

TECHNISCHE UNIVERSITÄT DARMSTADT

SICHERHEITSPARAMETER FÜR DAS
OKAMOTO-TANAKA-UCHIYAMA
QUANTEN-PUBLIC-KEY-KRYPTOSYSTEM

Diplomarbeit

von

Sabine Köhler



März 2006

Unter Leitung von

Prof. Dr. Johannes Buchmann
Dr. Christoph Ludwig
Fachbereich Kryptographie
Technische Universität Darmstadt

Ehrenwörtliche Erklärung

Hiermit versichere ich die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31. März 2006

Sabine Köhler

Inhaltsverzeichnis

1	Einleitung	5
2	Mathematische Grundlagen	6
2.1	Algebraische Strukturen	6
2.1.1	Kongruenzen	6
2.1.2	Gruppen und Körper	7
2.1.3	Ideale und Primzahlen	9
2.1.4	Zahlkörper	10
2.2	Gitter	13
2.3	Zusätzliche Definitionen	13
3	Kryptographische Grundlagen	15
3.1	Einführung	15
3.2	Verschlüsselungsverfahren	16
3.3	Definitionen aus der Komplexitätstheorie	18
4	Das OTU Quanten-Public-Key-Kryptosystem	22
4.1	Einleitung	22
4.1.1	Public-Key-Kryptosystem mittels \mathcal{NP} -harter Probleme	22
4.1.2	Knapsack-basierende Kryptosysteme	23
4.2	Quanten-Public-Key-Kryptosysteme	24
4.3	Das Modell eines QPKC von Okamoto, Tanaka und Uchiyama	26
4.3.1	Die Idee	26
4.3.2	Notationen	27
4.3.3	OTU QPKC über den Körper der rationalen Zahlen	27
4.3.4	OTU QPKC über dem imaginär quadratischen Körper	29
4.3.5	Das allgemeine OTU QPKC	32
4.3.6	Bemerkungen zu den einzelnen Varianten	33
5	Wahl der kryptographischen Schlüsselgrößen	35
5.1	Einführung	35
5.2	Das Modell von Lenstra und Verheul	37
5.2.1	Lebensdauer	38
5.2.2	Sicherheitsgrad	38

5.2.3	Computerausstattung	39
5.2.4	Kryptoanalyse	40
5.2.5	Bemerkungen	40
5.3	Ergebnisse von Lenstra und Verheul	42
5.3.1	Methoden zur Berechnung	42
5.3.2	Benutzung der Ergebnis-Tabelle	43
5.3.3	Alternative Sicherheitsgrade	44
5.3.4	Ausstattungskosten äquivalent zur Schlüsselgröße	45
5.3.5	Konsequenzen der Tabelle für OTU	45
6	Angriffe auf OTU	46
6.1	Allgemeine Angriffsmöglichkeiten	46
6.1.1	Angriff auf privaten Schlüssel	46
6.1.2	Angriff auf Schlüsseltext	47
6.2	Dichteangriff auf OTU	48
7	Praktische Ergebnisse	53
8	Fazit	67
9	Literaturverzeichnis	68

1 Einleitung

Obwohl sich die Kryptographie in den Jahren sehr stark verändert hat, ist ihr Fundament doch gleich geblieben. Sie beruht auf der Existenz schwieriger Probleme. Ein Problem ist dann als schwer anzusehen, wenn die Lösung sich nicht mit vertretbarem Einsatz lösen lässt.

Quantencomputer haben in den vergangenen Jahren für Aufsehen gesorgt, weil sich mit ihnen einige Probleme theoretisch sehr viel effizienter lösen lassen als es bisher möglich ist. Ein Beispiel hierfür ist das Faktorisieren großer ganzer Zahlen in ihre Primfaktoren. Das Faktorisierungsproblem ist bei der Sicherheit mancher Kryptoverfahren wichtig wie beispielsweise bei RSA. Mit einem Quantencomputer soll sich das Faktorisierungsproblem mit Hilfe des Algorithmus von Shor effizient lösen lassen, so dass die Sicherheit von RSA auf dem Spiel stünde. Vorausgesetzt natürlich, dass es möglich ist, Quantencomputer zu realisieren. Daher ist es notwendig, neue Kryptoverfahren zu entwickeln, die den Fähigkeiten der Quantencomputer Stand halten und deren Sicherheit nicht auf dem Faktorisierungsproblem beruht. Eine mögliche Alternative haben Okamoto, Tanaka und Uchiyama mit ihrem Kryptosystem gegeben. Es basiert auf einem Knapsack-Problem und soll in dieser Arbeit vorgestellt und auf seine Sicherheit hin untersucht werden.

In den ersten Kapiteln werden die mathematischen und kryptographischen Grundlagen erwähnt, die für das Verständnis dieser Arbeit wichtig sind. Anschließend wird das System von Okamoto, Tanaka und Uchiyama (kurz OTU) vorgestellt und auf die Wahl der Schlüsselgrößen von Kryptosystemen eingegangen. Im Anschluß werden die möglichen Angriffe auf OTU sowie die praktischen Ergebnisse vorgestellt.

2 Mathematische Grundlagen

In diesem Kapitel werden die mathematischen Grundlagen behandelt, die für das Verständnis der nachfolgenden Kapitel benötigt werden. Die Definitionen und verschiedenen mathematischen Sätze stammen aus den Büchern von Johannes Buchmann [4], Helmut Hasse [11], [23], [16] sowie aus Wikipedia [31].

2.1 Algebraische Strukturen

In der Kryptographie wird viel in algebraischen Strukturen gerechnet. Einige sollen hier kurz vorgestellt werden.

2.1.1 Kongruenzen

Definition 2.1 Eine ganze Zahl a heißt kongruent zu b modulo m , in Zeichen:

$$a \equiv b \pmod{m},$$

wenn m die Differenz $b - a$ teilt.

Die Kongruenz modulo m ist eine Äquivalenzrelation auf der Menge der ganzen Zahlen und hat deshalb die Eigenschaften der

- Reflexivität ($a \equiv a \pmod{m}$),
- Symmetrie (aus $a \equiv b \pmod{m}$ folgt $b \equiv a \pmod{m}$) und
- Transitivität (aus $a \equiv b \pmod{m}$ und $b \equiv c \pmod{m}$ folgt $a \equiv c \pmod{m}$).

Die Äquivalenzklasse von a besteht dann aus allen ganzen Zahlen, die sich durch die wiederholte Addition mit m ergeben, und schreibt sich $\{b : b \equiv a \pmod{m}\} = a + m\mathbb{Z}$. Sie wird als Restklasse von $a \pmod{m}$ bezeichnet.

Mit $\mathbb{Z}/m\mathbb{Z}$ wird die Menge aller Restklassen mod m bezeichnet. Die Menge hat m Elemente, weil sich bei der Division durch m die Reste $0, 1, 2, \dots, m - 1$ ergeben. Eine Menge, die aus jeder Restklasse mod m genau ein Element enthält, wird Vertretersystem für diese Äquivalenzrelation genannt. Ein solches Vertretersystem wird auch als volles Restsystem mod m bezeichnet. Ein Beispiel dafür ist \mathbb{Z}_m , das aus der Menge $\{0, 1, 2, \dots, m - 1\}$ besteht.

2.1.2 Gruppen und Körper

Definition 2.2 Sei X eine Menge. Eine Abbildung $\circ : X \times X \rightarrow X$ heißt eine innere Verknüpfung auf X , wenn sie jedem Paar (x_1, x_2) aus X ein Element $x_1 \circ x_2$ zuordnet. Diese innere Verknüpfung \circ heißt

- assoziativ, wenn für alle $a, b, c \in X$ gilt: $(a \circ b) \circ c = a \circ (b \circ c)$.
- kommutativ, wenn für alle $a, b \in X$ gilt: $a \circ b = b \circ a$.

Die Addition ist eine innere Verknüpfung auf der Menge der rationalen Zahlen, die assoziativ und kommutativ ist.

Definition 2.3 Sei H eine nicht leere Menge und \circ eine assoziative innere Verknüpfung auf H . (H, \circ) heißt dann Halbgruppe. Die Halbgruppe heißt kommutativ oder abelsch, wenn die innere Verknüpfung \circ kommutativ ist.

$(\mathbb{Z}, +)$ ist beispielsweise eine kommutative Halbgruppe.

Definition 2.4 Sei e ein Element aus H . Das Element e wird neutrales Element der Halbgruppe (H, \circ) genannt, wenn für alle $a \in H$ gilt: $e \circ a = a \circ e = a$. Eine Halbgruppe mit einem neutralen Element wird auch als Monoid bezeichnet.

Ein Element $b \in H$ heißt Inverses von a , wenn $a \circ b = b \circ a = e$ gilt. a heißt invertierbar in der Halbgruppe, wenn a ein Inverses besitzt.

Eine Halbgruppe hat allerdings höchstens ein neutrales Element und in Monoiden besitzt jedes Element höchstens ein Inverses.

Definition 2.5 Eine Gruppe ist eine Halbgruppe, die ein neutrales Element besitzt und in der jedes Element invertierbar ist. Die Gruppe heißt kommutativ oder abelsch, wenn die Halbgruppe kommutativ ist.

Die Gruppe $(\mathbb{Z}, +)$ ist abelsch. Ihr neutrales Element ist die 0 und das Inverse von a ist $-a$.

Definition 2.6 Eine Teilmenge U von einer Gruppe G heißt Untergruppe von G , wenn U mit der Verknüpfung von G selbst eine Gruppe ist.

Sei G eine Gruppe. Die Menge $\{g^k : k \in \mathbb{Z}\}$ bildet für jedes $g \in G$ eine Untergruppe von G . Sie wird die von g erzeugte Untergruppe genannt und als $\langle g \rangle$ geschrieben.

Definition 2.7 Sei $G = \langle g \rangle$ für ein $g \in G$, dann heißt G zyklisch und g wird als der Erzeuger von G bezeichnet. Die Gruppe G ist demnach die von g erzeugte Gruppe.

Definition 2.8 Ein Ring ist ein Tripel $(R, +, \cdot)$ mit zwei Operationen. $(R, +)$ ist eine abelsche Gruppe und (R, \cdot) ist eine Halbgruppe. Im Ring gelten für alle $x, y, z \in R$ die Distributivgesetze

$$\begin{aligned}x \cdot (y + z) &= (x \cdot y) + (x \cdot z) \text{ und} \\(x + y) \cdot z &= (x \cdot z) + (y \cdot z).\end{aligned}$$

Der Ring heißt kommutativ, wenn die Halbgruppe (R, \cdot) kommutativ ist. Ein Einselement des Ringes ist ein neutrales Element der Halbgruppe (R, \cdot) . Ein Nullelement ist das neutrale Element der Halbgruppe $(R, +)$.

Das Tripel $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ ist ein kommutativer Ring mit dem Einselement $1 + m\mathbb{Z}$, der auch als Restklassenring modulo m bezeichnet wird.

Definition 2.9 Sei R ein Ring mit Einselement. Ein Element a von R heißt invertierbar oder Einheit, wenn es in der multiplikativen Halbgruppe von R invertierbar ist. Das Element a heißt Nullteiler, wenn es von Null verschieden ist und es ein von Null verschiedenes Element $b \in R$ gibt mit $ab = 0$ oder $ba = 0$. Enthält R keine Nullteiler, so heißt R nullteilerfrei.

Definition 2.10 Ein Körper ist ein kommutativer Ring mit Einselement, in dem jedes von Null verschiedene Element invertierbar ist.

Wenn von einem endlichen Körper gesprochen wird, ist ein Körper mit nur endlich vielen Elementen gemeint.

Für die Division im Restklassenring muss die Kongruenz $ax \equiv 1 \pmod{m}$ gelöst werden, also die Restklasse $a + m\mathbb{Z}$ gefunden werden, die ein multiplikatives Inverses besitzt.

Theorem 2.1 Die Restklasse $a + m\mathbb{Z}$ ist genau dann in $\mathbb{Z}/m\mathbb{Z}$ invertierbar, wenn der größte gemeinsame Teiler $\text{ggT}(a, m) = 1$ ist. Für $\text{ggT}(a, m) = 1$ ist das Inverse von $a + m\mathbb{Z}$ eindeutig bestimmt und somit ist die Lösung x von $ax \equiv 1 \pmod{m}$ eindeutig bestimmt modulo m .

Beweis siehe [4].

Theorem 2.2 Der Restklassenring $\mathbb{Z}/m\mathbb{Z}$ ist genau dann ein Körper, wenn m eine Primzahl ist.

Beweis siehe [4].

Definition 2.11 Sei R ein Ring mit Einselement. Die Menge aller multiplikativ invertierbaren Elemente (Einheiten) von R bildet mit der Ringmultiplikation eine Gruppe. Sie wird Einheitengruppe von R genannt. Man schreibt die Einheitsgruppe meist als R^* oder R^\times .

2.1.3 Ideale und Primzahlen

Sei R ein Ring.

Definition 2.12 Seien $a, b, g \in R$. b heißt Teiler von a , wenn es ein g gibt, so dass $a = gb$ ist. Man schreibt dann $b|a$.

Eine andere Ausdrucksweise ist b teilt a . Jedes a hat die trivialen Teiler ± 1 und $\pm a$.

Definition 2.13 Eine natürliche Zahl p heißt Primzahl, wenn $p \neq 1$ ist und keine nicht-trivialen Teiler hat.

Definition 2.14 Ganze Zahlen a, b, \dots heißen zueinander teilerfremd oder prim, wenn ihr größter gemeinsamer Teiler $\text{ggT}(a, b, \dots) = 1$ ist; sie heißen paarweise teilerfremd, wenn dies für jedes aus ihnen herausgegriffene Paar der Fall ist.

Theorem 2.3 Jede natürliche Zahl $a > 1$ besitzt mindestens einen Primteiler p (das bedeutet: eine Primzahl p mit $p|a$) und zwar ist der kleinste natürliche Teiler $p > 1$ von a Primzahl.

Beweis siehe [11].

Und nun der Hauptsatz der elementaren Zahlentheorie.

Satz 2.1 Jede natürliche Zahl a besitzt eine Darstellung

$$a = p_1 \dots p_n$$

als Produkt einer Anzahl $n \geq 0$ von Primzahlen p_1, \dots, p_n . Von der willkürlichen Reihenfolge der Faktoren abgesehen ist diese Darstellung eindeutig.

Die eindeutige Darstellung $a = p_1 \dots p_n$ heißt Primfaktorzerlegung. Für den Fall, dass $n = 0$ ist, ist $a = 1$.

Beweis siehe [11].

Bevor die Primideale definiert werden, soll hier kurz erklärt werden, was unter einem Ideal verstanden wird.

Definition 2.15 Eine Teilmenge \mathcal{I} eines kommutativen Ringes R heißt Ideal von R , wenn es die folgenden Eigenschaften erfüllt:

- Die Null des Ringes liegt in \mathcal{I} .
- Wenn $a, b \in \mathcal{I}$ sind, so auch $a + b \in \mathcal{I}$ und $a - b \in \mathcal{I}$.
- Wenn $a \in \mathcal{I}$ und $r \in R$, so auch $r \cdot a \in \mathcal{I}$.

Jeder Ring hat die trivialen Ideale $\{0\}$ und R . Ein Ideal \mathcal{I} eines Ringes R heißt *Hauptideal*, wenn \mathcal{I} von einem Element $a \in R$ erzeugt wird: $\mathcal{I} = Ra = (a)$. Ein Ring R heißt *Hauptidealring*, wenn jedes Ideal in R ein Hauptideal ist. So ist beispielsweise jeder Körper K ein Hauptidealring, da (0) und K seine einzigen Ideale sind.

Definition 2.16 Sei R ein kommutativer Ring mit Einselement und $\mathfrak{p} \subset R$ ein Ideal in R . Dann heißt \mathfrak{p} *Primideal*, wenn für alle $x, y \in R$ gilt: aus $xy \in \mathfrak{p}$ folgt $x \in \mathfrak{p}$ oder $y \in \mathfrak{p}$.

Der quadratische Rest ist der Rest der Division einer Quadratzahl durch eine natürliche Zahl.

Definition 2.17 Eine Zahl a heißt *quadratischer Rest modulo m* , falls es eine Zahl x gibt mit

$$x^2 \equiv a \pmod{m}.$$

Das Legendre-Symbol gibt von einer natürlichen Zahl a an, ob sie quadratischer Rest oder quadratischer Nichtrest modulo einer Primzahl p mit $p > 2$ ist. Es ist nach dem französischen Mathematiker Adrien-Marie Legendre benannt.

Definition 2.18 Für eine Primzahl p und eine dazu teilerfremde, natürliche Zahl a ist das Legendre-Symbol definiert als

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{wenn } a \text{ ein quadratischer Rest zu } p \text{ ist} \\ -1, & \text{wenn } a \text{ kein quadratischer Rest zu } p \text{ ist} \\ 0, & \text{wenn } a \text{ und } p \text{ nicht teilerfremd sind und } p \text{ } a \text{ teilt} \end{cases}$$

2.1.4 Zahlkörper

Definition 2.19 Sei R ein kommutativer Ring mit Einselement. Ein Polynom in einer Variablen über R ist ein Ausdruck

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

wobei x die Variable ist und die Koeffizienten a_i zu R gehören. Die Menge aller Polynome über R in der Variablen x wird mit $R[x]$ bezeichnet. Sei $a_n \neq 0$, dann heißt n der Grad des Polynoms. Man schreibt $n = \deg f$.

Definition 2.20 Ist $r \in R$, so heißt $f(r) = a_n r^n + \dots + a_0$ der Wert von f an der Stelle r . Ist $f(r) = 0$, so heißt r *Nullstelle* von f .

Ein Polynom $f \in K[x]$, $f \neq 0$, hat höchstens $\deg f$ viele Nullstellen.

Definition 2.21 Sei L ein Körper, und sei K eine Teilmenge von L , die 0 und 1 enthält und mit den auf K eingeschränkten Verknüpfungen Addition und Multiplikation selbst ein Körper ist. Somit ist die 0 das Nullelement und 1 das Einselement des Körpers K . In diesem Fall heißt K Unterkörper (oder Teilkörper) von L und L heißt Oberkörper (oder Erweiterungskörper) von K . Das Paar L und K wird als Körpererweiterung bezeichnet und als L/K geschrieben.

Die Körpererweiterung kann dabei endlich oder unendlich sein, je nachdem ob der Grad der Erweiterung endlich oder unendlich ist. L kann dabei als Vektorraum über K aufgefasst werden. Die Körperaddition in L entspricht dann der Vektoraddition und Multiplikation von $a * b$ mit $a \in L$ und $b \in K$ entspricht der Skalarmultiplikation. Die Dimension des Vektorraums wird demnach Grad der Erweiterung genannt.

Definition 2.22 Ein Vektorraum V über einem Körper K oder kurz K -Vektorraum ist eine abelsche Gruppe $(V, +, 0)$ mit einer als Skalarmultiplikation bezeichneten Verknüpfung $\cdot : K \times V \rightarrow V$. Die Skalarmultiplikation muss die folgenden Eigenschaften für alle $v, w \in V$ und $a, b \in K$ erfüllen:

- $a(v + w) = av + aw$
- $(a + b)v = av + bv$
- $(ab)v = a(bv)$
- $1v = v$

Die Elemente von V werden als Vektoren bezeichnet und die Elemente von K als Skalare. Die Addition wird Vektoraddition genannt.

Definition 2.23 Sei V ein K -Vektorraum, dann heißt die Teilmenge $U \subseteq V$ Untervektorraum, wenn die folgenden Bedingungen erfüllt sind:

- $0 \in U$
- Es gilt für alle $x, y \in U$: $x + y \in U$.
- Es gilt für alle $x \in U$ und $a \in K$: $ax \in U$.

Definition 2.24 Sei $S = v_1, v_2, \dots, v_n$ eine Teilmenge von einem Vektorraum V über einem Körper K . Dann gilt folgendes:

- Eine Linearkombination von S ist ein Ausdruck der Form $a_1v_1 + a_2v_2 + \dots + a_nv_n$ mit $a_i \in K$.
- Der Span von S , $\langle S \rangle$, ist die Menge aller Linearkombinationen von S . Somit ist der Span von S ein Untervektorraum von V .

- Wenn U ein Untervektorraum von V ist, dann spannt S U auf, wenn $\langle S \rangle = U$ ist.
- Die Menge S ist linear abhängig von K , wenn Skalare a_1, \dots, a_n existieren, die nicht alle Null sind, und es gilt: $a_1v_1 + a_2v_2 + \dots + a_nv_n = 0$. Gibt es keine solchen Skalare, so ist S linear unabhängig von K .
- Eine Menge linear unabhängiger Vektoren, die V aufspannt, wird Basis von V genannt.

Definition 2.25 Wenn ein Vektorraum V eine Basis hat, dann heißt die Anzahl der Elemente einer Basis Dimension von V , $\dim V$.

Definition 2.26 Es sei L / K eine endliche Körpererweiterung. Ein fest gewähltes Element $a \in L$ definiert eine K -lineare Abbildung $L \rightarrow L, a \rightarrow xa$. Ihre Determinante heißt die Norm von a , $N_{L/K}(a)$. Sie ist ein Element von K ; die Norm ist also eine Abbildung $N_{L/K} : L \rightarrow K, a \rightarrow N_{L/K}$

Die Norm eines Primideals $\mathcal{N}(\mathfrak{p})$ ist die p -Potenz einer ganzrationalen Primzahl p laut [24]. In unserem Beispiel aus Satz 4.1 ist die Norm $\mathcal{N}(\mathfrak{p}) = p^f$.

Definition 2.27 Ein algebraischer Zahlkörper ist eine endliche Erweiterung des Körpers der rationalen Zahlen \mathbb{Q} .

Die Elemente von Zahlkörpern sind Nullstellen von Polynomen mit rationalen Koeffizienten. Diese Zahlkörper enthalten den ganzen Zahlen analoge Teilmengen, die Ganzheitsringe. Sie verhalten sich in vieler Hinsicht wie der Ring der ganzen Zahlen, aber manche Eigenschaften nehmen eine etwas andere Form an. Beispielsweise gibt es im Allgemeinen keine eindeutige Zerlegung in Primzahlen mehr, sondern nur noch in Primideale.

Definition 2.28 Ein quadratischer Zahlkörper ist eine Körpererweiterung K/\mathbb{Q} der Form $K = \mathbb{Q}(\sqrt{D})$ mit einer rationalen Zahl D , die kein Quadrat in \mathbb{Q} ist. Dies sind genau die Erweiterungen vom Grad 2 über \mathbb{Q} . Ist $D > 0$, so heißt K reell-quadratischer Zahlkörper, sonst imaginär-quadratischer Zahlkörper.

Ohne Einschränkung kann D als quadratfreie ganze Zahl angenommen werden. Die Diskriminante des Zahlkörpers ist dann D , wenn $D \equiv 1 \pmod{4}$ ist, und andernfalls $4D$, wenn $D \equiv 2, 3 \pmod{4}$.

Die Diskriminante ist in der Algebra eine einem Polynom zugeordnete Kennzahl, die Auskunft über mehrfache Nullstellen geben kann.

Definition 2.29 Gesucht ist eine Lösung für die quadratische Gleichung $ax^2 + bx + c = 0$ mit $a \neq 0$. Mit Hilfe der Mitternachtsformel ergibt sich die Lösungsformel $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Die Diskriminante ist $D = b^2 - 4ac$ und bestimmt für eine Gleichung mit reellen Koeffizienten, wie viele reellwertige Lösungen die Gleichung hat. Es werden drei Fälle unterschieden:

1. $D > 0$: Es gibt zwei verschiedene Nullstellen, x_1 und x_2 .
2. $D = 0$: Es gibt eine Lösung, da $x_1 = x_2$ ist.
3. $D < 0$: Es gibt keine Lösung für die quadratischen Gleichung.

Definition 2.30 Es sei K ein algebraischer Zahlkörper. Der Ganzheitsring \mathcal{O}_K von K ist definiert als die Teilmenge derjenigen $x \in K$, die eine Gleichung der Form

$$x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0 = 0$$

mit $c_i \in \mathbb{Z}$ erfüllen. Man beachte, dass der Koeffizient von x_n gleich 1 sein muss.

Definition 2.31 Sei $l \in \mathbb{N}$ und seien $\omega_1, \dots, \omega_l \in \mathcal{O}_K$. $[\omega_1, \dots, \omega_l]$ heißt Ganzheitsbasis von \mathcal{K} , wenn sich jede Zahl α eindeutig darstellen lässt als

$$\alpha = a_1\omega_1 + \dots + a_l\omega_l$$

mit $a_i \in \mathbb{Z}$ ($1 \leq i \leq l$).

2.2 Gitter

Ein Gitter $L \subset \mathbb{R}^n$ ist eine diskrete additive Untergruppe des \mathbb{R}^n derart, dass

$$L = \left\{ \sum_{i=1}^k x_i \underline{b}_i \mid x_i \in \mathbb{Z}, i = 1, \dots, k, k \leq n \right\},$$

wobei $\underline{b}_1, \underline{b}_2, \dots, \underline{b}_k \in \mathbb{R}^n$ linear unabhängige Vektoren sind. Wir bezeichnen mit

$$[\underline{b}_1, \dots, \underline{b}_k] \in \mathbb{R}^{k \times n}$$

die Basis B des Gitters $L = L(B) = L([\underline{b}_1, \dots, \underline{b}_k])$ mit Dimension k .

2.3 Zusätzliche Definitionen

Bei der Verschlüsselung von OTU wird ein Wert benötigt, der als Hamming Gewicht bekannt ist.

Definition 2.32 Das Hamming Gewicht w eines Wortes $x = x_1 \dots x_n$ mit $n \in \mathbb{N}$ aus einer endlichen Zeichenfolge $\{0, 1\}^n$ ergibt sich aus

$$w(x) := \#\{i \in \{1, 2, \dots, n\} \mid x_i \neq 0\}.$$

D.h. das Hamming Gewicht ist die Anzahl Bits, die innerhalb eines Wortes Eins sind. Für das Wort $x = 1001010$ ist das Hamming Gewicht $w(1001010) = 3$.

In [28] findet sich der Algorithmus von Shor für die Berechnung des diskrete Logarithmus. Im allgemeinen wird ein diskrete Logarithmus wie folgt definiert, wobei die Ordnung einer Gruppe die Anzahl ihrer Elemente ist.

Definition 2.33 *Gegeben ist eine endlich zyklische Gruppe G der Ordnung n . Der Erzeuger dieser Gruppe sei g und das neutrale Element in G sei 1 . Sei A ein Gruppenelement. Gesucht wird die kleinste nicht negative Zahl a für die gilt: $A = g^a$. Der Exponent a heißt diskreter Logarithmus von A zur Basis g .*

Oft wird der diskrete Logarithmus in einer primen Restklassengruppe ($\text{mod } p$) gesucht. Dann ist g eine prime Primitivwurzel ($\text{mod } p$) und der diskrete Logarithmus lässt sich wie folgt schreiben. $A \equiv g^a \pmod{p}$ oder $a = \log_g A$.

3 Kryptographische Grundlagen

In diesem Kapitel sollen die grundlegenden Eigenschaften eines Kryptosystems erläutert werden. Grundlage dieses Kapitels sind die Bücher von Johannes Buchmann [4] und Claudia Eckert [8], in denen auch ausführlicher auf die einzelnen Aspekte eingegangen wird, und Definitionen aus den Quellen [2, 16, 27, 26, 31].

3.1 Einführung

Die Kryptographie ist im traditionellen Sinn die Lehre der Verschlüsselung von Informationen. Verschlüsselung ist die Transformation von Daten in eine Form, so dass es unmöglich ist, die Daten ohne zusätzliches Wissen (Schlüssel) zu lesen. Die moderne Kryptographie kann man als eine Wissenschaft bezeichnen, die sich mit mathematischen Techniken beschäftigt, um Information zu schützen und sich insbesondere mit den Aspekten Vertraulichkeit, Datenintegrität und Authentifikation von Objekten (Daten) und Subjekten (Personen) befasst. Sie hat vier Hauptziele:

- *Vertraulichkeit*: Man spricht von Vertraulichkeit, wenn keine unauthorisierte Informationsgewinnung möglich ist. Das bedeutet bei einer verschlüsselten Nachricht, dass nur der gewünschte Empfänger in der Lage ist, den Inhalt zu lesen. Es ist nicht möglich, Informationen über den Nachrichteninhalt zu erlangen, selbst wenn man Zugang zu der verschlüsselten Nachricht hat.
- *Datenintegrität*: Datenintegrität ist gewährleistet, wenn unbefugte Änderungen von Informationen erkennbar sind. Bei einer verschickten Nachricht bedeutet dies, dass der Empfänger einer Nachricht in der Lage ist festzustellen, ob die Nachricht seit ihrer Übertragung verändert – Informationen wurden hinzugefügt, gelöscht oder ausgetauscht – wurde.
- *Authentifikation*: Bei der Authentifikation wird die Identität von Daten oder Personen überprüft. Beispielsweise kann der Empfänger den Absender einer Nachricht eindeutig identifizieren und es ist überprüfbar, ob die Nachricht tatsächlich vom Absender stammt.
- *Verbindlichkeit*: Verbindlichkeit verhindert, dass eine Person im Nachhinein die Durchführung einer Handlung abstreiten kann. So kann zum Beispiel ein Absender nicht abstreiten, dass er eine Nachricht gesendet hat.

Ein Ziel der Kryptographie ist es, diese vier Ziele adäquat in der Theorie und Praxis umzusetzen. Natürlich muss nicht jedes kryptographische System oder Verfahren diese Ziele erreichen. Dies hängt immer auch von deren Einsatzgebieten ab, da einige Ziele in bestimmten Umgebungen überflüssig oder nicht anwendbar sind.

3.2 Verschlüsselungsverfahren

Verschlüsselungsverfahren werden benötigt, um Nachrichten oder ganz allgemein Informationen geheim zu halten und deren Vertraulichkeit zu wahren. Es geht also darum, eine Nachricht zwischen zwei Partnern (Sender und Empfänger) auszutauschen, ohne dass Dritte Informationen über den Nachrichteninhalte erlangen können. Hierbei werden die Verschlüsselungsverfahren zuerst in der Art unterschieden, ob der Sender und Empfänger ein gemeinsames Geheimnis (Schlüssel) teilen oder nicht. Bei symmetrischen Verschlüsselungsverfahren teilen sich beide Partner einen gemeinsamen Schlüssel beziehungsweise der Schlüssel des einen Partners ist leicht aus dem Schlüssel des anderen Partners zu berechnen. Bei asymmetrischen Verschlüsselungsverfahren, die auch als Public-Key-Kryptosysteme bezeichnet werden, sind die Schlüssel der Partner verschieden und darüber hinaus ist der Schlüssel des Empfängers auch nicht mit vertretbarem Aufwand aus dem Schlüssel des Senders zu berechnen. Das Verschlüsselungsverfahren von OTU ist ein asymmetrisches Verschlüsselungsverfahren.

Definition 3.1 *Ein Public-Key-Kryptosystem (PKC) ist ein Tupel $(\mathcal{P}, \mathcal{C}, \mathcal{EK}, \mathcal{DK}, \mathcal{E}, \mathcal{D})$ mit den folgenden Eigenschaften:*

1. \mathcal{P} ist eine nicht leere endliche Menge von Klartexten und wird als der Klartext-raum bezeichnet.
2. \mathcal{C} ist eine nicht leere endliche Menge von Schlüsseltexten und wird als der Schlüsseltextraum bezeichnet.
3. \mathcal{EK} ist eine nicht leere Menge von Verschlüsselungsschlüsseln und wird als der öffentliche Schlüsselraum bezeichnet.
4. \mathcal{DK} ist eine nicht leere Menge von Entschlüsselungsschlüsseln und wird als der private Schlüsselraum bezeichnet.
5. $\mathcal{E} = \{E_k : k \in \mathcal{EK}\}$ ist eine Familie von Funktionen $E_k : \mathcal{P} \rightarrow \mathcal{C}$, die einen Klartext in einen Schlüsseltext umwandelt. Ihre Elemente werden als Verschlüsselungsfunktionen bezeichnet.
6. $\mathcal{D} = \{D_k : k \in \mathcal{DK}\}$ ist eine Familie von Funktionen $D_k : \mathcal{C} \rightarrow \mathcal{P}$, die einen Schlüsseltext in einen Klartext verwandelt. Ihre Elemente werden als Entschlüsselungsfunktionen bezeichnet.

7. Es existiert für jeden Schlüssel $e \in \mathcal{EK}$ ein Schlüssel $d \in \mathcal{DK}$, so dass für alle Nachrichten $m \in \mathcal{P}$ gilt: $D_d(E_e(m)) = m$.

Ein Benutzer eines Public-Key-Kryptosystems erzeugt sich mit Hilfe eines vorgegebenen Algorithmus zwei Schlüssel. Den öffentlichen Schlüssel e wird der Benutzer in einem jedem zugänglichen Verzeichnis bekannt geben. Den privaten Schlüssel d muß der Benutzer geheimhalten und daher an einem sicheren Ort vor dem Zugriff Dritter verwahren. Eine Person, die dem Benutzer eine Nachricht m senden möchte, verschlüsselt die Nachricht mit dem öffentlichen Schlüssel e des Benutzers. Anschließend entschlüsselt der Benutzer die verschlüsselte Nachricht c mit seinem privaten Schlüssel d . Ein Problem bei der Entwicklung von

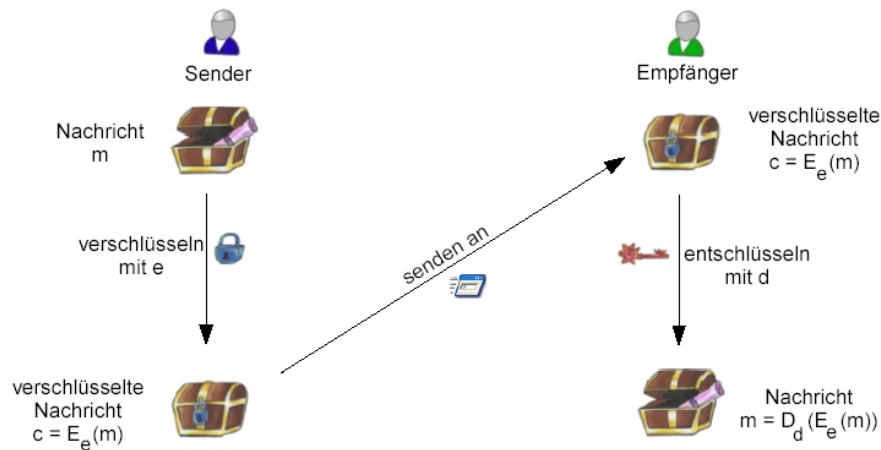


Abbildung 3.1: Beispiel für eine Ver- und Entschlüsselung einer Nachricht¹

Public-Key-Kryptosystemen besteht darin, eine Funktion zu finden, die es einem ermöglicht, effizient öffentliche Schlüssel zu erstellen, und es Angreifern unmöglich macht, aus dem öffentlichen Schlüssel den privaten Schlüssel zu berechnen. Die Lösung zu diesem Problem bieten die Einwegfunktionen.

Definition 3.2 Eine Einwegfunktion $f : X \rightarrow Y$ hat die folgenden Eigenschaften:

1. Es existiert ein Algorithmus A , der den Funktionswert $y = f(x)$ für alle $x \in X$ der Funktion f in polynomieller Zeit berechnet.
2. Es gibt kein effizientes Verfahren, dass aus einem Bild $y = f(x)$ ein Urbild x berechnet.

¹Leicht veränderte Graphik aus [30].

Einen effizienten Algorithmus zur Berechnung des Urbildes von y gibt es nicht, da jeder Algorithmus, der das Urbild berechnen soll, zuviel Zeit benötigt. Es gibt allerdings keinen Beweis für die Existenz von Einwegfunktionen. In der Praxis gibt es aber Funktionen, die den Bedingungen ausreichend genügen. Ein Beispiel für eine Einwegfunktion ist die Multiplikation zweier Primzahlen, da die Primfaktorzerlegung ein schwieriges Problem darstellt.

Eine Besonderheit bei den Einwegfunktionen stellt die Trapdoor-Einwegfunktion dar. Ihre Funktion ist leicht zu berechnen und sie kann auch effizient invertiert werden, wenn man ein Geheimnis kennt. Allerdings kann die Trapdoor-Einwegfunktion ohne das Geheimnis nicht in vertretbarer Zeit invertiert werden und macht es somit Unbefugten unmöglich, die Umkehrfunktion zu berechnen.

Definition 3.3 *Eine Trapdoor-Einwegfunktion $f : X \rightarrow Y$ hat die folgenden Eigenschaften:*

1. *Es existiert ein Algorithmus B , der den Funktionswert $y = f(x)$ für alle $x \in X$ der Funktion f in polynomieller Zeit berechnet.*
2. *Es gibt ein effizientes Verfahren, um die Umkehrfunktion $f^{-1}(y)$ für alle $y \in f[X]$ zu berechnen. Das Urbild x ist allerdings von y nur aus dem Zusammenhang $y = f(x)$ ohne eine zusätzliche Information nicht bestimmbar.*

Eine solche Trapdoor-Einwegfunktion wird zum Beispiel bei RSA benutzt. Bei der Verschlüsselung bei RSA wird mittels des öffentlichen Schlüssels (n, e) die Nachricht m mit $c = m^e \bmod n$ verschlüsselt. Für die Umkehrfunktion wird der private Schlüssel d benötigt und die Nachricht m wird dann mit $m = c^d \bmod n = (m^e)^d \bmod n$ berechnet.

3.3 Definitionen aus der Komplexitätstheorie

Ein Alphabet Σ ist eine endliche, nicht-leere Menge. Die Elemente dieser Menge heißen Zeichen. Ein Beispiel für ein Alphabet ist $\Sigma = a, b, c$. Werden mehrere Zeichen hintereinander geschrieben, so nennt man dies ein Wort. Es gibt natürlich auch das leere Wort. Die Menge aller Wörter für ein Alphabet wird mit Σ^* beschrieben. Eine Sprache ist eine beliebige Teilmenge von Σ^* .

Alan Turing entwickelte ein mathematisches Modell, mit der die Menge der berechenbaren Funktionen gebildet werden kann. Seine Turingmaschine kann alle von Menschen berechenbaren mathematischen Funktionen berechnen, nicht jedoch alle mathematischen Funktionen. Zudem kann die Turingmaschine jeden klassischen Computer simulieren.

Definition 3.4 *Eine Turingmaschine (TM) ist gegeben durch ein 7-Tupel*

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E) \quad \text{mit}$$

- Z ist die endliche Zustandsmenge,
- Σ ist das Eingabealphabet,
- $\Gamma \supset \Sigma$ ist das Arbeitsalphabet,
- $\delta : Z \times \Gamma \longrightarrow Z \times \Gamma \times \{L, R, N\}$ ist im deterministischen Fall (bzw. $\delta : Z \times \Gamma \longrightarrow \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$ im nichtdeterministischen Fall) die Überföhrungsfunktion,
- $z_0 \in Z$ ist der Startzustand,
- $\square \in \Gamma - \Sigma$ ist das Leerzeichen,
- $E \subseteq Z$ ist die Menge der Endzustände.

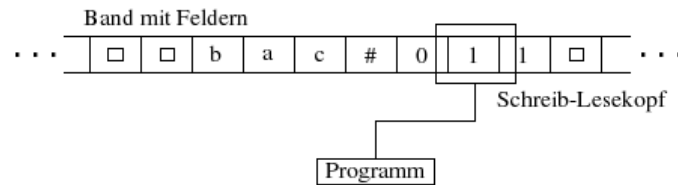


Abbildung 3.2: Turingmaschine

Die deterministische Turingmaschine besteht aus einem unendlichen Band, welches in Felder unterteilt ist. In einem Feld kann ein Zeichen des Arbeitsalphabets gespeichert werden. Auf dem Band bewegt sich ein Schreib-Lesekopf, der nur um eine Position nach rechts oder links bewegt werden kann. Es kann dabei lediglich der Inhalt des Feldes verändert werden, auf dem sich der Schreib-Lesekopf befindet. Felder, über die sich der Schreib-Lesekopf nicht bewegt hat oder die er nicht verändert hat, enthalten das Leerzeichen. Eine Turingmaschine arbeitet ein sogenanntes Programm ab, das eine Abfolge von Befehlen ist, wie sich der Schreib-Lesekopf über das Band bewegen soll. Zu Anfang steht die Turingmaschine auf dem ersten Zeichen der Eingabe und befindet sich somit im Startzustand z_0 . Sie liest das Zeichen vom Band und macht dann – abhängig von dem gelesenen Zeichen und dem jeweiligen Zustand, in dem sich die TM befindet – eine oder mehrere der folgenden Aktionen: Sie schreibt ein Zeichen auf das Band, bewegt den Schreib-Lesekopf nach links oder rechts und ändert ihren Zustand. Dies wird mittels der Überföhrungsfunktion beschrieben. Die Bewegungen des Schreib-Lesekopfes werden in der Überföhrungsfunktion mit L für eine Bewegung nach links, R für rechts und N für ein Stehenbleiben des Kopfes beschrieben. Nachdem die Turingmaschine ein Programm beendet hat, hat die Turingmaschine einen Endzustand erreicht und das Ergebnis befindet sich als Ausgabe auf dem Band der Turingmaschine.

Die nicht-deterministische Turingmaschine ist ebenso aufgebaut wie die deterministische Turingmaschine, allerdings verhält sie sich bei der Überföhrungsfunktion anders. Der nächste Zustand, der sich aus dem Zeichen auf dem Band und dem aktuellen Zustand ergibt, ist nicht mehr eindeutig. Es gibt für jeden Zustand im allgemeinen eine Anzahl von Folgezuständen. Dadurch gibt es verschiedene Möglichkeiten für den nächsten Rechenweg der Turingmaschine. Welcher dieser Rechenwege gewählt wird, kann vorher nicht eindeutig bestimmt werden. Auch eine nichtdeterministische Turingmaschine kann über einen der verschiedenen Rechenwege in einen Endzustand gelangen.

Von einer probabilistischen Turingmaschine spricht man, wenn einer nichtdeterministischen Turingmaschine für die verschiedenen Möglichkeiten der Überföhrungsfunktion Wahrscheinlichkeitswerte zugeordnet werden. Das heißt, die Turingmaschine wird eine bestimmte Zeit laufen gelassen und dann ist der Zustand, in dem sie sich zu der Zeit befindet, durch eine Wahrscheinlichkeitsverteilung über alle erreichbaren Zustände beschrieben.

Eine Quanten-Turingmaschine (QTM) unterscheidet sich von einer Turingmaschine insofern, dass alle Aktionen der QTM auf Basis quantenmechanischer Wechselwirkungen basieren. Das heißt, dass das Band als auch der Kopf der QTM durch einen Quantenzustand gegeben sind. Zudem kann eine QTM verschiedene Eingabewerte gleichzeitig lesen und alle Berechnungen für die Eingabewerte simultan ausführen. Eine QTM kann jede klassische Turingmaschine simulieren.

Um den Aufwand eines mathematischen Problems oder einer Aufgabe einschätzen zu können, hat man die Komplexitätsklassen eingeföhrt. In den einzelnen Komplexitätsklassen werden dann die verschiedensten Probleme zusammengefasst, die das gleiche Maß an Komplexität haben. Die Komplexität definiert sich durch den Ressourcenbedarf (also Laufzeit und Speicherplatz) für die Lösung eines Problem und dessen Abhängigkeit zu der Problemgröße. Die Problemgröße bezieht sich meist auf die Länge der Eingabe für ein Problem und verändert den Ressourcenverbrauch entsprechend. Für ein Knapsack-Problem wie es in 4.1.2 beschrieben wird, ist bei einer Eingabelänge von 2 trivial, da es nur eine mögliche Lösung gibt. Bei größeren Eingaben muss jedoch ein Lösungsalgorithmus wesentlich mehr Arbeit leisten. Es soll hier auf die wichtigsten Komplexitätsklassen für diese Arbeit kurz eingegangen werden.

Definition 3.5 *Ein Problem ist in polynomialer Zeit lösbar, wenn die benötigte Rechenzeit m höchstens polynomial mit der Problemgröße n wächst.*

Die Polynomialzeit wird als Grenze zwischen praktisch lösbaren (\mathcal{P}) und praktisch nicht lösbaren Problemen (\mathcal{NP}) betrachtet.

Die Komplexitätsklasse \mathcal{P} enthält alle Probleme, die von einer deterministischen Turingmaschine in polynomialer Zeit gelöst werden können.

Die Komplexitätsklasse \mathcal{NP} bezeichnet die Klasse von Problemen, die von einer nichtdeterministischen Turingmaschine in polynomieller Zeit gelöst werden können.

Die Komplexitätsklasse BQP (bounded error quantum polynomial time) bezeichnet die Klasse der Probleme, die auf einem Quantencomputer mit einer Fehlerwahrscheinlichkeit kleiner als $1/4$ in Polynomialzeit gelöst werden können.

In der Komplexitätstheorie werden Probleme betrachtet. Um jedoch die folgenden Definitionen beschreiben zu können, benötigen wir eine andere Sichtweise. Ein Problem kann als eine formale Sprache verstanden werden und die Lösung eines Problems entspricht dann der Entscheidung, ob ein Wort zu einer Sprache gehört oder nicht.

Definition 3.6 *Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A auf B polynomial reduzierbar ($A \leq_p B$), falls es eine total und mit polynomialer Komplexität berechnbare Funktion $f: \Sigma^* \rightarrow \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt:*

$$x \in A \iff f(x) \in B.$$

Nun lässt sich \mathcal{NP} -hart definieren als:

Definition 3.7 *Eine Sprache A heißt \mathcal{NP} -hart, falls für alle Sprachen $L \in \mathcal{NP}$ gilt: $L \leq_p A$.*

Definition 3.8 *Eine Sprache A heißt \mathcal{NP} -vollständig, falls A \mathcal{NP} -hart ist und $A \in \mathcal{NP}$ gilt.*

Ein Beispiel für ein \mathcal{NP} -vollständiges Problem ist das in dieser Arbeit behandelte Knapsack-Problem.

4 Das OTU Quanten-Public-Key-Kryptosystem

In diesem Kapitel wird das Quanten-Public-Key-Kryptosystem (QPKC) von Tatsua-ki Okamoto, Keisuke Tanaka und Shigenori Uchiyama [20] vorgestellt. Dabei werden zuerst zwei vereinfachte Versionen des QPKC betrachtet, bevor das eigentliche QPKC dargestellt wird. Beide Versionen sind in ihren Definitionsbereichen beschränkt, so dass damit die Versionen des QPKC in ihrer Sicherheit gegenüber dem vollständigen QPKC eingeschränkt sein könnten. Diese Sicherheitsbedenken und auch die allgemeine Sicherheit von dem QPKC werden in einem späteren Kapitel dann genauer beleuchtet.

4.1 Einleitung

In unserer heutigen Zeit spielen Public-Key-Kryptosysteme eine entscheidende Rolle bei der Sicherung und vertraulichen Weitergabe von Informationen und Daten. Fast alle bisherigen Public-Key-Kryptosysteme basieren auf einem Faktorisierungsproblem oder auf einem diskreten Logarithmus Problem. Seit einiger Zeit gibt es bei der Realisierung von Quantencomputern schon etliche entscheidende Erfolge zu vermelden, so dass die Zeit bis zu einem Quantencomputer mittelfristig scheint. Mit Quantencomputern ist es möglich, Probleme effizienter zu lösen als mit heutigen Rechnern. Es gibt einen Algorithmus von Shor [28], der eine Benutzung von Quantencomputern voraussetzt, mit dem es möglich es könnte, das Faktorisierungsproblem sowie auch diskrete Logarithmen von Public-Key-Kryptosystemen zu lösen. Somit wäre die Sicherheit von vielen Kryptosystemen gefährdet. Es ist daher notwendig, sich nach alternativen Public-Key-Kryptosystemen umzusehen bzw. sie zu entwickeln. Okamoto, Tanaka und Uchiyama versuchen, mit ihrem QPKC eine Alternative zu bieten und ihr System soll hier nun vorgestellt werden.

4.1.1 Public-Key-Kryptosystem mittels \mathcal{NP} -harter Probleme

Viele bisherigen Public-Key-Kryptosysteme basieren auf \mathcal{NP} -harten Problemen. Diese Systeme wurden nach dem klassischen PKC Modell erstellt, in dem unter anderem alle teilnehmenden Parteien eines PKC als klassische Turingmaschinen (TM) angenommen werden. Wenn ein solches PKC gegenüber einem Angreifer,

der als Quanten-Turingmaschine (QTM) angenommen wird, Stand hält, so könnte es das QPKC von Okamoto, Tanaka und Uchiyama sein. Das OTU QPKC verwendet Elemente der Quantenmechanik für die Schlüsselerstellung. Trotzdem benutzt es ein PKC Modell, in dem alle teilnehmenden Parteien und auch Angreifer klassische TMs sind. Der Fall, dass die Angreifer QTMs sind, ist selbst in dem QPKC Modell ein Spezialfall, solange die klassischen TM in den QTM enthalten sind.

Leider sind die meisten PKCs, die auf \mathcal{NP} -harten Problemen basieren, gebrochen worden und die Sicherheit der ungebrochenen Systeme scheint trügerisch wegen der fehlenden Einfachheit ihrer Trapdoor-Einwegfunktionen. Der Vorteil bei dem OTU Modell ist laut Okamoto, Tanaka und Uchiyama neben der Benutzung eines \mathcal{NP} -harten Problems als Basis, dass es Quantenmechanik bei der Schlüsselerstellung verwenden kann. Bei der Entwicklung von PKC gibt es durch die Benutzung der Quantenmechanik neue Freiheiten. Okamoto, Tanaka und Uchiyama haben dafür ein Beispiel mit ihrem Entwurf gegeben. Das OTU System basiert auf einem Knapsack (siehe 4.1.2) und benutzt eine sehr einfache Trapdoor-Einwegfunktion. Diese Trapdoor-Einwegfunktion sieht durch ihre Einfachheit sehr viel sicherer aus als alle anderen Knapsack-basierenden Systeme, die nach der traditionellen Methode entstanden sind.

4.1.2 Knapsack-basierende Kryptosysteme

Das Knapsack-Problem ist ein typisches \mathcal{NP} -hartes Problem und einige Kryptosysteme beruhen auf diesem Problem.

Definition 4.1 *Das Knapsack-Problem lautet:*

Gegeben sind eine Anzahl $n \in \mathbb{N}$ und Gewichte $g_1, g_2, \dots, g_n \in \mathbb{N}$ sowie eine Zahl $s \in \mathbb{N}$. Es wird ein Vektor $x = (x_1, x_2, \dots, x_n)$ mit $x_i \in \{0, 1\}^n$ gesucht, der die Gleichung

$$\sum_{i=1}^n g_i x_i = s$$

erfüllt oder es wird gezeigt, dass der Vektor x nicht existiert.

Das Problem wird auch als Rucksack- oder Subset-Sum-Problem bezeichnet.

Die Dichte d eines Knapsackes berechnet sich aus den Gewichten g_1, g_2, \dots, g_n und deren Anzahl mit

$$d = \frac{n}{\log_2 \max_{1 \leq i \leq n} g_i}.$$

Das System von Okamoto, Tanaka und Uchiyama basiert auf einem Knapsack und ist angelehnt an den multiplikativen Knapsack von Merkle-Hellman¹ [17] und

¹Es handelt sich hierbei nicht um den bekannten Merkle-Hellman Knapsack, der auf einem schnell wachsenden Vektor basiert.

dem System von Chor-Rivest [5]. Das System von Merkle-Hellman wurde gebrochen und zwar zum einen unter gewissen Bedingungen von Odlyzko [18] und zum anderen bei Angriffen auf seine niedrige Dichte, die asymptotisch gegen Null geht. Typische Realisierungen von dem Chor-Rivest System sind von Schnorr-Hoerner und Vaudenay (siehe [10], [25]) gebrochen worden wegen der geringen Dichte des Knapsackes und der Symmetrie der Informationen der Trapdoor-Einwegfunktion. Beide Systeme benutzen dabei eine Trapdoor-Einwegfunktion in der Schlüsselerstellung und zwar bei der Berechnung eines diskreten Logarithmus. Da keine Quantencomputer berücksichtigt werden konnten, war die Empfehlung für beide Systeme, eine spezielle Klasse eines diskreten Logarithmus zu benutzen, der sich leicht auf einer klassischen Maschine berechnen lässt. Das System von Okamoto, Tanaka und Uchiyama benutzt wesentlich allgemeinere Mathematik als die beiden anderen Systeme. Merkle-Hellman benutzen einen Ring der rationalen Zahlen \mathbb{Z} mit einer rationalen Primzahl p und berechnen ihren diskrete Logarithmus in $\mathbb{Z}/p\mathbb{Z}$. Chor-Rivest verwenden einen Polynomring über einem endlichen Körper \mathbb{F}_p mit einem nicht reduzierbaren Polynom $g(x)$ und ihr diskreter Logarithmus wird in $\mathbb{F}_p[x]/(g(x))$ berechnet. Das OTU System benutzt \mathcal{O}_K , den Ring der ganzen Zahlen in einem algebraischen Zahlkörper K , und einem Primideal \mathfrak{p} und der diskrete Logarithmus berechnet sich in $\mathcal{O}_K/\mathfrak{p}$. Alle Systeme berechnen ihren diskreten Logarithmus zwar in einem endlichen Körper, doch das System von Okamoto, Tanaka und Uchiyama hat dabei einige Vorteile gegenüber den anderen beiden:

- Es werden keine Informationen im öffentlichen Schlüssel über den zugrunde liegenden algebraischen Zahlkörper K gegeben im Gegensatz zu dem Merkle-Hellman System mit \mathbb{Z} und Chor-Rivest mit $\mathbb{F}_p[x]$, welche beide öffentlich sind. Hierbei ist zu beachten, dass es bei OTU exponentiell viele Kandidaten gibt, aus denen K gewählt werden kann.
- Es werden keine Informationen über den zugrunde liegenden endlichen Körper veröffentlicht. Bei Chor-Rivest wird der zugrunde liegende endliche Körper preisgegeben.
- Die Dichte des Knapsack-Problems von OTU ist mindestens 1, während sie beim Merkle-Hellman System asymptotisch gegen Null geht. Wenn die Parameter passend gewählt werden, liegt die Informationsrate bei OTU asymptotisch bei 1 (siehe 4.3.6).

4.2 Quanten-Public-Key-Kryptosysteme

In diesem Abschnitt wird auf die QPKCs und verwandte Definitionen eingegangen. Als erstes soll kurz auf den Unterschied zwischen Quantenkryptographie und QPKC erläutert werden. Beides sind Lösungen, die für Quantencomputer

ausgelegt sind. Die Quantenkryptographie kann als ein Ersatz für die herkömmliche Public-Key Verschlüsselung gesehen werden, um die Schlüssel zwischen zwei Kommunikationspartnern auszutauschen. Der wichtigste Unterschied zwischen Quantenkryptographie und QPKC ist, dass bei der Quantenkryptographie ein Quantenkanal zusätzlich zum klassischen Kanal verwendet wird, während ein QPKC nur einen klassischen Kanal benutzt. Die Sicherheitsannahmen bei der Quantenkryptographie beruhen auf der Quantenmechanik und beim QPKC auf Berechnungsannahmen wie beispielsweise der Existenz von Einwegfunktionen in einem QTM Modell.

Bei der Quantenkryptographie wird ein Quantenkanal benötigt, um die Nachrichten übertragen zu können. Der Sender erzeugt eine Nachricht und überträgt sie zu dem Empfänger. Der Empfänger misst anschließend die Nachricht von dem Quantenkanal. Jeder fremde Zugriff auf die Nachricht wird bemerkt, da aufgrund des Charakters der Quantenmechanik, die Nachricht durch das Abhören verändert würde. Im Bereich der QPKC sind viele Verschlüsselungs- und Schlüsselverteilungsanwendungen über klassische Kanäle möglich – ähnlich zu denen die zur Zeit durch PKC schon existieren.

Die Definitionen, die hier eingeführt werden sollen, sind vielfach von den klassischen PKC Definitionen abgeleitet, indem die klassische Turingmaschine durch eine Quanten-Turingmaschine ersetzt wurde. Somit lassen sich Quanten-Einwegfunktionen und Quanten-Public-Key Verschlüsselung definieren. Es ist zu beachten, dass bei den beiden folgenden Definitionen, alle Variablen in den Definitionen klassische Wörter sind und kein Quantenkanal zwischen den teilnehmenden Parteien angenommen wird.

Definition 4.2 *Eine Funktion f wird Quanten-Einwegfunktion genannt, wenn sie die beiden Bedingungen erfüllt:*

1. *Es existiert ein Algorithmus A für eine QTM, der für eine Eingabe x den Wert $f(x)$ in polynomial Zeit liefert. Also $A(x) = f(x)$.*
2. *Für jede probabilistische polynomial zeitbeschränkte QTM M und jedes Polynom h sowie für ausreichend große n gilt:*

$$\Pr[M(f(x)) \in f^{-1}(f(x))] < 1/h(n)$$

Die Wahrscheinlichkeit ist über die Verteilung von x , vom klassischen Münzenwurf von M und den Beobachtungen im Quantenbereich von M berechnet worden.

Definition 4.3 *Eine Quanten-Public-Key Verschlüsselung (QPKE) besteht aus drei probabilistischen polynomialen zeitbeschränkten QTMs (G , E , D), die sich wie folgt definieren:*

1. G ist eine probabilistische polynomial zeitbeschränkte QTM für die Erstellung von Schlüsseln. Auf die Eingabe 1^n gibt G das Paar (e, d) zurück, das mit sehr hoher Wahrscheinlichkeit die Bitlänge n hat. e ist dann der öffentliche Schlüssel und d der private Schlüssel. n ist dabei der Sicherheitsparameter.
2. E ist eine probabilistische polynomial zeitbeschränkte QTM für die Verschlüsselung, die den Schlüsseltext c erzeugt. D ist eine probabilistische polynomial zeitbeschränkte QTM für die Entschlüsselung. Für jede Nachricht m der Länge $|m| = n$ und jedes Polynom h gilt für ausreichend große n :

$$\Pr[D(E(m, e), d) = m] > 1 - 1/h(n).$$

Die Wahrscheinlichkeit ist über den klassischen Münzenwurf und den Beobachtungen im Quantenbereich von (G, E, D) berechnet worden.

Okamoto, Tanaka und Uchiyama haben bei der Definition von QPKE die Sicherheitsbeschreibungen ausgelassen, weil die Definition leicht um diese Beschreibungen erweitert werden könnte. Der einzige Unterschied zwischen den klassischen und Quanten-Sicherheitsdefinitionen ist, dass in QPKC die Angreifer als probabilistische polynomiale zeitbeschränkte QTM angenommen werden.

4.3 Das Modell eines QPKC von Okamoto, Tanaka und Uchiyama

4.3.1 Die Idee

Die grundsätzliche Idee, ein QPKC zu realisieren, ist, ein passendes \mathcal{NP} -hartes Problem als Basis zu verwenden. Dies geht allerdings nur, solange die Annahme gilt, dass \mathcal{NP} -vollständig $\not\subseteq$ BQP ist.

Okamoto, Tanaka und Uchiyama halten das Knapsack-Problem für das am besten geeignete Problem für ein QPKC, weil Lösungen zu diesem Problem und vor allem Wege zur Realisierung von Knapsack-basierenden PKC in den letzten 20 Jahren umfassend untersucht worden sind.

Eine typische Trapdoor-Einwegfunktion für Knapsäcke ist ein schnell anwachsender Vektor. Ein solcher Vektor wird dann in einen öffentlichen Vektor umgewandelt, der zufällig erscheint. Trotzdem sind fast alle Umwandlungstricks für einen solchen Vektor wegen ihrer Linearität und ihrer geringen Dichte gebrochen worden. Eine erfolgversprechendere Idee für eine solche Umwandlung ist, eine nicht-lineare Umwandlung zu verwenden, die das Knapsack-Problem löst. Das könnte beispielsweise eine Exponentiation und ein Logarithmus sein, wenn die Berechnung des Logarithmus machbar ist. Die Systeme von Merkle-Hellman und

Chor-Rivest haben eine solche Umwandlung benutzt und sind gebrochen worden. Um die Schwächen dieser Systeme zu überwinden, benutzen Okamoto, Tanaka und Uchiyama \mathcal{O}_K , den Ring der ganzen Zahlen in einem algebraischen Körper K , der zufällig aus einer exponentiellen Anzahl von Kandidaten gewählt wird. Das OTU System basiert also somit auf dem Knapsack-Problem und verfügt über eine einfache Trapdoor-Einwegfunktion, mit der das Knapsack-Problem überbrückt werden kann. Als Orakel für den diskreten Logarithmus wird der Algorithmus von Shor benutzt. Die Varianten des OTU QPKC über dem Körper der rationalen Zahlen und dem imaginär quadratischen Körper sind vereinfachte Versionen des OTU QPKC, die zum besseren Verständnis dienen sollen.

4.3.2 Notationen

Hier werden einige Notationen sowie zwei Sätze der algebraischen Zahlentheorie vorgestellt, die später benutzt werden. K ist ein algebraischer Zahlkörper, \mathcal{O}_K der Ring der ganzen Zahlen in K , und $\mathcal{N}(\mathcal{I})$ die Norm von \mathcal{I} . In dieser Arbeit wird \mathcal{I} als eine Zahl oder ein Ideal von \mathcal{O}_K verstanden. Der Logarithmus von n zur Basis 2 wird mit $\log n$ beschrieben. Der Logarithmus zur Basis e mit $\ln n$.

Satz 4.1 *Wenn K ein algebraischer Zahlkörper und \mathfrak{p} ein Primideal von \mathcal{O}_K sind, dann ist $\mathcal{O}_K/\mathfrak{p}$ ein endlicher Körper, \mathbb{F}_{p^f} , und $\mathcal{N}(\mathfrak{p}) = p^f$. Es existiert eine Ganzheitsbasis $[\omega_1, \dots, \omega_l]$, so dass jede Restklasse von $\mathcal{O}_K/\mathfrak{p}$ eindeutig repräsentiert wird von*

$$a_1\omega_1 + \dots + a_l\omega_l$$

mit l als Grad von K , $0 \leq a_i < e_i$ ($i = 1, \dots, l$) und mit $[e_1\omega_1, \dots, e_l\omega_l]$ als Ganzheitsbasis von \mathfrak{p} . Man beachte, dass $\prod_{i=1}^l e_i = p^f$ ist.

Beweis siehe [11].

Durch die Benutzung der sogenannten Hermite-Normalform (HNF) zur Darstellung der Primideale, kann angenommen werden, dass $\omega_1 = 1$ und $e_1 = p$ ist. (Mehr Details in Abschnitt 4.7 und Übung 17 in [6])

Der folgende Satz ist eine Spezialisierung vom kleinen Satz von Fermat.

Satz 4.2 (Kleiner Satz von Fermat) *Sei \mathfrak{p} ein Primideal von \mathcal{O}_K und g ein Element von $\mathcal{O}_K/\mathfrak{p}$, welches ungleich Null ist. Dann gilt*

$$g^{\mathcal{N}(\mathfrak{p})-1} \equiv 1 \pmod{\mathfrak{p}}.$$

Beweis siehe [11].

4.3.3 OTU QPKC über den Körper der rationalen Zahlen

Okamoto, Tanaka und Uchiyama haben mit der Version eines QPKC über dem Körper der rationalen Zahlen eine Variante geschaffen, die leichter zu verstehen ist

als ihr allgemeines QPKC, das erst später vorgestellt wird. Diese einfache Variante ist nicht so sicher wie das allgemeine QPKC, trotzdem haben Okamoto, Tanaka und Uchiyama keine Angriffe gegen diese Version gefunden. Jedoch sollte diese Version ihrer Meinung nach nicht in der Praxis eingesetzt werden. Wir begrenzen den algebraischen Zahlkörper $\mathcal{K} = \{\mathbb{Q}\}$ auf den Körper der rationalen Zahlen. Dann ist der Ring \mathcal{O}_K von \mathbb{Q} die ganzen Zahlen \mathbb{Z} . Jede Primzahl ist dann aus dem Ring von \mathbb{Z} .

Schlüsselerzeugung

Der öffentliche und private Schlüssel für das QPKC werden wie folgt erzeugt:

1. Wähle die Parameter $n, k \in \mathbb{Z}$ mit fester Größe.
2. Wähle eine zufällige Primzahl p sowie einen Erzeuger g der Gruppe $(\mathbb{Z}/p\mathbb{Z})^\times$. Wähle anschließend n paarweise teilerfremde Zahlen $p_1, p_2, \dots, p_n \in \mathbb{Z}/p\mathbb{Z}$, so dass folgendes gilt: $\prod_{j=1}^k p_{i_j} < p$ für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$.
3. Benutze den Algorithmus von Shor, um den diskrete Logarithmus von $p_i \equiv g^{a_i} \pmod{p}$ für jedes i von $1 \leq i \leq n$ zu lösen. So erhält man die Zahlen $a_1, a_2, \dots, a_n \in \mathbb{Z}/(p-1)\mathbb{Z}$.
4. Wähle zufällig eine Zahl $d \in \mathbb{Z}/(p-1)\mathbb{Z}$.
5. Berechne $b_i = (a_i + d) \pmod{p-1}$ und zwar für jedes i mit $1 \leq i \leq n$.
6. Der öffentliche Schlüssel ist (n, k, b_1, \dots, b_n) und der private Schlüssel ist $(g, d, p, p_1, \dots, p_n)$.

Verschlüsselung

Die Länge des Klartextes bzw. der Nachricht M wird auf die Länge $\lfloor \log \binom{n}{k} \rfloor$ begrenzt. Anschließend wird die Nachricht M in der folgenden Weise zu einem Schlüsseltext c umgewandelt:

1. Verschlüssele den Klartext M in einen binären String $m = (m_1, m_2, \dots, m_n)$ der Länge n und mit dem Hamming Gewicht k (d.h. mit exakt k Einsen) wie folgt:
 - a) Setze $l = k$.
 - b) Für jedes i von 1 bis n tue nun folgendes: Wenn $M \geq \binom{n-i}{l}$ ist, dann setze $m_i = 1$, $M = M - \binom{n-i}{l}$ und $l = l - 1$. Andernfalls setze $m_i = 0$. Man beachte, dass $\binom{l}{0} = 1$ für $l \geq 0$ und $\binom{0}{l} = 0$ für $l \geq 1$.
2. Berechne nun den Schlüsseltext c mit $c = \sum_{i=1}^n m_i b_i$.

Entschlüsselung

Der Schlüsseltext c kann leicht wieder entschlüsselt werden.

1. Berechne $r = (c - kd) \bmod (p - 1)$.
2. Berechne $u = g^r \bmod p$.
3. Finde die Faktoren von u . Wenn p_i ein Faktor von u ist, dann setze $m_i = 1$, ansonsten setze $m_i = 0$.
4. Entschlüssele m zu dem Klartext M wie folgt:
 - a) Setze $M = 0$ und $l = k$.
 - b) Für jedes i von 1 bis n tue das folgende: Wenn $m_i = 1$ ist, dann setze $M = M + \binom{n-i}{l}$ und $l = l - 1$.

4.3.4 OTU QPKC über dem imaginär quadratischen Körper

Diese Variante des OTU QPKC ist gegenüber dem allgemeinen OTU QPKC in ihrem Definitionsbereich beschränkt. Die Beschränkung bezieht sich hier auf die Menge der imaginär quadratischen Körper. Doch bevor auf die Schlüsselberechnung sowie die Ver- und Entschlüsselung eingegangen wird, werden erst ein paar Informationen zu imaginär quadratischen Körpern zusammengefasst und ein neuer Satz eingeführt.

Imaginär quadratische Körper

Sei $K = \mathbb{Q}(\sqrt{-D})$ ein imaginär quadratischer Körper mit der Diskriminate $-D$. \mathcal{O}_K , der Ring der ganzen Zahlen in K , hat eine Ganzheitsbasis $[1, \omega]$ mit $\omega = \sqrt{-D/4}$, wenn $-D \equiv 0 \pmod{4}$ ist und ansonsten hat ω den Wert $\omega = \frac{1+\sqrt{-D}}{2}$. Dieser Ring wird auch als Standardbasis für \mathcal{O}_K bezeichnet. Sei \mathfrak{p} ein Primideal von \mathcal{O}_K mit dem Restklassengrad f , nämlich $\mathcal{N}(\mathfrak{p}) = p^f$ mit p als rationale Primzahl unter \mathfrak{p} . Dann wird eine Ganzheitsbasis von \mathfrak{p} wie $[p, e_2\omega_2]$ gewählt, wobei $e_2 = p^{f-1}$ ist, und $\omega_2 = b + \omega$ mit einer rationalen Zahl b ist. Wenn also beispielsweise $-D \equiv 0 \pmod{4}$ und $-D$ ein quadratischer Rest mod p ist, dann ist b eine Wurzel von $b^2 \equiv -D \pmod{p}$. Man kann diese Basis immer noch als Standardbasis von \mathfrak{p} bezeichnen. Jede Restklasse von $\mathcal{O}_K/\mathfrak{p}$ wird dann einheitlich von $x_1 + x_2\omega$ repräsentiert mit $-p/2 < x_1 < p/2$ und $-e_2/2 < x_2 < e_2/2$ (siehe Satz 4.1). Also kann man das vollständige Repräsentantensystem von $\mathcal{O}_K/\mathfrak{p}$ wie folgt festhalten:

$$R(\mathfrak{p}) = \{x_1 + x_2\omega_2 \in \mathcal{O}_K \mid -p/2 < x_1 < p/2, -e_2/2 < x_2 < e_2/2\}.$$

Daraus lässt sich der folgenden Satz herleiten.

Satz 4.3 Sei $K = \mathbb{Q}(\sqrt{-D})$ ein imaginär quadratischer Körper von der Diskriminate $-D$ und \mathfrak{p} ein Primideal von \mathcal{O}_K mit $\mathcal{N}(\mathfrak{p}) = p^f$, wobei $f = 1, 2$ ist. Sei $[1, \omega]$ die Standardbasis von \mathcal{O}_K und $[p, e_2\omega_2]$ die Standardbasis von \mathfrak{p} . Dann gilt:

1. Fall: $\mathcal{N}(\mathfrak{p}) = p$ ($f = 1$)

Für jede Zahl $x \in \mathcal{O}_K$ mit $x = x_1 + x_2\omega$ gilt, wenn $\mathcal{N}(x) < p^2/4$ und $x_2 = 0$ ist, dann gibt es ein $x \in R(\mathfrak{p})$. Also ist $R(\mathfrak{p}) = \{x \in \mathbb{Z} \mid -p/2 < x < p/2\}$

2. Fall: $\mathcal{N}(\mathfrak{p}) = p^2$ ($f = 2$)

Für jede Zahl $x \in \mathcal{O}_K$ mit $x = x_1 + x_2\omega$ gilt, wenn $-D \equiv 0 \pmod{4}$ und $\mathcal{N}(x) < p^2/4$ ist, dann gibt es ein $x \in R(\mathfrak{p})$. Gilt $-D \equiv 1 \pmod{4}$ und $\mathcal{N}(x) < \frac{(p-1)^2/D}{4(1+D)}$, dann ist $x \in R(\mathfrak{p})$.

Beweis siehe [20].

Schlüsselerzeugung

Der öffentliche und private Schlüssel für das QPKC werden wie folgt erzeugt:

1. Wähle eine Menge \mathcal{K} von imaginär quadratischen Körpern, die dem System zur Verfügung steht.
2. Wähle zufällig einen imaginär quadratischen Körper $K = \mathbb{Q}(\sqrt{-D})$ mit $-D$ als Diskriminate von K aus \mathcal{K} . Sei \mathcal{O}_K der Ring der ganzen Zahlen in K .
3. Wähle die Parameter $n, k \in \mathbb{Z}$ mit fester Größe.
4. Wähle ein Primideal \mathfrak{p} von \mathcal{O}_K . Anschließend wähle ein zufälliges Element g von \mathcal{O}_K , so dass g ein Erzeuger der multiplikativen Gruppe des endlichen Körpers $\mathcal{O}_K/\mathfrak{p}$ ist. Ein Element von $\mathcal{O}_K/\mathfrak{p}$ wird einheitlich von der Basis $[1, \omega_2]$ und dem Zahlenpaar (p, p) repräsentiert. Für ein $x \in \mathcal{O}_K$ existieren also die Zahlen $x_1, x_2 \in \mathbb{Z}$, $-p/2 < x_1, x_2 < p/2$, so dass $x \equiv x_1 + x_2\omega_2 \pmod{\mathfrak{p}}$ ist.
5. Wähle n Zahlen p_1, p_2, \dots, p_n aus $\mathcal{O}_K/\mathfrak{p}$, so dass $\mathcal{N}(p_1), \dots, \mathcal{N}(p_n)$ paarweise teilerfremd sind und für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$ folgendes gilt: Wenn $-D \equiv 0 \pmod{4}$ ist, dann ist $\prod_{j=1}^k \mathcal{N}(p_{i_j}) < \frac{p^2}{4}$. Ansonsten gilt: $\prod_{j=1}^k \mathcal{N}(p_{i_j}) < \frac{(p-1)^2 D}{4(1+D)}$.
6. Mittels des Algorithmus von Shor werden die diskreten Logarithmen für die Zahlen a_1, a_2, \dots, a_n gesucht, so dass folgendes gilt: $p_i \equiv g^{a_i} \pmod{\mathfrak{p}}$ mit $a_i \in \mathbb{Z}/(\mathcal{N}(\mathfrak{p}) - 1)\mathbb{Z}$ und $1 \leq i \leq n$.
7. Wähle zufällig eine rationale Zahl d in $\mathbb{Z}/(\mathcal{N}(\mathfrak{p}) - 1)\mathbb{Z}$.

8. Berechne $b_i = (a_i + d) \bmod (\mathcal{N}(\mathfrak{p}) - 1)$ und zwar für jedes i mit $1 \leq i \leq n$.
9. Der öffentliche Schlüssel ist $(\mathcal{K}, n, k, b_1, \dots, b_n)$ und der private Schlüssel ist $(K = \mathbb{Q}(\sqrt{-D}), g, d, \mathfrak{p}, p_1, \dots, p_n)$.

Verschlüsselung

Siehe 4.3.3.

Entschlüsselung

Der Schlüsseltext c kann leicht wieder entschlüsselt werden.

1. Berechne $r = (c - kd) \bmod (\mathcal{N}(\mathfrak{p}) - 1)$.
2. Berechne $u = g^r \pmod{\mathfrak{p}}$.
3. Sei $[1, \omega_2]$ die Basis und (p, p) das Zahlenpaar, dass in dem Satz 4.1 definiert wird. u kann von der Auswahl von p_1, p_2, \dots, p_n repräsentiert werden durch $u = a_1 + a_2\omega_2$ mit $a_1, a_2 \in \mathbb{Z}$ und $-p/2 < a_i < p/2$ ($i = 1, 2$).
Für jedes p_i für das gilt, dass $p_i | u$ ist, wird $m_i = 1$ gesetzt. Andernfalls wird $m_i = 0$ gesetzt. Nachdem dies für alle p_i ($1 \leq i \leq n$) durchgeführt wurde, setzt man $m = (m_1, m_2, \dots, m_n)$.
4. Entschlüssel m zu dem Klartext M wie folgt:
 - a) Setze $M = 0$ und $l = k$.
 - b) Für jedes i von 1 bis n tue das folgende: Wenn $m_i = 1$ ist, dann setze $M = M + \binom{n-i}{l}$ und $l = l - 1$.

Bemerkungen

Hier noch zwei Bemerkungen zu dieser Variante des QPKC. Die erste Bemerkung betrifft die Wahl des Primideals \mathfrak{p} . Das Primideal \mathfrak{p} vom Grad 2 kann leicht gewählt werden, indem wie folgt vorgegangen wird: Es wird eine rationale Primzahl p gewählt, so dass $-D$ ein kein quadratischer Rest mod p ist. Anschließend wird $\mathfrak{p} = p\mathcal{O}_K$ gesetzt. Mit anderen Worten p ist ein Primelement in \mathcal{O}_K . Ferner lässt es sich schnell überprüfen, ob p eine Primzahl in \mathcal{O}_K ist, indem das Legendre-Symbol $\left(\frac{-D}{p}\right)$ berechnet wird. Ist $\left(\frac{-D}{p}\right) = -1$, dann ist p ein Primelement in \mathcal{O}_K .

Die zweite Bemerkung bezieht sich auf den 5. Schritt in der Schlüsselgenerierung. Es sollte beachtet werden, dass für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$ gilt, dass $\prod_{j=1}^k \mathcal{N}(p_{i_j}) = \mathcal{N}(\prod_{j=1}^k p_{i_j}) < \frac{p^2}{4}$ (oder $\frac{(p-1)^2 D}{4(1+D)}$), dann ist $\prod_{j=1}^k p_{i_j} \in R(\mathfrak{p})$ nach Satz 4.3. Das bedeutet, dass es Zahlen $a_1, a_2 \in$

\mathbb{Z} ($-p/2 < a_1, a_2 < p/2$) gibt, so dass $u = a_1 + a_2\omega_2$ im 3. Schritt der Entschlüsselung ist. Die typische Selektion von p_1, p_2, \dots, p_n wie sie in Abschnitt 4.3.6 gezeigt wird, mag eingeschränkt sein. Tatsächlich nehmen Okamoto, Tanaka und Uchiyama p_1, p_2, \dots, p_n aus den rationalen Zahlen, aber die Selektion von oben ist etwas genereller als die typische aus Abschnitt 4.3.6.

4.3.5 Das allgemeine OTU QPKC

Das allgemeine OTU QPKC Modell ist nach Okamoto, Tanaka und Uchiyama sehr sicher, da es exponentiell viele Kandidaten für die Auswahl von einem algebraischen Zahlkörper K gibt.

Schlüsselerzeugung

Der öffentliche und private Schlüssel für das QPKC werden wie folgt erzeugt:

1. Wähle eine Menge \mathcal{K} von algebraischen Zahlkörpern, die dem System zur Verfügung steht.
2. Wähle zufällig einen algebraischen Zahlkörper K von \mathcal{K} . Sei \mathcal{O}_K der Ring der ganzen Zahlen in K .
3. Wähle die Parameter $n, k \in \mathbb{Z}$ mit fester Größe.
4. Wähle ein Primideal \mathfrak{p} von \mathcal{O}_K . Anschließend wähle ein zufälliges Element g von \mathcal{O}_K , so dass g ein Erzeuger der multiplikativen Gruppe des endlichen Körpers $\mathcal{O}_K/\mathfrak{p}$ ist. Ein Element von $\mathcal{O}_K/\mathfrak{p}$ wird einheitlich von der Basis $[1, \omega_2, \dots, \omega_l]$ und dem Zahlentupel (e_1, e_2, \dots, e_l) (wobei $e_1 = p$) von Satz 4.1 definiert. Für ein $x \in \mathcal{O}_K$ existieren die rationalen Zahlen $x_1, x_2, \dots, x_l \in \mathbb{Z}$ ($0 \leq x_i < e_i$), so dass $x \equiv x_1 + x_2\omega_2 + \dots + x_l\omega_l \pmod{\mathfrak{p}}$ ist. Dabei ist p eine rationale Primzahl unter \mathfrak{p} .
5. Wähle n Zahlen p_1, p_2, \dots, p_n von $\mathcal{O}_K/\mathfrak{p}$, so dass $\mathcal{N}(p_1), \dots, \mathcal{N}(p_n)$ paarweise teilerfremd sind und für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$ es rationale Zahlen a_1, a_2, \dots, a_n ($0 \leq a_i < e_i$) existieren, so dass $\prod_{j=1}^k p_{i_j} = a_1 + a_2\omega_2 + \dots + a_l\omega_l$.
6. Mittels des Algorithmus von Shor werden die diskreten Logarithmen für die Zahlen a_1, a_2, \dots, a_n gesucht, so dass folgendes gilt: $p_i \equiv g^{a_i} \pmod{\mathfrak{p}}$ mit $a_i \in \mathbb{Z}/(\mathcal{N}(\mathfrak{p}) - 1)\mathbb{Z}$ und $1 \leq i \leq n$.
7. Wähle zufällig eine rationale Zahl d in $\mathbb{Z}/(\mathcal{N}(\mathfrak{p}) - 1)\mathbb{Z}$.
8. Berechne $b_i = (a_i + d) \pmod{\mathcal{N}(\mathfrak{p}) - 1}$ und zwar für jedes i mit $1 \leq i \leq n$.
9. Der öffentliche Schlüssel ist $(\mathcal{K}, n, k, b_1, \dots, b_n)$ und der private Schlüssel ist $(K, g, d, \mathfrak{p}, p_1, \dots, p_n)$.

Verschlüsselung

Siehe 4.3.3.

Entschlüsselung

Der Schlüsseltext c kann leicht wieder entschlüsselt werden.

1. Berechne $r = (c - kd) \bmod (\mathcal{N}(\mathfrak{p}) - 1)$.
2. Berechne $u = g^r \pmod{\mathfrak{p}}$.
3. Finde m wie folgt: Wenn $p_i | u$ ist, dann setze $m_i = 1$. Ansonsten setze $m_i = 0$. Nach Anwendung dieses Verfahrens für alle p_i mit $1 \leq i \leq n$ wird $m = (m_1, m_2, \dots, m_n)$.
4. Entschlüssel m zu dem Klartext M wie folgt:
 - a) Setze $M = 0$ und $l = k$.
 - b) Für jedes i von 1 bis n tue das folgende: Wenn $m_i = 1$ ist, dann setze $M = M + \binom{n-i}{l}$ und $l = l - 1$.

4.3.6 Bemerkungen zu den einzelnen Varianten

Als erstes wird nun gezeigt, dass die Entschlüsselung korrekt funktioniert. Dann wird auf einige Feinheiten in den einzelnen Varianten der QPKCs eingegangen.

Die Entschlüsselung des Schlüsseltextes c erfolgt mit dem privaten Schlüssel. Dabei ist der wichtigste Schritt bei der Entschlüsselung, nachdem die Zahlen r und u berechnet werden, das Finden der Faktoren der Zahl u . Dass die paarweise teilerfremden Zahlen p_1, \dots, p_n Faktoren von u sind, lässt sich durch folgende Umformungen zeigen:

$$\begin{aligned}
 u &\equiv g^r \equiv g^{c-kd} \pmod{\mathfrak{p}} \\
 &\equiv g^{(\sum_{i=1}^n m_i b_i) - kd} \equiv g^{\sum_{i=1}^n m_i a_i} \pmod{\mathfrak{p}} \\
 &\equiv \prod_{i=1}^n (g^{a_i})^{m_i} \pmod{\mathfrak{p}} \\
 &\equiv \prod_{i=1}^n p_i^{m_i} \pmod{\mathfrak{p}} \\
 &= \prod_{i=1}^n p_i^{m_i},
 \end{aligned}$$

Diese Umformungen sind gültig, solange $\prod_{i=1}^n p_i^{m_i}$ dargestellt werden kann durch $a_1 + a_2\omega_2 + \dots + a_l\omega_l$ für die rationalen Zahlen a_1, a_2, \dots, a_l ($0 \leq a_i < e_i$). Da \mathcal{O}_K

nicht immer einen eindeutig Faktorisierungsbereich hat, wählt man p_1, \dots, p_n , so dass $\mathcal{N}(p_1), \dots, \mathcal{N}(p_n)$ paarweise teilerfremd sind. Daraus folgt, dass ein Produkt von p_1, \dots, p_n eindeutig faktorisiert werden kann, wenn man nur diese Elemente als Faktoren benutzt. Das bedeutet, ein Schlüsseltext kann eindeutig entschlüsselt werden, wenn das Produkt von p_1, \dots, p_n wieder korrekt zusammengesetzt wird.

Nun zu den Zahlkörpern. Ein typisches Beispiel für einen Zahlkörper, der die Sicherheit und Effizienz von OTU gewährleisten kann, ist laut Okamoto, Tanaka und Uchiyama eine Menge von quadratischen Körpern $\mathbb{Q}(\sqrt{D})$. Insbesondere die Menge der imaginär quadratischen Körper wird dringend für \mathcal{K} empfohlen (siehe Kapitel 4.3.4). Selbst in der Menge von Körpern $\mathcal{K} = \mathbb{Q}(\sqrt{-D})$ gibt es exponentiell viele Kandidaten.

Es soll jetzt auf einige spezielle Parameter eingegangen werden. Obwohl schon im Kapitel 4.3.4 auf die Wahl der Parameter im imaginär quadratischen Körper eingegangen wurde, soll hier noch ein anderer Weg beschrieben werden, wie man die Parameter p_1, \dots, p_n in einem allgemeineren Körpern wählen kann. Rationale Primzahlen p_1, \dots, p_n werden so ausgesucht, dass $\prod_{j=1}^k p_{i_j} < e_1 = p$ für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$ gilt. In diesem Fall existiert für jede Teilmenge $\{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$ von $\{p_1, p_2, \dots, p_n\}$ eine rationale Zahl a_1 ($0 < a_1 < e_1$), so dass $\prod_{j=1}^k p_{i_j} = a_1$. Das bedeutet $\prod_{j=1}^k p_{i_j}$ kann durch $\prod_{j=1}^k p_{i_j} = a_1 \omega_1 + \dots + a_l \omega_l$ dargestellt werden, wobei $a_2 = \dots = a_l = 0$ ist. Wenn man die HNF zur Darstellung der Primideale benutzt, so kann angenommen werden, dass $\omega_1 = 1$ und $e_1 > 1$ ist (Mehr Details in Abschnitt 4.7 und Übung 17 in [6]). Ein Fehler oder ein Mangel in diesem Fall ist, dass $\prod_{j=1}^k p_{i_j} < e_1 = p = (\mathcal{N}(\mathfrak{p}))^{1/f}$ ist, wenn $\mathcal{N}(\mathfrak{p}) = p^f$. In dem Fall, dass f größer als 1 ist, sollte die Dichte und Informationsrate kleiner sein. Es ist zu beachten, dass $\prod_{j=1}^k p_{i_j} \approx \mathcal{N}(\mathfrak{p})$ im Fall des imaginär quadratischer Körper gilt.

Es werden nun die Dichte und Informationsrate für den imaginär quadratischen Fall betrachtet. Die Größe der b_i ist $|b_i| = |\mathcal{N}(\mathfrak{p})|$, $k \times |\mathcal{N}(p_i)| \approx |\mathcal{N}(\mathfrak{p})|$, und $|\mathcal{N}(p_i)| \approx 2 \log n$. Also erhält man $|b_i| \approx |\mathcal{N}(\mathfrak{p})| \approx 2k \log n$. Demnach wird die Dichte des Systems auf $D = \frac{n}{2k \log n}$ und die Informationsrate auf $R = \frac{k \log n - k \log k}{2k \log n}$ geschätzt. Wenn $k = 2^{(log n)^c}$ für jede Konstante $c < 1$ ist, dann geht die Informationsrate R asymptotisch gegen $1/2$ und die Dichte D asymptotisch gegen ∞ .

5 Wahl der kryptographischen Schlüsselgrößen

In diesem Kapitel wird betrachtet, wie die Schlüsselgrößen von Kryptosystemen zu wählen sind, damit die Kryptosysteme als sicher gelten können. Eine wichtige Arbeit über die Wahl von Schlüsselgrößen stammt aus dem Jahr 1999 von Arjen K. Lenstra und Eric R. Verheul [14], deren Richtlinien hier zur Anwendung kommen sollen. Zunächst wird erklärt, wie diese Richtlinien zustande kommen; anschließend wird auf die Wahl der Schlüsselgrößen eingegangen.

5.1 Einführung

Für die Wahl der kryptographischen Schlüsselgrößen ist unter anderem auch die zeitliche Dauer eines Angriffes wichtig und damit die Laufzeit eines Rechners zur Berechnung beziehungsweise zum Brechen eines Schlüssels. Alle Laufzeiten von Lenstra und Verheul basieren auf der realen beziehungsweise geschätzten Laufzeit eines 450MHz Pentium II Prozessors. Deshalb sind die Empfehlungen von Lenstra und Verheul nur als untere Schranken zu verstehen, die für die heutigen Systeme erhöht werden müssen. Die Rechenzeit wird mittels Mips- (Million instructions per second) Jahren gemessen. Ein Mips-Jahr ist definiert als die Anzahl der Berechnungen, die ein einzelner DEC VAX 11/780 Prozessor in einem Jahr berechnen kann. Diese Maßeinheit wurde oft kritisiert, weil es unklar ist, wie diese Maßeinheit für Prozessoren angewandt werden kann, die einen anderen Befehlssatz haben als der VAX Prozessor. Trotzdem wird diese Maßeinheit aufgrund ihrer hohen Akzeptanz verwendet. Als Konvention gilt: Ein Berechnungsjahr auf einem wie oben definierten PC ist gleich zu 450 Mips-Jahren. Es ist zu beachten, dass alle Schätzungen auf der Laufzeit eines PC beruhen und nicht auf der obigen Definition von Mips-Jahren. Die Definition ist vergleichbar mit anderen Definitionen von Mips-Jahren. Die Abkürzung MMY steht für eine Million Mips-Jahre. Die Richtlinien werden als untere Schranken für Schlüsselgrößen verstanden und zwar im Sinne von gleich zu oder größer als die empfohlene Größe für eine bestimmte Sicherheitsstufe. Vom Sicherheitsstandpunkt ist es besser, größere Schlüssel zu verlangen als zu diesem Zeitpunkt benötigt werden. Die meisten Schlüsselrichtlinien bei Lenstra und Verheul unterschätzen daher systematisch den Berechnungsaufwand für einen erfolgreichen Angriff. Deshalb scheinen die Schlüssel schwächer, als sie in der Realität sind. In einigen Fällen werden

die Angriffserfolge überschätzt, hier helfen aber andere Faktoren, ein gewünschtes Sicherheitslevel zu erreichen.

Lenstra und Verheul beziehen sich bei ihrer Untersuchung auf die kryptographischen Primitive der Wassenaar Vereinbarung. Die Wassenaar Vereinbarung ist eine Fortschreibung der COCOM (Coordinating Committee for Multilateral Export Controls) Regelungen. Es ist eine internationale Vereinbarung zwischen den Mitgliedstaaten – beispielsweise den europäischen Ländern und den USA – über die Beschränkungen bei Exportprodukten. Darunter fällt auch der Export kryptographischer Produkte (Primitive). Es soll dadurch sichergestellt werden, dass die nationale Sicherheit eines Landes nicht gefährdet wird. Dabei werden die Schlüsselgrößen von kryptographischen Produkten auf eine bestimmte Größe beschränkt, die dann exportiert werden dürfen, ohne dass dazu eine Lizenz benötigt wird. Es ist natürlich klar, dass die Beschränkungen der Wassenaar Vereinbarung bei den meisten kommerziellen Produkten nicht den nötigen Schutz zulassen. Bei der Wahl der Schlüsselgröße werden nur zwei kommunizierende Partner (Sender und Empfänger) betrachtet, die eine vertrauliche Unterhaltung führen wollen. OTU ist ein asymmetrisches Kryptosystem, so dass man auf den ersten Blick auf die Idee kommen könnte, dass der private Schlüssel von OTU nach der Wassenaar Vereinbarung nur 512 Bits groß sein darf. Die Größe des öffentlichen Schlüssels ist hierbei unerheblich. Allerdings ist OTU nicht für die Wassenaar Vereinbarung betrachtet worden. Da die Sicherheit von OTU auf einem Knapsack-Problem beruht und noch keine Quantencomputer existieren, ist unklar, wie groß die Schlüsselgröße von OTU sein muss, um das Kryptosystem vor Angriffen zu schützen, und welche Schlüsselgröße die Wassenaar Vereinbarung zulassen würde. Daher soll hier die Methodik von Lenstra und Verheul etwas genauer betrachtet werden.

Lenstra und Verheul gehen bei ihrer Schlüsselgrößenbestimmung auf einige Angriffstypen für die verschiedenen Kryptosysteme ein und beziehen diese auch in ihr Modell ein. Da allerdings keiner der betrachteten Angriffe wirklich eine ernstzunehmende Gefahr für OTU darstellt, werden sie hier auch nicht aufgeführt. Eine genaue Betrachtung der Angriffe gegen OTU findet in dem Kapitel 6 über die Sicherheit von OTU statt. Es sei hier noch ein Angriff erwähnt, da er später bei den Berechnungen von Lenstra und Verheul eine Rolle spielt. Der Faktorisierungsalgorithmus, der maßgeblich benutzt wird, ist der Number Field Sieve (NFS) Algorithmus, der ein RSA Modul n von 512 Bit in einer Laufzeit von weniger als 10^4 Mips-Jahren faktorisiert. Nach heuristischen Grundlagen benötigt der NFS eine Zeit proportional zu

$$L[n] = e^{(1.9229 + o(1))} * \ln(n)^{1/3} * \ln(\ln(n))^{2/3},$$

wobei $o(1)$ gegen 0 geht, sobald n gegen unendlich geht. Diese Laufzeit nennt man teilexponentiell zu n , weil, wenn n gegen unendlich geht, die Laufzeit $L[n]$ weniger als n^c für jedes $c > 0$ ist. Der Speicher, der von NFS benötigt wird, ist proportional zu $\sqrt{L[n]}$. Die Laufzeit $L[n]$ kann aber nicht direkt genutzt werden,

um auf die Anzahl der Operationen zu schließen, die nötig sind, um die Faktorisierung beziehungsweise den diskreten Logarithmus (Wenn p eine Primzahl ist, dann gibt es eine Variante des NFS (DLNFS), die den diskreten Logarithmus in einem endlichen Feld F_p berechnet.) zu berechnen. Aber man kann sie für eine beschränkte Extrapolation nutzen. Wenn man aus Experimenten weiß, dass die Faktorisierung von n mit dem NFS die Zeit t benötigt, dann wird die Faktorisierung einer Zahl m mit $m > n$ ungefähr die Zeit $t * L[m]/L[n]$ benötigen. Natürlich nur, wenn man die $o(1)$ Konstanten auslässt und die Größen m und n nicht zu weit auseinander liegen. Liegen die Werte für m und n zu weit auseinander, dann kann man den Effekt der $o(1)$, die gegen 0 gehen, nicht mehr außer Acht lassen und $t * L[m]/L[n]$ wird um den Zeitfaktor m zu hoch bewertet. Gleiches gilt für die Zeitextrapolation von DLNFS.

Spezielle Hardware für die Angriffe gegen OTU gibt es zur Zeit nicht. Anders als bei anderen asymmetrischen Kryptosystemen sind mögliche Quantencomputer mit ihrer Fähigkeit zur Faktorisierung keine Gefahr für OTU.

Auch ein Angriff durch Raten des Schlüssels bei aktuellen Schlüsselgrößen ist nicht sehr vielversprechend. Unter Raten versteht man das Erraten der Bits eines privaten Schlüssels beziehungsweise eines Teils von ihm. Bei einem privaten RSA Schlüssel mit einer Schlüsselgröße von 512 Bits werden inzwischen unter gewissen Vorbedingungen nur noch 128 richtig geratene Bits benötigt [15]. Die Schlüssel von RSA mit ihren Schlüsselgrößen nach dem Jahr 2000 gelten für Lenstra und Verheul als unmöglich zu erraten. Bei OTU gestaltet sich das Erraten der Bits schon bei dem einfachen Modell von OTU (siehe Kapitel 4.3.3) um einiges schwieriger. Neben den Größen p und g sind zudem die teilerfremden Zahlen p_i zu erraten. Dies ist ohne Hinweise so gut wie unmöglich.

5.2 Das Modell von Lenstra und Verheul

Die Wahl kryptographischer Schlüssellängen hängt primär von den folgenden Punkten ab:

1. *Lebensdauer*: Die Lebensdauer ist die erwartete Zeit, in der die Informationen geschützt sein müssen.
2. *Sicherheitsgrad*: Der Sicherheitsgrad ist die Erwartung an den Aufwand eines erfolgreichen Angriffes.
3. *Computerausstattung*: Die Computerausstattung betrifft die erwarteten Entwicklungen im Hardwarebereich, die für Angreifer verfügbar sind.
4. *Kryptoanalyse*: Die Kryptoanalyse bezieht sich auf die erwarteten Entwicklungen in diesem Bereich.

Überlegungen bezüglich der Effizienz und dem Speicher mögen die Wahl von Schlüsselgrößen beeinflussen, werden hier aber – solange sie nicht sicherheitsrelevant werden – vernachlässigt.

5.2.1 Lebensdauer

In der Tabelle 5.1 im nächsten Abschnitt werden Schlüsselgrößen für asymmetrische Kryptosysteme vorgeschlagen, die von der erwartenden Lebensdauer der kryptographischen Anwendung abhängen. Dabei liegt es im Ermessen des Betrachters, bis für welches Jahr die Sicherheit gewährleistet sein soll oder welchem Sicherheitsmaß (kurzfristige, mittelfristige oder langfristige Sicherheit) die erwartete Lebensdauer entspricht.

5.2.2 Sicherheitsgrad

Ein Kryptosystem wird nur dann als sicher angenommen, wenn ein erfolgreicher Angriff als praktisch unmöglich angesehen wird. Unglücklicherweise ist es schwer zu beschreiben, was man unter praktisch unmöglich versteht. So könnte man beispielsweise entscheiden, dass eine Schlüsselgröße für ein Kryptosystem sicher für eine aktuelle Anwendung ist, wenn es zu brechen 10^6 mal schwieriger wäre als die größte Schlüsselgröße, die aktuell für dieses Kryptosystem gebrochen werden kann. Es gibt verschiedene Probleme mit diesem Ansatz. Als erstes ist 10^6 willkürlich gewählt. Zweitens gibt es keinen Grund zu glauben, dass die größte gebrochene Schlüsselgröße das Beste ist, was man aktuell tun kann. Drittens gibt es für manche kryptographischen Primitive gar keine Daten oder sie sind veraltet, so dass dies kein guter Ansatz ist. Lenstra und Verheul haben sich daher für einen anderen Ansatz entschieden und stellen im folgenden Hypothesen auf, aus denen sie dann ihre Schlüsse ziehen.

Hypothese I: Als Basis der Extrapolation wird angenommen, dass das Kryptosystem DES (Data Encryption Standard) bis 1982 ausreichend sicher für kommerzielle Anwendungen war. Daher wird angenommen, dass im Jahr 1982 eine Computerleistung von 0.5 MMY aufgebracht werden musste, um eine kommerzielle DES Anwendung mit einem Software-Angriff zu brechen. Für Hardware-Angriffe nimmt man die “\$50 Millionen und 2 Tage“ DES Schlüsselsuchmaschine von 1980 nicht als ernstzunehmende Bedrohung für kommerzielle DES Anwendungen bis 1982 an. Unter einer “\$50 Millionen und 2 Tage“ DES Schlüsselsuchmaschine versteht man, dass die Kosten für die spezielle Hardware dieser Maschine bei \$50 Millionen lagen und nur noch 2 Tage benötigt wurden, um einen DES Schlüssel zu brechen. Weil \$50 Millionen und 2 Tage im Jahre 1980 kein unmögliches Hindernis für gewisse Organisationen wie beispielsweise Geheimdienste waren, wird hier von kommerziellen Anwendungen gesprochen. Daraus folgt, dass man den Standardwert für den Sicherheitsgrad s wie folgt setzen kann: $s = 1982$.

5.2.3 Computerausstattung

Hypothese II: Zur Abschätzung der Computerleistung, die Angreifern im Laufe der Zeit zur Verfügung steht, wird das Gesetz von Moore benutzt. Moores Gesetz lautet, dass sich die Komponentendichte alle 18 Monate verdoppelt. Eine weit verbreitete Interpretation dieses Gesetzes ist, dass sich die Computerleistung pro Chip alle 18 Monate verdoppelt. Natürlich ist nicht zu sagen, ob das Gesetz immer gelten wird, da immer wieder neue Technologien benötigt werden, um es einhalten zu können. Daher wird eine leichte Variation des Mooreschen Gesetzes angenommen, die weniger Technologie abhängig und trotzdem ausreichend genau ist: Jede 18 Monate verdoppeln sich die Prozessorgeschwindigkeit und die Speichergröße, die man für einen Dollar bekommt. Daraus folgt, dass man für die gleichen Kosten einen um den Faktor $2^{10+12/18} \approx 100$ mal mehr Rechenleistung und schnelleren Speicher alle 10 Jahre bekommt, entweder in Form von Software auf Multifunktionschips (PCs) oder als spezielle Hardware.

Mittels eines Beispiels soll diese Hypothese verdeutlicht werden. Es ist nicht unrealistisch anzunehmen, dass eine billigere und langsamere Version der “\$50 Millionen und 2 Tage“ DES Schlüsselsuchmaschine eine “\$1 Million und 100 Tage“ Maschine wäre, die 50 mal langsamer wäre und weniger Hardware zur Verfügung hätte. Betrachten wir diese Maschine für das Jahr 2005 und 2006 statt für das Jahr 1999 wie Lenstra und Verheul, so würde die \$1 Millionen Maschine im Jahr 2005 83 Sekunden und im Jahr 2006 nur noch 52 Sekunden benötigen, um einen DES Schlüssel zu suchen. Die Berechnung des Ganzen ist einfach. Zwischen dem Jahr 1980 und 2005 liegen 25 Jahre oder anders ausgedrückt $25 \cdot 12$ Monate. Nach dem Mooreschen Gesetz wäre die \$1 Millionen Maschine im Jahr 2005 $2^{16.67}$ mal schneller, weil $25 \cdot 12 = 18 \cdot 16.67$ ist. Da $2^{16.67} \approx 104272$ ist, würde die Maschine aus dem Jahr 2005 nur $100/104272$ Tage benötigen. Rechnet man die $100/104272$ Tage in Sekunden um, so erhält man 83 Sekunden. Die gleiche Rechnung für das Jahr 2006 führt zu den 52 Sekunden von oben.

Hypothese III: Natürlich impliziert die oben benutzte Variation des Mooreschen Gesetzes den Verfall des Geldwertes. Das US Gross National Product (GNP; entspricht dem deutschen Bruttosozialprodukt und wurde inzwischen durch das Gross National Income (GNI) ersetzt) zeigte den Trend des Verdoppelns alle 10 Jahre: \$1630 Billionen im Jahr 1975 gemessen im Dollarwert von 1975, \$4189 Billionen im Jahr 1985 gemessen im Dollarwert von 1985 und \$7269 Billionen im Jahr 1995 gemessen im Dollarwert von 1995. Für das Jahr 2005 wird das US GNP auf \$11391 Billionen geschätzt. Für das Jahr 2004 betrug der GNP \$12150 Billionen. Somit sollte die Schätzung für das Jahr 2005 realistisch sein. Das führt zu der Hypothese, dass sich das Budget von Organisationen, auch von denen die kryptographische Schlüssel brechen wollen, alle 10 Jahre verdoppelt.

Wenn man alle obigen drei Hypothesen kombiniert, so kann man folgendes fest-

stellen. Wenn im Jahr 1982 eine Rechenleistung von 0.5 MMY als unmöglich angesehen wurde, um eine kommerzielle kryptographische Anwendung zu brechen, so sind $100 (\approx 2 * 100 * 0.5)$ MMY unmöglich für das Jahr 1992. Ebenso sind $2 * 10^4 (\approx 200 * 100)$ MMY unmöglich im Jahr 2002 und $4 * 10^6$ MMY unmöglich für das Jahr 2012. Diese Zahlen stimmen auch mit denen von Odlyzko überein, die auf der Rechenleistung basieren, die aus dem Internet verfügbar wären [19].

5.2.4 Kryptoanalyse

Hypothese IV: Es ist unmöglich vorausszusagen, welche kryptographischen Entwicklungen es geben wird oder welche schon heimlich stattgefunden haben. Daher ist es als realistisch anzunehmen, dass das Tempo von veröffentlichten kryptographischen Ergebnissen und ihr Einfluss sich nicht dramatisch ändern werden, verglichen mit dem, was zwischen 1970 und 1999 passiert ist. Für klassische asymmetrische Systeme ist der Effekt von kryptographischen Entwicklungen ähnlich zu dem Mooreschen Gesetz: Alle 18 Monate werden die Angriffe auf die gleichen asymmetrischen Systeme nur noch die Hälfte der Rechenleistung benötigen wie heutzutage.

5.2.5 Bemerkungen

Die von Lenstra und Verheul vorgeschlagenen Schlüsselgrößen im nächsten Abschnitt sind eine Kombination aus den Hypothesen I - IV mit den auf Software-Angriffen basierenden Datenpunkten in Mips-Jahren. Es soll hier erläutert werden, wie die Ergebnisse von Lenstra und Verheul interpretiert werden können.

Lenstra und Verheul betrachten für die Bestimmung der Schlüsselgrößen neben den verschiedenen Angriffen auf die unterschiedlichen Kryptosysteme auch den Unterschied zwischen Software-Angriffen und speziellen gezielten Hardware-Angriffen. Da allerdings Lenstra und Verheul für die klassischen asymmetrischen Systeme keine speziellen gezielten Hardware-Angriffe annehmen, wird dies hier auch nicht genauer beleuchtet. Soviel sei dazu nur gesagt, die empfohlenen Schlüsselgrößen von Lenstra und Verheul bieten Sicherheit gegen Software- als auch gegen Hardware-Angriffe und sind äquivalent zu der Sicherheit von DES im Jahr 1982.

Um die anfallende Kosten der verschiedenen Systeme mit einbeziehen zu können, haben sich Lenstra und Verheul entschlossen, den Berechnungsaufwand als Basis für ihre Resultate zu wählen. Darunter kann man auch die Anzahl von Mips-Jahren verstehen, die für einen erfolgreichen Angriff benötigt werden. Lenstra und Verheul haben sich für den Berechnungsaufwand entschieden, weil zum einen die Software-Angriffe im wesentlichen kostenlos sind, da die Hardware meist schon vorhanden ist. Zum anderen fehlen Analysen zu den finanziellen Kosten der verschiedenen Angriffstypen, um präzise Kostenunterschiede machen zu können. Eine Analyse zu der Kostengleichheit eines bestimmten Angriffstyps ist abhängig

von subjektiven Einschätzungen bzw. Entscheidungen des Analyisten, so dass die Ergebnisse davon entscheidend beeinflusst wären. Hinzukommt, dass diese Einschätzungen sich im Laufe der Zeit ändern können. Mit Kostengleichheit ist gemeint, dass die Hardware zweier unterschiedlicher Systeme für einen erfolgreichen Angriff ungefähr gleich viel kostet und die Zeitdauer des Angriffes auch ungefähr gleich lang ist. Trotzdem skizzieren beide Autoren, was bei Kostengleichheit herausgekommen wäre.

Lenstra und Verheul gehen von Zeitungsanzeigen aus, wenn sie den Preis für einen voll ausgestatteten PC zwischen \$0 und \$450 beziffern. Diese "kostenlosen" Maschinen unterstützen den Blickpunkt, dass Software-Angriffe kostenlos seien. Nimmt man an, dass ein zerlegter PC (bspw. Motherboard mit Kommunikations-Hardware) für \$100 zu kaufen wäre. Dann würde daraus folgen, dass eine "\$81 Millionen und 1 Tag, 1999"-Hardware gleichbedeutend mit mindestens einer Millionen Software Mips-Jahren wäre. Verglichen mit einer "\$12500, 1 Tag, 1996" exhaustive Search-Hardware, was einer "\$31000, 1 Tag, 1999"-Hardware entspricht, ist 1 Software Mips-Jahre etwa 2500 mal teurer. Unter einer exhaustive Search wird eine Schlüsselsuche verstanden, die erst endet, wenn der gesuchte Schlüssel gefunden ist. Dabei wird jede mögliche Bitkombination des Schlüssel ausprobiert.

Es folgt, dass für die Zwecke von Lenstra und Verheul ein Software Mips-Jahr etwa 2500 mal teurer ist als ein Mips-Jahr für eine speziell gezielte Hardware. In Kapitel 5.3.4 wird gezeigt, wie der Faktor 2500 benutzt wird, um aus den Ausstattungskosten äquivalente Schlüsselgrößen abzuleiten. Man beachte, dass der Faktor 2500 mit großer Vorsicht betrachtet werden sollte, da es sich hierbei um rein heuristische Werte handelt, die auf einen geschätzten Preis für zerlegte PCs und der angenommenen Unwahrscheinlichkeit eines speziellen gezielten Hardware-Angriffes auf RSA basieren.

Noch ein paar Bemerkungen zu den Hypothesen: Es ist klar, dass nicht jeder den Hypothesen von Lenstra und Verheul zustimmt. Vor allem über die erste Hypothese kann man diskutieren. Man beachte, dass nicht angenommen wird, dass DES 1977 oder 1982 unbrechbar war. Es wird nur angenommen, dass er ausreichend Sicherheit für kommerzielle Anwendungen bot, und nicht, dass es für zahlungskräftige Regierungsbehörden unmöglich war, DES 1977 zu brechen. Der endgültige Beweis, dass DES unsicher ist, wurde im Jahre 1998 von der Electronic Frontier Foundation angetreten. Die Gesellschaft konnte mittels spezieller Hardware einen DES Schlüsseltext innerhalb von 56 Stunden brechen. Als Ersatz wurde dann Triple-DES genutzt, der anschließend im Jahr 2000 von Advanced Encryption Standard (AES) abgelöst wurde. Auf jeden Fall kann jeder, der denkt, dass die Unwahrscheinlichkeitsannahme für das Jahr 1982 für DES zu schwach oder zu stark sei, trotzdem die Annäherung von Lenstra und Verheul benutzen. In Kapitel 5.3.3 wird erklärt, wie das funktioniert.

Lenstra und Verheul gehen auch nicht davon aus, dass jeder der zweiten Hypothese zustimmt. Manche argumentieren, dass das Gesetz von Moore nicht mehr lange hält, andere finden das Gesetz von Moore für große Maschinen zu pessimi-

stisch. Hypothese II ist folglich für Lenstra und Verheul ein realistischer Kompromiss. Allerdings scheint sich die Geschwindigkeit des Mooreschen Gesetzes seit der Arbeit von Lenstra und Verheul zu verlangsamen. Das hat zwei Gründe. Erstens wird allmählich die technische Grenze erreicht, bei der ein Transistor nur noch aus wenigen Atomen besteht, so dass eine weitere Verkleinerung der Bauteile nicht mehr möglich ist. Zweitens wächst der finanzielle Aufwand zur Entwicklung und Herstellung der Bauteile bald schneller als die Komponentendichte. Daher würden sich Investitionen nicht mehr lohnen. In einigen Bereichen der Halbleiterindustrie hat die Kurve von Moores Gesetz schon eine Sättigung erfahren[29]. Sieht man das Gesetz von Moore so wie Lenstra und Verheul als eine Verdopplung der Rechenleistung, so könnte das Gesetz weiterhin halten, wenn die jetzige Technologie durch eine neue leistungsfähigere abgelöst werden würde. Bisher wurden immer alle Technologiesprünge realisiert, bevor sie für das Mooresche Gesetz nötig wurden. Die heutigen Chips haben eine Struktur von 130 bis 90 nm (Nanometer) und es existieren erste Prototypen von Transistoren mit einer Gatelänge von nur 10 nm. Zum Vergleich: Ein Transistor von 90 nm hat eine Gatelänge von ungefähr 50 nm und ist circa so groß wie ein Influenzavirus. [31].

Auch Hypothese IV sollte man kritisch betrachten. Die Entwicklungen in der Kryptoanalyse sind doch eher als sprunghaft anzusehen. So kann diese These schon bei einer genialen Idee außer Kraft gesetzt werden. Die Entwicklungen seit 1999 sind immer noch so sprunghaft wie damals und lassen sich genauso wenig vorhersagen. So wäre es ein riesiger Sprung in der Kryptoanalyse, wenn eine Faktorisierung für RSA oder ähnliche Systeme gefunden würde, die nur noch die Hälfte der Zeit benötigt. Jeder, der sich mit dieser Thematik beschäftigt, ist sich der Tatsache bewusst, dass ein neuer Faktorisierungsalgorithmus einige der bisherigen Kryptosysteme stark in Frage stellen kann. Somit bleibt diese Hypothese nur eine Hypothese. Darüber waren sich Lenstra und Verheul im Klaren. Auch heutzutage gibt es deshalb keine bessere Abschätzung als die Hypothese der beiden.

5.3 Ergebnisse von Lenstra und Verheul

5.3.1 Methoden zur Berechnung

Zur Berechnung der Schlüsselgröße für ein bestimmtes Jahr y geht man wie folgt vor. Zuerst berechnet man für das Jahr y den Wert $IMY(y)$. $IMY(y)$ ist die Anzahl von Mips-Jahren, die für das Jahr y als unmöglich zu berechnen gelten und zwar basierend auf den Hypothesen I - III:

$$IMY(y) = 0.5 * 10^6 * 2^{2(y-1982)/3} * 2^{(y-1982)/10}.$$

Das Ergebnis wird benutzt, um davon die Schlüsselgröße abzuleiten, die ausreichende Sicherheit für das entsprechende kryptographische Primitiv bis zu diesem Jahr geben soll.

Für klassische asymmetrische Systeme wird eine asymptotische Laufzeit $L[n]$ von NFS (unter Auslassung der $o(1)$, siehe 5.1) mit dem Datenpunkt kombiniert, dass ein 512-Bit Schlüssel im Jahre 1999 mit den Kosten von weniger als 10^4 Mips-Jahren gebrochen werden konnte, und der Hypothese IV, die einen kryptoanalytischen Prozess à la Moore erwartet. Daraus kann man die kleinste Schlüsselgröße s festlegen, so dass

$$L[2^s] * 10^4 \geq L[2^{512}] * 2^{2(y-1999)/3} * IMY(y)$$

ist. Trotzdem sollte die Schlüsselgröße zum Beispiel von RSA größer sein als die, die in der Tabelle 5.1 angegeben wird, und zwar aus zwei Gründen. Zum einen ist der Datenpunkt für die Kosten der Faktorisierung eines 512-Bit Schlüssels zu hoch bewertet und zum anderen wird durch Auslassung der $o(1)$ die Schwierigkeit überschätzt, mit der ein klassisches asymmetrisches System mit der Schlüsselgröße von s gebrochen werden kann. Lenstra und Verheul haben nicht versucht, dieses Probleme zu korrigieren, weil es als eine realistische Kompensation des zweiten Berechnungsschrittes von NFS betrachtet werden kann (siehe [14]).

5.3.2 Benutzung der Ergebnis-Tabelle

Zunächst müssen zwei Bemerkungen zur Tabelle gemacht werden. Die Daten in der Tabelle ändern sich kaum merklich, wenn der von Lenstra und Verheul eingesetzte Datenpunkt “512-Bit, 10^4 Mips-Jahre, 1999“ gegen einen anderen ausgetauscht wird. Selbst wenn man den neusten Datenpunkt von RSA-200 einsetzt, der im Mai 2005 geknackt wurde, ändern sich die Schlüsselgrößen nicht signifikant. Die Schlüsselgrößen weichen von denen von Lenstra und Verheul nur um 6 Bit (für das Jahr 2005) bis maximal 13 Bit (für das Jahr 2050) ab. Der RSA Schlüssel, der bei RSA-200 benutzt wurde, hat dabei eine Länge von 663 Bit und benötigt ungefähr 120000 Mips-Jahre [9, 3]. Die neuen Schlüsselgrößen nach dem Datenpunkt von RSA-200 sind von mir in die ursprüngliche Tabelle von Lenstra und Verheul hinzugefügt worden und finden sich in der zweiten Spalte wieder.

Die Austauschbarkeit der Datenpunkte bestätigt nach Lenstra und Verheul auch die Entscheidung, ein Moore ähnliches Gesetz für die kryptographische Entwicklung anzunehmen, die die klassischen asymmetrischen Systeme beeinflusst.

Wenn man mit den Hypothesen übereinstimmt, so kann man die Tabelle 5.1 wie folgt benutzen: Angenommen man entwickelt eine kommerzielle Anwendung, in der die Vertraulichkeit oder Integrität von elektronischen Daten für die nächsten 20 Jahre garantiert sein muss, beispielsweise bis ins Jahr 2020. Dann findet man in der Tabelle in der Zeile für das Jahr 2020 den Eintrag, dass die Berechnungsanzahl von $2.9 * 10^{14}$ Mips-Jahren im Jahr 2020 als unmöglich anzusehen ist, genauso wie $0.5 * 10^6$ es im Jahre 1982 war. Das berechenbare Sicherheitsäquivalent (siehe Kapitel 5.3.3) zu der Sicherheit, die durch DES im Jahre 1982 erzielt wurde, ist im Jahr 2020 ein privater Schlüssel mit mindestens 1881 Bits Schlüssellänge. Die Bedeutung des zweiten Eintrags in Klammer ‘1472’ wird in Kapitel 5.3.4 erklärt.

Tabelle 5.1: Untere Schranken für asymmetrische Schlüsselgrößen aus [14]

Jahr	Klassische asymmetrische Schlüsselgröße		$IMY(y)^1$	Untere Schranke der Hardware-Kosten in US \$ für einen Angriffstag	Entsprechende Anzahl von Jahren auf einem 450MHz PentiumII PC
	nach Lenstra und Verheul (SDL Feldgröße)	nach neuem Datenpunkt RSA-200			
1982	417 (288)		$5.00 * 10^5$	$3.97 * 10^7$	$1.11 * 10^3$
1985	488 (320)		$2.46 * 10^6$	$4.90 * 10^7$	$5.47 * 10^3$
1990	622 (448)		$3.51 * 10^7$	$6.93 * 10^7$	$7.80 * 10^4$
1995	777 (544)		$5.00 * 10^8$	$9.81 * 10^7$	$1.11 * 10^6$
2000	952 (704)		$7.13 * 10^9$	$1.39 * 10^8$	$1.58 * 10^7$
2005	1149 (864)	1155	$1.02 * 10^{11}$	$1.96 * 10^8$	$2.26 * 10^8$
2006	1191 (896)	1197	$1.73 * 10^{11}$	$2.10 * 10^8$	$3.84 * 10^8$
2007	1235 (928)	1241	$2.94 * 10^{11}$	$2.25 * 10^8$	$6.54 * 10^8$
2008	1279 (960)	1285	$5.01 * 10^{11}$	$2.41 * 10^8$	$1.11 * 10^9$
2009	1323 (1024)	1330	$8.52 * 10^{11}$	$2.59 * 10^8$	$1.89 * 10^9$
2010	1369 (1056)	1376	$1.45 * 10^{12}$	$2.77 * 10^8$	$3.22 * 10^9$
2015	1613 (1248)	1620	$2.07 * 10^{13}$	$3.92 * 10^8$	$4.59 * 10^{10}$
2020	1881 (1472)	1888	$2.94 * 10^{14}$	$5.55 * 10^8$	$6.54 * 10^{11}$
2025	2174 (1728)	2182	$4.20 * 10^{15}$	$7.84 * 10^8$	$9.33 * 10^{12}$
2030	2493 (2016)	2503	$5.98 * 10^{16}$	$1.11 * 10^9$	$1.33 * 10^{14}$
2035	2840 (2336)	2850	$8.53 * 10^{17}$	$1.57 * 10^9$	$1.90 * 10^{15}$
2040	3214 (2656)	3224	$1.22 * 10^{19}$	$2.22 * 10^9$	$2.70 * 10^{16}$
2045	3616 (3008)	3627	$1.73 * 10^{20}$	$3.14 * 10^9$	$3.85 * 10^{17}$
2050	4047 (3392)	4060	$2.47 * 10^{21}$	$4.44 * 10^9$	$5.49 * 10^{18}$

5.3.3 Alternative Sicherheitsgrade

Nach der Hypothese I stellt DES genügend Sicherheit für kommerzielle Anwendungen bis in Jahr 1982 zur Verfügung, aber nicht über das Jahr 1982 hinaus. Für Unternehmen, die DES über 1982 hinaus oder sogar bis Ende der späten 90er Jahre benutzt haben, sind die Unmöglichkeitsschätzung von 0.5 MMY für das Jahr 1982 möglicherweise zu stark. Für andere zu schwach. Es wird nun beschrieben, wie Tabelle 5.1 für Schlüsselgrößen für das Jahr y genutzt werden kann, wenn jemand DES bis zum Jahr $1982 + x$ traut, wobei x negativ ist für den Fall, dass die Unmöglichkeitsschätzung als zu schwach eingeschätzt wird, und positiv

¹ $IMY(y)$ ist die Anzahl von Mips-Jahren, die für das Jahr y als unmöglich zu berechnen gelten.

für den umgekehrten Fall. Beispielsweise für $y = 2005$ und $x = 13$, falls jemand DES bis in Jahr 1995 vertraut hat: Für die klassischen asymmetrischen Schlüssel wird der Eintrag für das Jahr $y - 23 * x/43$ genommen. Also für das Beispiel: $2005 - 23 * 13/43 \approx 1998$. So sollte ein 879 Bit Schlüssel benutzt werden. Die Korrektheit dieser Methode folgt leicht aus den Formeln in Kapitel 5.3.1.

5.3.4 Ausstattungskosten äquivalent zur Schlüsselgröße

Angenommen, ein zerlegter PC kostet \$100 und der Ergebnisfaktor von 2500 ist akzeptabel, so kann die Tabelle 5.1 die Ausstattungskosten äquivalent zur Schlüsselgröße in folgender Art festlegen: Eine untere Schranke für die Ausstattungskosten für einen erfolgreichen Tagesangriff wird in der zweitletzten Spalte der Tabelle angegeben und zwar im Jahr y in Dollar vom Jahr y .

Für klassische asymmetrische Systeme sind Mips-Jahre angeblich 2500 mal teurer. Dies ist für Lenstra und Verheul nur Berechnungszweck, äquivalent zu der Annahme, dass DES eine akzeptable Sicherheit bis 1997 anbietet, da $1997 - 1982 = 15$ und $2^{(15*23/30)}$ nahe an 2500 ist. Benutzt man die Überlegungen aus Kapitel 5.3.3, so können klassische asymmetrische Schlüsselgrößen für das Jahr y , die von den Ausstattungskosten äquivalent zu symmetrischen und EC (Elliptische Kurven) Schlüsselgrößen sind, in der Tabelle in der Spalte für klassische asymmetrische Schlüsselgrößen für $y - (23 * 15)/43 = y - 8$ gefunden werden. Die sich ergebenden Schlüsselgrößen, aufgerundet zu dem nächsten Vielfachen von 32, werden als zweiter Eintrag in Klammern in der Spalte für klassische asymmetrische Schlüsselgrößen in der Tabelle angegeben. Um solche Schlüssel zu brechen, wird eine wesentlich kleine Anzahl von Mips-Jahren benötigt als die IMY für das Jahr y , aber die benötigten Mips-Jahre werden als unbezahlbar angenommen.

Eine ähnliche offene Analyse kann jeder zu anderen PC Preisen durchführen. Beispielsweise für ein \$10 oder \$1000 PC sollte $y - 8$ durch ein $y - 6$ beziehungsweise $y - 10$ ausgetauscht werden.

5.3.5 Konsequenzen der Tabelle für OTU

Zur Festlegung einer Schlüsselgröße für eine OTU Anwendung ist es wichtig zu wissen, welche Daten mit der Anwendung geschützt werden sollen. Je nach Höhe des Sicherheitsgrades kann man nun eine Schlüsselgröße für OTU mit Hilfe der Tabelle von Lenstra und Verheul wählen. Im nächsten Kapitel und im praktischen Teil dieser Arbeit wird geklärt, welche Schlüsselgrößen sinnvoll für eine OTU Anwendung sein könnten und wie sicher diese dann wirklich sind.

6 Angriffe auf OTU

In diesem Kapitel sollen die verschiedenen Angriffsmöglichkeiten auf das OTU QPKC System und ihre Erfolgsaussichten vorgestellt werden. Die Informationen dafür kommen aus den Papier von Okamoto, Tanaka und Uchiyama [20] sowie aus den Quellen [7, 12, 21].

6.1 Allgemeine Angriffsmöglichkeiten

6.1.1 Angriff auf privaten Schlüssel

Bevor die Angriffsmöglichkeiten auf den privaten Schlüssel erörtert werden, hier zur Erinnerung: Der öffentliche Schlüssel ist $(\mathcal{K}, n, k, b_1, \dots, b_n)$ und der private Schlüssel ist $(K, g, d, \mathfrak{p}, p_1, \dots, p_n)$. Die b_i beziehungsweise die p_i sind definiert als $b_i = (a_i + d) \bmod (\mathcal{N}(\mathfrak{p}) - 1)$ und $p_i \equiv g^{a_i} \pmod{\mathfrak{p}}$. Die Größen n und k geben die Problemgröße an.

Sollte ein Angreifer versuchen, den privaten Schlüssel nur aufgrund der Kenntnis des öffentlichen Schlüssel zu brechen, so müßte er folgendes tun: Zuerst müßte er den Körper K bestimmen. Dies scheint unmöglich, wenn man \mathcal{K} nur ausreichend groß wählt, da es dann exponentiell viele Möglichkeiten für K gibt. Selbst wenn er diese Hürde nimmt, so müßte er danach g und d bestimmen. Allein für d gibt es auch wieder exponentiell viele Möglichkeiten. Gleiches gilt für die Werte g und \mathfrak{p} .

Sollte der Angreifer, nachdem er den Körper K bestimmt hat, versuchen die Zahlen p_i zu erraten, so hat er die gleichen Probleme, wie bei den anderen Werten. Es gibt exponentiell viele Möglichkeiten. Selbst wenn er es schaffen sollte, eine Teilmenge der p_i zu bestimmen, so muss diese ziemlich groß sein, um sie für einen Angriff verwenden zu können. Es gibt nämlich von Odlyzko [18] einen Angriff auf multiplikative Knapsack-Probleme, den man hier benutzen könnte. Mit Hilfe des Angriffes von Odlyzko könnte dann versucht werden, g zu berechnen. Der Angriff von Odlyzko benötigt dafür aber eine relative große Teilmenge der p_i . Doch selbst wenn g damit berechnet werden könnte, so muss der Angreifer anschließend die p_i den entsprechenden b_i zuordnen. Dies scheint schwierig, weil die b_i durch die vorher berechneten diskreten Logarithmen zufällig erscheinen.

Es scheint also unmöglich, aus den Informationen des öffentlichen Schlüssels den privaten Schlüssel angreifen zu können, ohne K, g, d und \mathfrak{p} zu kennen.

Die Möglichkeit, aus den Schlüsseltexten Informationen auf den privaten Schlüssel zu ziehen, scheint unmöglich, denn neben verschiedenen m_i besteht der Schlüsseltext nur noch aus den b_i und die sind ja schon durch den öffentlichen Schlüssel bekannt. Hat ein Angreifer einen Klartext, so nützt ihm dies ebenso wenig. Bei der Verschlüsselung werden lediglich n, k und die b_i benutzt. Also führt auch diese Variante nicht zum gewünschten Ergebnis.

Eine andere Möglichkeit, einen privaten Schlüssel zu knacken, ist ein Brute Force Angriff. Bei einem solchen Angriff werden alle möglichen privaten Schlüssel einfach ausprobiert. Jedoch bringt das im Fall von OTU nicht viel, da es einfach zu viele mögliche Schlüsselvariationen aufgrund der vielen Variablen gibt.

Somit scheint es unmöglich zu sein den privaten Schlüssel anzugreifen ohne K, g, d und \mathfrak{p} zu kennen. Dabei ist es auch gleich, ob es sich um das allgemeine OTU QPKC oder eine abgeschwächte Variante handelt.

6.1.2 Angriff auf Schlüsseltext

Die nächste Angriffsmöglichkeit ist den Schlüsseltext anzugreifen. Viele Knapsack-basierten Kryptosysteme sind anfällig gegenüber Angriffen auf ihre geringe Knapsack-dichte. Also könnte das bei OTU eine Möglichkeit sein, das System zu knacken. Hierbei wird versucht, den Klartext direkt aus dem Schlüsseltext zu berechnen, indem Lösungen zu dem Knapsack-Problem gesucht werden. Bei der Verschlüsselung wird aus einer Nachricht M ein Schlüsseltext c berechnet. Im Fall von OTU wird M mittels einer Berechnung in eine Zeichenkette $m = (m_1, \dots, m_n)$ umgewandelt und mit den Zahlen b_i verrechnet. Das ganze kann man wie folgt schreiben:

$$c = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}^T.$$

Die verschiedenen m_i haben einen Wert von 0 oder 1 und die Werte der b_i sind ja schon durch den öffentlichen Schlüssel bekannt.

Bei einem Knapsack-Problem wie diesem geht es darum herauszufinden, ob es eine Lösung für das Problem gibt. Wir wissen ja, dass es eine entsprechende Lösung geben muss, so dass wir uns auf die Dichte des Knapsackes konzentrieren können. Bei einem Angriff auf die geringe Dichte eines Knapsackes wird ein kürzester Lösungsvektor gesucht, der in dem unten beschriebenen Gitter liegt. Ein Algorithmus, der sich zum Knacken des Schlüsseltext anbietet, ist der von J.C.

Lagarias und A.M. Odlyzko. Ein Gitter lässt sich wie folgt aufbauen

$$\begin{aligned} x_1 &:= (1, 0, \dots, 0, Nb_1) \\ x_2 &:= (0, 1, \dots, 0, Nb_2) \\ &\vdots \\ x_n &:= (0, 0, \dots, 1, Nb_n) \\ x_{n+1} &:= \left(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}, Nc\right) \end{aligned}$$

und anschließend sucht man in diesem Gitter den kürzesten Vektor, der nicht Null ist. Dies haben Lagarias und Odlyzko in ihrem Papier [12] gezeigt. Sie haben es geschafft, Knapsäcke zu lösen, die eine Dichte kleiner 0,6463 haben. Ein paar Jahre später haben M.J. Coster, A. Joux, B.A. LaMacchia, A. Odlyzko, C.P. Schnorr und J. Stern mit ihrem Papier [7] gezeigt, dass auch Knapsäcke mit einer Dichte kleiner als 0,9804 lösbar sind.

Mittels des Algorithmus von Lenstra, Lenstra und Lóvasz [13], kurz LLL, kann für den Fall von OTU der kürzeste Gittervektor sehr einfach gefunden werden.

6.2 Dichteangriff auf OTU

Nachdem oben gezeigt wurde, dass OTU ein Problem mit Angriffen auf eine Dichte kleiner als 0,9804 hat, soll nun gezeigt werden, wie es mit Dichteangriffen größer als 0,9804 aussieht. Ein Gitterorakel ist ein Orakel, das bei Eingabe einer Gitterbasis, einen kürzesten, nichtrivialen Gittervektor für dieses Gitter liefert.

Theorem 6.1 *Sei A eine positive Zahl und $a_1, \dots, a_n \in_R \{1, \dots, A\}$ gleichverteilte Zufallszahlen. Sei $e = (e_1, \dots, e_n) \in \{0, 1\}^n$, so dass $\sum_{i=1}^n e_i \leq \beta n$ für ein $0 < \beta \leq \frac{1}{4}$, und sei $s = \sum_{i=1}^n e_i a_i$. Wenn die Dichte $d < d_a$ wie unten beschrieben ist, dann ist das Knapsack-Problem, das durch a_1, \dots, a_n und s definiert wird, für hinreichend große n in polynomieller Zeit mit einem Aufruf eines Gitterorakel lösbar:*

$$d_a = \max_{u \in \mathbb{R}} \frac{1}{(\log_2 e) \delta(\beta, u)}, \quad \delta(\beta, u) = u\delta + \ln \theta(e^{-u}), \quad \theta(z) = 1 + 2 \sum_{k=1}^{\infty} z^{k^2}$$

Die Aussage des Theorems ist entscheidend für die Sicherheit des OTU Kryptosystems, da die Dichte des Knapsackes für die Sicherheit nicht mehr die entscheidende Rolle spielt. Das System ist nicht mehr sicher, wenn ein Gitterorakel einen kürzesten Vektor finden kann, der der obigen Definition genügt. Diese Aussage des Theorems werden wir nun im folgenden beweisen. Als erstes wird der allgemeine Knapsack untersucht und wie man die Lösungsmenge eingrenzen kann. Dann wird der Lösungsvektor betrachtet. Zum Schluss werden alle Beobachtungen zusammengesetzt, um die Wahrscheinlichkeit zu berechnen, wann eine Lösung für ein Knapsack-Problem gefunden werden kann.

Der folgende Beweis ist aus drei verschiedenen Beweisen [7, 12, 21] zusammengetragen worden.

Betrachten wir nun den oben angegebenen Knapsack. Die Summe aller Gewichte des Knapsackes sei mit $t = \sum_{i=1}^n a_i$ definiert. t/n ist dann der Durchschnitt aller Gewichte des Knapsackes. Nun können wir folgende Aussage machen.

Lemma 6.1 *Sei (s, a_1, \dots, a_n) ein Knapsack-Problem wie in Theorem 6.1 definiert. Wenn $s \leq t/n$ oder $s \geq (1 - (1/n))t$, dann können wir das Knapsack-Problem (s, a_1, \dots, a_n) auf ein Knapsack-Problem $(s, a'_1, \dots, a'_{n-1})$ mit $n - 1$ Gewichten reduzieren.*

Beweis. Betrachten wir zuerst den Fall, dass $s \leq t/n$ ist. Es existiert ein $j \in \{1, \dots, n\}$ derart, dass $a_j \geq t/n$ ist. Wenn $s < t/n$, dann ist $s < a_j$ und wir können a_j trivialerweise als potentielles Element des gesuchten Knapsackes ausschließen und das gesamte Knapsack-Problem durch eines mit weniger Gewichten ersetzen. Wir können deshalb im folgenden o.B.d.A. $s \leq t/n$ annehmen. Gleiches gilt nun für den Fall, dass $s \geq (1 - (1/n))t$. Betrachten wir das inverse Knapsack-Problem mit $(t - s, a_1, \dots, a_n)$ und sei $s' := t - s$. Wenn $s' \leq t/n$ ist, dann existiert ein j , so dass $a_j \geq t/n$. Ist nun $s' < t/n$, dann ist $s' < a_j$ und daher auch kein Element des gesuchten Knapsackes. Daher lässt sich ein äquivalentes Knapsack-Problem mit weniger Gewichten finden und wir können o.B.d.A. $s' \leq t/n$ annehmen. Somit lässt sich $s' \leq t/n$ zu $s \geq \frac{n-1}{n}t$ wie folgt umformen:

$$\begin{aligned} s' &\leq t/n \\ t - s &\leq t/n \\ t - t/n &\leq s \end{aligned}$$

Somit werden mit den beiden betrachteten Fällen die kleineren und größeren Gewichte des Problems entfernt und die Aufgabe mit jedem Mal reduziert. ■

Im folgenden können wir nun annehmen, dass

$$\frac{1}{n}t \leq s \leq \frac{n-1}{n}t \tag{6.1}$$

gilt.

Für die Lösungssuche unseres Problems definieren wir Vektoren $v_1, \dots, v_{n+1} \in \mathbb{Z}^{n+1}$ wie folgt:

$$\begin{aligned} v_1 &:= (1, 0, \dots, 0, Na_1) \\ v_2 &:= (0, 1, \dots, 0, Na_2) \\ &\vdots \\ v_n &:= (0, 0, \dots, 1, Na_n) \\ v_{n+1} &:= \left(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}, Ns\right). \end{aligned}$$

N ist eine hinreichend große Zahl und wird hier auf $N > \sqrt{n}$ gesetzt. Mit den Vektoren v_1, \dots, v_{n+1} wird ein Gitter L aufgespannt. Der Lösungsvektor für unser betrachtetes Knapsack-Problem ist $\hat{e} = (e_1 - \frac{1}{2}, \dots, e_n - \frac{1}{2}, 0)$.

Wir suchen nun den kürzesten Vektor $\hat{x} = (x_1, \dots, x_n, x_{n+1})$, der eine Lösung für unser betrachtetes Knapsack-Problem darstellt und kleiner ist als der Lösungsvektor \hat{e} , dass heißt er soll folgende Eigenschaften erfüllen:

$$\|\hat{x}\| \leq \|\hat{e}\|, \quad \hat{x} \in L, \quad \|\hat{x}\| \notin \{0, \hat{e}, -\hat{e}\}. \quad (6.2)$$

Da $N > \sqrt{n}$ ist, ist $x_{n+1} = 0$ und somit sei $x := (x_1, \dots, x_n)$.

Lemma 6.2 *Das Knapsack-Problem (s, a_1, \dots, a_n) sei wie im Theorem 6.1 und in Lemma 6.1 angegeben, \hat{x} wie oben. Sei $y := \sum_{i=1}^n x_i a_i$ mit $x := (x_1, \dots, x_n)$. Dann ist $|y| \leq n\sqrt{\beta n}$.*

Beweis. Betrachten wir y und wenden die Gleichungen 6.1 und 6.2 darauf an.

$$|y| = \frac{1}{s} \left| \sum_{i=1}^n x_i a_i \right| \leq \frac{\|\hat{x}\|}{s} \left| \sum_{i=1}^n a_i \right| = \frac{t}{s} \|\hat{x}\| \leq \frac{t\sqrt{\beta n}}{s}$$

Also

$$|y| \leq \frac{t\sqrt{\beta n}}{(t/n)} = n\sqrt{\beta n}.$$

■

Wir wissen, dass die Dichte eines Knapsackes sich aus den Gewichten a_1, \dots, a_n und deren Anzahl wie folgt berechnet:

$$d = \frac{n}{\log_2 \max_{1 \leq i \leq n} a_i}.$$

Wir definieren uns eine Konstante $c := \log_2(A)/n$ mit $A = \max_{1 \leq i \leq n} a_i$. Es wird noch eine weitere Konstante, c_0 , benötigt, die erst später genauer betrachtet werden soll. c_0 hängt von der Dichte ab, ist aber für eine feste Dichte jedoch konstant. Wir berechnen jetzt die Wahrscheinlichkeit dafür, dass das oben definierte Gitter L einen kürzesten Vektor gemäß Gleichung (6.2) beinhaltet.

Lemma 6.3 *Die Wahrscheinlichkeit, dass ein Gitter gemäß Gleichung (6.2) einen kürzesten Gittervektor beinhaltet, ist*

$$P = Pr(\exists \hat{x} : \|\hat{x}\| \leq \|\hat{e}\|, \quad \hat{x} \in L, \quad \hat{x} \notin \{0, \hat{e}, -\hat{e}\}) \leq n(2n\sqrt{\beta n} + 1) \frac{2^{c_0 n}}{A}$$

Beweis. Um das Lemma 6.3 zu beweisen, nehmen wir das vorher definierte $x = (x_1, \dots, x_n) \in \mathbb{Z}^n$, das ein Element von \mathbb{Z}^n ist. (Wenn $\hat{x} = (x_1, \dots, x_n, 0)$, dann ist $\|\hat{x}\| = \|x\|$ und als Spezialfall $\|\hat{e}\| = \|e\|$.)

Nun berechnen wir die Wahrscheinlichkeit mit

$$\begin{aligned}
 P &= Pr(\exists \hat{x}, \text{ welches Bedingung (6.2) erfüllt}) \\
 &= Pr(\exists \hat{x} \in L, y \in \mathbb{Z}, \text{ so dass: } \|\hat{x}\| \leq \|\hat{e}\|, \|\hat{x}\| \notin \{0, \hat{e}, -\hat{e}\}, |y| \leq n\sqrt{\beta n}, ys = \sum_{i=1}^n x_i a_i) \\
 &\leq \underbrace{Pr\left(\begin{array}{l} ys = \sum_{i=1}^n x_i a_i, x \in \mathbb{Z}^n + \mathbb{Z}(\frac{1}{2}, \dots, \frac{1}{2}) \text{ und } y \in \mathbb{Z} \text{ fest : } \\ \|x\| \leq \|\hat{e}\|, \|x\| \notin \{0, \hat{e}, -\hat{e}\}, |y| \leq n\sqrt{\beta n} \end{array}\right)}_{\text{Faktor1}} \\
 &\quad \cdot \underbrace{|\{x \in \mathbb{Z}^n + \mathbb{Z}(\frac{1}{2}, \dots, \frac{1}{2}) : \|x\| \leq \|\hat{e}\| \leq \sqrt{\beta n}\}|}_{\text{Faktor2}} \cdot \underbrace{|\{y \in \mathbb{Z} : |y| \leq n\sqrt{\beta n}\}|}_{\text{Faktor3}}
 \end{aligned}$$

Es werden nun die einzelnen Faktoren der letzten Zeile betrachtet. Der erste Faktor wird mit $ys = \sum_{i=1}^n x_i a_i$ umschrieben als: $\sum_{i=1}^n a_i z_i = 0$ mit $z_i := x_i - y e_i$. Solange $x \neq 0$ und $\|x\| \leq \|e\|$, dann ist $z = (z_1, \dots, z_n) \neq 0$ und man kann ohne Verlust auf die Allgemeingültigkeit annehmen, dass $z_1 \neq 0$. Wenn z' definiert ist als $-(\sum_{i=1}^n a_i z_i / z_1)$, dann ist

$$\begin{aligned}
 Pr\left(\sum_{i=1}^n a_i z_i = 0\right) &= Pr(a_1 = z') \\
 &= \sum_{j=1}^A Pr(a_1 = z' | z' = j) Pr(z' = j) \\
 &= \sum_{j=1}^A Pr(a_1 = z') Pr(z' = j) \quad (a_1 \text{ und } j \text{ sind unabhängig}) \\
 &= \sum_{j=1}^A \frac{1}{A} Pr(z' = j) \leq \frac{1}{A}
 \end{aligned}$$

Der zweite Faktor lässt sich wie folgt berechnen: Von Lagarias und Odlyzko ([12]) weiß man, dass

$$|\{\hat{x} : \|\hat{x}\| \leq \|\hat{e}\|\}| \leq |\{x : \|x\| \leq \sqrt{\beta n}\}| \leq 2^{c_0 n}.$$

Lagarias und Odlyzko haben in dem Papier [12] unter anderem die Anzahl N der Gitterpunkte von \mathbb{Z}^n in einer Kugel mit dem Radius $\sqrt{\beta n}$ um den Ursprung

$$N(n, \beta) := |\{x \in \mathbb{Z}^n : \|x\|^2 \leq \beta n\}|$$

betrachtet. Sie zeigten dabei für hinreichend große n , dass für jedes $u > 0$ gilt

$$N(n, \beta) \leq 2^{(\log_2 e)\delta(\beta, u)n}$$

mit $\delta(\beta, u) = u\beta + \ln \theta(e^{-u})$ und $\theta(z) = 1 + 2 \sum_{i=1}^{\infty} z^{i^2}$. Für feste β kann die Minimalstelle u_0 von $\delta(\beta, u)$ numerisch angenähert werden. c_0 wird berechnet mit $c_0 = \min_{u \in \mathbb{R}} (\log_2 e) \{u\beta + \ln\{1 + 2 \sum_{i=1}^{\infty} (e^{-u})^{i^2}\}\}$. Daher können wir den zweiten Faktor wie oben beschrieben mit $2^{c_0 n}$ abschätzen.

Der dritte Faktor ist mit $2n\sqrt{\beta n} + 1$ abzuschätzen.

Somit ergibt sich die Gleichung (6.3) mit $P = Pr(n) \leq n(2n\sqrt{\beta n} + 1) \frac{2^{c_0 n}}{A}$.

■

Daraus folgt, wenn $A = 2^{cn}$ mit $c > c_0$ ist, dann ist $\lim_{n \rightarrow \infty} P = 0$ und Dichte d lässt sich wie folgt berechnen:

$$d = \frac{n}{\log_2 \max_{1 \leq i \leq n} a_i} \approx \frac{n}{\log_2 A} < \frac{n}{\log_2 2^{c_0 n}} = \frac{1}{c_0} = d_a.$$

Als Beispiel für einen Wert für c_0 nehmen wir die Knapsackdichte von 0,6463 aus dem Beweis von Lagarias und Odlyzko. c_0 hat dabei einen Wert von 1,5473. Bei einer Dichte von 0,9408 wie bei Coster, Joux, LaMacchia, Odlyzko, Schnorr und Stern ist $c_0 = 1,0629$.

Die Dichte d_k von OTU QPKC hängt somit von n und β ab, so dass $\sum_{i=1}^n e_i = \beta n$ für den verschlüsselten Text $e = (e_1, \dots, e_n)$ ist. Somit haben wir eine Dichte d_k mit

$$d_k = \frac{n}{\beta n \log n} = \frac{1}{\beta \log n}.$$

Mittels dem Theorem 6.1 und der Dichte d_a mit

$$d_a = \max_{u \in \mathbb{R}} \frac{1}{(\log_2 e)\delta(\beta, u)}$$

konnte gezeigt werden, dass für jede Parameterwahl von n und β , der Knapsack von OTU gebrochen werden kann, wenn $d_a - d_k > 0$ ist. Somit ist das OTU System nicht mehr sicher.

7 Praktische Ergebnisse

Nachdem im vorherigen Kapitel gezeigt wurde, dass OTU gegen Angriffe mit einer Knapsackdichte von über 0,9804 nicht Stand halten kann, wenn es ein entsprechendes Gitterorakel gibt, soll dies nun in der Praxis überprüft werden. Es wird mittels des LLL Algorithmus ein kürzester Vektor in unserem Gitter gesucht.

Zur Erinnerung die Verschlüsselungsfunktion von OTU: Aus der Nachricht M wird ein Vektor $m = (m_1, \dots, m_n)$ mit $m_i \in \{0, 1\}$ errechnet. Anschließend wird dieser Vektor mit den Zahlen b_1, \dots, b_n aus dem öffentlichen Schlüssel zu dem Schlüsseltext $c = \sum_{i=1}^n m_i b_i$ verrechnet.

Die Dichte des OTU Knapsackes ist, wie wir wissen, $d_k = \frac{n}{\beta n \log n} = \frac{1}{\beta \log n}$. Außerdem wissen wir, dass $k = \sum_{i=1}^n e_i = \beta n$ ist (siehe [21]). Wenn wir nun k in die Dichtegleichung mit einbringen, ergibt sich $d_k = \frac{n}{k \log n}$. Es gibt also einen Zusammenhang zwischen n und k , wodurch beide als Sicherheitsparameter für die Dichte des Knapsackes einzustufen sind. Dies muss bei der Wahl der beiden Parameter bei der Schlüsselerstellung berücksichtigt werden. Um die Bitlänge für die Primzahl p festzulegen, wird sie auf $2 * k * \log n$ abgeschätzt.

Aus diesen Informationen wird ein Gitter zusammengebaut, so dass mit Hilfe des LLL Algorithmus ein kürzester Vektor in diesem Gitter gesucht werden kann. Bei dem Aufbau des Gitters halten wir uns an die Idee von Coster, Joux, LaMacchia, Odlyzko, Schnorr und Stern [7]. Also sieht das Gitter wie folgt aus:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & Nb_1 \\ 0 & 1 & \dots & 0 & Nb_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & Nb_n \\ \frac{1}{2} & \frac{1}{2} & \dots & \frac{1}{2} & Nc \end{pmatrix}$$

Es wird jetzt der Vektor $e = (e_1, \dots, e_n)$ mit $e_i \in \{0, 1\}$ gesucht, der die Gleichung $c = \sum_{i=1}^n e_i b_i$ erfüllt. Der kürzeste Vektor, der dies erfüllt und in dem obigen Gitter liegt, ist $(e_1, \dots, e_n, 0)$.

Die Programmierung des Angriffes wurde in C++ durchgeführt und dabei auf den LLL Algorithmus aus der NTL Bibliothek [1] zurückgegriffen. Der zugrundeliegende Rechner bei der Ausführung des Programmes hat einen 1,2 GHz Prozessor mit 512MB Arbeitsspeicher. Der allgemeine Angriff sieht wie folgt aus:

INPUT Anzahl der Elemente: $n \in \mathbb{Z}$

FOR (i=1, i <= n, i++) DO

```
k berechnen mit  $k = n / (0.9408 * \log n)$ 
Bitlänge berechnen mit  $bitlength = 2 * k * \log n$ 
FOR (test_k = k; test_k >= 2; test_k--) DO
  Dichte berechnen mit  $density = n / test\_k * \log n$ 
  FOR (testrun = 0; testrun < 100; testrun++) DO
    Gitterbasis  $b_1, \dots, b_n$  mittels Zufallszahlen setzen
    Vektor  $m$  setzen und Gitter aufbauen
    Matrix erstellen
    Aufruf des LLL Algorithmus
OUTPUT kurzer Vektor  $e_1, \dots, e_n, e_{n+1}$ 
```

Die Vektorerstellung von m ist der einzige Unterschied in den verschiedenen Programmvarianten. Jede Variante der Parameterwahl von n , Bitlänge und Dichte wird 100 mal durchgespielt und zwar mit immer einem anderen Gitter. Sollte bei einem der 100 Angriffe ein Angriff erfolgreich sein, so ist das System für diese Parameterwahl nicht sicher.

Um die Sicherheit richtig einschätzen zu können, wurde zuerst ein sehr naiver Angriff benutzt. Bei dem Vektor m wurden immer die ersten k Werte auf 1 gesetzt und die restlichen Werte auf 0, also $m_1 = 1, m_2 = 1, \dots, m_k = 1, m_{k+1} = 0, \dots, m_n = 0$. Somit kann man testen, wie anfällig das OTU System in einem Extremfall von dem Vektor m ist und allgemeine Zusammenhänge zwischen den einzelnen Parametern finden, falls es welche gibt.

Mit dieser naiven Variante des Angriffes konnte in fast 50% aller Fälle ein kürzester Vektor gefunden werden, der den Ansprüchen genügte, sprich das OTU System wäre in diesem Fall unsicher. In Tabelle 7.1 kann man dies gut ablesen. Allgemein lässt sich sagen, dass es dabei keinen richtigen Zusammenhang zwischen der Dichte und der Erfolgswahrscheinlichkeit für einen erfolgreichen Angriff gibt. Es gibt eine Tendenz dazu, dass je höher die Dichte ist, desto sicherer gelingt ein Angriff. Die Empfehlung von Okamoto, Tanaka und Uchiyama die Dichte größer 1 zu wählen, ist also nur bedingt zu befolgen. Die Angriffschancen waren eigentlich schlechter, wenn die Dichte etwas größer als 0,9804 lag. Zudem scheint die größer werdende Bitlänge auch kein wirkliches Problem für einen erfolgreichen Angriff zu sein. Zwar sinkt die Erfolgswahrscheinlichkeit bei größeren Bitlängen, aber selbst bei einer Bitlänge von 743 Bits ist die Erfolgswahrscheinlichkeit immer noch über 40%. Das ist einfach zuviel. Hinzukommt, dass ein Angriff mit dieser Bitlänge nicht länger als 8 Minuten für ein Gitter braucht, um ein Ergebnis zu liefern.

Wegen den oben genannten Problemen wurde nun überlegt, wie man das ganze System wieder sicherer gemacht konnte. Dies scheint nur über die Wahl des Vektors m möglich zu sein. Daher werden für eine feste Wahl von n, k , der Bitlänge und Dichte immer wieder der Vektor m verändert. Der Vektor wird so verändert,

dass die Einsen im Abstand von i vorkommen. Der Parameter i wird der Reihe nach mit 2 bis 15 gesetzt. Also für beispielsweise $i = 3$ wird jede dritte Stelle im Vektor mit einer Eins gesetzt. Wie man in Tabelle 7.2 sehen kann, hat dies einen enormen Einfluss auf die Erfolgswahrscheinlichkeit eines Angriffes. Mit $i = 4$ kann die Erfolgswahrscheinlichkeit auf 0 gebracht werden. Je höher i ist, desto sicherer kann das OTU System gemacht werden.

Dies zeigt sich auch in der letzten Variante des Angriffes. Hier werden die verschiedenen Parameter beleuchtet unter Rücksichtnahme des Vektors m . Ab $n = 130$ und einer Bitlänge von 266 Bit kann man OTU sicher vor LLL-Angriffen machen, indem man $i = 4$ wählt. Wie aus Tabelle 7.3 ersichtlich wird, ist der Vektor m ausschlaggebend für die Sicherheit des OTU Systems. Einen geringen Anteil haben natürlich auch die Wahl von n, k , der Bitlänge und der Dichte des Knapsackes. Dabei sollte n entsprechend groß gewählt werden und dann möglichst ein k in der Nähe von $k = n/(0.9408 * \log n)$.

Um zu einem Schluss über die Sicherheit von OTU zu kommen, muss die Art und Weise, wie die Nachricht M in den Vektor m umgewandelt wird, betrachtet werden. Schließlich ist der Vektor m entscheidend für die Sicherheit. Das Problem bei der Umwandlung ist, dass die Verteilung der Einsen im Vektor m dahin tendiert, dass die Einsen nicht weit genug gestreut werden. Es ist nicht gerade unwahrscheinlich, da es irgendwo im Vektor zu einer Häufung von Einsen kommt. Damit wird die Erfolgswahrscheinlichkeit eines Angriffes nach oben getrieben, da die Vektoren dann denen im naiven Ansatz wieder näher kommen.

Es spielt dabei keine Rolle, welche Variante von dem OTU QPKC benutzt wird, da alle drei Varianten auf die gleiche Art verschlüsseln. Das komplette OTU QPKC bietet nur zusätzlichen Schutz gegen mögliche anderen Angriffe. Somit lässt sich festhalten, dass das System von OTU für normale Anwendungen wie sie beispielsweise durch RSA erledigt werden, nicht geeignet ist.

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
10	3	19	1.00343	100
10	2	19	1.50515	100
15	4	31	0.959843	100
15	3	31	1.27979	100
15	2	31	1.91969	100
20	4	34	1.15689	100
20	3	34	1.54252	100
20	2	34	2.31378	100

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
25	5	46	1.07669	100
25	4	46	1.34586	100
25	3	46	1.79449	100
25	2	46	2.69173	100
30	6	58	1.01898	100
30	5	58	1.22277	100
30	4	58	1.52846	100
30	3	58	2.03795	100
30	2	58	3.05693	100
35	7	71	0.974795	100
35	6	71	1.13726	100
35	5	71	1.36471	100
35	4	71	1.70589	100
35	3	71	2.27452	100
40	7	74	1.07372	99
40	6	74	1.25268	98
40	5	74	1.50321	100
40	4	74	1.87902	100
40	3	74	2.50536	100
45	8	87	1.02424	99
45	7	87	1.17057	99
45	6	87	1.36566	99
45	5	87	1.63879	100
45	4	87	2.04849	100
45	3	87	2.73132	100
50	9	101	0.984355	94
50	8	101	1.1074	93
50	7	101	1.2656	99
50	6	101	1.47653	98
50	5	101	1.77184	100
50	4	101	2.2148	100
50	3	101	2.95306	100
55	10	115	0.951333	98
55	9	115	1.05704	97
55	8	115	1.18917	98
55	7	115	1.35905	99
55	6	115	1.58556	99
55	5	115	1.90267	99

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
55	4	115	2.37833	100
55	3	115	3.17111	100
60	10	118	1.01576	84
60	9	118	1.12863	91
60	8	118	1.2697	95
60	7	118	1.45109	97
60	6	118	1.69294	99
60	5	118	2.03153	99
60	4	118	2.53941	100
65	11	132	0.981191	78
65	10	132	1.07931	86
65	9	132	1.19923	90
65	8	132	1.34914	90
65	7	132	1.54187	92
65	6	132	1.79885	96
65	5	132	2.15862	98
65	4	132	2.69827	100
70	12	147	0.951715	79
70	11	147	1.03824	79
70	10	147	1.14206	87
70	9	147	1.26895	91
70	8	147	1.42757	92
70	7	147	1.63151	96
70	6	147	1.90343	96
70	5	147	2.28412	97
70	4	147	2.85515	99
70	3	147	3.80686	100
75	12	149	1.0034	67
75	11	149	1.09462	75
75	10	149	1.20408	81
75	9	149	1.33787	85
75	8	149	1.5051	90
75	7	149	1.72012	96
75	6	149	2.0068	96
75	5	149	2.40816	99
75	4	149	3.0102	99
75	3	149	4.0136	100
75	2	149	6.0204	100

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
80	13	164	0.973413	69
80	12	164	1.05453	75
80	11	164	1.1504	77
80	10	164	1.26544	77
80	9	164	1.40604	85
80	8	164	1.5818	89
80	7	164	1.80777	91
80	6	164	2.10906	96
80	5	164	2.53087	99
80	4	164	3.16359	100
80	3	164	4.21812	100
80	2	164	6.32718	100
85	14	179	0.947271	65
85	13	179	1.02014	69
85	12	179	1.10515	76
85	11	179	1.20562	84
85	10	179	1.32618	86
85	9	179	1.47353	90
85	8	179	1.65772	96
85	7	179	1.89454	94
85	6	179	2.2103	97
85	5	179	2.65236	98
85	4	179	3.31545	100
85	3	179	4.4206	100
85	2	179	6.6309	100
90	13	181	1.06643	70
90	12	181	1.15529	77
90	11	181	1.26032	79
90	10	181	1.38635	82
95	15	197	0.963999	51
95	14	197	1.03286	60
95	13	197	1.11231	69
95	12	197	1.205	74
95	11	197	1.31454	80
95	10	197	1.446	83
100	15	199	1.00343	71
100	14	199	1.07511	71
100	13	199	1.15781	77

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
100	12	199	1.25429	82
100	11	199	1.36832	83
100	10	199	1.50515	83
105	16	214	0.977399	67
105	15	214	1.04256	72
105	14	214	1.11703	77
105	13	214	1.20295	77
110	17	230	0.954173	70
110	16	230	1.01381	75
110	15	230	1.0814	76
110	14	230	1.15864	82
115	17	232	0.988199	68
115	16	232	1.04996	74
115	15	232	1.11996	78
115	14	232	1.19996	83
120	18	248	0.96522	65
120	17	248	1.022	72
120	16	248	1.08587	76
120	15	248	1.15826	79
125	19	264	0.944466	62
125	18	264	0.996936	65
125	17	264	1.05558	67
125	16	264	1.12155	73
130	19	266	0.97433	71
130	18	266	1.02846	73
130	17	266	1.08896	71
130	16	266	1.15702	82
135	20	283	0.953819	66
135	19	283	1.00402	69
135	18	283	1.0598	72
135	17	283	1.12214	75
140	20	285	0.981866	55
140	19	285	1.03354	62
140	18	285	1.09096	65
140	17	285	1.15514	73
145	21	301	0.961678	64
145	20	301	1.00976	66
145	19	301	1.06291	65

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
145	18	301	1.12196	73
150	22	318	0.943194	52
150	21	318	0.988108	56
150	20	318	1.03751	56
150	19	318	1.09212	62
155	22	320	0.968298	57
155	21	320	1.01441	64
155	20	320	1.06513	68
155	19	320	1.12119	73
160	23	336	0.950094	51
160	22	336	0.99328	59
160	21	336	1.04058	60
160	20	336	1.09261	70
165	23	338	0.97388	53
165	22	338	1.01815	62
170	24	355	0.955994	51
170	23	355	0.997559	57
175	24	357	0.978588	59
175	23	357	1.02114	61
180	25	374	0.961044	61
180	24	374	1.00109	63
185	26	391	0.944765	54
185	25	391	0.982555	58
190	26	393	0.965367	51
190	25	393	1.00398	55
195	27	410	0.949377	49
195	26	410	0.985891	56
200	27	412	0.969067	57
200	26	412	1.00634	66
205	28	430	0.953376	52
210	28	431	0.972227	53
215	29	449	0.956842	55
220	30	466	0.942423	55
225	30	468	0.959843	51
230	31	486	0.945684	52
235	31	488	0.962436	50
240	32	506	0.94854	42
245	32	507	0.964672	51

Tabelle 7.1: Naiver Angriff mit LLL Algorithmus

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %
250	33	525	0.951037	48
255	33	527	0.966591	54
260	34	545	0.953217	45
265	34	547	0.968232	49
270	35	565	0.955115	46
275	36	583	0.94269	41
280	36	585	0.956761	39
285	37	603	0.944558	46
293	38	622	0.94091	45
300	38	625	0.959401	43
350	44	743	0.941231	42

Tabelle 7.2: Angriff bei verschiedenem Aufbau des Vektors m

n	k	i	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
130	19	2	266	0.97433	53	12352.9
130	18	2	266	1.02846	64	12310.8
130	17	2	266	1.08896	68	12316.4
130	16	2	266	1.15702	78	12311.9
130	15	2	266	1.23415	81	12311.5
130	19	3	266	0.97433	1	12311.6
130	18	3	266	1.02846	2	12311.5
130	17	3	266	1.08896	6	12315.2
130	16	3	266	1.15702	23	12311.9
130	15	3	266	1.23415	34	12448.7
130	19	4	266	0.97433	0	12300
130	18	4	266	1.02846	0	12340.3
130	17	4	266	1.08896	0	12295.6
130	16	4	266	1.15702	0	12300.7
130	15	4	266	1.23415	0	12482.1
130	14	4	266	1.32231	0	13623.8
130	13	4	266	1.42402	6	13564.1
130	19	5	266	0.97433	0	12285.4
130	18	5	266	1.02846	0	12281.4

Tabelle 7.2: Angriff bei verschiedenem Aufbau des Vektors m

n	k	i	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
130	17	5	266	1.08896	0	12282.2
130	16	5	266	1.15702	0	12285.4
130	15	5	266	1.23415	0	13122.2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	12	5	266	1.54269	0	12285.2
130	11	5	266	1.68293	2	12293.8
130	19	6	266	0.97433	0	12288.6
130	18	6	266	1.02846	0	13267.9
130	17	6	266	1.08896	0	13567.8
130	16	6	266	1.15702	0	13619.2
130	15	6	266	1.23415	0	14651.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	10	6	266	1.85123	0	12287
130	9	6	266	2.05692	5	12294.1
130	19	7	266	0.97433	0	12287.3
130	18	7	266	1.02846	0	12283.4
130	17	7	266	1.08896	0	12283.4
130	16	7	266	1.15702	0	12287.4
130	15	7	266	1.23415	0	13982.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	9	7	266	2.05692	0	13580.7
130	8	7	266	2.31403	2	13772.5
130	19	8	266	0.97433	0	13662.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	8	8	266	2.31403	0	12456.5
130	7	8	266	2.64461	1	12460.2
130	19	9	266	0.97433	0	13452
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	7	9	266	2.64461	0	13718.1
130	6	9	266	3.08538	5	13727.7
130	19	10	266	0.97433	0	12258.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
130	7	10	266	2.64461	0	16503.5
130	6	10	266	3.08538	1	14262.4
130	19	11	266	0.97433	0	12874.8

Tabelle 7.2: Angriff bei verschiedenem Aufbau des Vektors m

n	k	i	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
⋮		⋮	⋮		⋮	
130	6	11	266	3.08538	0	13558.2
130	5	11	266	3.70245	4	13685.7
130	19	12	266	0.97433	0	13611.7
⋮		⋮	⋮		⋮	
130	5	12	266	3.70245	0	17168.3
130	4	12	266	4.62807	34	17197.4
130	19	15	266	0.97433	0	17181.9
⋮		⋮	⋮		⋮	
130	4	15	266	4.62807	0	17188
130	3	15	266	6.17076	54	17182.7

Tabelle 7.3: LLL Angriff mit verbessertem Vektor m ($i = 4$)

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
10	3	19	1.00343	0	3.52
10	2	19	1.50515	21	2.63
15	4	31	0.959843	1	9.21
15	3	31	1.27979	24	16.15
20	4	34	1.15689	18	23.53
20	3	34	1.54252	67	20.47
25	5	46	1.07669	17	42.7
25	4	46	1.34586	66	49.19
30	6	58	1.01898	10	89.6
30	5	58	1.22277	60	78.41
35	7	71	0.974795	10	156.62
35	6	71	1.13726	46	172.93
40	7	74	1.07372	43	205.39
40	6	74	1.25268	76	201.01
45	8	87	1.02424	25	382.61
45	7	87	1.17057	62	329.69

Tabelle 7.3: LLL Angriff mit verbessertem Vektor m ($i = 4$)

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
50	9	101	0.984355	27	562.4
50	8	101	1.1074	57	580.17
55	10	115	0.951333	22	776.34
55	9	115	1.05704	53	793.18
60	10	118	1.01576	32	910.38
60	9	118	1.12863	65	1052.3
65	11	132	0.981191	30	1230.73
65	10	132	1.07931	54	1217.65
70	12	147	0.951715	21	1745.77
70	11	147	1.03824	52	1738.04
75	12	149	1.0034	18	1913.63
75	11	149	1.09462	40	1918.14
80	13	164	0.973413	11	2512.76
80	12	164	1.05453	23	2513.48
85	14	179	0.947271	5	3379.3
90	13	181	1.06643	9	3791.87
95	15	197	0.963999	8	4769.68
100	15	199	1.00343	2	5228.95
105	16	214	0.977399	4	6184.9
110	17	230	0.954173	1	78418.7
115	17	232	0.988199	4	8556.58
120	18	248	0.96522	0	9953.35
120	17	248	1.022	1	10240.4
120	16	248	1.08587	10	10634.4
120	15	248	1.15826	29	10690.4
125	19	264	0.944466	0	12610.4
125	18	264	0.996936	0	12677.9
125	17	264	1.05558	0	12763.6
125	16	264	1.12155	0	12648.2
125	15	264	1.19632	0	12716.2
125	14	264	1.28178	2	12753.9
130	19	266	0.97433	0	12800.5
130	18	266	1.02846	0	13357.8
130	17	266	1.08896	0	13866.7
130	16	266	1.15702	0	13847.6
⋮		⋮		⋮	

Tabelle 7.3: LLL Angriff mit verbessertem Vektor m ($i = 4$)

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
130	14	266	1.32231	0	13898.4
130	13	266	1.42402	6	13741.1
135	20	283	0.953819	0	22846.5
135	19	283	1.00402	0	15875.9
135	18	283	1.0598	0	15879.4
135	17	283	1.12214	0	15863.7
⋮	⋮	⋮	⋮	⋮	⋮
135	15	283	1.27176	0	22770.6
135	14	283	1.3626	4	15907.2
140	20	285	0.981866	0	27603
140	19	285	1.03354	0	24055.5
140	18	285	1.09096	0	16963
140	17	285	1.15514	0	16914.1
⋮	⋮	⋮	⋮	⋮	⋮
140	15	285	1.30915	0	17108
140	14	285	1.40267	1	17198.3
145	21	301	0.961678	0	20340.8
145	20	301	1.00976	0	20754.7
145	19	301	1.06291	0	20691.3
145	18	301	1.12196	0	20689.1
⋮	⋮	⋮	⋮	⋮	⋮
145	16	301	1.2622	0	20685.3
145	15	301	1.34635	1	20590.5
150	22	318	0.943194	0	23232.7
150	21	318	0.988108	0	23429.3
150	20	318	1.03751	0	23315.1
150	19	318	1.09212	0	23392.3
⋮	⋮	⋮	⋮	⋮	⋮
150	17	318	1.2206	0	23427.1
150	16	318	1.29689	1	23300.1
160	23	336	0.950094	0	27687.8
160	22	336	0.99328	0	27621.5
160	21	336	1.04058	0	28223.7
160	20	336	1.09261	0	27621.6
⋮	⋮	⋮	⋮	⋮	⋮

Tabelle 7.3: LLL Angriff mit verbessertem Vektor m ($i = 4$)

n	k	Bitlänge	Dichte	Erfolgswahrscheinlichkeit für einen Angriff in %	Durchschnittliche Angriffszeitdauer in ms für ein Gitter
160	15	336	1.45681	0	27589.7
160	14	336	1.56087	8	28097.9

8 Fazit

Nachdem in dem Kapitel über die verschiedenen Angriffsvarianten auf das OTU System gezeigt wurde, dass OTU gegen Angriffe auf den kürzesten Vektor eines Knapsack-Problems mit der Dichte größer als 0,9804 nicht Stand halten konnte, war nicht viel mehr zu erwarten. Es hatte sich damit auch gezeigt, dass es bei einem Knapsack-basierenden System nicht bloß auf die Dichte ankommt, sondern man auch auf andere Faktoren schauen muss. Dies zeigt sich auch in den praktischen Tests. Die naive herangehensweise an den Angriff mittels LLL zeigt, dass OTU nicht mehr sicher ist, wenn man nur auf die Dichte schaut. Selbst bei einer Problemgröße von 350 und einer Bitlänge von 743 Bits für die Primzahl p lag die Erfolgswahrscheinlichkeit für einen erfolgreichen Angriff bei 42%. Nachdem ich mir jedoch die Verteilung der k Einsen im Vektor m angeschaut habe, konnte man sehr wohl das System wieder sicherer machen. Dies setzt allerdings voraus, dass die Nachricht, die verschlüsselt werden soll, so aufgebaut ist, dass sie eine gewisse Verteilung der Einsen hervorruft. Womit man bei dem Knackpunkt des Systems ist. Die Verschlüsselung ist unglücklich bei dem Aufbau des Vektors m , da es immer wieder zu einer Häufung von Einsen kommen kann. Diese Häufung bietet die Möglichkeit das OTU System zu knacken und dabei ist es egal, welche Variante des OTU QPKC benutzt wird.

Es wäre allerdings ein Versuch wert, die Verschlüsselung der Nachricht dahingehend zu verändern, dass es nicht mehr zu einer Häufung von Einsen im Vektor kommt.

9 Literaturverzeichnis

- [1] NTL - A library for doing number theory. <http://www.shoup.net>.
- [2] Der berechenbarkeitsbegriff. Vorlesungsskript, 2002. http://www.kr.tuwien.ac.at/courses/ss2002/Kapitel_2.pdf.
- [3] RSA-200 is factored!, Mai 2005. <http://www.rsasecurity.com/rsalabs/node.asp?id=2879>.
- [4] Johannes Buchmann. *Einführung in die Kryptographie*. Springer Verlag, 2001.
- [5] B. Chor and R. L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. on Information Theory* 34, pages 901–909, 1988.
- [6] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer Verlag, 1993.
- [7] M.J. Coster, A. Joux, B.A. LaMacchia, A. Odlyzko, C.P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity* 2, pages 111 – 128, 1992.
- [8] Claudia Eckert. *IT Sicherheit*. Oldenbourg Wissenschaftsverlag GmbH, 2003.
- [9] Dirk Fox, Stefan Gora, Stefan Kelm, and Jochen Schlichting. Sercovo security news, Mai 2005.
- [10] L. C. Guillou and J. J. Quisquater. Advances in cryptology. In *Lecture Notes in Computer Science*, volume 921. EUROCRYPT 95, SP, 21-25 May 1995.
- [11] Helmut Hasse. *Zahlentheorie*. Akademie-Verlag, 1949.
- [12] J.C. Lagarias and A.M. Odlyzko. Solving low-density subset sum problems. *Journal of the Association for Computing Machinery* 32-1, pages 229–246, 1985.
- [13] A.K. Lenstra, H.W. Lenstra, and L. Lóvasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, Band 261:515–534, 1982.

- [14] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptography*, 14(4), October 27 1999.
- [15] Alexander May. *New RSA Vulnerabilities using Lattice Reduction Methods*. PhD thesis, Universität Paderborn, Oktober 2003.
- [16] Alfred J. Menezes, Paul C. von Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [17] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. on Information Theory* 24, pages 525–530, 1978.
- [18] A.M. Odlyzko. Cryptoanalytic attacks on the multiplicative knapsack cryptosystem and on shamir’s fast signature scheme. *IEEE Trans. on Information Theory* IT-30, pages 594–601, 1984.
- [19] A.M. Odlyzko. *The future of integer factorization*, volume 1. RSA Laboratories’ Cryptobytes, 1995. <http://www.research.att.com/~amo/doc/crypto.html>.
- [20] Tatsuaki Okamoto and Keisuke Tanaka und Shigenori Uchiyama. Quantum public-key cryptosystems, 1999.
- [21] Keiji Oomura and Keisuke Tanaka. Density attack on the knapsack cryptosystem with enumerative source encoding (extended abstract). <http://www.citeseer.com>, February 2003.
- [22] Raphael Overbeck. Potential und grenzen der anwendung von gitterreduktionsalgorithmen in der kryptographie. Master’s thesis, TU Darmstadt, 2004.
- [23] R. Remmert and P. Ullrich. *Elementare Zahlentheorie*. Birkhäuser Verlag, 1995.
- [24] Katja Schmidt-Samoa. Das number field sieve – entwicklung, varianten und erfolge. Master’s thesis, Universität Kaiserslautern, 2002.
- [25] C. P. Schnorr and H. H. Hörner. Attacking the chor-rivest cryptosystem by improved lattice reduction. pages 1–12.
- [26] Uwe Schöning. *Theoretische Informatik – kurzgefaßt*. Spektrum Akademischer Verlag, 1997.
- [27] RSA Security. What is cryptography? <http://www.rsasecurity.com/rsalabs/node.asp?id=2157>, September 2005.

- [28] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing*, 26(5):1484–1509, October 1997.
- [29] Andreas Stiller. Goldener Oktober. *c't Magazin*, (23):38, 2005.
- [30] Tsuyoshi Takagi. Beweisbare sichere kryptographie, 2002. Vorlesungsskript.
- [31] Internetlexikon Wikipedia. <http://www.wikipedia.de>, März 2006.