein Gitter erzeugt, welches in (fast) allen Experimenten keine (linear unabhängigen) Vektoren in der Nähe von dem kürzesten Vektor hat. Also findet der BKZ-Algorithmus entweder einen Vektor, welcher nicht länger als der Zielvektor ist, oder nur ungeeignete Vektoren (siehe dazu [31]).

Der Angriff scheint für die Praxis von geringer Relevanz zu sein, jedoch ist zu bemerken, daß wir nicht den Zielvektor selbst finden müssen. So sind z.B. alle Vektoren aus Bemerkung 6.1.1 geeignete alternative Schlüssel.

6.4 „Zero-Run Lattices“ und „Zero-Forced Lattices“

A. May hat in [46] und [47] sogenannte „Zero-Run Lattices“ eingeführt, welche die hohe Anzahl von Nullen in dem Zielvektor der NTRU-Matrix ausnutzen.

Eine Verallgemeinerung dieser sind die von J.H. Silverman in [68] vorgestellten „Zero-Forced Lattices“ (ZFL), deren Effektivität in [31] analysiert wird. Letztere sind aus praktischer Sicht erfolgsversprechender als die Matrizen von A. May. Auch sie nutzen die hohe Anzahl von Nullen in dem Zielvektor aus. Dabei raten sie die Abfolge der Nullen in dem g -Anteil des Zielvektors. Dieses reicht nach Bemerkung 6.1.1 aus, egal an welcher Stelle diese Abfolge von Nullen beginnt.

Bemerkung 6.4.1 Bei einem Angriff mittels ZFL auf die Nachricht versagt das Gitter aus Bemerkung 6.2.1, da in diesem Gitter keine der zyklisch vertauschten Varianten des Zielvektors enthalten ist. Man müßte also die exakten Positionen der Nullen von m raten, falls man es dennoch verwenden will.

Der erste Schritt zur Generierung einer ZFL ist die Auswahl von einer Zahl $r \in \{0, 1, \dots, N\}$ und einer Indexmenge

$$J \subseteq \{0, \dots, N-1\} \text{ mit } |J| = r .$$

Man sucht also nach Zielvektoren, die an den Stellen $j \in J$ Nullen als Einträge haben. Um diese zu finden, löst man die Kongruenzen, welche man aus der ursprünglichen Matrix L ablesen kann:

$$f_0 h_j + f_1 h_{j+1} + \dots + f_{N-1} h_{j-1} \equiv g_j \pmod{q}, \quad 0 \leq j < N,$$

wobei man $g_j = 0$ für alle $j \in J$ setzt. Diese kann man dann in die restlichen Kongruenzen resubstituieren und erhält ein neues System von Kongruenzen. Nehmen wir an, wir hätten die Kongruenzen nach den letzten r Koordinaten von f aufgelöst, dann gilt:

$$a_{0j} f_0 + a_{1j} f_1 + \dots + a_{(N-1-r)j} f_{N-1-r} \equiv g_j \pmod{q}, \quad j \notin J .$$

Wir erhalten nun ein System mit $N - r$ modularen Gleichungen und $2(N - r)$ Unbekannten. Wir können dieses wieder mit Hilfe einer Matrix formulieren:

$$B^{ZF_J} := \left(\begin{array}{cccc|ccc} \alpha & 0 & \cdots & 0 & a_{11} & a_{12} & \cdots \\ 0 & \alpha & \cdots & 0 & a_{21} & \ddots & \\ \vdots & & \ddots & \vdots & \vdots & & \\ 0 & 0 & \cdots & \alpha & & & \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & & & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array} \right)$$

Das zugehörige Gitter definieren wir als $L^{ZF_J} = L(B^{ZF_J})$. Der neue Zielvektor hat dann die Form:

$$\tau^{ZF_J} = \left((f_i)_{i \in \{0, \dots, N-r-1\}}, (g_j)_{j \notin J} \right)$$

und analog zu Bemerkung 6.1.1 wären

$$\tau^{ZF_J(k)} = \left((f_{i+k})_{i \in \{0, \dots, N-r-1\}}, (g_{j+k})_{j \notin J} \right)$$

in dem Gitter enthalten.

Nach der Heuristik von Gauss ist der kürzeste Gittervektor nun größer als

$$s^{ZF_J} = \sqrt{\frac{N}{2\pi e}} \cdot (\alpha q)^{(N-r)/2(N-r)}$$

zu erwarten, während die Länge des neuen Zielvektors nicht grösser als die des alten ist. Damit ergibt sich ein Verhältnis von

$$\left(c_h^{ZF_J} \right)^{-1} := \frac{s^{ZF_J}}{\|\tau\|} = \sqrt{\frac{N\alpha q}{2\pi e(\|\alpha f\|^2 + \|g\|^2)}}.$$

Wenn wir also die Überlegungen für eine optimale Wahl von α wiederholen, dann gelangen wir zu dem gleichen Verhältnis $c_h^{ZF_J} = c_h$, d.h. alle Überlegungen zur Annäherung des Zielvektors laufen analog zu denen in Kapitel 6.1. Es ist klar, daß ein Gitterbasisreduktionsalgorithmus in dieser Matrix wegen der geringeren Dimension wesentlich schneller läuft. Weiterhin kann der Zielvektor durch die oben generierten modularen Gleichungen aus dem kürzesten Gittervektor des Gitters L_J^{ZF} rekonstruiert werden.

Bemerkung 6.4.2 Es ergibt sich die Möglichkeit eines parallelisierten Angriffs, denn verschiedene Computer könnten unabhängig voneinander die Nullstellen raten.

Um zu sehen, wie hoch die Wahrscheinlichkeit ist, daß man die Abfolge der Nullen in g richtig erraten hat brachen wir den folgenden Satz:

Satz 6.4.3 Seien $N, m, r \in \mathbb{N}$ gegeben, $b = (g_0, \dots, g_{N-1})$ ein zufällig gewählter Vektor, der $N - m$ Einträge gleich 0 und die restlichen m Einträge ungleich 0 hat und $J = \{j_1, j_2, \dots, j_r\}$ mit $0 \leq j_1 < j_2 < \dots < j_r < N$ eine zufällig gewählte Menge von Indizes, dann gilt:

1. $\text{Prob}(g_{j_1} = g_{j_2} = \dots = g_{j_r} = 0) = \frac{\binom{N-r}{m}}{\binom{N}{m}} = \prod_{i=0}^{m-1} \left(1 - \frac{r}{N-i}\right)$
2. $\text{Prob}\left(\begin{array}{l} g_{j_1+k} = g_{j_2+k} = \dots = g_{j_r+k} = 0 \\ \text{für ein } 0 \leq k < N \end{array}\right) \approx 1 - \left(1 - \prod_{i=0}^{m-1} \left(1 - \frac{r}{N-i}\right)\right)^N$

Beweis. Siehe [68] ■

Der erwartete Zeitvorteil ist demnach:

$$\text{Prob}\left(\begin{array}{l} L^{ZF_J} \text{ enthält} \\ \text{einen Zielvektor} \end{array}\right) \cdot \left(\frac{\text{Laufzeit in } L}{\text{Laufzeit in } L^{ZF_J}}\right),$$

wobei hier mit „Laufzeit“ die Zeit gemeint ist, die benötigt wird, um den Zielvektor zu finden.

6.5 „Dimension-Reduced Lattices“

Auch diese Idee geht auf A. May zurück, die er in [47] beschreibt. Der Grundgedanke basiert auf der folgende Annahme, welche man auf der Grundlage von Beobachtungen von J.H. Hoffstein (siehe u.a. [31]) macht:

Annahme 3 Für das NTRU-Gitter gilt, daß der kürzeste Vektor, der länger als einer der Zielvektoren $\tau^{(k)}$ und von diesen linear unabhängig ist, länger als s ist.

Gilt diese Annahme, so reicht es eventuell aus, nicht alle der n modularen Gleichungen zu erfüllen, sondern nur für die meisten davon, da es nur wenige kurze Gittervektoren gibt. Man hofft, danach den Zielvektor rekonstruieren zu können.

Zur Generierung einer solchen Dimensionsreduzierten Matrix wählt man zwei Konstanten $\alpha, \beta \in]0, 1]$ und betrachtet bei der Suche nach dem Zielvektor nur $\lfloor \beta N \rfloor$ Koeffizienten von f sowie $\lfloor \alpha N \rfloor$ von g . J.H. Silverman bemerkt in [68], daß man dieses Verfahren sowohl auf L , als auch auf L^{ZF_J} anwenden kann.

Wir wollen hier Beispielhaft von L ausgehen. Man bildet nun

$$B^{\alpha,\beta} := \left(\begin{array}{cccc|cccc} \lambda & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{\lceil \alpha N \rceil - 1} \\ 0 & \lambda & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lceil \alpha N \rceil} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda & h_{\lceil \beta N \rceil - 1} & h_{\lceil \beta N \rceil} & \cdots & h_{\lceil \beta N \rceil + \lceil \alpha N \rceil - 2} \\ 0 & 0 & \cdots & 0 & h_{\lceil \beta N \rceil} & h_{\lceil \beta N \rceil + 1} & \cdots & h_{\lceil \beta N \rceil + \lceil \alpha N \rceil - 1} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{\lceil \alpha N \rceil - 2} \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & & & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array} \right),$$

wobei die Indizes von h modulo N zu verstehen sind. Das Gitter $L^{\alpha,\beta} = L(B^{\alpha,\beta})$ der Zeilenvektoren von $B^{\alpha,\beta}$ hat also eine Dimension von $\lceil \alpha N \rceil + \lceil \beta N \rceil$. Die Gitterdeterminante kann man mit Satz B.1.3 durch $\lambda^{\beta n} q^{(\alpha+\beta-1)n}$ nach unten abschätzen. Damit ergibt sich nach der Volumenheuristik von Gauß (Bemerkung 2.1.6), daß die die Länge des kürzesten Gittervektors größer als

$$s^{\alpha,\beta} = \sqrt{\frac{(\alpha + \beta) n}{2\pi e}} \cdot (\lambda^{\beta} q^{\alpha+\beta-1})^{\frac{1}{\alpha+\beta}}$$

ist, und der Zielvektor $\tau^{\alpha,\beta}$ die Länge $\sqrt{\beta\|\lambda f\|^2 + \alpha\|g\|^2}$ haben wird. Wenn der Angreifer geschickterweise wieder $\lambda = \|g\|/\|f\|$ wählt, ist das Verhältnis der Länge des Zielvektors und $s^{\alpha,\beta}$:

$$\left(c_h^{\alpha,\beta}\right)^{-1} = \sqrt{\frac{q^2 N}{2\pi e (\|f\|^\beta \|g\|^\alpha q)^{2/(\alpha+\beta)}}}.$$

Je nach Parameterwahl hat man nun also den Vorteil der geringeren Dimension, aber den Nachteil, daß das Verhältnis $c_h^{\alpha,\beta}$ kleiner wird, d.h. daß man den Zielvektor besser annähern muß.

Hat man auf diese Weise Teilinformationen über g und f erhalten, bleibt das Problem, die restlichen $(N - \lceil \beta N \rceil)$ Koeffizienten von f bzw. $(N - \lceil \alpha N \rceil)$ von g zu bestimmen, was zu einem modularem Gleichungssystem führt. Genauere Analysen dieses Angriffs finden sich in [31], [46], [47] und [68].

6.6 Entschlüsselungsfehler und ihre Folgen

In vielen Sicherheitsanalysen (z.B. [55]) wurden Entschlüsselungsfehler als vernachlässigbar angenommen. Diese Annahme ist jedoch falsch, wie der folgende Abschnitt zeigt. Wir unterscheiden zwischen „wrap“ und „gap failures“, welche verschiedene Auswirkungen auf die Sicherheit von NTRU haben.

Definition 6.6.1 Ein „wrap failure“ (W-Fehler) liegt vor, wenn $\|a\|_w \leq q$, aber $a \neq p \odot r \odot g + f \odot m \in \mathbb{Z}[X]$. Hier ist nicht die Kongruenz gemeint, die selbstverständlich gilt, sondern daß der gewählte Repräsentant a von $f \odot \hat{e} \in (\mathbb{Z}/q\mathbb{Z})[X]$ die Gleichung nicht erfüllt. (Die Koeffizienten von a liegen nicht in dem richtigen Intervall.)

Liegt ein W-Fehler vor und ist a der gewählte Repräsentant von $f \odot \hat{e}$ in $(\mathbb{Z}/q\mathbb{Z})[X]$, dann gilt $a = p \odot r \odot g + f \odot m + q \cdot k \in \mathbb{Z}[X]$ für ein Polynom k aus R . Damit ist $b = F_p \odot a \pmod{p}$ der Wert, mit dem eine Identifizierung der Nachricht vorgenommen wird, wobei jedoch

$$\begin{aligned} b' &= F_p \odot a - q \cdot k \pmod{p} \\ &= b - q \cdot k \pmod{p} \end{aligned}$$

für eine korrekte Entschlüsselung nötig wäre. In [32] werden zwei Methoden vorgestellt, um die Wahrscheinlichkeit eines W-Fehlers zu begrenzen, bzw. auftretende W-Fehler zu korrigieren. Sie werden auch von EESS in [18] für die Implementation empfohlen.

Eine Methode, um W-Fehler zu erkennen, ist die Verwendung einer Hashfunktion $H : \{0, 1\}^{N-r} \rightarrow \{0, 1\}^r$ für ein $0 < r < N$. Man kann dann für eine Nachricht $M \in \{0, 1\}^{N-r}$ den Klartext $(M, H(M)) \in \mathcal{L}_m$ zur Verschlüsselung wählen. Beim Entschlüsseln wird dann erkannt, ob der Hashwert korrekt ist. Verschiedene Padding-Techniken werden unter anderem in [30], [18], [55] sowie [33] vorgestellt. Allein die letzte Padding-Technik ermöglicht einen Sicherheitsbeweis, solange die Fehlerwahrscheinlichkeiten sehr gering sind, die anderen vernachlässigen sie.

Definition 6.6.2 Ein „gap failure“ (G-Fehler) liegt vor, wenn bei der Entschlüsselung einer Nachricht m durch die Berechnung von $a = p \odot r \odot g + f \odot m$ gilt, daß

$$\|a\|_w > q.$$

Tritt ein G-Fehler auf, so besteht keine Möglichkeit, die Nachricht dennoch zu entschlüsseln. Die Wahrscheinlichkeit eines G-Fehlers hängt allein von der Wahl der NTRU-Parameter $N, p, q, \mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\Phi$ und \mathcal{L}_m ab.

Bevor wir zu einer Darstellung kommen, wie man die Entschlüsselungsfehler in einer wirksamen Attacke auf das NTRU-Cryptosystem nutzen kann, wollen wir eine vereinfachte Schreibweise einführen.

Definition 6.6.3 Für ein $c \in R$ sei das „reversal“ \bar{c} als das Polynom aus R definiert, für das $\bar{c}(X) \equiv c(X^{-1})$ in R gilt. Mit \hat{c} bezeichnen wir das Produkt $c \odot \bar{c}$.

Wenn wir c als den Vektor $(c_0, c_1, \dots, c_{N-1})$ darstellen, dann ist die Darstellung von \bar{c} der Vektor $(c_0, c_{N-1}, c_{N-2}, \dots, c_1)$. Für die Koeffizienten von \hat{c} ergibt sich $\hat{c}_i = \langle c, (X_i \odot c) \rangle$. Ferner ist $\hat{c}_0 = \sum_{i=0}^{N-1} c_i^2 = \|c\|^2$, während $\hat{c}_i \approx \|c\|$. Wendet man diese Beobachtung auf f an, so ergibt sich $\|f \odot \bar{f}\|_w \geq \|f\|$, was

im Vergleich zu dem Produkt von f mit einem zufälligen Vektor gleicher Größe sehr groß ist. Wenn also ein G-Fehler auftritt dann können wir davon ausgehen, daß m stark mit \bar{f} und Φ stark mit \bar{g} korreliert ist.

Wir wollen nun einen Angriff aus [34] betrachten. Er entspricht einer „Chosen Ciphertext Attack“. Man generiert so lange (gültige) Schlüsseltexte, bis bei der Entschlüsselung ein G-Fehler auftritt. Damit haben wir ein 3-Tupel

$$(m, \Phi, \hat{e}) \text{ mit } \|p \odot \Phi \odot g + f \odot m\|_w \geq q .$$

Die Überlegungen, welche wir zuvor gemacht haben, legen nahe, daß für ein $0 \leq i \leq N - 1$ die Polynome Φ und m zu $X^i \odot \bar{f}$ bzw. $X^i \odot \bar{g}$ ähnlich sind. Unglücklicherweise kennen wir dieses i nicht, sonst wäre es einfach, f aus den $X^{-i} \odot m$ zu rekonstruieren, wenn man genügend G-Fehler beobachtet.

Wenn man jedoch den umgekehrten Ansatz verfolgt, so sieht man, daß wenn m und Φ wie $X^i \odot \bar{f}$ bzw. $X^i \odot \bar{g}$ aussehen, \hat{m} und $\hat{\Phi}$ entsprechend wie \hat{f} bzw. \hat{g} aussehen. Hat man beide letztere durch ausreichend viele Beobachtungen identifiziert, dann können f und g nach [23] in Polynomzeit ermittelt werden. Um das Verfahren aus [23] anwenden zu können, muß zunächst das von f erzeugte Ideal $\langle f \rangle$ berechnet werden. Dies ist z.B. dann möglich, wenn es $x, y \in R$ gibt für die

$$x \odot \bar{g} + y \odot \bar{f} = 1 \in R$$

gilt. Dies wird in [34] gefordert, aber nicht in der Version aus [29]. Die Wahrscheinlichkeit, daß ein beliebiges g modulo q invertierbar ist, ist sehr groß, so daß ein Angreifer oBdA davon ausgehen kann, daß obige Bedingung erfüllt ist. Ist dem der Fall, so ist

$$\begin{aligned} \langle \hat{f} \rangle \odot \langle \bar{h} \rangle &= \langle f \odot \bar{f} \rangle \odot \langle \bar{h} \rangle = \langle f \odot \bar{g} \rangle \text{ und} \\ \langle f \odot \bar{g} \rangle \odot \langle \hat{f} \rangle &= \langle f \odot \bar{g} \rangle \odot \langle f \odot \bar{f} \rangle = \langle f \rangle . \end{aligned}$$

Um den oben beschriebenen Angriff durchzuführen, braucht man eine relativ große Anzahl von G-Fehlern. In [34] wird jedoch vorgeschlagen, statt dem üblichen G-Fehler-Orakel für die „Chosen Ciphertext Attacke“ ein Orakel $\mathcal{O}_{w(y)}$ zu verwenden, welches zu einem gegebenen Schlüsseltext zurückgibt, ob

$$\|p \odot \Phi \odot g + f \odot m\|_w \geq y ,$$

oder nicht. Das experimentelle Ergebnis der Autoren spricht dafür, daß es ausreichend ist, dieses Orakel zu betrachten. Auch hier genügt es anscheinend selbst bei y , die im Vergleich mit q relativ klein sind, aus, das Mittel der erhaltenen \hat{m} bzw. $\hat{\Phi}$ zu nehmen, um auf \hat{f} und \hat{g} zu schließen.

Bemerkung 6.6.4 Dieser Angriff funktioniert unabhängig von der verwendeten Padding-Technik. Sein Erfolg beruht allein auf der Häufigkeit von G-Fehlern.

Ein Angriff auf NTRU über die W-Fehler ist ebenfalls möglich (siehe [49] bzw. [34]). Dies ist deshalb interessant, da diese häufiger auftreten als G-Fehler.

Die meisten dieser Angriffe beruhen allerdings auf speziellen Padding-Techniken. In [34] wird bemerkt, daß man W-Fehler eventuell ausnutzen könnte, wenn man eine Reihe von Schlüsseltexten \hat{e} betrachtet, die W-Fehler verursachen. Falls man genügend dieser Fehler findet, so konvergiert das Mittel der \hat{e} gegen $A+B\hat{f}+C\hat{g}$. Diese Idee für einen (von der Padding-Technik unabhängigen) Angriff haben die Autoren leider nicht bis zum Ende verfolgt.

Es ist wichtig, die Wahrscheinlichkeit für Entschlüsselungsfehler abschätzen zu können. Erste Ergebnisse hierzu kann man in [69] finden. Damit man gegenüber einer solchen Attacke sicher ist, sollte ein Angreifer mindestens 2^{80} Anfragen an das Entschlüsselungsorakel stellen müssen, da wir dann davon ausgehen können, daß er dieses nicht in angemessener Zeit durchführen kann. Doch selbst wenn es nicht gelingt, f oder g vollständig zu ermitteln, kann ein Angreifer die Kenntnis von einigen Bits von f bzw. g für einen Gitterangriff analog zu den „Zero-Forced-“ oder „Dimension-Reduced Lattices“ nutzen.

6.7 Einige Überlegungen

Das in 6.1 beschriebene Verhältnis c_h bzw. c_m gibt also an, wie gut die reduzierte Basis den Zielvektor annähern muß, damit der Angriff erfolgreich ist:

$$\|b_1\| < \|\tau\| c_h^{-1} = s$$

Falls Annahme 3 gilt ist dieses genau dann gesichert, falls $\|b_1\| < \lambda_1(L) c_h^{-1}$. Wir erinnern uns an die Definition von c_h^{-1} und c_m^{-1} und die Forderung, daß eine Nachricht genau so schwer anzugreifen sein sollte, wie der private Schlüssel:

$$\sqrt{\frac{Nq}{2\pi e \cdot \|f\| \cdot \|g\|}} = c_h^{-1} \approx c_m^{-1} = \sqrt{\frac{Nq}{2\pi e \cdot \|\Phi\| \cdot \|m\|}}$$

und damit $\|f\| \cdot \|g\| \approx \|\Phi\| \cdot \|m\|$. Weiter wird man, um zu viele G-Fehler zu vermeiden, Beschränkungen für die Normen von f , g , Φ und m in Abhängigkeit von N , q , p und der gewünschten Fehlerrate \mathcal{P}_G für G-Fehler finden müssen, damit

$$\mathcal{P}(\|p \odot \Phi \odot g + f \odot m\|_w \geq q) \leq \mathcal{P}_G .$$

Hier scheint es sinnvoll, analog zu den Überlegungen aus [29] eine Beschränkung der Form

$$\|f\| \cdot \|m\| \leq q/\gamma \text{ und } \|\Phi\| \cdot \|p \odot g\| \leq q/\gamma$$

für ein $\gamma > 0$ zu wählen. Falls $P = X+2$, dann gilt $\|p \odot g\| = \|X \odot g + 2g\| \leq 3\|g\|$. Wir wollen im Folgenden davon ausgehen, daß $P = 3$ oder $P = X + 2$ und alle verwendeten Polynome binär sind. Damit ergibt sich, daß

$$(\|f\| \cdot \|g\|)^2 \approx (\|\Phi\| \cdot \|m\|)^2 \approx \|f\| \cdot \|g\| \cdot \|\Phi\| \cdot \|m\| \leq q^2 / (3\gamma^2)$$

als Forderung ausreichend ist. Andererseits gilt $\|\Phi\| \leq \sqrt{N}$ und $\|m\| \leq \sqrt{N}$, da es sich um binäre Polynome handelt. Damit macht nur $N^2 \geq q^2 / (3\gamma^2)$

Sinn. Wenn wir davon ausgehen, daß q größtmöglich gewählt wird, damit wenig G-Fehler auftreten, ergibt sich:

$$c_h^{-1} \approx c_m^{-1} \geq \sqrt{\frac{q}{2\pi e}} \approx \sqrt{\frac{\sqrt{3}\gamma N}{2\pi e}} \approx 0.318\sqrt{N\gamma}.$$

Die Frage ist daher, wie sich γ verhält. Nehmen wir zunächst einmal an, daß sich γ wie $1/\ln N$ verhält. Dann ergibt sich als Vergleich mit den Daten aus [29] (veraltet, $\gamma = 4\gamma_2$) bzw. dem Aktuellen EESS1 Empfehlungen [18]:

(N, p, q)	(107, 3, 64)	(167, 3, 128)	(503, 3, 256)	(251, 2, 239)
\mathcal{L}_f	$\mathcal{L}(15, 14)$	$\mathcal{L}(61, 60)$	$\mathcal{L}(216, 215)$	$1 + p \odot \mathcal{L}(72, 0)$
\mathcal{L}_g	$\mathcal{L}(12, 12)$	$\mathcal{L}(20, 20)$	$\mathcal{L}(72, 72)$	$\mathcal{L}(72, 0)$
\mathcal{L}_Φ	$\mathcal{L}(5, 5)$	$\mathcal{L}(18, 18)$	$\mathcal{L}(55, 55)$	$\mathcal{L}(72, 0)$
\mathcal{L}_m	$\{m \in \mathbb{R} : m_i \leq (p-1)/2\}$			$\mathcal{L}(125, 0)$
γ	1.4	1.08	0.68	≈ 1.279
c_h^{-1}	3.891	4.237	5.495	5.848
$0.318\sqrt{N\gamma}$	3.892	4.270	5.881	5.697
$\gamma \cdot \ln N$	6.542	5.527	4.230	7.067
\mathcal{P}_G	2^{-31}	2^{-31}	2^{-32}	2^{-25}

Diese Daten lassen leider keine Aussage über die Entwicklung von γ zu. Alle Parametersätze weisen zudem eine zu hohe Fehlerrate bei der Entschlüsselung auf (siehe dazu [34] bzw. [70]). Eine Abschätzung von γ wäre allerdings für eine Aussage über die Sicherheit von NTRU wichtig, denn einerseits ist nach O. Goldreich (siehe [24]) zu erwarten, daß eine Annäherung des kürzesten Gittervektors auf den Faktor $\sqrt{N/\ln N}$ nicht \mathcal{NP} -schwer ist, andererseits ist die Annäherung bis auf einen Faktor kleiner als $\sqrt{2}$ nach [51] sehr wohl \mathcal{NP} -schwer. Eine Übersicht über die Leistungsfähigkeit der aktuellen Reduktionsalgorithmen gibt die folgende Tabelle (vergleiche [43]):

Reduktionsalgorithmus	Laufzeit	Speicherplatz	Approximationsfaktor SVP
LLL [41]	$n^5 A$	n^2	$2^{(n-1)/2}$
Segment-LLL [38]	$n^3 \log(n) A$	n^2	$(\frac{4}{3} + \epsilon)^{(n-1)/2}$
2k-Reduktion [59]	$n^3 k^{k+\mathcal{O}(k)} + n^4 A$	n^2	$(\frac{k}{3})^{n/k}$
Primal-Dual (Koy)	$n^3 k^{k/2+\mathcal{O}(k)} + n^4 A$	n^2	$(\frac{k}{6})^{n/k}$
RSR [63]	$n^3 (\frac{k}{6})^{k/4} A + n^4 A$	n^2	$(\frac{k}{6})^{n/2k}$
QSR [43]	$n^3 (\frac{k}{6})^{k/8} A + n^4 A$	n^2	$(\frac{k}{6})^{n/2k}$

A = Bit-Operationen bei Berechnungen mit ganzen Zahlen der Eingabelänge $\mathcal{O}(n^2)$.

Die Autoren von [31] orientieren sich an den Größen

$$a := N/q \text{ und } c := \sqrt{\frac{4\pi e \cdot \|f\| \cdot \|g\|}{q}}.$$

(Es gilt $c_m^{-1} = c^{-1}\sqrt{aq}$.) Sie stellen fest, daß extrem kleine a und c auf ein leichtes SVP schließen lassen. Selbst der LLL-Algorithmus hat dann Chancen, den Zielvektor zu finden.

7 Rucksackprobleme und Gitter

Von den verschiedenen Kryptosystemen (z.B. [11]), welche auf dem Rucksackproblem basieren, gilt nur noch das von Okamoto [58] als nicht gebrochen.

Definition 7.0.1 Das *Rucksackproblem* lautet:

- Gegeben $n \in \mathbb{N}$, Gewichte $a_1, \dots, a_n \in \mathbb{N}$ und $s \in \mathbb{N}$.
- Finde $x \in \{0, 1\}^n$ mit $\sum_{i=1}^n x_i a_i = s$ oder zeige, daß kein solcher Vektor existiert.

Das Rucksackproblem nennt man auch Subsetsum- oder Knapsack-Problem. Es ist \mathcal{NP} -vollständig [20]. Es ist klar, daß man das Rucksackproblem für die Dimension n auf die Dimension $n - 1$ reduzieren kann, falls $\max_i a_i > s$ bzw. $\max_i a_i > \sum_i a_i - s$, da man dann das maximale Gewicht entfernen kann. Kann man ein Rucksackproblem nicht in diesem Sinne weiter reduzieren, so heißt es *reduziert*. Es gilt dann:

$$\frac{1}{n}t \leq s \leq \frac{n-1}{n}t,$$

wobei $t := \sum_{i=1}^n a_i$. Mit dem *inversen Rucksackproblem* bezeichnen wir das Rucksackproblem $n, a_1, \dots, a_n, t - s$.

Definition 7.0.2 Zu einem Rucksackproblem mit Gewichten a_1, \dots, a_n definieren wir die *Dichte* als

$$d := \frac{n}{\log_2(\max_i a_i)}.$$

Für Rucksackprobleme, deren Dichte viel grösser als 1 ist, existieren „in der Regel“ viele Lösungen. Als am schwierigsten zu lösen gelten zufällige Rucksackprobleme mit $d \approx 1$. Aus der gegebenen Dichte erhalten wir eine Schranke für die Gewichte:

$$\max_i a_i \geq 2^{\frac{n}{d}}.$$

7.1 Lösungen bei geringer Dichte

Wir wollen hier den Ansatz von Lagarias-Odlyzko [39] vorstellen und die Erweiterung von Coster, Joux, LaMaccia, Odlyzko, Schnorr und Stern [15] ansprechen.

Für ein reduziertes Rucksackproblem mit n, a_1, \dots, a_n und s gegeben, definieren wir

$$B := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ b_{n+1} \end{bmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 & Na_1 \\ 0 & 1 & & 0 & Na_2 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & Na_n \\ 0 & 0 & \cdots & 0 & Ns \end{pmatrix}$$

Wenn wir das Gitter $L(B)$ betrachten, dann ist klar: Wenn der Vektor $e := (e_1, \dots, e_n)$ eine Lösung des gegebenen Problems ist, also

$$\sum_i e_i a_i = s$$

gilt, daß dann der Vektor $(e_1, \dots, e_n, 0)$ in dem Gitter enthalten ist. Wir können weiterhin oBdA davon ausgehen, daß $\sum_i e_i \leq \frac{1}{2}n$, sonst betrachten wir das inverse Rucksackproblem.

Sei $N > \sqrt{n/2}$. Wir wollen die Wahrscheinlichkeit abschätzen, mit der e der kürzeste Vektor in dem Gitter $L(B)$ ist. Für alle Vektoren $(x_1, \dots, x_{n+1}) := x \in L(B)$, welche als Lösungen des SVP in Frage kommen, gilt:

- (i) $\|x\| \leq \|e\| \leq \sqrt{\frac{1}{2}n}$
- (ii) $y := \frac{1}{s} \sum_{i=1}^n x_i a_i \in \mathbb{Z}$
- (iii) $|y| \leq n\sqrt{\frac{1}{2}n}$.

Da x_{n+1} ein ganzzahliges Vielfaches von N ist, folgt aus (i) und $N > \sqrt{n/2}$, daß $x_{n+1} = 0$ und damit auch (ii) gilt. Die Behauptung (iii) ergibt sich aus der Reduziertheit des Problems und (i):

$$|y| = \frac{1}{s} \sum_{i=1}^n x_i a_i \leq \frac{\|x\|}{s} \sum_{i=1}^n a_i = \frac{\|x\|}{s} t \leq n\sqrt{\frac{1}{2}n}.$$

Nach [39] kann man die Wahrscheinlichkeit der Existenz eines solchen x folgendermaßen abschätzen:

$$\begin{aligned} & \mathcal{P}(\exists_{x \in L(B), y \in \mathbb{Z}} \text{ mit } x \notin \{0, \pm e\}, \text{ (i), (ii) und (iii)}) \\ & \leq \mathcal{P}(\sum_{i=1}^n x_i a_i = ys : x \notin \{0, \pm e\}, \text{ (i) und (iii)}) \cdot |\{x \in \mathbb{Z}^n : \text{(i)}\}| \cdot |\{y : \text{(iii)}\}| \\ & \leq \mathcal{P}(\sum_{i=1}^n x_i a_i = ys : x \notin \{0, \pm e\}, \text{ (i) und (iii)}) \cdot 2^{n \cdot 1.54725} \cdot \left(1 + 2n\sqrt{\frac{1}{2}n}\right) \\ & \leq n \cdot A^{-1} \cdot 2^{n \cdot 1.54725} \cdot \left(1 + 2n\sqrt{\frac{1}{2}n}\right), \end{aligned}$$

wobei $A \geq \max_i a_i \geq 2^{\frac{n}{d}}$. Deswegen gilt mit $\frac{1}{d} > 1.54725$, also $d < 0.6463$ und Gewichten $a_i < A$, daß für ein festes A

$$\lim_{n \rightarrow \infty} \mathcal{P}(\exists_{x \in L(B)} \text{ mit } x \notin \{0, \pm e\}, \text{ (i), (ii) und (iii)}) = 0.$$

Die Lösung des Rucksackproblems kann folglich durch zweifaches Befragen eines SVP-Orakels gefunden werden (Man muß ein Gitter für das Problem und das inverse Problem konstruieren), wenn die Dichte gering genug ist. Nach [15] kann man Rucksackprobleme mit Dichte $d < 0.9408$ lösen, wenn man b_{n+1} in der oben angegebenen Basismatrix durch

$$b'_{n+1} = \left(\frac{1}{2}, \dots, \frac{1}{2}, Ns\right)$$

ersetzt. Der Beweis ist nahezu analog. Durch Fallunterscheidung kann erreicht werden, daß $\|t - 2s\| \geq \frac{1}{2} \max_i a_i$. Es ergeben sich die folgenden Bedingungen für ein x , welches das SVP in $L(B)$ löst:

$$\begin{aligned} \text{(i)'} \quad & \|x\| \leq \|e\| \leq \frac{1}{2}\sqrt{n} \\ \text{(ii)'} \quad & y := \frac{2}{t-2s} \sum_{i=1}^n x_i a_i \in \mathbb{Z} \\ \text{(iii)'} \quad & |y| \leq 2n\sqrt{n}. \end{aligned}$$

Man erhält folgende veränderte Abschätzungen:

$$\left| \left\{ x \in \frac{1}{2}\mathbb{Z}^n : \text{(i)'} \right\} \right| \leq 2^{n \cdot 1.0628} \quad \text{und} \quad |\{y : \text{(iii)'}\}| \leq (1 + 4n\sqrt{n}) ,$$

woraus sich die angegebene Grenze ergibt.

7.2 Das Kryptosystem von Okamoto

Okamoto konstruiert ein Kryptosystem in dem Szenario, daß er einen Quantencomputer (QTM - Quantum Turing Machine) zur Verfügung hat, welchen er als Orakel zum Bestimmen von diskreten Logarithmen benutzt. Falls man eine QTM zur Verfügung hat, so kann ein solches Orakel mittels des Algorithmus von Shor [67] konstruiert werden. Die Sicherheit des Kryptosystems basiert auf der Annahme, daß man eine QTM nicht zum Lösen von \mathcal{NP} -schweren Problemen nutzen kann.

Bevor wir das Kryptosystem beschreiben, wollen wir noch auf ein paar Sätze und Begriffe eingehen, welche für das Verständnis von zentraler Bedeutung sind. Einen guten Überblick über die Theorie der algebraischen Zahlkörper findet man z.B. in [28] oder [52]. Ein *algebraischer Zahlkörper* sei im folgenden eine endliche separable Körpererweiterung über \mathbb{Q} . Für einen algebraischen Zahlkörper K sei \mathcal{O}_K der Ring der ganzen Zahlen in K , das heißt der Elemente aus K die Nullstellen eines normierten Polynoms $f(X) \in \mathbb{Z}[X]$ sind.

Definition 7.2.1 Sei l der Grad eines algebraischen Zahlkörpers K und N die kleinste K enthaltende normale Erweiterung von \mathbb{Q} . Sei \mathfrak{S} die Galoisgruppe von $N | \mathbb{Q}$ und \mathfrak{K} ihre zu K gehörige Untergruppe. Dann induzieren die Restklassen $\mathfrak{K}S$ für $S \in \mathfrak{S}$ genau l verschiedene Isomorphismen σ^i , $i = 0, \dots, l-1$ von K auf Teilkörper von N , bei denen $\sigma^i(q) = q$ für alle $q \in \mathbb{Q}$ gilt. Für $k \in K$ sei die *Norm* von k definiert als

$$\mathcal{N}(k) := \prod_{i=0}^{l-1} \sigma^i(k).$$

Für ein Primideal \wp von \mathcal{O}_K ist $\mathcal{O}_K/\wp = \mathbb{F}_{p^f}$ ein endlicher Körper und die *Absolutnorm* von \wp ergibt sich als $\mathcal{N}(\wp) = p^f$.

Satz 7.2.2 Sei K ein algebraischer Zahlkörper und \wp ein Primideal von \mathcal{O}_K . Es existiert eine ganze Basis (engl.: integral-basis) w_1, \dots, w_l , so daß jede Restklasse von \mathcal{O}_K/\wp eindeutig durch

$$v_1 w_1 + \dots + v_l w_l$$

dargestellt werden kann, wobei l der Grad von K ist, $0 \leq v_i < e_i$ für $i = 1, \dots, l$ und $[e_1 w_1, \dots, e_l w_l]$ eine ganze Basis von \wp ist.

Beweis. siehe [28] ■

Satz 7.2.3 (Kleiner Satz von Fermat) Sei \wp ein Primideal von \mathcal{O}_K und $g \neq 0$ ein Element aus \mathcal{O}_K/\wp , dann gilt

$$g^{\mathcal{N}(\wp)-1} \equiv 1 \pmod{\wp}$$

Beweis. siehe [28] ■

Zur Schlüsselerzeugung wählt man eine Menge algebraischer Zahlkörper \mathcal{K} und aus dieser ein Element K . Desweiteren wählt man ein Primideal \wp aus dem Ring der ganzen Zahlen \mathcal{O}_K von K , sowie $n, k \in \mathbb{Z}$ und ein erzeugendes Element g der multiplikativen Gruppe in dem Endlichen Körper \mathcal{O}_K/\wp . Man beachte, daß ein Element $x \in \mathcal{O}_K/\wp$ nach Satz 7.2.2 mit der Basis w_1, w_2, \dots, w_l eindeutig durch ein Tupel (v_1, v_2, \dots, v_l) mit $0 \leq v_i < e_i$ beschrieben werden kann. Aus \mathcal{O}_K/\wp wählt man nun n ganze Zahlen p_1, \dots, p_n , so daß $\mathcal{N}(p_1), \dots, \mathcal{N}(p_n)$ paarweise teilerfremd sind und für alle $m \in \{0, 1\}^n$ mit $\sum_{i=1}^n m_i = k$ gilt, daß $\prod_{i=1}^n p_i^{m_i} = v_1 w_1 + v_2 w_2 + \dots + v_l w_l$ mit $0 \leq v_i < e_i$. Mit dem Algorithmus von Shor können nun die diskreten Logarithmen a_i der p_i zur Basis g berechnet werden, so daß

$$g^{a_i} \equiv p_i \pmod{\wp}.$$

Danach wählt man zufällig $d \in \mathbb{Z}/(\mathcal{N}(\wp) - 1)\mathbb{Z}$ und berechnet

$$b_i = (a_i + d) \pmod{(\mathcal{N}(\wp) - 1)}.$$

Der öffentliche Schlüssel ist nun

$$(\mathcal{K}, n, k, b_1, \dots, b_n),$$

der private

$$(K, g, d, \wp, p_1, \dots, p_n).$$

Die Verschlüsselung einer Nachricht M mit Länge $\lfloor \binom{n}{k} \rfloor$ erfolgt über die Repräsentation durch ein $m \in \{0, 1\}^n$ mit $\sum_{i=1}^n m_i = k$, welches auf die folgende Weise mit $l_i := k - \sum_{h=1}^{i-1} m_h$ gebildet wird:

$$m_i = \begin{cases} 1 & \text{falls } \binom{n-i}{l_i} \leq M - \sum_{j=1}^{i-1} m_j \binom{n-j}{l_j} \\ 0 & \text{sonst.} \end{cases}$$

Man beachte, daß $\binom{l}{0} = 1$ falls $l \geq 0$ und $\binom{0}{l} = 0$ falls $l > 0$. Die m_i sind die „Bits“ des Klartextes m , und der Schlüsseltext c berechnet sich dann als

$$c = \sum_{i=1}^n m_i b_i.$$

Zur Entschlüsselung berechnet man $r = (c - kd) \pmod{(\mathcal{N}(\varphi) - 1)}$ und

$$u = g^r \pmod{\varphi} .$$

Aus u konstruiert man $m \in \{0, 1\}^n$ durch

$$m_i = \begin{cases} 1 & \text{falls } p_i | u \\ 0 & \text{sonst.} \end{cases}$$

Aus diesem m erhält man die Nachricht

$$M = \sum_{i=1}^n m_i \binom{n-i}{l_i}$$

Beispiel 7.2.4 Falls K der Ring der ganzen Zahlen ist, so wählt man eine Primzahl p und wählt n teilerfremde $p_i \in \mathbb{Z}/p\mathbb{Z}$ so, daß das Produkt einer Auswahl von k dieser p_i kleiner als p ist. Dann wählt man ein erzeugendes Element g von $(\mathbb{Z}/p\mathbb{Z})^\times$ und ein $d \in \mathbb{N}$. Zur Generierung des öffentlichen Schlüssels berechnet man die diskreten Logarithmen a_i der p_i zur Basis g und bildet $b_i = a_i + d$. und veröffentlicht $(\mathbb{Z}, n, k, b_1, \dots, b_n)$. Der private Schlüssel ist folglich $(\mathbb{Z}, g, d, p, p_1, \dots, p_n)$.

7.3 Angriffe auf Okamoto's Kryptosystem

Das Problem eine Nachricht zu entschlüsseln kann als das Rucksackproblem mit c und den Gewichten b_1, \dots, b_n gesehen werden. Nach [58] divergiert die Dichte d , falls $k \approx 2^{(\log n)^\alpha}$ für ein $\alpha < 1$. Damit ist der Ansatz zur Lösung des Rucksackproblems, den wir oben präsentiert haben, nicht anwendbar. Eine Möglichkeit, um diesen trotzdem anwenden zu können, wäre das Raten von Werten einzelner Bits des Klartextes. Dies führt zu einer Verringerung der Dichte um maximal d/n pro geratenen m_i . Um die Dichte auf $d' < d$ zu reduzieren muß man also mindestens $n(1 - d'/d)$ Bits richtig raten, was sehr schwer zu sein scheint. Ein weiterer Ansatz wäre das Ausnutzen der Information, daß genau k Bits des Klartextes den Wert 1 haben. Man könnte zum Beispiel die folgende Gitterbasis betrachten:

$$B := \begin{pmatrix} 1 & 0 & \dots & 0 & Nb_1 & N \\ 0 & 1 & & & Nb_2 & N \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & Nb_n & N \\ \frac{1}{2} & \frac{1}{2} & \dots & \frac{1}{2} & Nc & kN \end{pmatrix}$$

Dann wäre klar, daß der Vektor $e = (\frac{1}{2} - m_1, \dots, \frac{1}{2} - m_n, 0, 0)$ in dem Gitter $L(B)$ enthalten ist. Man kann nun zeigen, daß mit $N \geq \frac{1}{2}\sqrt{n}$ und $t := \sum_{i=1}^n b_i$ die Bedingungen

- (i) $\|x\| \leq \|e\| \leq \frac{1}{2}\sqrt{n}$
- (ii) $y := \frac{2}{t-2c} \sum_{i=1}^n x_i b_i \in \mathbb{Z}$
- (iii) $\sum_{i=1}^n x_i = \frac{y(n-2k)}{2}$
- (iv) $|y| \leq \frac{1}{2}n\sqrt{n}$ falls $k \neq \frac{n}{2}$

für jeden Vektor x erfüllt sein müssen, der das SVP in $L(B)$ löst. Wir gehen im folgenden davon aus, daß $k \neq \frac{n}{2}$ und die Ereignisse (ii) und (iii) für zufällige $x \in \frac{1}{2}\mathbb{Z}^n$ stochastisch unabhängig sind. Wir brauchen noch das folgende Lemma:

Lemma 7.3.1 Für alle $n \in \mathbb{N}$ gilt:

$$\left| \left\{ x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|y\| \leq \frac{\sqrt{n}}{2} \right\} \right| \leq (1+2n) \cdot \left| \left\{ x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|y\| \leq \frac{\sqrt{n-1}}{2} \right\} \right|$$

Beweis. Wir definieren:

$$M_1 := \left\{ x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|y\| \leq \frac{\sqrt{n}}{2} \right\} \quad \text{und} \quad M_2 := \left\{ x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|y\| \leq \frac{\sqrt{n-1}}{2} \right\} .$$

Es genügt zu zeigen, daß $M_1 \subseteq M_2 + \{y \in \frac{1}{2}\mathbb{Z}^{n-1} : \|y\| \leq \frac{1}{2}\}$. Dazu wählen wir ein $z = (z_1, \dots, z_{n-1}) \in M_1 \setminus M_2$, falls ein solches existiert. Wir können oBdA davon ausgehen, daß ein $i \in \{1, \dots, n-1\}$ existiert, so daß $z_i > 0$ (Falls nicht, so gelten die Überlegungen analog für $-z$). Sei \vec{e}_i der i -te Einheitsvektor im \mathbb{Z}^{n-1} (der Vektor mit 1 in der i -ten Komponente, 0 sonst), dann gilt:

$$\left\| z - \frac{1}{2}\vec{e}_i \right\|^2 \leq \|z\|^2 - \frac{1}{4} \leq \frac{1}{4}(n-1) .$$

Folglich ist $(z - \frac{1}{2}\vec{e}_i) \in M_2$ und die Behauptung folgt. ■

Wir wollen nun die Wahrscheinlichkeit betrachten, daß die Lösung des SVP in $L(B)$ keine Lösung für das Rucksackproblem impliziert. Sie läßt sich mit

$$\begin{aligned} & \mathcal{P}(\exists_{x \in L(B), y \in \mathbb{Z}} \text{ mit } x \notin \{0, \pm e\}, \text{ (i), (ii), (iii) und (iv) }) \\ & \leq \mathcal{P}(\text{(ii)} \cap \text{(iii)} : x \notin \{0, \pm e\}, \text{ (i) und (iv)}) \\ & \quad \cdot |\{x \in \frac{1}{2}\mathbb{Z}^n : \text{(i)}\}| \cdot |\{y \in \frac{1}{2}\mathbb{Z} : \text{(iv)}\}| \\ & \leq \mathcal{P}(\text{(iii)} : x \notin \{0, \pm e\}, \text{ (i) und (iv)}) \\ & \quad \cdot \mathcal{P}(\text{(ii)} : x \notin \{0, \pm e\}, \text{ (i) und (iv)}) \\ & \quad \cdot |\{x \in \frac{1}{2}\mathbb{Z}^n : \text{(i)}\}| \cdot |\{y \in \frac{1}{2}\mathbb{Z} : \text{(iv)}\}| \\ & = \frac{|\{x \in \frac{1}{2}\mathbb{Z}^n : \langle (1, \dots, 1), x \rangle = y^{\frac{n-2k}{2}} \text{ und } \|x\| \leq \frac{1}{2}\sqrt{n}\}|}{|\{x \in \frac{1}{2}\mathbb{Z}^n : \|x\| \leq \frac{1}{2}\sqrt{n}\}|} \\ & \quad \cdot n \cdot A^{-1} \cdot |\{x \in \frac{1}{2}\mathbb{Z}^n : \text{(i)}\}| \cdot (1 + 2n\sqrt{n}) \\ & \leq |\{x \in \frac{1}{2}\mathbb{Z}^n : \langle (1, \dots, 1), x \rangle = 0 \text{ und } \|x\| \leq \frac{1}{2}\sqrt{n}\}| \cdot n \cdot A^{-1} \cdot (1 + 2n\sqrt{n}) \\ & \leq \frac{1}{\sqrt{n}} \cdot |\{x \in \frac{1}{2}\mathbb{Z}^n : x_n = 0 \text{ und } \|x\| \leq \frac{1}{2}\sqrt{n}\}| \cdot n \cdot A^{-1} \cdot (1 + 2n\sqrt{n}) \\ & \leq \frac{1}{\sqrt{n}} \cdot |\{x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|x\| \leq \frac{1}{2}\sqrt{n}\}| \cdot n \cdot A^{-1} \cdot (1 + 2n\sqrt{n}) \\ & \leq \frac{1}{\sqrt{n}} \cdot (2n+1) \cdot |\{x \in \frac{1}{2}\mathbb{Z}^{n-1} : \|x\| \leq \frac{1}{2}\sqrt{n-1}\}| \cdot n \cdot A^{-1} \cdot (1 + 2n\sqrt{n}) \\ & \leq A^{-1} \cdot 2^{(n-1) \cdot 1.0628} \cdot \sqrt{n} \cdot (2n+1) \cdot (1 + 2n\sqrt{n}), \end{aligned}$$

abschätzen, wobei $A \geq \max_i b_i \geq 2^{\frac{n}{d}}$. Deshalb können wir auch mit diesem Ansatz keine Rucksackprobleme mit einer höheren Dichte als $d = 0.9408$ lösen. Die Rucksackprobleme, die durch ein Okamoto-Kryptosystem konstruiert werden, weisen nur eindeutige Lösungen (mit k Gewichten) auf, nicht mehrere, wie es von Rucksackproblemen mit hoher Dichte zu erwarten wäre. Multiplikative Vielfache von Gewichten treten ebenfalls nicht auf, so daß man die Struktur dieser speziellen Rucksackprobleme vielleicht doch für einen Angriff ausnutzen kann.

Ein Angriff auf den privaten Schlüssel scheint undenkbar, solange \mathcal{K} genügend groß ist, denn ein Angreifer müßte zunächst den Körper K bestimmen. Selbst wenn in Angreifer K bestimmen könnte, wäre das erzeugende Element g bzw. φ nur schwer zu bestimmen. Gleiches gilt für den Parameter d oder die ganzen Zahlen p_i . Nehmen wir an, man hätte ein Teilmenge der p_i erraten, so darf diese nicht zu klein sein, um den Angriff auf multiplikative Rucksackprobleme von A.M. Odlyzko [56] anwenden zu können. Man würde versuchen eine Lösung von $1 = \sum_i x_i a_i$ zu finden, so daß für alle $x_i \neq 0$ das zugehörige p_i bekannt ist, denn dann könnte man $g = \prod_{i=1}^n p_i^{x_i} \pmod{\varphi}$ berechnen. Das Zuordnen der geratenen p_i zu den b_i bzw. a_i wird schwierig sein, da letztere die Diskreten Logarithmen der p_i zur Basis g bilden, und somit zufällig erscheinen. Es scheint also unmöglich zu sein, den privaten Schlüssel anzugreifen, ohne K, g, d und φ zu bestimmen, was als sehr schwierig angenommen werden kann.

Literaturverzeichnis

- [1] L.M. Adleman „On breaking generalized knapsack public key cryptosystems“, Proc. of 15th STOC, S. 402-412, ACM, 1983.
- [2] M. Ajtai und C. Dwork, „A public-key cryptosystem with worst-case/average-case equivalence“, Proc. of 29th STOC, S. 284-293, ACM 1997, Erhältlich unter [17].
- [3] L. Babai, „On Lovász lattice reduction and the nearest lattice point problem“, *Combinatorica* 6, S. 1-13, 1986.
- [4] C.H. Bennett, E. Bernstein, G. Brassard und U. Vazirani „Strengths and Weaknesses of Quantum Computing“, preprint 1994
- [5] J. Blömer, A. May, „Low Secret Exponent RSA Revisited“, Proc. of CaLC 2001, LNCS, Springer-Verlag 2001.
- [6] J. Blömer, Alexander May, „New Partial Key Exposure Attacks on RSA“, to appear in *Advances in Cryptology (Crypto 2003)*, vol. 2729 of LNCS, Springer Verlag, 2003.
- [7] D. Boneh und G. Durfee, „Cryptanalysis of RSA with private key d less than $N^{0.292}$ “, Proc. of Eurocrypt '99, volume 1592 of LNCS, S. 1-11, IACR, Springer-Verlag 1999.
- [8] D. Boneh, G. Durfee und Y. Frankel, „An attack on RSA given a small fraction of the private key bits“ Proc. of Asiacrypt '98, volume 1514 of LNCS, S. 25-34, Springer-Verlag 1998.
- [9] E.T. Browne „Introduction to the Theory of Determinants and Matrices“, The University of North Carolina Press, Chapel Hill 1958
- [10] J. Buchmann, „Einführung in die Kryptographie“, Springer-Verlag Heidelberg 1999.
- [11] B. Chor und R. Rivest, „A Knapsack Type Public Key Cryptosystem Based on Arithmetic in Finite Fields“, *IEEE Trans. Information Theory*, Vol. 34, No. 5, S. 901-909, September 1988.
- [12] D. Coppersmith „Finding small solutions to small degree polynomials“, Proc. of CALC '01, LNCS, Springer-Verlag 2001.
- [13] D. Coppersmith und A. Shamir, „Lattice attacks on NTRU“, Proc. of Eurocrypt '97, LNCS, Springer-Verlag 1997
- [14] D. Coppersmith, „Small solutions to polynomial equations, and low exponent RSA vulnerabilities“, *J. of Cryptology*, 10(4), S. 233-260, 1997. Revised version of two articles from Eurocrypt '96.

- [15] M.J. Coster, B.A. LaMacchia, A.M. Odlyzko und C.P. Schnorr, „An Improved Low-Density Subset Sum Algorithm“, volume 547 of LNCS, S. 54ff, Springer-Verlag 1991.
- [16] C. Coupé, P.Q. Nguyen und J. Stern, „The effectiveness of lattice attacks against low-exponent RSA“, Proc. of PKC '98, volume 1431 of LNCS, Springer-Verlag 1999.
- [17] ECCC „<http://www.eccc.uni/trier.de/eccc>“, The Electronic Colloquium on Computational Complexity.
- [18] EESS: Consortium for Efficient Embedded Security. „Efficient Embedded Security Standards # 1: Implementation Aspects of NTRUEncrypt and NTRUSign“, Version 2.0, available at <http://www.ceesstandards.org>, May 2003.
- [19] M.L. Furst und R. Kannan „Succinct certificates for almost all subset sum problems“, SIAM Journal on Computing, S. 550-558, 1989.
- [20] M.R. Garey und D.S. Johnson „Computer and Intractability: A Guide to the Theory of \mathcal{NP} -completeness“, W.H. Freeman and Company, San Francisco 1978.
- [21] J. von zur Gathen und M. Sieveking, „A Bound on Solution of Linear Integer Equations and Inequations“, Proc. of the American Mathematical Society, Band 72, S. 155-158, 1978.
- [22] C. Gentry „Key Recovery and Message Attacks on NTRU-Composite“, In Eurocrypt '01, vol. 2045 of LNCS, S. 182-194, Springer-Verlag 2001.
- [23] C. Gentry und M. Szydło. „Cryptanalysis of the Revised NTRU Signature Scheme“, Proc. of Eurocrypt '02, vol. 2332 of LNCS, S. 299-320, Springer-Verlag 2002.
- [24] O. Goldreich und S. Goldwasser, „On the limits of non-approximability of lattice problems“, Proc. of 30th STOC, ACM 1998, erhältlich bei [17] unter TR97-031.
- [25] O. Goldreich, S. Goldwasser und S. Halevi, „Public-Key Cryptosystems from Lattice Reduction Problems“, Proc. of Crypto '97, volume 1294 of LNCS, S. 112-131, IACR, Springer-Verlag 1997, erhältlich bei [17].
- [26] O. Goldreich, D. Micciancio, S. Safra and J.-P. Seifert „Approximating shortest lattice vectors is not harder than approximation closest lattice vectors“, Report No. 2, The Electronic Colloquium on Computational Complexity 1999, erhältlich bei [17].
- [27] M. Grötschel, L. Lovász und A. Schrijver, „Geometric Algorithms and Combinatorial Optimization“, Springer Verlag Berlin Heidelberg 1988.

- [28] H. Hasse „Zahlentheorie“, Akademie-Verlag, Berlin 1949
- [29] J. Hoffstein, J. Pipher und J.H. Silverman, „NTRU: a ring based public key cryptosystem“, Proc. of ANTS III, volume 1423 of LNCS, S. 267-288, Springer-Verlag 1998.
- [30] J. Hoffstein and J. H. Silverman, „Optimizations for NTRU“, Proc. of Public Key Cryptography and Computational Number Theory, de Gruyter, Warschau, September 2000.
- [31] J. Hoffstein, J.H. Silverman und W. Whyte, „Estimated Breaking Times for NTRU Lattices“, NTRU Cryptosystems Technical Report 012, Version 2, 2003. Erhältlich bei www.ntru.com.
- [32] N. Howgrave-Graham, „Finding small roots of univariate modular equations revisited“, Proc. of Cryptography and Coding, volume 1355 of LNCS, Springer 1997.
- [33] N. Howgrave-Graham, J. H. Silverman, A. Singer, W. Whyte „NAEP: Provable Security in the Presence of Decryption Failures“, IEEE P1363.1
- [34] N. Howgrave-Graham, P.Q. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer und W. Whyte, „The Impact of Decryption Failures on the Security of NTRU Encryption“, To appear in Proc. of CRYPTO '03, vol. 2729 of LNCS, S. 226-246, Springer-Verlag 2003
- [35] E.Jaulmes und A.Joux, „A Chosen Ciphertext Attack on NTRU“, In Crypto '00, vol. 1880, S. 20-35. Springer-Verlag 2000.
- [36] P.Klein „Finding the closest lattice vector, when it's unusually close“, Proc. of 11th Symposium on Discrete Algorithms, SF, California, Januar 2000 SIAM.
- [37] H.J. Kowalsky, „Lineare Algebra“, 7. Auflage, Walter de Gruyter & Co., Berlin 1975.
- [38] H. Koy, C.P. Schnorr „Segment and Strong Segment LLL-Reduction of Lattice Bases“, 2002
- [39] J.C. Lagarias und A.M. Odlyzko „Solving low-density subset sum problems“, J. Assoc. Comp. Mach, 32(1), S. 229-246, Januar 1985.
- [40] A.K. Lenstra und H.W. Lenstra Jr., „The Development of the Number Field Sieve“, volume 1554 of Lecture Notes in Mathematics, Springer-Verlag 1993.
- [41] A.K. Lenstra, H.W. Lenstra Jr. und L. Lóvasz, „Factoring Polynomials with Rational Coefficients“, Mathematische Annalen, Band 261, S. 515-534, Springer 1982.

- [42] D. Lieman (Editor) „Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices“, IEEE P1363.1/D2 (Draft Version 2), 2001.
- [43] C. Ludwig „A Faster Lattice Reduction Method Using Quantum Search“, Technical Report No. TI-3/03, TU-Darmstadt 2003.
- [44] C. Ludwig, „The Security and Efficiency of Micciancio’s Cryptosystem“, Technical Report No. TI-7/02, TU-Darmstadt 2002.
- [45] E.E. Mahassni, P.Q. Nguyen, I.E. Shparlinski „The Insecurity of Nyberg-Rueppel and Other DSA-Like Signature Schemes with Partially Known Nonces“CaLC 2001, volume 2146 of LNCS, S. 97-109, Springer-Verlag 2001.
- [46] A. May, J.H. Silverman, „Dimension reduction methods for convolution modular lattices“, Proc. of CaLC 2001, volume 2146 of LNCS, S. 111-127, Springer-Verlag 2001.
- [47] A. May, „Cryptanalysis of NTRU“, preprint February 1999.
- [48] A.J. Menezes, P.C. van Oorschot und S.A. Vanstone „Handbook of Applied Cryptography“, CRC-Press, Boca Raton, Florida 1997.
- [49] T.Meskanen und A.Renvall, „A Wrap Error Attack against NTRUEncrypt“, TUCS Technical Report No. 507, Jan 2003. (To appear in Proc. of WCC’03)
- [50] D. Micciancio, „Improving lattice-based cryptosystems using the Hermite normal form“, Prov. of CALC ’01, LNCS, Springer-Verlag 2001.
- [51] D. Micciancio, „The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant (preliminary version)“, Report No 16, ECCC 1998. Erhältlich unter TR98-16 bei [17].
- [52] J. Neukirch „Algebraische Zahlentheorie“, Springer-Verlag, Berlin 1992
- [53] P.Q. Nguyen, „Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto ’97“, Proc. of Crypto ’98, volume 1666 of LNCS, Springer-Verlag 1998.
- [54] P.Q. Nguyen, „The Two Faces of Lattices in Cryptology“, Proc. of Cryptography and Lattices Conference, volume 2146 of LNCS, Springer 2001
- [55] P.Q. Nguyen und D.Pointcheval, „Analysis and Improvements of NTRU Encryption Paddings“, In Crypto ’02, vol. 2442 of LNCS, S. 210-225, Springer-Verlag 2002.
- [56] A.M. Odlyzko „Cryptoanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir’s fast signature scheme.“, IEEE Trans. on Information Theory IT-30, S. 594-601, 1984

- [57] A.M. Odlyzko „The rise and fall of knapsack cryptosystems“, Cryptology and Computational Number Theory, volume 42 of Proc. of Symposia in Applied Mathematics, S. 75-88, A.M.S. 1990.
- [58] T.Okamoto, K. Tanaka und S.Uchiyama, „Quantum Public Key Cryptosystems“, Proc. Of CRYPTO 2000, vol. 1880 of LNCS, S. 147-165, Springer-Verlag 2000.
- [59] C.P. Schnorr, „A hierarchy of polynomial lattice basis reduction algorithms“, Theoretical Computer Science 53, S. 201-224, 1987.
- [60] C.P. Schnorr, „Block KorkinZolotarev Bases and Successive Minima“ International Computer Science Institute, Berkeley, TR92-063, 1992.
- [61] C.P. Schnorr, „Factoring integers and computing discrete logarithms via diophantine approximation“, In Proc. of Eurocrypt '91, volume 547 of LNCS, S. 171-181, IACR, Springer-Verlag 1991.
- [62] C.P. Schnorr, Vorlesungen „Gittertheorie und algorithmische Geometrie“ an der Johann Wolfgang Goethe-Universität Frankfurt/Main im Wintersemester 2001/2002. Mitschrift erhältlich unter „http://www.mi.informatik.uni-frankfurt.de/teaching/lecture_notes/schnorr.gitter.ps“
- [63] C.P. Schnorr, „Lattice reduction by random sampling and birthday methods“, Proc. of STACS 2003, volume 2607 of LNCS, S. 145-156, Springer-Verlag 2003.
- [64] C.P. Schnorr, M. Euchner, „Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum“, Math. Programming, 66, S. 181-199, 1994.
- [65] C.P. Schnorr und H.H. Hörner, „Attacking the Chor-Rivest cryptosystem by improved lattice reduction“, Proc. of Eurocrypt '95, volume 921 of LNCS, S. 1-12 IACR, Springer 1995.
- [66] A. Schrijver, „Theory of Linear and Integer Programming“, Wiley, Chichester, 1986.
- [67] P.W. Shor „Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer“, SIAM J. of Comput., 26(5), S. 1484-1509, 1997.
- [68] J.H. Silverman, „Dimension reduced lattices, zero-forced lattices, and the NTRU public key cryptosystem“, NTRU Technical Report 013, 1999. Erhältlich bei www.ntru.com.
- [69] J.H. Silverman und W. Whyte, „Estimating Decryption Failure Probabilities for NTRUEncrypt“, Technical Report, NTRU Cryptosystem, Mai 2003. Report # 018, version 1, erhältlich bei www.ntru.com

- [70] J.H. Silverman „Wraps, Gaps, and Lattice Constants“, Technical Report, NTRU Cryptosystem, März 2001. Report # 011, version 2, erhältlich bei www.ntru.com
- [71] B.L. van der Waerden „Algebra“, Springer-Verlag, Berlin 1967.
- [72] M.Wiener, „Cryptanalysis of short RSA secret exponents“, IEEE Transactions on Information Theory, vol. 36, Nr. 3, S. 553-558, 1990.

Anhang

A Notationen und Konventionen

\mathbb{N}	die Menge $\{1, 2, 3, \dots\}$
\mathbb{N}_0	die Menge $\mathbb{N} \cup \{0\}$.
$\lfloor a \rfloor$ (bzw. $\lceil a \rceil$)	Bezeichnet für $a \in \mathbb{R}$ die größte (kleinste) Zahl in \mathbb{Z} , die kleiner (größer) gleich a ist.
$\lceil a \rceil$	Bezeichnet für $a \in \mathbb{R}$ die Zahl aus \mathbb{Z} , welche den kleinsten Abstand zu a hat. Ist diese nicht eindeutig ($a - \lfloor a \rfloor = 0.5$), so wählen wir die Zahl $\lceil a \rceil$.
$\text{ggT}(a, b)$	für $a, b \in \mathbb{N}$ ist die größte Zahl aus \mathbb{N} , die a und b teilt.
$\varphi(n), n \in \mathbb{N}$	Gruppenordnung von $(\mathbb{Z}/n\mathbb{Z})^\times$, also die Anzahl der Elemente $a < n$, so daß $\text{ggT}(a, n) = 1$.
$a \bmod b \equiv c$	bezeichnet eine Kongruenz,
$a \pmod{b} = c$	$(\bmod b) : \mathbb{Z} \rightarrow \{0, 1, \dots, b-1\}$, $a \pmod{b} = c$ ist Funktion, welche a die kleinste Zahl $c \in \mathbb{N}_0$ zuordnet, die $a \bmod b \equiv c$ erfüllt.
Id_n	die Identität auf dem \mathbb{R}^n
$\langle x, y \rangle$	das Skalarprodukt zweier Vektoren x, y aus \mathbb{R}^n .
$R^{n \times m}$	Die Menge aller $n \times m$ -Matrizen über dem Ring R .
$(m_{i,j}) \in R^{n \times m}$	Eine Matrix mit Koeffizienten $m_{i,j}$ in der i -ten Zeile und j -ten Spalte
B_i .	Die i -te Zeile einer Matrix
$B_{\cdot j}$	Die j -te Spalte einer Matrix
B_I .	Die Matrix, die aus den Zeilen B_i mit $i \in I$ besteht.
$B_{\cdot J}$	Die Matrix, die aus den Spalten $B_{\cdot j}$ mit $j \in J$ besteht.
$[b_1, \dots, b_m]$	Die Matrix, die aus den Spaltenvektoren b_1, \dots, b_m besteht.
$\text{odef}(B)$	Der Orthogonalisierungsdefekt einer Matrix, siehe Definition 5.1.1.
$L(B)$	$= L(B_{\cdot 1}, \dots, B_{\cdot m})$ Das Gitter mit Basismatrix $B \in \mathbb{R}^{n \times m}$.
\hat{b}_i	Der von b_1, \dots, b_{i-1} linear unabhängige Anteil von b_i .
b_i^*	Der i -te Basisvektor der dualen Gitterbasis
$M(B)$	Bezeichnet die Grundmasche, falls B eine Gitterbasismatrix ist. Siehe Seite 5.
$\lambda_{i,p}(L)$	Die sukzessiven Minima eines Gitters in L^p -Norm.
γ_n	Die Hermite-Konstante für Gitter der Dimension n .
\odot	Die Multiplikation in dem Ring $\mathbb{Z}[X]/(X^N - 1)$.
$\text{Res}(f, g)$	Die Resultante der Polynome f und g .
$\mathcal{L}(d_1, d_2)$	siehe Seite 33.
$\bar{c}(X)$	„reversal“ von $c \in \mathbb{Z}[X]/(X^N - 1)$, siehe Definition 6.6.3.
\dot{c}	$= c \odot \bar{c}$
$\mathcal{P}(E)$	ist die Wahrscheinlichkeit, daß das Ereignis E eintritt.
$A + B$	bezeichnet für zwei Mengen A und B die Menge $\{a + b : a \in A, b \in B\}$.

B Grundlagen

B.1 Matrizen

Eine Definition der grundlegenden Begriffe aus der Linearen Algebra findet man u.a. in [37]. In der ganzzahligen Optimierung bzw. der Gittertheorie sind die Begriffe der *Hermiteschen Normalform* und der *Unimodularität* einer Matrix zentral, weswegen wir hier noch einmal darauf eingehen wollen.

Definition B.1.1 Eine Matrix A heißt unimodular, falls

$$A \in \mathbb{Z}^{n \times n} \text{ und } \det(A) = \pm 1 .$$

Eine Matrix A ist demnach genau dann unimodular, wenn $A, A^{-1} \in \mathbb{Z}^{n \times n}$.

Definition B.1.2 Eine Matrix $(m_{i,j}) = M \in \mathbb{R}^{n \times n}$ ist in Hermitescher Normal Form (HNF), wenn

1. M eine obere Dreiecksmatrix ist und
2. $\forall_{0 \leq i < j \leq n} 0 \leq m_{i,j} < m_{i,i}$

Es kann gezeigt werden, daß jede rationale Matrix A eine eindeutige Darstellung durch das Produkt einer unimodularen Matrix U und einer Matrix M in HNF hat: $A = U \cdot M$ (siehe [66])

Für Gitter wird die *Gram-Determinante* eine wichtige Kenngröße sein. Der folgende Satz ermöglicht eine Berechnung, ohne diese explizit zu bilden.

Satz B.1.3 Seien $A, B \in \mathbb{R}^{n \times m}$, $m \leq n$. Für die Gram-Determinante $\det(A^\top B)$ gilt:

$$\det(A^\top B) = \sum_{J \subseteq \{1, \dots, n\}, |J|=m} \det(A_{J \cdot}) \cdot \det(B_{J \cdot})$$

Beweis. Siehe [9] (Theorem 9.1). ■

Eine weitere Determinante, welche uns besonders interessiert, ist die *Resultante*. Wir definieren sie für zwei Polynome.

Definition B.1.4 Sei A ein Ring. Die Resultante (resultant) zweier Polynome $f = \sum_{i=0}^n f_i X^i, g = \sum_{j=0}^m g_j X^j$ aus dem Ring $A[X]$ ist die Determinante

$$\begin{array}{c} m \\ \left\{ \begin{array}{cccccc} f_0 & f_1 & \cdots & f_n & & \\ & f_0 & f_1 & \cdots & f_n & \\ & & \ddots & & \dots & \ddots \\ & & & f_0 & f_1 & \cdots & f_n \\ n \\ \left\{ \begin{array}{cccccc} g_0 & g_1 & \cdots & & g_m & & \\ & \ddots & & \dots & & \ddots & \\ & & g_0 & g_1 & \cdots & & g_m \end{array} \right. \\ \underbrace{\hspace{10em}}_{m+n} \end{array} \right. \end{array}$$

und wird mit $\text{Res}(f, g)$ bezeichnet.

Wir werden die Resultante zur Berechnung von Nullstellen bivariater Polynome verwenden. Sind f und g bivariate Polynome in $A[X][Y]$, so ist $\text{Res}_x(f, g)$ die Resultante der entsprechenden Polynome F und G aus dem Ring $(A[Y])[X]$ mit Koeffizienten aus $A[Y]$. Verschwindet deren Resultante nicht, so sind die y -Werte der gemeinsamen Nullstellen von f und g Nullstellen der Resultante, welche aus $A[Y]$ ist. Verschwindet die Resultante zweier Polynome, dann haben sie einen gemeinsamen Faktor. Siehe auch [71].

B.2 Komplexität, \mathcal{NP} -Vollständigkeit

Wir wollen die Schwierigkeit von mathematischen Problemen in Beziehung zur Eingabelänge des Problems setzen. Dazu betrachtet man die Zeit, die ein Algorithmus zum Lösen des Problems braucht, wobei ein Algorithmus eine Folge von elementaren Handlungsanweisungen ist. Elementare Handlungsanweisungen können z.B. als Bit-Operationen bzw. Addition und Multiplikation definiert werden. Diese Kapitel soll einen Überblick zu den komplexitätstheoretischen Begriffen dienen, welche im folgenden verwendet werden. Eine ausführlichere Beschreibung findet man u.a. in [20].

Definition B.2.1 Wir definieren die *Bitlänge* l für Zahlen bzw. Matrizen (ohne Vorzeichen) als:

- $l(0) := 1$
- $l(n) := \lfloor \log_2(n) \rfloor + 1$ für $n \in \mathbb{N}$
- $l\left(\frac{p}{q}\right) := l(p) + l(q)$ für teilerfremde $p, q \in \mathbb{N}$.
- $l((a_{i,j})) := \sum_{i,j} l(a_{i,j})$ für eine Matrix $(a_{i,j}) \in \mathbb{Q}^{n \times m}$.

Wir haben die Eingabelänge eines Problems definiert und wollen nun die Algorithmen nach ihrer Laufzeit charakterisieren. Die *Polynomialzeit*-Algorithmen sind für uns von besonderem Interesse.

Definition B.2.2 Ein Algorithmus ist in Polynomialzeit, falls die Schrittzahl (Turing-Maschine oder Anzahl Bitoperationen) polynomiell in der Länge der (gültigen) Eingaben beschränkt ist, d.h. es existiert ein Polynom P aus $\mathbb{R}[X]$, so daß

$$\text{Schrittzahl}(\text{Eingabe}) \leq P(l(\text{Eingabe}))$$

Ähnlich definiert man exponentielle bzw. subexponentielle Algorithmen. Allgemein spricht man von einer Laufzeit von $\mathcal{O}(f(l))$, falls l die Eingabelänge und $f: \mathbb{N} \rightarrow \mathbb{R}$ eine Funktion ist, so, daß $f(l)$ eine obere Schranke für die Schrittzahl ist. Eine Funktion ist in Polynomialzeit berechenbar, wenn ein Algorithmus existiert, welcher den Funktionswert für alle möglichen Eingabewerte aus dem Urbildbereich in Polynomialzeit berechnet.

Definition B.2.3 Ein (Polynomialzeit-)Orakel für ein Problem A (A -Orakel) ist ein nicht spezifizierter Algorithmus, welcher A (in Polynomialzeit) löst.

Die Idee ist es, anzunehmen, daß ein Polynomialzeitalgorithmus zum Lösen eines Problems A existiert, welchen man als Orakel bezeichnet. Diesen (unbekannten) Algorithmus verwendet man zur Lösung eines anderen Problems B . Ein Algorithmus in *Orakel-Polynomialzeit mit Orakel für A* zur Lösung des Problems B darf eine polynomiell in der Länge der Eingabe beschränkte Schrittzahl haben und in jedem Schritt höchstens eine Anfrage an das Orakel stellen. Kann man einen Algorithmus in Orakel-Polynomialzeit mit einem Orakel für A zum Lösen von B angeben und hat man ein Polynomialzeit-Algorithmus zum Lösen von A , dann kann man auch einen Polynomialzeitalgorithmus zum Lösen von B angeben.

Die Komplexität eines (Entscheidungs-) Problems wird über die *Sprache*, in der es formuliert ist, beschrieben. Wir wollen hier vereinfachte Definitionen geben.

Definition B.2.4 Ein *Wort* sei definiert als eine endliche, geordnete Kombination von Elementen aus dem zugehörigen *Alphabet* \mathcal{A} (einer beliebigen Menge). Die Menge aller Wörter bezeichnen wir mit \mathcal{A}^* . Eine *Sprache* $A := (\mathcal{A}, W_A, F_A)$ (mit Alphabet \mathcal{A}) sei die Menge der in Ihr gültigen Wörter $W_A \subseteq \mathcal{A}^*$, verbunden mit einer Menge von Funktionen F_A .

Definition B.2.5 Für die Sprache A mit Alphabet \mathcal{A} ist die *charakteristische Funktion* $\chi_A : \mathcal{A}^* \rightarrow \{0, 1\}$ definiert durch

$$\chi_A(a) = 1 \Leftrightarrow a \in W_A .$$

Im folgenden wollen wir uns auf die Sprachen A mit dem Alphabet $\mathcal{A} = \{0, 1\}$ und einer (eielementigen) Funktionenmenge, die nur die charakteristische Funktion enthält, (also von der Form $A = (\{0, 1\}, W_A, \{\chi_A\})$ sind,) beschränken. Wir schreiben $x \in A$ für ein $x \in \mathcal{A}^*$, wenn $\chi_A(x) = 1$ und $x \notin A$ respektive.

Definition B.2.6 Die Klasse \mathcal{P} der Polynomialzeit-Sprachen besteht genau aus den Sprachen A , für welche die charakteristische Funktion χ_A in Polynomialzeit berechenbar ist.

Definition B.2.7 Die Klasse \mathcal{NP} der nichtdeterministischen Polynomialzeitsprachen A ist erklärt durch

$$A \in \mathcal{NP} : \Leftrightarrow \left\{ \exists_{B \in \mathcal{P}, W_B \subseteq \{0,1\}^* \times \{0,1\}^*, P \in \mathbb{R}[X]} : W_A = \left\{ x \in \{0,1\}^* \mid \exists_{y \in \{0,1\}^{P(|x|)}} : (x,y) \in W_B \right\} \right\} .$$

Ist $(x, y) \in B$, heißt y Zeuge für $x \in A$.

Für jedes Wort aus der Sprache A gibt es einen Zeugen, so daß wir in Polynomialzeit entscheiden können, ob $(x, y) \in B$ gilt. Ist in Polynomialzeit ein Zeuge berechenbar, bzw. zu entscheiden, daß es keinen Zeugen gibt, dann gilt $A \in \mathcal{P}$. Es folgt $\mathcal{P} \subseteq \mathcal{NP}$. Existiert ein Zeuge y für $x \in A$, so gibt es einen nichtdeterministischen Polynomialzeitalgorithmus, welcher $x \in A$ entscheiden kann, daraus folgt jedoch nicht, daß $A \in \mathcal{P}$, da man für $x \notin A$ nicht garantiert ist, daß man die Nichtexistenz eines Zeugen in Polynomialzeit beweisen kann.

Definition B.2.8 (Cook-Karp-Reduktion) Seien A, B Sprachen:

$$A \leq_{\text{pol}} B \iff \exists h: A \rightarrow B : \forall x \in \{0,1\}^* : x \in A \iff h(x) \in B,$$

wobei h eine Abbildung ist, so daß für alle $x \in A$ der Funktionswert $h(x)$ in Polynomialzeit berechenbar ist.

Die Relation \leq_{pol} ist transitiv, und es gilt $A \leq_{\text{pol}} B$ wenn man $x \in A$ in Polynomialzeit mit einmaliger Anwendung eines Orakels für B entscheiden kann.

Definition B.2.9 (NP-schwer) Eine Sprache $A \subseteq \{0,1\}^*$ heißt \mathcal{NP} -schwer, wenn

$$\forall B \in \mathcal{NP} : B \leq_{\text{pol}} A$$

Hat man ein Orakel für ein \mathcal{NP} -schweres Problem, so kann man jedes Problem in \mathcal{NP} in Orakel-Polynomialzeit lösen.

Definition B.2.10 (NP-vollständig) Eine Sprache $A \subseteq \{0,1\}^*$ heißt \mathcal{NP} -vollständig, wenn $A \in \mathcal{NP}$ und \mathcal{NP} -schwer ist.

Könnte man also ein Polynomialzeit-Algorithmus für eine \mathcal{NP} -schweres Problem angeben, hätte man $\mathcal{P} = \mathcal{NP}$ gezeigt. Man vermutet jedoch, daß $\mathcal{P} \neq \mathcal{NP}$.

Beispiel B.2.11 Die Sprache IP der *ganzahligen Programmierung* ist definiert durch $IP := (\mathbb{Z}, W_{IP}, \{\chi_{IP}\})$ mit

$$W_{IP} = \{(m, n, A, b) \in \mathbb{N} \times \mathbb{N} \times \mathbb{Z}^{m \times n} \times \mathbb{Z}^m \mid \exists x \in \mathbb{Z}^n : Ax \leq b\}$$

Nach [21] gilt, daß IP \mathcal{NP} -vollständig ist. Ein Problem liegt in \mathcal{P} (bzw. in \mathcal{NP} , ...) wenn die Sprache, die es beschreibt in der entsprechenden Klasse liegt, also heißt das Entscheidungsproblem der ganzahligen Programmierung ebenfalls \mathcal{NP} -schwer:

Beispiel B.2.12 Das *Entscheidungsproblem der ganzahligen Programmierung* lautet:

- Gegeben $m, n \in \mathbb{N}$, $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$.
- Entscheide, ob ein $x \in \mathbb{Z}^n$ existiert mit $Ax \leq b$.

Hat man ein Orakel für IP , so kann man das Problem der ganzzahligen Programmierung in Orakel-Polynomialzeit lösen. Dieses fordert neben der Entscheidung der Existenz eines x mit $Ax \leq b$ die Angabe eines solchen, falls es existiert.

B.3 Public-Key Kryptosysteme und Sicherheitsbegriffe

Alle hier behandelten Kryptosysteme sind *Public-Key-Kryptosysteme*. Im Gegensatz zu den symmetrischen Verschlüsselungsverfahren müssen Ver- und Entschlüsseler nicht über ein gemeinsames Geheimnis verfügen, um miteinander kommunizieren zu können.

Definition B.3.1 Ein Public-Key-Kryptosystem (PKC) Π wird durch ein 5-Tupel von Mengen $\{N, K_p, K_s, M, C\}$ und ein 3-Tupel $\{K, E_{k_p}, D_{k_s}\}$ von Algorithmen beschrieben, so daß

- K ein probabilistischer Schlüsselerzeugungsalgorithmus ist, der auf die Eingabe eines *Sicherheitsparameters* $n \in N$ einen *öffentlichen Schlüssel* $k_p \in K_p$ und einen *privaten Schlüssel* $k_s \in K_s$ in Abhängigkeit von dem Sicherheitsparameter n erzeugt.
- E_{k_p} ein deterministischer Verschlüsselungsalgorithmus ist, der mit der Eingabe von p_k und einer Nachricht m aus dem *Klartextraum* M einen Schlüsseltext c aus dem *Schlüsseltextraum* C zurückgibt.
- D_{k_s} ein deterministischer Entschlüsselungsalgorithmus ist, der auf die Eingabe von s_k and c , einen Wert $m \in M$ zurückgibt. Wir fordern weiter, daß $D_{s_k} \circ E_{p_k} = \text{Id}$.

Definition B.3.2 Ein *Probabilistisches PKC* Π wird durch ein 6-Tupel von Mengen $\{N, K_p, K_s, M, R, C\}$ und ein 3-Tupel von Algorithmen $\{K, E_{k_p}, D_{k_s}\}$ beschrieben, so daß

- K ein probabilistischer Schlüsselerzeugungsalgorithmus ist, der auf die Eingabe eines *Sicherheitsparameters* $n \in N$ einen *öffentlichen Schlüssel* $k_p \in K_p$ und einen *privaten Schlüssel* $k_s \in K_s$ in Abhängigkeit von dem Sicherheitsparameter n erzeugt.
- E_{k_p} ein deterministischer Verschlüsselungsalgorithmus ist, der mit der Eingabe von p_k , einer Nachricht $m \in M$, und einem zufällig aus R gewählten r einen Schlüsseltext $c \in C$ zurückgibt.
- D_{k_s} ein deterministischer Entschlüsselungsalgorithmus ist, der auf die Eingabe von s_k and c , einen Wert $m \in M$ zurückgibt. Wir fordern weiter, daß die Wahrscheinlichkeit, daß $(D_{s_k} \circ E_{p_k})(m, r) \neq m$ zu vernachlässigen ist.

Ein Beispiel für ein probabilistisches PKC ist NRTU, RSA dagegen ist es nicht. Es wurden verschiedene Sicherheitsbegriffe für Kryptosysteme entwickelt,

von denen einige speziell die Situation eines probabilistischen PKC fordern. Wichtig ist dabei oft die Rolle des Angreifers. Dieser könnte versuchen, sich neben einigen Schlüsseltexten und der Beschreibung des Kryptosystems („*Ciphertext Only Attack*“) weitere Informationen zu beschaffen. Ein gängiges Szenario ist zum Beispiel die „*Chosen Ciphertext Attack*“, bei der der Angreifer selbst gewählte Schlüsseltexte von einem Orakel entschlüsseln lassen kann, ohne den zugehörigen Schlüssel zu kennen. Er versucht aus diesen Informationen den Schlüssel zu ermitteln. Dieses könnte auch durch eine sogenannte „*Brute Force Attack*“ geschehen, bei der man alle in Frage kommenden Schlüssel oder Klartexte gegenüber dem vorliegenden Schlüsseltext testet. Ein Kryptosystem gilt als unsicher (gebrochen), wenn es einem Angreifer gelingt, in angemessener Zeit den privaten Schlüssel teilweise bzw. vollständig zu ermitteln, oder zu einem beliebigen Schlüsseltext einen Anteil bzw. dem gesamten Klartext zu berechnen.

Definition B.3.3 Ein Kryptosystem heißt *perfekt sicher*, wenn die Wahrscheinlichkeit, daß ein bestimmter Schlüsseltext c auftritt und daß ein bestimmter Klartext m vorliegt, unabhängig sind. (Vgl. [10])

Definition B.3.4 Ein Kryptosystem heißt *beweisbar sicher*, wenn gezeigt werden kann, daß derjenige, der es brechen kann, auch ein bekanntes (und als schwierig angenommenes mathematisches) Problem zu lösen vermag. Gilt auch die Umkehrung, sagt man, die Sicherheit ist *äquivalent* zu dem primitiven Problem.

Für die probabilistischen Public-Key-Kryptosysteme wurden die zueinander äquivalenten Begriffe der *polynomiellen* bzw. *semantischen Sicherheit* gebildet. Ein *Angreifer* sei hierbei ein probabilistischer Polynomzeitalgorithmus:

Definition B.3.5 Ein Public-Key-Kryptosystem (PKS) heißt *polynomiell sicher*, wenn kein passiver Angreifer zwei Klartexte m_1 und m_2 aus genügend großen Klartexträumen wählen und dann zwischen den Verschlüsselungen von m_1 und m_2 in Polynomzeit mit einer signifikant größeren Wahrscheinlichkeit als $\frac{1}{2}$ korrekt unterscheiden kann. Siehe auch [48].

Oft ist die oben stehende Definition als semantische Sicherheit angegeben, doch die folgende Definition ist der Idee der semantischen Sicherheit näher: Ein Angreifer versucht, Informationen über den Klartext aus der Kenntnis des Schlüsseltextes und des Verschlüsselungsverfahrens zu erhalten. Kann er dabei in Polynomzeit nur Informationen errechnen, die er auch ohne den Schlüsseltext erschließen könnte (z.B. aus der Wahrscheinlichkeitsverteilung über dem Klartextrraum), so gilt das Verfahren als sicher. Es ist klar, daß jedes semantisch sichere PKS probabilistisch sein sollte (sind m_1 und m_2 deterministisch verschlüsselt, kann man sie immer unterscheiden). Die semantische Sicherheit kann als Polynomzeit-Version der perfekten Sicherheit angesehen werden.

Viele Kryptosysteme, die nicht semantisch sicher sind, versucht man durch sogenannte *Padding*s semantisch sicher zu machen. Die Anwendung solcher Pad-

dings ist jedoch vielfältiger, da man z.B. auch Übertragungs- oder Entschlüsselungsfehler mit ihrer Hilfe identifizieren kann.

Definition B.3.6 Ein Padding für ein PKC Π ist eine probabilistische Funktion $P : M' \rightarrow M$ mit $M' \subset M$. Wir bezeichnen dann $m' \in M'$ als *Nachricht* und $P(m') = m$ als *Klartext*.

Die Definition eines Paddings für ein probabilistisches PKC ist analog. Für $M = \{0, 1\}^{10}$ ist die Funktion $P : \{0, 1\}^5 \rightarrow M, m \mapsto (m, m)$ ein mögliches Padding, um Übertragungsfehler zu erkennen. Jedoch ist anzunehmen, daß diese Padding-Technik Angriffe erleichtert. Die Erkennung von Übertragungsfehlern ist auch eine Anwendung von *Hash-Funktionen*, jedoch bilden diese Objekte beliebiger Länge auf Objekte fester Länge ab, was z.B. Kapazitäten bei der Übertragung einspart.

Definition B.3.7 Eine Hash-Funktion ist eine Abbildung

$$h : \{0, 1\}^* \longrightarrow \{0, 1\}^n, \quad n \in \mathbb{N}$$

Der Funktionswert einer Hash-Funktion soll in Polynomialzeit berechenbar sein. Bei der Verwendung in der Kryptographie fordert man unter anderem, daß man zu einem Bild kein (unbekanntes) Urbild in Polynomialzeit finden kann (Einwegfunktion, Kollisionsresistenz, siehe z.B. [10]).