

Universität des Saarlandes
Fachrichtung 6.2, Informatik
Lehrstuhl für Kryptographie und Sicherheit
Prof. Dr. Birgit Pfitzmann



Diplomarbeit

Neue zahlentheoretische Annahmen in der Kryptographie

Michael Backes
<mbackes@cs.uni-sb.de>

Betreuer:
Prof. Dr. Birgit Pfitzmann

22. Februar 2001

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbständig verfasst, nur die angegebenen Quellen verwendet und sie noch keinem anderen Prüfungsamt vorgelegt habe.

Saarbrücken, den 22. Februar 2001

Zusammenfassung

Die Diplomarbeit behandelt neue zahlentheoretische Annahmen mit ihren Anwendungen in der Kryptographie. Nach einer kurzen Einführung in die erforderlichen kryptographischen sowie mathematischen Grundlagen werden bekannte Annahmen vorgestellt, die die Grundlage für die Sicherheit zahlreicher aktueller Systeme bilden.

Im darauffolgenden Kapitel werden neue zahlentheoretische Annahmen eingeführt, d.h. Annahmen, die erst vor kurzer Zeit aufgestellt wurden und deren Sicherheit noch nicht in größerem Maße untersucht wurde. Desweiteren wird zu jeder Annahme ein auf ihr basierendes Kryptosystem vorgestellt, um einen praktischen Nutzen dieser Annahme aufzuzeigen.

Die Annahmen selbst werden in Reduktionsbeweisen miteinander verglichen mit dem Ziel, zwischen ihnen eine hierarchische Struktur bzgl. ihrer Schwierigkeit zu erstellen und somit auch die aufgeführten Systeme bzgl. ihrer Sicherheit zu vergleichen.

Abschließend wird auf die Orakelbitsicherheit sowie auf die partielle Bitsicherheit ausgewählter Annahmen eingegangen.

Abstract

This diploma thesis deals with new number-theoretic assumptions and how they are used in cryptography. After a short introduction to some necessary basics of mathematics and cryptography well-known assumptions are presented that serve as the foundation of the security of many cryptosystems.

The following chapter introduces new number-theoretic assumptions, i.e. assumptions that were presented in the recent past and whose security has not been investigated very much. Furthermore, to show a practical use, for every assumption a new cryptosystem is described that can be proved to be secure under this assumption.

The assumptions themselves are compared using reduction proofs in order to develop a hierarchy according to their computational hardness. This also yields a comparison between the security of the described cryptosystems.

Finally some selected assumptions are analysed with regard to their oracle bit security and their security against partial bit loss.

Dank

An erster Stelle möchte ich Frau Prof. Dr. Birgit Pfitzmann für die Ausgabe des interessanten Themas und für die ausgezeichnete Betreuung danken, die die Erstellung dieser Arbeit erst möglich gemacht hat. Bei den zahlreichen Problemen und Fragen, die sich mir im Laufe der Arbeit stellten, hat sie sich stets die Zeit genommen, mir diese ausführlich zu erklären. Insbesondere ihre Korrekturen und Anmerkungen zu den Vorabversionen der Arbeit waren zur Fertigstellung sehr hilfreich. Für Korrekturen und Anmerkungen danke ich weiterhin Dirk Leinenbach, Alexander Geraldy und Peter Kneifel ganz herzlich.

Meinen Eltern möchte ich für ihre finanzielle und ideelle Unterstützung meines Studiums danken. Ein besonderer Dank geht an meine Freunde, die mich des öfteren daran erinnern mussten, dass es andere, ebenso wichtige (oder wichtigere) Dinge wie das Studium gibt.

Saarbrücken, 22. Februar 2001

Michael Backes

Inhaltsverzeichnis

1	Einleitung	1
2	Grundlagen	3
2.1	Grundlagen der Algebra	3
2.2	Grundlagen der elementaren Zahlentheorie	6
2.3	Grundlagen der Wahrscheinlichkeitsrechnung	8
2.4	Public-Key Verschlüsselung	9
2.5	Signaturssysteme	11
2.6	Probabilistische Algorithmen	12
2.7	Hashfunktionen	12
2.8	Definitionen von Sicherheit	14
2.8.1	Sicherheit gegen passive Angriffe	15
2.8.2	Sicherheit gegen aktive Angriffe	16
2.9	Das Random Oracle Modell	19
3	Bekannte kryptographische Annahmen	21
3.1	Das Faktorisierungsproblem (<i>Integer Factorisation Problem</i>)	21
3.2	Das Diskreter Logarithmus Problem (<i>Discrete Logarithm Problem</i>)	23
3.3	Das RSA Problem	24
3.4	Das Diffie Hellman Problem	26
3.5	Das Wurzel-Problem (<i>Square Root Problem</i>)	26
3.6	Das Quadratische-Reste-Problem (<i>Quadratic Residuosity Assumption</i>)	27
3.7	Nichtzahlentheoretische Probleme	27
4	Einführung neuer Annahmen	29
4.1	Übersicht	29
4.2	Das <i>Diffie Hellman Decision Problem</i>	29
4.2.1	Definition	29
4.2.2	Das Verschlüsselungssystem von R. Cramer, V. Shoup	31
4.3	Höhere Reste (<i>Higher Residues</i>)	32
4.3.1	Einführung	32
4.3.2	Definition der Annahmen	36
4.3.3	Das Verschlüsselungssystem von D. Naccache und J.Stern	43
4.3.4	Das Verschlüsselungssystem von T. Okamoto und S. Uchiyama	44
4.3.5	Das Verschlüsselungssystem von Pascal Paillier	46
4.3.6	Die Systeme in der Übersicht	48
4.4	Die Starke RSA Annahme (<i>Strong RSA Assumption</i>)	49

4.4.1	Definitionen	49
4.4.2	Das Signatursystem von R. Cramer, V. Shoup	50
4.5	Die Dependent RSA Annahme (<i>Dependent RSA Assumption</i>)	52
4.5.1	Definitionen	52
4.5.2	Das Verschlüsselungssystem von D. Pointcheval	53
4.6	Die ϕ -Hiding Assumption	55
4.6.1	Definition und Notation	55
4.6.2	Das Private Information Retrieval System von C. Cachin, S. Micali, M. Stadler	56
4.7	Das Nullstellenproblem (<i>Root Finding Problem</i>)	58
4.7.1	Definitionen	59
4.7.2	Das Verschlüsselungssystem von J. Schwenk, J. Einfeld	59
4.7.3	Das Signatursystem von J. Schwenk, J. Einfeld	60
5	Die verschiedenen Annahmen im Vergleich	63
5.1	Einleitung	63
5.2	Übersicht	63
5.3	Der RSA Graph	65
5.3.1	<i>Integer Factorisation Problem</i> \Rightarrow <i>RSA Problem</i>	65
5.3.2	<i>Integer Factorisation Problem</i> \Rightarrow <i>Root Finding Problem</i>	65
5.3.3	<i>RSA Problem</i> \Rightarrow <i>Strong RSA Problem</i>	65
5.3.4	<i>RSA Problem</i> \Rightarrow <i>Computational Dependent RSA Problem</i>	65
5.3.5	<i>Computational Dependent RSA Problem</i> \Rightarrow <i>RSA Problem</i> ?	66
5.4	Der Diskreter Log. Graph	70
5.4.1	<i>Discrete Logarithm Problem</i> \Rightarrow <i>Diffie Hellman Problem</i>	70
5.4.2	<i>Diffie Hellman Problem</i> \Rightarrow <i>Diffie Hellman Decision Problem</i>	70
5.5	Der Höhere Reste Graph	70
5.5.1	<i>Higher Residuosity Problem</i> \Rightarrow <i>Decisional Higher Residuosity Problem</i>	71
5.5.2	$\text{Class}[P^2 Q^2] \Rightarrow D\text{-Class}[P^2 Q^2]$	71
5.5.3	$D\text{-Class}[P^2 Q^2] \Leftrightarrow CR[P^2 Q^2]$?	71
5.5.4	$\text{Class}[P^2 Q^2] \Rightarrow PDL[P^2 Q^2, g]$	72
5.5.5	$D\text{-Class}[P^2 Q^2] \Rightarrow D\text{-PDL}[P^2 Q^2, g]$	72
5.6	Graphverbindungsbeispiele	73
5.6.1	<i>Integer Factorisation Problem</i> \Leftrightarrow <i>Square Root Problem</i>	73
5.6.2	<i>Integer Factorisation Problem</i> \Rightarrow <i>Higher Residuosity Problem</i>	75
5.6.3	<i>Integer Factorisation Problem</i> \Rightarrow <i>Decisional Composite Residuosity Problem</i>	76
5.6.4	<i>RSA Problem</i> \Rightarrow $CR[P^2 Q^2]$?	76
5.6.5	<i>Composite Discrete Logarithm Problem</i> \Rightarrow <i>Strong RSA Problem</i>	77
6	Orakel-Bitsicherheiten der einzelnen Annahmen	79
6.1	Einleitung	79
6.2	Notation und Definitionen	80
6.3	Bitsicherheit der RSA-Funktion	81
6.3.1	Übersicht	81
6.3.2	Die Sicherheit des niederwertigsten Bits	82
6.3.3	Die Sicherheit der $O(\log l)$ niederwertigsten Bits	87
6.3.4	Die Sicherheit der mittleren RSA-Bits	88

6.3.5	Die Sicherheit der $O(\log l)$ höherwertigsten Bits	92
6.3.6	Simultane Bitsicherheit	94
6.3.7	Bitsicherheit der abgeleiteten Annahmen	94
6.4	Bitsicherheit der Diskreter Log. Annahme	94
7	Partielle Bitsicherheit der einzelnen Annahmen	97
7.1	Einleitung	97
7.2	Bitsicherheit der Faktorisierungsannahme	97
7.3	Bitsicherheit der RSA-Annahme	98
7.3.1	Überblick	98
7.3.2	Notation und ein bekanntes Theorem	99
7.3.3	Beweis der Theoreme 6 und 7	100
A	Tabellarische Übersicht der Annahmen	111

Kapitel 1

Einleitung

In unserer heutigen Zeit nimmt die Kryptographie einen immer größer werdenden Stellenwert ein. Besonders das stetig wachsende Internet und die damit verbundenen zahlreichen Möglichkeiten wie Internetshopping, Onlinebanking etc. fordern geradezu einen Schutz der ehrlichen Benutzer, den die Kryptographie liefern kann. Der wohl wichtigste Aspekt neben der Korrektheit eines kryptographischen Algorithmus ist seine Sicherheit. Sie kann die ehrlichen Benutzer in ihrem Vertrauen bestärken, dass durch das verwendete System ihre geheimen Daten, wie z.B. ihre Kontonummer oder ein geheimes Passwort, effektiv vor Angriffen geschützt sind. Wie in nahezu allen technischen und naturwissenschaftlichen Gebieten gibt man sich auch in der Kryptographie nicht mit anschaulichen Begründungen bzgl. der Sicherheit eines betrachteten Systems zufrieden, sondern man ist stets bemüht, diese auch zu *beweisen*.

In der Kryptographie werden solche Beweise mit Reduktionsbeweisen auf sogenannte Annahmen durchgeführt. Annahmen sind Probleme meist mathematischer Herkunft, die bereits seit langer Zeit untersucht wurden und zu deren Lösungen immer noch kein effizienter Algorithmus bekannt ist. Folglich ist es sinnvoll davon auszugehen, dass ein solcher Algorithmus auch von keinem Angreifer gefunden wird, und man baut die Sicherheitsbeweise auf dieser Hoffnung auf. Formal zeigt man, dass ein Angreifer, der das betrachtete System brechen kann, auch die zugrundeliegende Annahme brechen kann und hofft somit auf einen Widerspruch. Es stellt sich folglich die Frage, welche Probleme als Annahmen in Frage kommen.

Kapitel 2 stellt eine kurze Einführung in die für diese Arbeit notwendigen kryptographischen und mathematischen Grundlagen dar. Insbesondere aus dem großen Gebiet der Mathematik werden lediglich grundlegende Sachverhalte der linearen Algebra sowie der Zahlentheorie vorgestellt.

Kapitel 3 beschäftigt sich mit den seit langem bekannten, wichtigen zahlentheoretischen Problemen, wie z.B. dem “Faktorisierungsproblem” oder dem “Diskreter Logarithmus Problem”. Neben einer anschaulichen Erklärung werden die Probleme formal eingeführt, um den Leser mit der in den folgenden Kapiteln benutzten Notation vertraut zu machen.

Kapitel 4 behandelt schließlich neue zahlentheoretische Annahmen und ihre Anwendungen in modernen Kryptosystemen. Hier finden sich Annahmen, die im Laufe der letzten Jahre aufgestellt wurden und die die Grundlage für die Sicherheit zahlreicher kryptographischer Systeme bilden. Um ihre möglichen Anwendung zu verdeutlichen, wird zu jeder Annahme ein darauf aufbauendes System vorgestellt.

Da diese Probleme offensichtlich noch nicht in dem Maße untersucht wurden wie die seit langem bekannten Annahmen, bleibt die Frage bestehen, wie schwer ein solches Problem nun

wirklich ist. Um dies zu untersuchen, werden die Annahmen in *Kapitel 5* durch Reduktionsbeweise miteinander verglichen und in einer hierarchischen Struktur angeordnet. Dieses Kapitel bildet im wesentlichen den Kern der Arbeit, da es neben einer Fülle an bekannten Reduktionsbeweisen auch einen neuen Beweis enthält, der die bis zu diesem Zeitpunkt bestehende hierarchische Struktur um eine Reduktion erweitert.

Bis zu dieser Stelle der Arbeit werden Geheimnisse im wesentlichen als Ganzes betrachtet; interessant ist jedoch die Frage, welchen partiellen Informationsgewinn ein Angreifer erlangen kann, was zum Begriff der Bitsicherheit führt. Bitsicherheitsbetrachtungen werden seit jeher durchgeführt, da sich ihre Ergebnisse oftmals unmittelbar auf die semantische Sicherheit von Systemen übertragen läßt. Ein weiteres, neueres Anwendungsgebiet besteht darin, dem Angreifer ein gewisses "Vorwissen" mitzugeben, was in der Praxis oftmals realistisch ist. Dem Angreifer könnte es zum Beispiel möglich sein, bei der Eingabe eines geheimen Passwortes einen kurzen Blick auf die Tastatur zu erhaschen, bzw. im elektronischen Sinn könnte es ihm möglich sein, einen Teil eines geheimen Schlüssels durch sogenannte *timing attacks* für sich zu gewinnen. Mit dem Begriff der Bitsicherheit und der Sicherheit der Annahmen bei einem Vorwissen des Angreifers beschäftigen sich *Kapitel 6* und *7*.

Abschliessend sollte gesagt werden, dass insbesondere alle asymmetrischen Systeme in der Kryptographie auf Annahmen aufbauen. Die Sicherheit der Annahmen spielt eine Schlüsselrolle in fast allen in der Praxis verwendeten Systemen und sollte in Zukunft weiter untersucht werden.

Kapitel 2

Grundlagen

Dieses Kapitel stellt eine kurze Einführung in die Grundlagen der Mathematik, der Wahrscheinlichkeitsrechnung sowie der Kryptographie dar, die zum Verständnis der folgenden Kapitel nötig sind. Insbesondere aus dem großen Gebiet der Mathematik werden lediglich die für die Kryptographie wichtigen Teilgebiete der Algebra sowie der elementaren Zahlentheorie herausgegriffen.

2.1 Grundlagen der Algebra

Elementare Kenntnisse der Algebra sind heutzutage in der Kryptographie zu einer unverzichtbaren Voraussetzung geworden. Mit ihrer Hilfe läßt sich die Korrektheit von Algorithmen beweisen, neue Systeme werden üblicherweise aufgrund einer algebraischen Intuition konstruiert, um dann wiederum mit algebraischen Mitteln bewiesen zu werden. Der folgende Abschnitt enthält einen kurzen Überblick über wichtige algebraische Definitionen sowie bekannte Sätze, die im weiteren Verlauf der Arbeit Anwendung finden:

Definition 2.1 (Gruppe). Sei G eine Menge. Ein Tupel (G, \circ) mit einer Verknüpfung $\circ : G \times G \rightarrow G$ heißt eine Gruppe, wenn gilt:

1. *Assoziativität:* $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$.
2. *Existenz des neutralen Elementes:* $\exists e \in G \forall a \in G : e \circ a = a \circ e = a$.
3. *Existenz des Inversen:* $\forall a \in G \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$.

Gilt des weiteren $a \circ b = b \circ a$ für alle $a, b \in G$, so heißt \circ kommutativ. Die Gruppe (G, \circ) heißt dann kommutativ oder abelsch. Anstelle von (G, \circ) wird im allgemeinen kurz G geschrieben, die dazugehörige Verknüpfung ergibt sich aus dem Kontext. Um weitere Schreibarbeit zu sparen, hat es sich eingebürgert, die Verknüpfung \circ durch eine übliche Multiplikation zu ersetzen, d.h. man definiert $ab := a \cdot b := a \circ b$. Man spricht von einer multiplikativen Schreibweise. Das neutrale Element der Gruppe wird gemäß dieser Notation mit 1 bezeichnet. Nun kann man die natürlichen Potenzen eines Elementes a definieren durch $a^0 := 1$ und $a^{n+1} := aa^n$. Um auch negative Exponenten zuzulassen definiert man weiter $a^{-n} := (a^{-1})^n$.

Definition 2.2 (Generator). Ein Element $a \in G$ heißt ein Generator der Gruppe G , wenn gilt: $\forall b \in G \exists x \in \mathbb{Z} : a^x = b$.

Definition 2.3 (Erzeugnis). Sei G eine Gruppe. Zu einem $a \in G$ definiert man das Erzeugnis von a durch $\langle a \rangle := \{a^x \mid x \in \mathbb{Z}\}$.

Offensichtlich ist damit a ein Generator der Gruppe $\langle a \rangle$.

Definition 2.4 (Zyklische Gruppe). Eine Gruppe G heißt genau dann zyklisch, wenn sie einen Generator besitzt.

Definition 2.5 (Ordnung einer Gruppe). Sei G eine Gruppe. Unter der Ordnung von G versteht man die Anzahl ihrer Elemente. Die Ordnung wird mit $|G|$ bezeichnet.

Definition 2.6 (Ordnung eines Elementes). Sei G eine endliche Gruppe. Unter der Ordnung $\text{ord}(a)$ eines Elementes $a \in G$ versteht man die kleinste natürliche Zahl x , so dass gilt $a^x = 1$, d.h. $\text{ord}(a) = \min\{x \in \mathbb{N} \mid a^x = 1\}$.

Definition 2.7 (Untergruppe). Ein Teilmenge $S \subseteq G$ einer Gruppe G heißt Untergruppe von G , wenn S bzgl. der auf S eingeschränkten Verknüpfung von G wieder eine Gruppe ist.

Satz 2.8 (Lagrange). Sei eine Gruppe G und eine Untergruppe S von G gegeben. Dann teilt die Ordnung der Untergruppe die Ordnung der Gesamtgruppe, d.h. es existiert ein $k \in \mathbb{N}$ mit $k \cdot |S| = |G|$.

Der genaue Begriff der Teilbarkeit wird im zahlentheoretischen Abschnitt eingeführt.

Definition 2.9 (Ring). Sei R eine Menge. Ein Tupel $(R, +, \cdot)$ mit Verknüpfungen $+, \cdot : R \times R \rightarrow R$ heißt Ring, wenn gilt

1. $(R, +)$ ist eine abelsche Gruppe.
2. Assoziativität von $\cdot : \forall a, b, c \in R : a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
3. Distributivgesetze: $\forall a, b, c \in R : a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ und $\forall a, b, c \in R : (a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

Existiert desweiteren ein neutrales Element bzgl. der Verknüpfung \cdot so heißt der Ring unitär, ist die Verknüpfung \cdot kommutativ, so heißt der Ring kommutativ. Auch hier kürzt man das Tupel $(R, +, \cdot)$ einfach durch R ab, das eventuell vorhandene neutrale Element der Multiplikation wird mit 1 bezeichnet. Ein wichtiges Beispiel ist der Ring \mathbb{Z}_n , auf den in Abschnitt 2.2 näher eingegangen wird.

Definition 2.10 (Einheit). Sei R ein unitärer Ring. Ein Element $a \in R$ heißt Einheit, wenn ein $a' \in R$ existiert mit $a' \cdot a = a \cdot a' = 1$.

Die Menge aller Einheiten eines Ringes bildet selbst wieder eine Gruppe mit der Multiplikation als Verknüpfung. Sie wird mit R^* bezeichnet.

Definition 2.11 (Nullteiler). Sei R ein Ring und $a \in R$. Existiert ein $b \in R$ mit $b \neq 0$ und $a \cdot b = 0$ bzw. $b \cdot a = 0$, so heißt ein a Nullteiler von R , sofern R kommutativ ist. Im nichtkommutativen Fall heißt a Linksnullteiler bzw. Rechtsnullteiler.

Ein Ring, der keine Nullteiler $a \neq 0$ besitzt, heißt nullteilerfreier Ring.

Definition 2.12 (Integritätsbereich). *Ein kommutativer, unitärer, nullteilerfreier Ring heißt Integritätsbereich bzw. Integritätsring.*

Definition 2.13 (Irreduzibles Element). *Sei R ein kommutativer, unitärer Ring. Ein Element $a \in R$ heißt irreduzibel, wenn gilt:*

1. $a \notin R^*$.
2. $a = bc \Rightarrow b \in R^*$ oder $c \in R^*$.

Die Definition der irreduziblen Elemente eines Ringes ist eng verbunden mit den sogenannten Primelementen bzw. im Fall $R = \mathbb{Z}$ den Primzahlen und ihren additiven Inversen. Sie werden im Abschnitt 2.2 eingeführt.

Definition 2.14 (Polynome). *Sei R ein Ring und X ein neues, nicht zu R gehöriges Symbol. Die Ausdrücke $f(X) = \sum_{i=1}^t a_i X^i$ mit endlich vielen $a_i \in R$ für $i = 1, \dots, t$ heißen Polynome über R .*

Auf Polynomen kann man wie folgt eine Ringstruktur einführen: Zu den Polynomen $f(X) = \sum a_\nu X^\nu$, $g(X) = \sum b_\mu X^\mu$ definiert man die Addition $f(X) + g(X) := \sum c_\sigma X^\sigma$ durch

$$c_\nu = a_\nu + b_\nu.$$

bzw. die Multiplikation $f(X) \cdot g(X) := h(X) = \sum c_\sigma X^\sigma$ durch

$$c_\sigma = \sum_{\nu+\mu=\sigma} a_\nu b_\mu.$$

Definition 2.15 (Polynomring). *Sei R ein Ring. Der Ring mit den oben eingeführten Verknüpfungen wird als Polynomring, i.Z. $R[X]$ bezeichnet.*

Definition 2.16 (Grad eines Polynoms). *Sei R ein Ring. Der Grad eines Polynoms $f(X) = \sum a_\nu X^\nu \in R[X]$, i.Z. $\deg(f)$, wird definiert durch $\deg(f) := \max\{\nu \mid a_\nu \neq 0\}$. Der Koeffizient $a_{\deg(f)}$ heißt der höchste Koeffizient bzw. der Leitkoeffizient des Polynoms.*

Definition 2.17 (Körper). *Sei K eine Menge. Ein Tupel $(K, +, \cdot)$ mit Verknüpfungen $+, \cdot : K \times K \rightarrow K$ heißt Körper, wenn gilt*

1. $(K, +, \cdot)$ ist ein kommutativer, unitärer Ring.
2. Für alle $a \in K \setminus \{0\}$ existiert ein Inverses bzgl. der multiplikativen Verknüpfung.

Wie bereits bei Ringen bezeichnet man auch hier den Körper nur mit K .

Satz 2.18. *Sei K ein Körper. Dann hat jedes Polynom $f(X) \neq 0$ über $K[X]$ mit Grad $d = \deg(f)$ höchstens d Nullstellen.*

2.2 Grundlagen der elementaren Zahlentheorie

Neben den im vorigen Abschnitt eingeführten Grundlagen der Algebra spielt die elementare Zahlentheorie in der Kryptographie eine entscheidende Rolle. Sie beschäftigt sich mit den Eigenschaften der ganzen Zahlen, das heißt im folgenden wird stets der Ring \mathbb{Z} als Grundlage angesehen. Der grundlegende Begriff in der Zahlentheorie ist der Begriff der Teilbarkeit:

Definition 2.19 (Teilbarkeit). Sei $m, n \in \mathbb{Z}$, $m \neq 0$. Dann heißt m ein Teiler von n , in Zeichen $m|n$, wenn gilt: $\exists q \in \mathbb{Z} : n = mq$. Gleichbedeutend damit sind Sprechweisen wie: n ist durch m teilbar oder n ist ein Vielfaches von m .

Definition 2.20 (Primelemente). Sei R ein Ring. Ein Element $a \in R$ heißt Primelement oder prim, wenn gilt:

1. $a \notin R^*$.
2. $a|bc \Rightarrow a|b$ oder $a|c$.

Die Primelemente im Ring \mathbb{Z} sind genau die Primzahlen und ihre additiven Inversen. Im Ring \mathbb{Z} besitzt jedes Element $n \neq 0$ eine eindeutige Zerlegung in Primfaktoren, d.h. es gibt eindeutig bestimmte Primzahlen p_i und $\nu_i \in \mathbb{N}$ mit $n = \pm \prod_{i=1}^t p_i^{\nu_i}$.

Definition 2.21 (ZPE-Ring). Ein Ring R , bei dem jedes Element eine eindeutige Zerlegung in Primelemente besitzt, heißt ZPE-Ring (engl. unique factorisation domain).

Definition 2.22 (Kongruenzen). Seien $n, a, b \in \mathbb{Z}$, $n \neq 0$. Dann heißt a kongruent (zu) b modulo n , in Zeichen $a \equiv b \pmod{n}$, wenn gilt: $n|(a - b)$.

Die Relation "kongruent modulo n " ist eine Äquivalenzrelation auf \mathbb{Z} und unterteilt \mathbb{Z} somit in disjunkte Klassen, die sogenannten Restklassen modulo n . Zu jedem $a \in \mathbb{Z}$ wird die korrespondierende Restklasse durch $\bar{a} = \{b \in \mathbb{Z} | b \equiv a \pmod{n}\}$ bestimmt. Die Menge \mathbb{Z}_n wird durch $\mathbb{Z}_n := \{\bar{a} | a \in \mathbb{Z}\}$ zu einem kommutativen, unitären Ring, wobei die Verknüpfungen zweier Restklassen komponentenweise wohldefiniert durchgeführt werden. Diese komponentenweise Definition führt zum Begriff des Repräsentantensystems bzw. Restsystems.

Definition 2.23 (Restsystem). Eine Menge $R := \{a_0, \dots, a_{n-1}\}$ heißt Repräsentantensystem von \mathbb{Z}_n , wenn für alle $i \neq j$ mit $i, j \in \{0, \dots, n-1\}$ die Elemente a_i und a_j verschiedenen Restklassen angehören.

Das am häufigsten benutzte Restsystem von \mathbb{Z}_n ist die Menge $R = \{0, \dots, n-1\}$, das sogenannte kleinste nichtnegative Restsystem. Im folgende wird stets dieses Repräsentantensystem verwendet, d.h. der Ausdruck $a \in \mathbb{Z}_n$ entspricht dem Element $a_i \in R$, das der Restklasse von a angehört.

Zu dem Ring \mathbb{Z}_n wird gemäß (2.1) die multiplikative Gruppe der Elemente, die ein Inverses besitzen, mit \mathbb{Z}_n^* bezeichnet. Es gilt das folgende einfache Kriterium: $a \in \mathbb{Z}_n^* \Leftrightarrow ggT(a, n) = 1$.

Satz 2.24. Die Gruppe \mathbb{Z}_n^* ist genau dann zyklisch, wenn für n einer der folgenden Fälle gilt:

1. $n = 1, 2, 4$.
2. $n = p^\nu$ mit $\nu \in \mathbb{N}$, $p \neq 2$ und p prim.

3. $n = 2p^\nu$ mit $\nu \in \mathbb{N}$, $p \neq 2$ und p prim.

Satz 2.25. Der Ring \mathbb{Z}_n ist genau dann ein Körper, wenn n eine Primzahl ist.

Definition 2.26 (Eulersche Phi-Funktion). Sei $n = p_1^{\nu_1} \cdots p_t^{\nu_t}$ die Primfaktorzerlegung von n . Die Funktion $\varphi : \mathbb{N} \rightarrow \mathbb{N}$ mit

$$\varphi(n) := \prod_{i=1}^t (p_i^{\nu_i} - p_i^{\nu_i-1})$$

heißt Eulersche Phi-Funktion.

Es gilt $\varphi(n) = |\mathbb{Z}_n^*|$.

Theorem 1 (Chinesischer Restsatz). Seien natürliche Zahlen $a_1, \dots, a_t, m_1, \dots, m_t$ gegeben. Die Zahlen m_1, \dots, m_t seien paarweise teilerfremd, $m := \prod_{i=1}^t m_i$. Dann besitzt das System der Kongruenzen

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ &\vdots \\ x &\equiv a_t \pmod{m_t} \end{aligned}$$

eine eindeutige Lösung modulo m .

Der Beweis des Satzes liefert desweiteren ein konstruktives Verfahren zur Bestimmung der gesuchten Lösung.

Ein weiterer wichtiger Begriff aus der elementaren Zahlentheorie ist der Begriff eines quadratischen Restes.

Definition 2.27 (Quadratischer Rest). Sei $n \in \mathbb{N}$. Ein Element $a \in \mathbb{Z}_n^*$ heißt quadratischer Rest modulo n , wenn ein $x \in \mathbb{Z}_n^*$ existiert mit $x^2 \equiv a \pmod{n}$; ansonsten heißt a ein quadratischer Nichtrest modulo n .

Die Menge aller quadratischen Reste modulo n wird mit QR_n bezeichnet, die Menge der quadratischen Nichtreste bisweilen mit NQR_n . Anhand dieser Definition lassen sich nun das Legendresymbol und das Jacobisymbol definieren.

Definition 2.28 (Legendre-Symbol). Seien $c \in \mathbb{Z}$, $2 \neq p$ prim. Desweiteren sei c nicht durch p teilbar. Dann setzt man

$$\left(\frac{c}{p}\right) := \begin{cases} 1, & \text{falls } c \in QR_p \\ -1, & \text{falls } c \in NQR_p \end{cases}$$

Das Symbol $\left(\frac{c}{p}\right)$ heißt Legendre-Symbol.

Satz 2.29 (Satz von Euler). Sei $c \in \mathbb{Z}$, $2 \neq p$ prim mit $p \nmid c$, dann gilt $\left(\frac{c}{p}\right) \equiv c^{(p-1)/2} \pmod{p}$.

Satz 2.30. Es gilt: $c \in \mathbb{Z}_p^*$ ist quadratischer Rest modulo $p \Leftrightarrow \left(\frac{c}{p}\right) = 1$.

Definition 2.31 (Jacobi-Symbol). Sei $c \in \mathbb{Z}$ und $2 \nmid n \in \mathbb{N}$ mit $\text{ggT}(c, n) = 1$, so definiert man das Jacobi-Symbol $\left(\frac{c}{n}\right)$ für $n = p_1^{\nu_1} \cdots p_k^{\nu_k}$ durch

$$\left(\frac{c}{n}\right) := \left(\frac{c}{p_1}\right)^{\nu_1} \cdots \left(\frac{c}{p_k}\right)^{\nu_k}$$

Satz 2.32. Für $c \in \mathbb{Z}_n^*$, $2 \nmid n \in \mathbb{N}$ gilt: $c \in QR_n \Rightarrow \left(\frac{c}{n}\right) = 1$.

Die Menge der Elemente c mit $\left(\frac{c}{n}\right) = 1$ wird mit \mathbb{Z}_n^{+1} bezeichnet.

Satz 2.33. Sei $n = pq$ mit p, q prim. Es gilt $|\mathbb{Z}_n^{+1} \setminus QR_n| = |QR_n|$, d.h. es gibt genauso viele quadratische Reste wie quadratische Nichtreste mit Jacobi-Symbol $= 1$.

Definition 2.34 (e-ter Rest modulo n). Sei $n \in \mathbb{N}$. Eine Zahl $z \in \mathbb{Z}_n^*$ heißt ein e -ter Rest modulo n , wenn ein $x \in \mathbb{Z}_n^*$ existiert mit $z \equiv x^e \pmod{n}$. Für $e > 2$ spricht man von einem höheren Rest.

2.3 Grundlagen der Wahrscheinlichkeitsrechnung

Definition 2.35 (Wahrscheinlichkeitsverteilung). Sei $S = \{s_i \mid i \in I\}$ eine Menge für eine Indexmenge I . Eine Wahrscheinlichkeitsverteilung P auf S ist eine Familie von nichtnegativen Zahlen $(p_i)_{s_i \in S}$ mit $\sum_{s_i \in S} p_i = 1$.

Die Zahl p_i wird als die Wahrscheinlichkeit definiert, dass das Elementarereignisses s_i das Ergebnis des Experimentes darstellt. Der Ausdruck $s \in_P S$ bedeutet, dass ein Element aus S bzgl. der Wahrscheinlichkeitsverteilung P gezogen wird. Ist die Verteilung P aus dem Kontext bekannt, so schreibt man einfach $s \leftarrow S$. Im Falle einer endlichen Menge S spricht man von einer endlichen Wahrscheinlichkeitsverteilung. Sei im folgenden eine endliche Menge S vorausgesetzt.

Definition 2.36 (Uniforme Wahrscheinlichkeitsverteilung). Eine Wahrscheinlichkeitsverteilung P heißt uniform, wenn für alle Elementarereignisse s_i gilt: $P(s_i) = \frac{1}{|S|}$. In diesem Fall schreibt man $s \in_R S$ anstelle von $x \in_P S$.

Im folgenden sei S ein Wahrscheinlichkeitsraum mit einer Wahrscheinlichkeitsverteilung P .

Definition 2.37 (Ereignis). Ein Ereignis E ist eine Teilmenge des Wahrscheinlichkeitsraums S .

Definition 2.38 (Wahrscheinlichkeit eines Ereignisses). Die Wahrscheinlichkeit, dass Ereignis E bzgl. P eintritt, i.Z. $P(E)$, ist die Summe aller Wahrscheinlichkeiten p_i der Elementarereignisse s_i mit $s_i \in E$, also $P(E) = \sum_{s_i \in E} p_i$. Ist $s_i \in S$, so schreibt man $P(s_i)$ anstelle von $P(\{s_i\})$.

Definition 2.39 (Komplementärereignis). Sei E ein Ereignis. Unter dem Komplementärereignis \bar{E} versteht man die Wahrscheinlichkeit des Mengenkomplements von E in S .

Definition 2.40 (Bedingte Wahrscheinlichkeit). Seien E_1 und E_2 Ereignisse mit $P(E_2) > 0$. Die bedingte Wahrscheinlichkeit von E_1 bei gegebenem Ereignis E_2 , i.Z. $P(E_1|E_2)$, wird durch

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)}$$

definiert.

$P(E_1|E_2)$ entspricht der Wahrscheinlichkeit, dass E_1 eintritt unter der Voraussetzung, dass E_2 eingetreten ist.

Definition 2.41 (Zufallsvariablen). Eine Zufallsvariable X auf S ist eine Funktion $X : S \rightarrow \mathbb{R}$, also eine Funktion, die jedem Elementarereignis s_i eine reelle Zahl $X(s_i)$ zuweist.

Definition 2.42 (Erwartungswert). Sei X eine Zufallsvariable auf S . Der Erwartungswert von X , i.Z. $E(X)$, wird definiert durch

$$E(X) = \sum_{s_i \in S} X(s_i)P(s_i).$$

Definition 2.43 (Varianz). Sei X eine Zufallsvariable auf S mit $\mu = E(X)$. Die Varianz von X , i.Z. $\text{Var}(X)$ wird definiert durch

$$\text{Var}(X) = E((X - \mu)^2).$$

Theorem 2 (Tschebychevsche Ungleichung). Sei X eine Zufallsvariable auf S mit Erwartungswert $\mu = E(X)$ und Varianz $\sigma^2 = \text{Var}(X)$. Dann gilt für alle $t > 0$

$$P(|X - \mu| \geq t) \leq \frac{\sigma^2}{t^2}.$$

In der Kryptographie führt man die folgenden Notationen ein, die das weitere Verständnis erheblich erleichtern. Sei S_1 ein Wahrscheinlichkeitsraum, S_2, \dots, S_n seien Übergangswahrscheinlichkeiten für $n \in \mathbb{N}$, d.h. Wahrscheinlichkeitsräume S_i , deren Wahrscheinlichkeitsverteilungen von vorherigen Zuweisungen $x_j \leftarrow S_j$ für $j < i$ abhängen. Ist nun p ein n -stelliges Prädikat, dann bezeichnet

$$P(p(x_1, \dots, x_n) :: x_1 \leftarrow S_1; \dots; x_n \leftarrow S_n)$$

die Wahrscheinlichkeit, dass das Prädikat $p(x_1, \dots, x_n)$ wahr ist unter der Voraussetzung, dass ein aus S_i gewählter Wert der Variable x_i zugewiesen wurde für $i = 1, \dots, n$ in genau dieser Reihenfolge. Hierbei wird wie im folgenden Verlauf der Arbeit nicht mehr exakt zwischen dem Wahrscheinlichkeitsraum S und seiner Wahrscheinlichkeitsverteilung unterschieden.

Im streng mathematischen Sinn würde man obige Wahrscheinlichkeit wie folgt definieren

$$P(p(x_1, \dots, x_n) :: x_1 \leftarrow S_1; \dots; x_n \leftarrow S_n) := \sum_{x_1, \dots, x_n : p(x_1, \dots, x_n)} P_{S_1}(x_1) \cdots P_{S_n}(x_n).$$

d.h. man definiert die neu eingeführte Notation als Produkt von Wahrscheinlichkeiten der verschiedenen Räume bzw. der Übergangswahrscheinlichkeiten.

2.4 Public-Key Verschlüsselung

Die folgenden Abschnitte stellen eine kurze Einführung in verschiedene Teilgebiete der Kryptographie dar. Im Gegensatz zu den vorangehenden mathematischen Abschnitten wird auf eine streng formale Einführung der Kryptographie verzichtet mit dem Ziel, dem Leser eine leicht verständliche Übersicht zu liefern. Die formalen Definitionen werden, sofern sie benötigt werden,

erst nach der eigentlichen Einführung vorgestellt, so dass im weiteren Verlauf der Arbeit darauf Bezug genommen werden kann.

Man beginnt mit den wohl am häufigsten betrachteten kryptographischen Systemen, den Verschlüsselungssystemen. Ein Verschlüsselungssystem besteht im wesentlichen aus den drei Algorithmen *gen*, *ver* und *ent*, die für die Schlüsselgenerierung, die Ver- und die Entschlüsselung von Nachrichten bestimmt sind. Eine mögliche Unterteilung von Verschlüsselungssystemen ist die Unterteilung in symmetrische und asymmetrische Systeme:

- In einem symmetrischen System teilen beide Parteien das gleiche Geheimnis, d.h. es existiert nur ein Schlüssel k , der von allen teilnehmenden Parteien benutzt wird. Ein wichtiger Punkt ist demzufolge die Geheimhaltung sowie die Integrität des Schlüssels.
- In einem asymmetrischen System existieren unterschiedliche Schlüssel für die verschiedenen Anwendungen, z.B. für Ver- und Entschlüsselung. Man spricht im Beispiel eines Verschlüsselungssystems von einem gültigen Schlüsselpaar (sk, pk) . Der Schlüssel sk dient der Entschlüsselung und ist nur dem Erzeuger des Schlüsselpaares bekannt. Der Schlüssel pk dient der Verschlüsselung und ist öffentlich zugänglich, d.h. jeder kann Nachrichten verschlüsseln, eine Entschlüsselung ist jedoch nur mit dem geheimen Schlüssel sk möglich. Die Verteilung des Schlüssels pk muss folglich integer, jedoch nicht geheim sein. Diese Verfahren bezeichnet man mit public-key Kryptographie.

Bis zum Jahre 1976 beschäftigte man sich ausschließlich mit den symmetrischen Systemen. Gegenüber den später entdeckten asymmetrischen Systemen liegt der Vorteil der symmetrischen hauptsächlich in ihrer zum Teil wesentlich höheren Effizienz sowie in der möglichen informationstheoretischen Geheimhaltung. Die Nachteile liegen jedoch klar auf der Hand. Für jeden Kommunikationspartner wird ein neuer Schlüssel benötigt, und der Schlüsselaustausch stellt hierbei oftmals das größte Problem dar. Insbesondere durch das stetig wachsende Internet und die damit verbundenen Möglich- und Notwendigkeiten, wie z.B. Sicherheit bei Onlinebanking etc., sind die asymmetrischen Systeme in der Praxis weit verbreitet.

Die eigentliche public-key Kryptographie wurde von Diffie und Hellman eingeführt [DiHe76]. Sie führten die Bezeichnung von trapdoor-Einwegfunktionen ein, Funktionen, deren Berechnung einfach durchzuführen ist, deren Invertierung jedoch ohne die Benutzung einer zusätzlichen Information (wie z.B. eines Schlüssels sk) schwer sein sollte. Mittels einer solchen Funktion f wurde eine Nachricht m mittels $f(m)$ verschlüsselt und anhand der Identität $f^{-1}(f(m)) = m$ wieder entschlüsselt. Nicht alle trapdoor-Einwegfunktionen sind jedoch per se als Verschlüsselungsfunktion geeignet, da sie große Teile des Klartextes unverändert lassen könnten, was dem Sinn einer Verschlüsselung widerspricht.

Ein weiterer sehr wichtiger Unterschied der beiden Verschlüsselungsarten besteht in der maximal erreichbaren Sicherheit. Im Fall eines symmetrischen Systems ist eine sogenannte informationstheoretische Sicherheit möglich, d.h. selbst ein Angreifer mit unbeschränkter Rechenleistung ist nicht in der Lage anhand eines gegebenen Schlüsseltextes c den dazu korrespondierenden Klartext m zu bestimmen. Dies ist nur möglich, da der Schlüssel k geheim ist und dem Angreifer somit keine Möglichkeit gegeben wird, von einem m aus dem Nachrichtenraum M den dazugehörigen Schlüsseltext c zu bestimmen und ihm somit eine Testmöglichkeit seiner Antwort zu entziehen. Im asymmetrischen Fall ist eine informationstheoretische Sicherheit unmöglich. Ein dermaßen mächtiger Angreifer kann alle $m \in M$ mittels des öffentlichen Schlüssels pk verschlüsseln und den erhaltenen Schlüsseltext im deterministischen Fall mit dem gegebenen c vergleichen. Findet er eine solche Übereinstimmung, so hat er das gesuchte m gefunden. Folglich führt eine

vollständige Suche im Nachrichtenraum immer zum Erfolg. Im probabilistischen Fall könnte ein derart mächtiger Angreifer z.B. den Schlüsselgenerieralgorithmus *gen* mit allen Zufallszahlen als Eingabe aufrufen und für jedes erhaltene Schlüsselpaar (sk', pk') testen, ob $pk = pk'$ gilt. Der in diesem Fall berechnete geheime Schlüssel sk' kann folglich zur Entschlüsselung genutzt werden. In der Praxis ist eine vollständige Suche bei einer entsprechenden Größe von M jedoch nicht durchführbar, da ein solches Vorgehen selbst mit der heutzutage maximal verfügbaren Rechenleistung nicht in akzeptabler Zeit möglich wäre. Für eine anschauliche Definition der Sicherheit sollte man also verlangen, dass es nicht effizient möglich ist, zu einem Schlüsseltext c den dazugehörigen Klartext m zu finden bzw. partielle Informationen über m zu erlangen. Um den Begriff "effizient" zu formalisieren, betrachtet man Algorithmen mit polynomieller Laufzeit in einem Sicherheitsparameter l des Systems als effizient, alle asymptotisch langsameren Algorithmen, z.B. subexponentielle bzw. exponentielle, als ineffizient.

Sicherheit wird nun so definiert, dass kein polynomieller Algorithmus einen Vorteil zur Entschlüsselung von c mit sich bringt. Man sagt: Ein solcher Algorithmus bricht das System. Dies führt jedoch zu dem Problem, dass man in Sicherheitsbeweisen über alle polynomiellen Algorithmen allquantifizieren muss, was Beweise wohl unmöglich macht, da ein solcher Beweis insbesondere $P \neq NP$ implizieren würde. Stattdessen betrachtet man bestimmte, meist zahlentheoretische Probleme, zu deren Lösung bis jetzt kein polynomieller Algorithmus gefunden wurde und zeigt, dass ein beliebiger polynomieller Algorithmus A , der das Kryptosystem bricht, zu einem polynomiellen Algorithmus A' umgewandelt werden kann, der wiederum das zugrundeliegende zahlentheoretische Problem löst. Da ein solcher Algorithmus trotz zum Teil erheblichen Aufwandes noch nicht gefunden wurde, hofft man auf einen Widerspruch im obigen Argument, d.h. ein Angreifer wird keinen passenden Algorithmus A finden. Man spricht von der sogenannten kryptographischen bzw. komplexitätstheoretischen Sicherheit des Systems. Die heuristische Sicherheit des zugrundeliegenden schweren Problems wird als Annahme bezeichnet, wobei die Begriffe Annahme und Problem im folgenden synonym gebraucht werden. Neben bekannten Problemen, die zum Teil seit über 2000 Jahren in der Mathematik untersucht wurden, wurden in jüngster Vergangenheit zahlreiche neue Annahmen aufgestellt, deren Sicherheit sich erst noch mit fortschreitender Zeit im heuristischen Sinn bzw. in Reduktionsbeweisen bewähren muss. Mit eben diesen Annahmen beschäftigt sich diese Arbeit.

2.5 Signatursysteme

Ähnlich wie die im vorigen Abschnitt eingeführten asymmetrischen Verschlüsselungssysteme werden Signatursysteme eingeführt. Eine Signatur ist ein nachrichtenbezogener Wert, der nur von dem Signierer effizient berechnet werden kann und der mitsamt der Nachricht übermittelt wird. Das Ziel eines Signatursystems besteht darin, der korrespondierenden Partei versichern zu können, dass die Nachricht m wirklich von dem Signierer gesendet wurde. Auch hierbei wird ein Schlüssel (sk, pk) vom Benutzer erzeugt, der Schlüssel sk heißt Signierschlüssel und wird geheimgehalten, der Schlüssel pk heißt Testschlüssel und ist zusammen mit einem Algorithmus *test* öffentlich zugänglich. Der Besitzer des geheimen Schlüssels sk kann eine Nachricht m mittels eines Algorithmus *sig* signieren, d.h. er erzeugt einen Beweis $sig(sk, m)$, dass die Nachricht m von ihm stammt. Anhand des öffentlichen Schlüssels pk und des Algorithmus *test* kann nun jeder die Signatur $sig(sk, m)$ bei gegebener Nachricht m auf ihre Gültigkeit testen. Das Prinzip der kryptographischen Sicherheit sowie der zugrundeliegenden Annahmen gilt für Signatursysteme analog.

2.6 Probabilistische Algorithmen

In der Kryptographie werden probabilistische Algorithmen üblicherweise durch deterministische Turingmaschinen repräsentiert, denen ein weiteres Band, das sogenannte Zufallsband, zu Verfügung gestellt wird. Dieses Band enthält bereits zum Startzeitpunkt der Turingmaschine unendlich viele Zufallsbits; um eine Mehrfachnutzung der gleichen Bandzelle auszuschließen, darf das Band ausschließlich von links nach rechts gelesen werden, auch ein Stehenbleiben des Lesekopfes ist nicht erlaubt. Das Ergebnis eines probabilistischen Algorithmus ist stets eine Wahrscheinlichkeitsverteilung auf dem betrachteten Raum.

Ist A ein probabilistischer Algorithmus, so bezeichnet $A(i)$ den Wahrscheinlichkeitsraum auf den Ausgaben, wenn A auf Eingabe i gestartet wird. Ein weiterer wichtiger Begriff ist der Begriff einer vernachlässigbaren Wahrscheinlichkeit:

Definition 2.44 (vernachlässigbar). *Eine Wahrscheinlichkeit $P(l)$ heißt vernachlässigbar bzgl. l , in Zeichen $P(l) \leq \frac{1}{\text{poly}(l)}$, wenn für alle Polynome pol über \mathbb{N} gilt:*

$$\exists l_0 \forall l \geq l_0 : P(l) \leq \frac{1}{\text{pol}(l)}.$$

Analog spricht man von einer signifikanten Wahrscheinlichkeit $P(l)$, wenn gilt:

$$\exists \text{Polynom } \text{pol} \forall l_0 \exists l \geq l_0 : P(l) > \frac{1}{\text{pol}(l)}$$

Probabilistische Algorithmen werden in nahezu allen Kryptosystemen an den verschiedensten Stellen benutzt. Das einfachste Beispiel stellt der Schlüsselgenerieralgorithmus gen dar, der zur Schlüsselerzeugung auf Berechnungen, die auf dem ‘‘Zufallsprinzip’’ beruhen, angewiesen ist. Der Algorithmus gen bekommt als Eingabe eine Zahl $l \in \mathbb{N}$, den sogenannten Sicherheitsparameter, der in unärer Form, also als String von l Einsen übergeben wird. Dies ist nötig, da die Komplexität eines Algorithmus bezüglich seiner Eingabengänge definiert ist, folglich benötigt man eine Eingabe der Länge l , um Begriffe wie ‘‘polynomial in l ’’ zu gebrauchen, was zu der besagten unären Darstellung von l führt. Die Ausgabe des Algorithmus gen bezeichnet man als Schlüssel oder Key.

Ein weiteres Beispiel eines probabilistischen Algorithmus ist oftmals bei einer asymmetrischen Verschlüsselung gegeben: Im Falle eines asymmetrischen Verschlüsselungssystems besteht z.B. ein generelles Problem darin, dass ein Angreifer ein eventuell vorhandenes Vorwissen über die Nachricht besitzt, was die verbleibenden Möglichkeiten der Nachricht stark einschränkt. Ein typisches Beispiel wären sehr kurze Nachrichten, z.B. Zahlungsbeträge beim Onlinebanking oder kurze E-Mails, die eine vollständige Suche innerhalb der verbliebenen Möglichkeiten sinnvoll machen. Um dies zu vermeiden werden probabilistische Algorithmen bei der Verschlüsselung benutzt, d.h. es wird ein Zufallswert in die Verschlüsselung der Nachricht m mit einbezogen.

In Fall eines probabilistischen, asymmetrischen Verschlüsselungssystems führen offensichtlich zwei Verschlüsselungen derselben Nachricht m zu zwei unterschiedlichen Schlüsseltexten c und c' .

2.7 Hashfunktionen

Eine in der Kryptographie häufig anzutreffende Klasse von Funktionen sind die sogenannten Hashfunktionen. Hashfunktionen entsprechen im Prinzip den bereits angesprochenen Einweg-

funktionen, wobei oftmals weitere Eigenschaften der Funktion erwünscht sind. Sei im folgenden eine Familie von Nachrichtenräumen $(M_{pk})_{pk \in [gen(\cdot)]}$ gegeben. Desweiteren seien effiziente Algorithmen für " $\in_R M_{pk}$ " und " $m \in M_{pk}$?" bekannt. Je nach zusätzlichen Eigenschaften der Hashfunktion H unterscheidet man folgende Fälle:

- H heißt einweg, wenn es schwer ist, zu einem gegebenen Funktionswert c ein m zu finden mit $H(m) = c$.

In kryptographischer Schreibweise bedeutet dies: $\forall prob. poly. Algo. A :$

$$\begin{aligned} P(H(pk, m') = y \\ \wedge m' \in M_{pk} \quad &:: \quad pk \leftarrow gen(l); \\ & \quad m \in_R M_{pk}; \\ & \quad y := H(pk, m); \\ & \quad m' \leftarrow A(l, pk, y)) \\ &\leq \frac{1}{poly(l)}. \end{aligned}$$

- Eine Familie von Hashfunktionen $(H_i)_{i \in I}$ heißt kollisionsresistent, wenn es schwer ist, ein Tupel (m, m') mit $m \neq m'$ zu finden mit $H_j(m) = H_j(m')$, wobei H_j zufällig aus der Familie $(H_i)_{i \in I}$ gezogen wurde.

In kryptographischer Schreibweise bedeutet dies: $\forall prob. poly. Algo. A :$

$$\begin{aligned} P(H_j(pk, m) = H_j(pk, m') \\ \wedge m, m' \in M_{pk} \wedge m \neq m' \quad &:: \quad pk \leftarrow gen(l); \\ & \quad j \in_R I; \\ & \quad m, m' \leftarrow A(l, pk, j)) \\ &\leq \frac{1}{poly(l)}. \end{aligned}$$

- Eine Familie von Hashfunktionen $(H_i)_{i \in I}$ heißt universal-oneway, wenn es schwer ist, zu einem gegebenen m ein $m' \neq m$ zu finden mit $H_j(m) = H_j(m')$, wobei H_j zufällig aus der Familie $(H_i)_{i \in I}$ gezogen wurde.

In kryptographischer Schreibweise bedeutet dies: $\forall prob. poly. Algo. A :$

$$\begin{aligned} P(H_j(pk, m) = H_j(pk, m') \\ \wedge m' \in M_{pk} \wedge m \neq m' \quad &:: \quad pk \leftarrow gen(l); \\ & \quad j \in_R I; \\ & \quad m \in_R M_{pk}; \\ & \quad m' \leftarrow A(l, pk, j, m)) \\ &\leq \frac{1}{poly(l)}. \end{aligned}$$

Offensichtlich können auch diese Aussagen nur im komplexitätstheoretischen Sinn gelten, da eine vollständige Suche im Definitionsbereich der Funktion keine der drei Eigenschaften zulassen würde.

2.8 Definitionen von Sicherheit

Mit zunehmender Anzahl von Kryptosystemen wurden Sicherheitsdefinitionen nötig, um ein System als sicher bzgl. einer bestimmten Definition zu klassifizieren. Wie bereits in Abschnitt (2.4) erwähnt, unterscheidet man die informationstheoretische und die kryptographische (komplexitätstheoretische) Sicherheit. Informationstheoretische Sicherheit besagt, dass selbst bei einem gegebenen Schlüsseltext c alle Klartexte m immer noch mit der ursprünglichen Wahrscheinlichkeit einer trivialen Ratestrategie vorkommen können. Man spricht von gleichen a-posteriori und a-priori Wahrscheinlichkeiten. Diese Art der Sicherheit spielt in dieser Arbeit keine weitere Rolle, stattdessen betrachtet man die sogenannte kryptographische Sicherheit, die die höchste erreichbare Sicherheit bei public-key Systemen darstellt und folglich in einem engen Zusammenhang mit den zugrundeliegenden Annahmen des betrachteten Systems steht. Man unterscheidet hierbei Sicherheit gegen passive sowie gegen aktive Angriffe. Bevor man sich die eigentlichen Sicherheitsdefinitionen betrachtet, wird noch eingeführt, welche Arten von erfolgreichen Angriffen es überhaupt geben kann:

Erfolgreiche Angriffe auf Verschlüsselungssysteme

Erfolgreiche Angriffe auf Verschlüsselungssysteme kann man in die folgenden vier Klassen einteilen:

Total break: Der Angreifer kann beliebige Schlüsseltexte entschlüsseln, indem er den geheimen Schlüssel sk herausfindet.

Universal break: Der Angreifer kann beliebige Schlüsseltexte entschlüsseln, indem er ein zur Entschlüsselung äquivalentes Verfahren findet.

Entschlüsselung: Der Angreifer kann einen gegebenen Schlüsseltext entschlüsseln.

Partielle Information: Der Angreifer kann anhand eines gegebenen Schlüsseltextes einer Nachricht m partielle Informationen der Nachricht herausfinden.

Erfolgreiche Angriffe auf Signatursysteme

Analog zu den Verschlüsselungssystemen können die Angriffe auf Signatursysteme ebenfalls in vier Klassen eingeteilt werden:

Total break: Der Angreifer kann beliebige Nachrichten signieren, indem er den geheimen Schlüssel sk herausfindet.

Universal break: Der Angreifer kann beliebige Nachrichten signieren, indem er ein zum Signieralgorithmus äquivalentes Verfahren findet.

Selektive Fälschung: Der Angreifer kann zu einer gegebenen Nachricht m eine gültige Signatur erzeugen.

Existentielle Fälschung: Der Angreifer kann sich ein Tupel (m, sig) erzeugen, so dass sig eine gültige Signatur der Nachricht m ist.

Nun kann man sich den eigentlichen Sicherheitsdefinitionen zuwenden. Man beginnt mit der Sicherheit gegen passive Angriffe.

2.8.1 Sicherheit gegen passive Angriffe

Ein Angreifer, der sich darauf beschränkt, die Leitungen zwischen den beiden kommunizierenden Parteien abzuhören und somit nur die gesendeten Schlüsseltexte bzw. Signaturen zu erhalten, nennt man passiv. Je nachdem, ob man Verschlüsselungs- oder Signatursysteme betrachtet, erhält man verschiedene Definitionen der Sicherheit.

Sicherheitsdefinitionen von Verschlüsselungssystemen

Die Definitionen der passiven Sicherheit bei Verschlüsselungssystemen basieren alle auf dem Gewinn von partiellen Informationen über die Nachricht m , da aus der Sicherheit gegen partiellen Informationsgewinn die Sicherheit gegen die verbliebenen drei Punkte direkt folgt. Da es jedoch durch vollständige Suche immer möglich ist, an solche Informationen zu gelangen, beschränkt man den Angreifer wie bereits erwähnt auf polynomielle Algorithmen. Nachdem der Angreifer den Schlüsseltext c gesehen hat, muss man ihm in der Theorie dennoch eine leichte Verbesserung seiner Chancen zugestehen, da die in der Praxis nicht anwendbaren exponentiellen Algorithmen in der Theorie dennoch einen geringen, sogenannten vernachlässigbaren, Vorteil liefern. Desweiteren kann man nicht zwangsläufig erreichen, dass die betrachtete Wahrscheinlichkeit für alle Instanzen vernachlässigbar ist.

Man unterscheidet drei unterschiedliche Sicherheitsdefinitionen, die jedoch nach geringfügigen Modifikationen als äquivalent bewiesen wurden [MiRS88]. Im folgenden wird sich auf eine Definition beschränkt.

Eine Möglichkeit passive Sicherheit zu definieren, ist durch das sogenannte Ununterscheidbarkeitskriterium (engl. *indistinguishability*) gegeben [GoMi84]. Man betrachtet folgenden Ablauf:

- Es wird ein Schlüssel (sk, pk) generiert und dem Angreifer der Schlüssel pk zugänglich gemacht.
- Der Angreifer wählt sich zwei Nachrichten m_0, m_1 des zugrundeliegenden Nachrichtenraumes und übergibt sie einem Verschlüsselungsortakel.
- Das Orakel wählt $b \in_R \{0, 1\}$, verschlüsselt m_b und übergibt $ver(m_b)$ dem Angreifer.
- Der Angreifer soll nun entscheiden, welche Nachricht verschlüsselt wurde, d.h. er gibt ein Bit b' aus, von dem er annimmt, dass $b' = b$ gilt.

Dann heißt das betrachtete Verschlüsselungssystem sicher, wenn gilt:

$$P(b = b') \leq \frac{1}{2} + \frac{1}{poly(l)}$$

bzw. in exakter kryptographischer Schreibweise: $\forall prob. poly. Algo. A_1, A_2 :$

$$\begin{aligned} P(b' = b) &:: (sk, pk) \leftarrow gen(l); \\ &(m_0, m_1, loc) \leftarrow A_1(l, pk); \\ &b \in_R \{0, 1\}; \\ &c := ver(pk, m_b); \\ &b' \leftarrow A_2(c, m_0, m_1, loc) \\ &\leq \frac{1}{2} + \frac{1}{poly(l)} \end{aligned}$$

wobei l den Sicherheitsparameter des Systems bezeichnet, was üblicherweise der Bitlänge eines Schlüssels entspricht. Mit *loc* wird eine lokale Variable bezeichnet, die dem Algorithmus A_2 das "Wissen" von A_1 zu Verfügung stellen soll. Diese Wahrscheinlichkeit besagt, dass der Angreifer nur einen vernachlässigbaren Vorteil im Gegensatz zur trivialen Ratestrategie besitzt bzw. dass ihm kein in polynomieller Zeit laufender Algorithmus einen Vorteil für seine Antwort liefert. Obige Schreibweise hat sich in der Kryptographie eingebürgert und wird im folgenden durchgehend benutzt werden.

Sicherheitsdefinitionen bei Signatursystemen

Auch hier definiert man Sicherheit anhand der schwächsten Forderung, also anhand der Sicherheit gegenüber existentiellen Fälschungen. Ein Signatursystem heißt sicher gegen passive Angriffe, wenn gilt: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned} P(\text{test}(pk, \text{Sig}, m) = \text{ok}) &:: (sk, pk) \leftarrow \text{gen}(l); \\ &(m, \text{Sig}) \leftarrow A(l, pk) \\ &\leq \frac{1}{\text{poly}(l)}. \end{aligned}$$

d.h. ein polynomieller Angreifer soll nicht in der Lage sein, eine existentielle Fälschung zu erzeugen.

2.8.2 Sicherheit gegen aktive Angriffe

Während sich die Betrachtungen des vorherigen Abschnittes ausschließlich auf passive Angriffe bezogen, beschäftigt man sich nun mit den in der Praxis sehr wichtigen aktiven Angriffen, d.h. mit Angriffen, bei denen der Angreifer auf eine oder mehrere Parteien des betrachteten Protokolls aktiv Einfluss nimmt. In der Praxis ist dies ein üblicher Vorgang, indem zum Beispiel die Unachtsamkeit oder die Gutgläubigkeit eines ehrlichen Benutzers ausgenutzt wird. Auf diese Art ist es dem Angreifer oftmals möglich, sich gewisse Schlüsseltexte von einem ehrlichen Benutzer entschlüsseln zu lassen, ohne dass dieser die damit verbundenen Gefahren erkennt. Obige Definitionen fordern keinerlei Sicherheit bei solchen aktiven Angriffen; aus diesem Grund wurden neue Definitionen notwendig, um die Bedrohungen in der realen Welt besser zu modellieren.

Sicherheitsdefinitionen von Verschlüsselungssystemen

Um mit diesen Angriffen umgehen zu können, haben Rackoff und Simon die Sicherheit gegen einen *adaptive chosen ciphertext attack* definiert [RaSi91]:

Sicherheit gegen *adaptive chosen ciphertext attack*

Man betrachtet folgenden Ablauf :

1. Es wird ein Verschlüsselungsschlüssel in Abhängigkeit eines Sicherheitsparameters l erzeugt.
2. Der Angreifer darf nun beliebige Schlüsseltexte seiner Wahl an ein Entschlüsselungsorakel geben, die ihm übersetzt werden.

3. Der Angreifer wählt nun zwei Nachrichten m_0^*, m_1^* und schickt sie an das Verschlüsselungsorakel.
4. Das Verschlüsselungsorakel wählt $b \in_R \{0, 1\}$ zufällig und verschlüsselt m_b^* . Die Wahl von b findet offensichtlich außerhalb der Sicht des Angreifers statt. Den Schlüsseltext der Nachricht erhält der Angreifer.
5. Der Angreifer darf sich nun vom Entschlüsselungsorakel wieder beliebige Nachrichten entschlüsseln lassen, mit der Einschränkung, dass diese verschieden von dem in Punkt (4) erhaltenen Schlüsseltext sein müssen.

Nach Beendigung von Punkt (5) muss der Angreifer sagen, welche Nachricht seiner Meinung nach in Punkt (3) verschlüsselt wurde, also er gibt $b' \in \{0, 1\}$ aus. Gilt für die Wahrscheinlichkeit $P(b' = b) = \frac{1}{2} + \epsilon$, so bezeichnet ϵ den Vorteil des Angreifers. Ein Verschlüsselungssystem heißt sicher gegen *adaptive chosen ciphertext attack*, wenn der Vorteil jedes Polynomialzeit-Angreifers als Funktion des Sicherheitsparameters l vernachlässigbar ist. Wenn man auf die Einführung von interaktiven Algorithmen und Orakeln verzichtet, könnte dies wie folgt in kryptographischer Schreibweise dargestellt werden: $\forall \text{prob. poly. Algo. } A_1, \dots, A_{2x+2}$:

$$\begin{aligned}
P(b' = b) &:: (sk, pk) \leftarrow \text{gen}(l); \\
&(c_1, loc_1) \leftarrow A_1(pk, l); \\
&m_1 \leftarrow \text{dec}(sk, c_1); \\
&(c_2, loc_2) \leftarrow A_2(m_1, loc_1); \\
&\vdots \\
&(c_x, loc_x) \leftarrow A_x(m_{x-1}, loc_{x-1}); \\
&m_x \leftarrow \text{dec}(sk, c_x); \\
&(m_0^*, m_1^*, loc_{x+1}) \leftarrow A_{x+1}(m_x, loc_x); \\
&b \in_R \{0, 1\}; c \leftarrow \text{enc}(pk, m_b^*); \\
&(c_{x+1}, loc_{x+2}) \leftarrow A_{x+2}(c, loc_{x+1}); \\
&\text{IF } c_{x+1} \neq c \\
&\quad \text{THEN } m_{x+1} \leftarrow \text{dec}(sk, c_{x+1}) \\
&\quad \text{ELSE } m_{x+1} := ' -'; \\
&(c_{x+2}, loc_{x+3}) \leftarrow A_{x+3}(m_{x+1}, loc_{x+2}); \\
&\vdots \\
&(c_{2x}, loc_{2x+1}) \leftarrow A_{2x+1}(m_{2x-1}, loc_{2x}); \\
&\text{IF } c_{2x} \neq c \\
&\quad \text{THEN } m_{2x} \leftarrow \text{dec}(sk, c_{2x}) \\
&\quad \text{ELSE } m_{2x} := ' -'; \\
&b' \leftarrow A_{2x+2}(m_{2x}, loc_{2x+1}) \\
&\leq \frac{1}{2} + \frac{1}{\text{poly}(l)}.
\end{aligned}$$

Diese Definition vereinfacht die aktiven Angriffe des Angreifers, indem sie ihm zugesteht, sich beliebige Schlüsseltexte durch ein Entschlüsselungsorakel dekodieren zu lassen. Um zu verhindern, dass der Angreifer den ihm gegebenen Schlüsseltext entschlüsseln lässt, muss man sein Verhalten jedoch einschränken. Er darf den gegebenen Schlüsseltext nicht dem Orakel geben, sondern muss sich auf sonstige Schlüsseltexte beschränken (er darf sich aber Texte entschlüsseln lassen, die dem gegebenen Schlüsseltext ähnlich sind). Man beachte, dass Sicherheit gegen *adaptive chosen ciphertext attack* die Sicherheit gegen *alle* in der Praxis vorkommenden Angriffe impliziert.

Eine andere Definition für Sicherheit gegen aktive Angriffe, mit dem Namen *non-malleability*, wurde von Dolev, Dwork and Naor entworfen [DoDN91].

Non-malleability Sicherheit

Der Angreifer hat auch hier Zugang zu einem Entschlüsselungsorakel, aber sein Ziel besteht nicht darin, partielle Informationen über die Nachricht herauszufinden, sondern selbst einen Schlüsseltext zu erzeugen, dessen Inhalt in einer bestimmten Beziehung zum Inhalt des gegebenen Schlüsseltextes steht. Sei z.B. eine Verschlüsselung von n gegeben. Dann soll es nicht möglich sein, eine Verschlüsselung von $n + 1$ zu erzeugen. Anfangs schien diese Definition stärker als die Sicherheit gegen *adaptive chosen ciphertext attack* zu sein, erwies sich jedoch später als äquivalent dazu [BDPR98].

Es gibt weitere Definitionen von Sicherheit, die zwischen semantischer Sicherheit und Sicherheit gegen *adaptive chosen ciphertext attack* anzusiedeln sind. Eine abgeschwächte Form der Sicherheit gegen *adaptive chosen ciphertext attack* stellt die von Naor und Yung definierte Sicherheit gegen *chosen ciphertext attack* dar [NaYu89], d.h. der Angreifer hat nur vor dem Erhalt des gegebenen Schlüsseltextes Zugang zu einem Entschlüsselungsorakel mit dem Ziel, Informationen zu erlangen, die es ihm später möglich machen sollen, partielle Informationen über die gegebene Nachricht herauszufinden. Diese Art von Angriff wird auch *lunch-time attack* oder *midnight attack* genannt.

Sicherheitsdefinitionen bei Signatursystemen

Analog zur Sicherheit gegen *adaptive chosen ciphertext attack* bei Verschlüsselungssystemen ist die Sicherheit gegen *adaptive chosen message attack* bei Signatursystemen definiert [GoMR88]:

Sicherheit gegen *adaptive chosen message attack*

Der Angreifer hat wiederum Zugriff zu einem Orakel, das in der Lage ist, beliebige vom Angreifer gewählte Nachrichten zu signieren. Man betrachtet analog zur Definition des vorigen Abschnittes folgenden Ablauf:

1. Es wird ein privater (Signier-)Schlüssel und ein öffentlicher (Test-)Schlüssel in Abhängigkeit eines Sicherheitsparameters l generiert.
2. Der Angreifer darf nun dem Orakel beliebige Nachrichten m_i übergeben und erhält die dazu passenden gültigen Signaturen.
3. Der Angreifer muss sich nun ein Tupel (m, Sig) errechnen, so dass gilt: $m \neq m_i$ für alle i und $test(pk, Sig, m) = ok$.

Ein Signatursystem heißt nun sicher gegen *adaptive chosen message attack*, wenn der Vorteil eines jeden Polynomialzeit-Angreifers als Funktion des Sicherheitsparameters l vernachlässigbar ist, also in kryptographischer Schreibweise: $\forall \text{prob. poly. Algo. } A_1, \dots, A_{x+1} :$

$$\begin{aligned}
 P(\text{test}(pk, \text{Sig}, m) = \text{ok} \\
 \wedge m \notin \{m_1, \dots, m_x\} \quad &:: (sk, pk) \leftarrow \text{gen}(l); \\
 &(m_1, loc_1) \leftarrow A_1(pk, l); \\
 &\text{Sig}_1 \leftarrow \text{sig}(sk, m_1); \\
 &(m_2, loc_2) \leftarrow A_2(loc_1, \text{Sig}_1); \\
 &\vdots \\
 &(m_x, loc_x) \leftarrow A_x(loc_{x-1}, \text{Sig}_{x-1}); \\
 &\text{Sig}_x \leftarrow \text{sig}(sk, m_x); \\
 &(m, \text{Sig}) \leftarrow A_{x+1}(loc_x, \text{Sig}_x);) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

2.9 Das Random Oracle Modell

In Sicherheitsbeweisen spielen die Anforderungen an eventuell verwendete Hashfunktionen eine große Rolle. Oftmals können Beweise nur durchgeführt werden, wenn man im Beweis anstelle der benutzten Hashfunktion ein Orakel einsetzt, das einen echt zufälligen Wert liefert. Man spricht vom sogenannten *Random Oracle* Modell [FiSh87]. Auf diese Weise werden eventuelle, noch nicht bekannte Schwächen der Hashfunktion umgangen. Man argumentiert anhand eines solchen Beweises, dass aus der Sicherheit im *Random Oracle* Modell die Sicherheit im realen Modell folgt, sofern die ersetzte Hashfunktion keine Schwächen aufweist. Dennoch sollte man das *Random Oracle* Modell lediglich als eine Heuristik ansehen, die man keinesfalls mit Beweisen im realen Fall gleichsetzen sollte.

Kapitel 3

Bekannte kryptographische Annahmen

3.1 Das Faktorisierungsproblem (*Integer Factorisation Problem*)

Das Faktorisierungsproblem stellt das älteste und damit auch das am meisten untersuchte zahlen-theoretische Problem dar. Wie bereits in Kapitel 2 erwähnt wurde, besitzt jede Zahl $n \in \mathbb{Z} \setminus \{0\}$ eine eindeutige Zerlegung in Primzahlen. Sei z.B. $n = 342$, dann besitzt n die Primzahlzerlegung $342 = 2 \cdot 3^2 \cdot 19$. Es ist offensichtlich, dass sich die ursprüngliche Zahl aus den gegebenen Primfaktoren wieder effizient berechnen lässt, die Umkehrung ist jedoch intuitiv “schwer”, d.h. es bereitet erhebliche Schwierigkeiten, eine (große) Zahl n in ihre Primfaktoren zu zerlegen. In der Mathematik beschäftigt man sich nun bereits seit über 2000 Jahren mit diesem Problem, ohne einen Algorithmus gefunden zu haben, der das Problem in polynomieller Laufzeit in der Bitlänge von n löst. In jüngster Vergangenheit wurden jedoch stetige Verbesserungen der Algorithmen erzielt, die die Grenze der momentan faktorisierten Zahlen weiter nach oben verschoben haben.

Die Definition der Faktorisierungsannahme und die Definitionen der übrigen vorgestellten Probleme werden in eine zahlentheoretische und eine kryptographische Definition aufgespalten.

Definition 3.1 (Integer Factorisation Problem).

Zahlentheoretisch: Sei eine Zahl $n \in \mathbb{N}$ gegeben. Finde die eindeutig bestimmten paarweise verschiedenen Primzahlen $p_1 \dots p_r$, so dass gilt: $n = p_1^{\alpha_1} \cdot \dots \cdot p_r^{\alpha_r}$ mit $\alpha_i \in \mathbb{N}$ für alle $i \in \{1, \dots, r\}$.

Bevor ich zur kryptographischen Definition des Problems komme, sind noch einige Vorüberlegungen nötig.

In der Kryptographie beschränkt man sich meist auf den Fall $n = pq$, wobei p und q als große Primzahlen von ungefähr der selben Bitlänge gewählt werden, d.h. man stellt an n die Bedingung, dass sie sich nur aus zwei Primzahlen zusammensetzt. Dieses Problem wird in einigen Artikeln als Splitting-Problem bezeichnet. In Wahrheit bildet dies jedoch zumindest in der Theorie nicht wirklich eine Einschränkung, da eine Faktorisierung in zwei Primzahlen eine sukzessive Zerlegung in beliebig viele Primzahlen möglich macht. In der Praxis jedoch erweist sich eine Zahl der obigen Form als wesentlich schwieriger zu faktorisieren, da sie keine kleinen Primfaktoren enthält und somit Algorithmen, die durch Probedivision (engl. *trial division*) faktorisieren, scheitern, wohingegen eine beliebig gewählte Zahl derselben Größe mit sehr hoher Wahrscheinlichkeit einen oder mehrere kleine Primfaktoren besitzt, was zumindest eine Abspaltung dieser Primfaktoren von der Zahl ermöglichen würde.

An dieser Stelle wiederhole ich noch einmal, was es nun eigentlich heißt, dass ein Problem schwer ist. Offensichtlich kann man jede gegebene Zahl n faktorisieren, indem man einfach für

die Primzahlen zwischen 2 und \sqrt{n} testet, ob sie n teilen. Das Problem liegt darin, dass die Zahl n so groß gewählt wurde, dass ein solcher Algorithmus die Faktorisierung der Zahl erst in einer inakzeptablen Zeit berechnen kann. Der oben vorgeschlagene Algorithmus würde bei einer 1024-Bitzahl n noch rechnen, wenn unser Sonnensystem längst der Vergangenheit angehört, sofern der Rechner dann noch existieren würde. Die oben beschriebenen Eigenschaften eines Problems sind genau diejenigen, die es anschaulich als schwer klassifizieren:

1. Das Problem ist lösbar.
2. Es existiert mindestens ein Algorithmus, welcher das Problem löst.
3. Jeder bis jetzt bekannte Algorithmus, der das Problem löst, hat inakzeptable Laufzeit.
4. Man geht davon aus bzw. man hofft, dass kein polynomieller Algorithmus existiert, der das Problem löst. Genauer: Man hofft, dass jeder polynomielle Algorithmus nur auf einer vernachlässigbaren Teilmenge Erfolg hat. Man spricht dann von einem *fast überall schweren* Problem.

Was man unter inakzeptabler Laufzeit zu verstehen hat, wurde bereits im vorangegangenen Kapitel genauer erläutert. Es ist nun nötig, diese anschauliche Definition in kryptographischer Schreibweise aufzuschreiben, um sie später in Beweisen nutzen zu können.

Definition 3.2 (Integer Factorisation Problem).

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(n = p'q' \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(p', q') \leftarrow A(l, n) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Inzwischen wird in unterschiedlichen Papers oftmals auch das Problem betrachtet, Zahlen n , deren Primzahlzerlegung von der üblichen $n = pq$ - Form abweicht zu faktorisieren. In [OkUc97] wird beispielsweise behauptet, dass das Faktorisieren von Zahlen der Form $n = p^2q$ ebenfalls schwer sei. Diese Annahme wird als modifiziertes Faktorisierungsproblem bezeichnet:

Definition 3.3 (Modified Integer Factorisation Problem).

Zahlentheoretisch: Gegeben: $n = p^2q$ mit p, q prim. Gesucht: p, q .

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(n = p'^2q' \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := p^2q; \\
 &(p', q') \leftarrow A(l, n) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Eine auf der Hand liegende Frage besteht nun darin, ob sich diese Behauptung auf Zahlen der Form $n = p^r q$ erweitern lässt für beliebige Exponenten r . Diese Frage wurde in [BoDH99] mit nein beantwortet. Für größer werdende Exponenten r existieren immer schneller Algorithmen, die schließlich für $r \approx \sqrt{\log p}$ in polynomieller Zeit in der Bitlänge von n ablaufen.

3.2 Das Diskreter Logarithmus Problem (*Discrete Logarithm Problem*)

Das zweite große zahlentheoretische Referenzproblem stellt das Diskreter Logarithmus Problem, kurz: Diskreter Log oder D-Log, dar. Es befasst sich mit der Berechnung des Logarithmus einer Zahl im Ring \mathbb{Z}_p mit p prim bzw. in einer beliebigen zyklischen Gruppe. Da dieses Problem im Vergleich zum Faktorisierungsproblem noch recht jung und demzufolge noch nicht in einem solch großen Ausmaß untersucht wurde, hat sich die Forschung der letzten Jahre verstärkt damit befaßt. Auch hier fand man keinen effizienten Algorithmus, der das Problem lösen konnte, jedoch wurden Parallelen zum Faktorisierungsproblem deutlich sichtbar. Dies äußerte sich z.B. in der übereinstimmenden asymptotischen Laufzeit des momentan besten Faktorisierungsalgorithmus, dem *number field sieve*, sowie des momentan besten Algorithmus zum Berechnen des diskreten Logarithmus, der ebenfalls unter dem Namen *number field sieve* bekannt ist.

Eine offene Frage besteht nach wie vor darin, ein Problem auf das andere zurückzuführen und damit ein Problem als mindestens so schwer wie das andere zu klassifizieren. Dieses Unterfangen ist zum momentanen Zeitpunkt noch nicht geglückt, und man kann damit wohl auch in naher Zukunft nicht rechnen. Beide Probleme werden in der Kryptographie als grundlegende Annahmen angesehen, auf denen man seine Kryptosysteme aufbauen kann. Das Diskreter Log Problem wird folgendermaßen definiert:

Definition 3.4 (Discrete Logarithm Problem in \mathbb{Z}_p^*).

Zahlentheoretisch: Sei p eine Primzahl, g ein Generator von \mathbb{Z}_p^* und $h \in \mathbb{Z}_p^*$. Finde $x \in \mathbb{N}$, $1 \leq x \leq p-1$, so dass gilt: $g^x \equiv h \pmod{p}$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(g^x \equiv h \pmod{p} \quad &:: \quad p \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &g \in_R \text{ Menge der Generatoren von } \mathbb{Z}_p^*; \\
 &h \in_R \mathbb{Z}_p^*; \\
 &x \leftarrow A(l, p, g, h) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Da zur sinnvollen Definition eines Logarithmus in \mathbb{Z}_p nur ein Generator der Gruppe \mathbb{Z}_p^* vorhanden sein muss, ist es möglich, das Problem auf bestimmte Familien zyklischer Gruppen ausweiten:

Definition 3.5 (Generalised Discrete Logarithm Problem).

Zahlentheoretisch: Sei $G = (G_{desc,n})_{desc,n \in Desc \times \mathbb{N}}$ eine Familie zyklischer Gruppen mit einer Indexmenge $Desc$, wobei die natürliche Zahl n der Ordnung der Gruppe $G_{desc,n}$ entspricht. Seien nun $(desc, n)$ durch eine vorgegebene Generierung bereits bestimmt. Es seien ein zufälliger Generator g von $G_{desc,n}$ und ein $h \in_R G_{desc,n}$ gegeben. Finde die natürliche Zahl x mit $1 \leq x \leq n-1$ mit $g^x = h$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(g^x = h \quad &:: \quad (desc, n) \leftarrow \text{gen}(l); \\
 &g \in_R \text{ Menge der Generatoren von } G_{desc, n}; \\
 &h \in_R G_{desc, n}; \\
 &x \leftarrow A(l, n, g, h, desc)) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Zum oben definierten allgemeinen Diskreten Log. gibt es sehr viele Varianten. Oftmals wird z.B. der diskrete Logarithmus modulo einer zusammengesetzten Zahl $n = pq$ betrachtet: Man spricht vom *composite discrete logarithm problem*. Da jedoch die Gruppe \mathbb{Z}_n^* für $n = pq$ nicht mehr zyklisch ist, muss die Wahl des Elementes h an das erzeugende Element g angepasst werden, d.h. es muss gelten $h \in \langle g \rangle$. Sinnvollerweise sollte man an das Element g noch die Anforderung stellen, dass die Größe der von ihm erzeugten Gruppe etwa in der Größenordnung der Gruppe \mathbb{Z}_n^* liegt. Dies ist jedoch nicht einheitlich festgelegt und wird in der Literatur ja nach Anwendung anders gebraucht. Im folgenden betrachten wir nur Elemente g mit maximaler Ordnung, d.h. es muss gelten $\text{ord}(g) = \text{kgV}(p-1, q-1)$. Damit können wir zur formalen Definition kommen:

Definition 3.6 (Composite Discrete Logarithm Problem).

Zahlentheoretisch: Sei $n = pq$ gegeben mit p, q prim, $g \in \mathbb{Z}_n^*$ mit $|\langle g \rangle| = \text{kgV}(p-1, q-1)$ und $h \in \langle g \rangle$. Finde $x \in \mathbb{N}$, $1 \leq x \leq |\langle g \rangle| - 1$, so dass gilt: $g^x \equiv h \pmod{n}$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(g^x \equiv h \pmod{n} \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen}; \\
 &n := pq; \\
 &g \in_R \mathbb{Z}_p^* \text{ mit } |\langle g \rangle| = \text{kgV}(p-1, q-1); \\
 &h \in_R \langle g \rangle; \\
 &x \leftarrow A(l, p, g, h)) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Neben der Faktorisierungsannahme und dem Diskreten Logarithmus Problem gibt es weitere abgeleitete Annahmen, von denen die beiden nächsten die am häufigsten benutzten darstellen.

3.3 Das RSA Problem

Das RSA Problem wird vom Faktorisierungsproblem abgeleitet. Das RSA Problem besagt informell, dass das Ziehen von Wurzeln in \mathbb{Z}_n schwer ist. Das Problem wird folgendermaßen definiert:

Definition 3.7 (RSA problem).

Zahlentheoretisch: Gegeben:

1. $n \in \mathbb{N}$ mit $n = pq$; p, q prim.
2. $e \in \mathbb{N}$, so dass gilt: $\text{ggT}(e, (p-1)(q-1)) = 1$.

3. $c \in \mathbb{N}$.

Gesucht: $m \in \mathbb{N}$ mit $m^e \equiv c \pmod{n}$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(m^e \equiv c \pmod{n} \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 e &\in_R \mathbb{Z}_{\varphi(n)}^*; \\
 c &\in_R \mathbb{Z}_n; \\
 m &\leftarrow A(l, n, e, c) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

In Worten bedeutet dies, dass das Ziehen e -ter Wurzeln in \mathbb{Z}_n schwer sein soll. Das Problem ist höchstens so schwer wie das Faktorisierungsproblem. Der Beweis dazu ist in Kapitel 5 zu finden. Das RSA Problem bildet die Grundlage für das wohl derzeit bekannteste Verschlüsselungsverfahren überhaupt: Das RSA-Verschlüsselungssystem von Rivest, Shamir und Adelman, das auch zur Namensgebung des hier vorgestellten Problems geführt hat [RiSA78]. In diesem System besteht der private Schlüssel aus (p, q) und der öffentliche Schlüssel aus (n, e) , d.h. der Exponent e wird bei jeder Schlüsselerzeugung wieder zufällig gewählt. Das Problem scheint sich jedoch als schwer für alle einzelnen Exponenten $e \in \mathbb{Z}_{\varphi(n)}^*$ mit $e \geq 2$ zu erweisen. Aus diesem Grund wird in der Praxis ein fester Exponent gewählt mit dem Ziel, den öffentlichen Schlüssel kürzer und die Entschlüsselung schneller zu machen. In der Praxis hat sich der Exponent 65537 als Standard durchgesetzt, da er durch sein Bitmuster $65537 = 1000000000000001_b$ die erforderlichen Square-and-Multiply Algorithmen zur Potenzierung stark beschleunigt. Man erhält das folgende modifizierte RSA Problem:

Definition 3.8 (RSA Problem mit festem e).

Zahlentheoretisch: Gegeben:

1. $e \in \mathbb{N}$.
2. $n \in \mathbb{N}$, so dass gilt $\text{ggT}(e, (p-1)(q-1)) = 1$ für $n = pq$ mit p, q prim.
3. $c \in \mathbb{N}$.

Gesucht: $m \in \mathbb{N}$ mit $m^e \equiv c \pmod{n}$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(m^e \equiv c \pmod{n} \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen mit} \\
 n &:= pq, \text{ so dass gilt: } \text{ggT}(e, \varphi(n)) = 1; \\
 c &\in_R \mathbb{Z}_n^*; \\
 m &\leftarrow A(l, n, c) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

3.4 Das Diffie Hellman Problem

Das Diffie Hellman Problem wird vom Diskreter Log. Problem abgeleitet:

Definition 3.9 (Diffie Hellman Problem).

Zahlentheoretisch: Gegeben:

1. eine Primzahl p .
2. ein Generator g von \mathbb{Z}_p^* , $a, b \in \mathbb{Z}_{p-1}$.
3. $g^a \bmod p$ und $g^b \bmod p$.

Gesucht: $g^{ab} \bmod p$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(\beta \equiv e \bmod p \quad &:: \quad p \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 & a, b \in_R \mathbb{Z}_{p-1}; \\
 & g \in_R \text{ Menge der Generatoren von } \mathbb{Z}_p^*; \\
 & c \equiv g^a \bmod p; \\
 & d \equiv g^b \bmod p; \\
 & e \equiv g^{ab} \bmod p; \\
 & \beta \leftarrow A(l, p, g, c, d) \\
 & \leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Bei diesem Problem ist die Polynomzeit-Reduktion auf das Diskreter Logarithmus Problem offensichtlich. Sie wird jedoch der Vollständigkeit halber in Kapitel 5 bewiesen.

3.5 Das Wurzel-Problem (*Square Root Problem*)

Das *Square Root Problem* benötigt den in Abschnitt (2.2) eingeführten Begriff eines quadratischen Restes.

Definition 3.10 (Square Root Problem).

Zahlentheoretisch: Sei $n = pq$ mit p, q prim, a sei ein quadratischer Rest modulo n . Finde $x \in \mathbb{Z}_n^$ mit $x^2 \equiv a \bmod n$.*

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(x^2 \equiv a \bmod n \\
 \wedge x \in \mathbb{Z}_n^* \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 & n := pq; \\
 & a \in_R QR_n; \\
 & x \leftarrow A(l, n, a) \\
 & \leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Wie man in Kapitel 5 sehen wird, ist das *Square Root Problem* äquivalent zum Faktorisierungsproblem. Die dem Wurzelproblem zugrundeliegende Funktion $f(x) = x^2 \bmod n$ wird Rabin-Funktion genannt.

3.6 Das Quadratische-Reste-Problem (*Quadratic Residuosity Assumption*)

Definition 3.11 (Quadratic Residuosity Assumption (QRA)).

Zahlentheoretisch: Sei $n = pq$ mit p, q prim, $a \in_R \mathbb{Z}_n^*$ mit $\left(\frac{a}{n}\right) = 1$. Entscheide, ob gilt: $a \in QR_n$.
Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned} P(b' = b) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\ &n := pq; \\ &a \in_R \mathbb{Z}_n^{+1}; \\ &b \in \{0, 1\} \text{ mit } b = 1 \Leftrightarrow a \in QR_n; \\ &b' \leftarrow A(l, n, a) \\ &\leq \frac{1}{2} + \frac{1}{\text{poly}(l)}. \end{aligned}$$

Die kryptographische Definition weicht hier von den weiter oben vorkommenden Definitionen ab, da die Lösung des betrachteten Problems nur ein Bit erfordert und der Angreifer bereits durch Raten eine nicht vernachlässigbare Erfolgswahrscheinlichkeit, nämlich $\frac{1}{2}$, besitzt. Es wird somit in der Definition behauptet, dass der Angreifer nichts signifikant Besseres tun kann, als ein Bit zufällig zu bestimmen.

3.7 Nichtzahlentheoretische Probleme

Alle bis jetzt betrachteten Annahmen beruhten ausschließlich auf zahlentheoretischen Problemen. Im Gegensatz dazu stehen die nichtzahlentheoretischen Probleme, die ihren Ursprung meist in der Komplexitätstheorie haben. Theoretisch genügt jedes als NP-vollständig klassifizierte Problem der in (3.1) beschriebenen intuitiven Anforderung. Allerdings wird in der Komplexitätstheorie stets die worst-case-Komplexität betrachtet, d.h. es gibt Instanzen, für die das zugrundeliegende Problem nicht gebrochen werden kann. In der Kryptographie hingegen benötigt man, dass ein Problem "fast überall schwer" ist, d.h. das Problem kann lediglich für eine nicht signifikante Anzahl von Instanzen gebrochen werden. Ein Problem aus der Komplexitätstheorie, das in der Kryptographie betrachtet wurde, ist das *Subset Sum Problem*:

Definition 3.12 (Subset Sum Problem). Sei eine natürliche Zahl s und eine Menge $M = \{a_1, \dots, a_n\}$ von natürlichen Zahlen gegeben. Bestimme, ob es eine Teilmenge $M' = \{a_{i_1}, \dots, a_{i_k}\}$ von M gibt mit $\sum_{j=1}^k a_{i_j} = s$.

Allerdings konnte unter dieser Annahme im Gegensatz zu den weiter oben vorgestellten Problemen keine Sicherheit eines Systems bewiesen werden. Es gibt weitere nichtzahlentheoretische Arten von Annahmen, die den Grundstein eines Kryptosystems bilden können, deren Betrachtung geht jedoch über den Rahmen dieser Arbeit hinaus. Als Beispiel seien hier Perceptrons

[Poin95] bzw. [KnMe99] und fehlerkorrigierende Codes [McEl79] bzw. [McSa81] erwähnt. Von besonderer Wichtigkeit ist noch die Annahme der Existenz einer Einwegfunktion, da diese Annahme die wohl schwächste bekannte Annahme ist und sich dennoch Verschlüsselungssysteme konstruieren lassen, deren Sicherheit auf dieser Annahme basiert.

Kapitel 4

Einführung neuer Annahmen

4.1 Übersicht

In diesem Kapitel werden nun die häufigsten neueren Annahmen vorgestellt. Neben den Definitionen wird verstärkt auf die intuitiven bzw. die heuristischen Gründe eingegangen, die im allgemeinen Fall wohl die Grundlage bilden, ein Problem als schwer anzusehen. Anhand dieses Gesichtspunktes bildet dieses Kapitel neben den Definitionen sozusagen die *intuitive* bzw. die *heuristische* Grundlage der vorgestellten Probleme, wohingegen im nächsten Kapitel die einzelnen Probleme durch mathematische Methoden analysiert und miteinander verglichen werden, was die *mathematischen* bzw. *analytischen* Grundlagen der Probleme liefert.

Desweiteren wird zu jeder Annahme ein darauf basierendes Kryptosystem vorgestellt, um dem Leser eine ungefähre Vorstellung zu liefern, wie diese Annahmen momentan in der Praxis umgesetzt werden.

Bei Kryptosystemen, die sich bisher als nicht beweisbar sicher gegen aktive Angriffe erwiesen haben, also Systeme, die nicht der Sicherheit gegen *adaptive chosen ciphertext attack* bzw. *adaptive chosen message attack* genügen, werden Angriffsmöglichkeiten auf das System vorgestellt. Im Fall der Sicherheit gegen aktive Angriffe wird gezeigt, wie man das Kryptosystem brechen könnte, vorausgesetzt man könnte die zugrunde liegende Annahme brechen. Auf Möglichkeiten, die Annahmen anhand gegebenen partiellen Informationen zu brechen, wird in Kapitel 6 und 7 näher eingegangen.

Ich beginne mit der Einführung der zur Zeit wohl am weitesten verbreiteten, am meisten genutzten und am besten untersuchten höheren Annahme.

4.2 Das *Diffie Hellman Decision Problem*

Das *Diffie Hellman Decision Problem* bildet die natürliche Erweiterung des *Diffie Hellman Problems* zum Entscheidungsproblem:

4.2.1 Definition

Definition 4.1 (Diffie Hellman Decision Problem).

Zahlentheoretisch: Sei $G = (G_{desc,p})_{desc,p \in Desc \times \mathbb{P}}$ eine Familie von Gruppen mit einer Indexmenge $Desc$, wobei die Primzahl p der Ordnung der Gruppe $G_{desc,p}$ entspricht. Seien nun $(desc, p)$ durch eine vorgegebene Generierung bereits bestimmt, desweiteren sei g ein Generator von $G_{desc,p}$.

Entscheide, ob für ein gegebenes Tupel (g^a, g^b, x) gilt: $x = g^{ab}$?

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(b' = b) &:: (desc, p) \leftarrow \text{gen}(l); \\
 &\alpha, \beta \in_R \mathbb{Z}_{p-1}; \\
 &g \in_R \text{Menge der Generatoren von } G_{desc,p}; \\
 &d \equiv g^\alpha \pmod{p}; \\
 &e \equiv g^\beta \pmod{p}; \\
 &f_0 \equiv g^{\alpha\beta} \pmod{p}; \\
 &f_1 \in_R G_{desc,p}; \\
 &b \in_R \{0, 1\}; \\
 &b' \leftarrow A(l, p, desc, g, d, e, f_b) \\
 &\leq \frac{1}{2} + \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Das *Diffie Hellman Decision Problem* ist offensichtlich verwandt zum

1. *Diffie Hellman Problem*: geg. : g, g^x, g^y ges.: g^{xy} , wie in (1.2.4) vorgestellt
2. *Discrete Logarithm Problem*: geg. : g, g^x ges.: x , wie in (1.2.2) vorgestellt.

Es gibt offensichtliche Polynomzeitreduktionen vom *Diffie Hellman Decision Problem* zu (1) und von (1) zu (2). Ob die Umkehrung auch gilt, ist noch nicht bekannt. Die Beziehungen der verschiedenen Annahmen untereinander werden in Kapitel 5 genauer untersucht. Alle drei Probleme werden als hart angesehen und demzufolge in verschiedenen Verschlüsselungssystemen eingesetzt.

Die Sicherheit des ElGamal Systems ist z.B. äquivalent zum *Diffie Hellman Decision Problem*. In ElGamal wird eine Nachricht $m \in G$ als $(g^r, h^r m)$ verschlüsselt, wobei h der public key ist [ElGa85].

Ist nun einerseits das *Diffie Hellman Decision Problem* schwer, so könnte man h^r durch ein beliebiges Gruppenelement ersetzen, ohne das Verhalten des Angreifers entscheidend zu verändern. Da die Multiplikation in einer Gruppe die Anforderungen eines One-Time-Pads [Shan49] erfüllt, kann man somit keinerlei Informationen über die Nachricht erhalten.

Könnte man andererseits das *Diffie Hellman Decision Problem* effizient lösen, so könnte man auf folgende Art ElGamal brechen: Der Angreifer wählt zwei Nachrichten m_0, m_1 , die er einem Verschlüsselungssorakel übergibt und wozu er eine Verschlüsselung $(u, e) = (g^r, h^r m_b)$ von m_b erhält, wobei b zufällig aus $\{0, 1\}$ gewählt wurde. Die Aufgabe des Angreifers besteht nun darin, b herauszufinden. Dazu muss er nur testen, ob $(u, h, \frac{e}{m_0})$ oder $(u, h, \frac{e}{m_1})$ ein Diffie-Hellman-Tripel ist.

In der Literatur finden sich mittlerweile zahlreiche Abwandlungen des Problems, von denen eine im folgenden vorgestellt wird, da sie sich als grundlegend für die Sicherheit des später betrachteten Cramer-Shoup-Verschlüsselungssystem erweist.

Definition 4.2 (Modifiziertes Diffie Hellman Decision Problem).

Zahlentheoretisch: Sei $G = (G_{desc,p})_{desc,p \in Desc \times \mathbb{P}}$ eine Familie von Gruppen mit einer Indexmenge $Desc$, wobei die Primzahl p der Ordnung der Gruppe $G_{desc,p}$ entspricht. Seien nun $(desc, p)$

durch eine vorgegebene Generierung bereits bestimmt, desweiteren seien g_1, g_2 Generatoren von $G_{desc,p}$. Entscheide, ob zu einem gegebenen Tupel (g_1, g_2, u_1, u_2) ein $r \in \mathbb{Z}_{p-1}$ existiert mit $u_1 = g_1^r$ und $u_2 = g_2^r$?

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned} P(b' = b \quad &:: \quad (desc, p) \leftarrow \text{gen}(l); \\ &r \in_R \mathbb{Z}_{p-1}; \\ &g_1, g_2 \in_R \text{ Menge der Generatoren von } G_{desc,p}; \\ &u_0 \equiv g_1^r \pmod{p}; \\ &v_0 \equiv g_2^r \pmod{p}; \\ &u_1, v_1 \in_R G_{desc,p}; \\ &b \in_R \{0, 1\}; \\ &b' \leftarrow A(l, p, desc, g_1, g_2, u_b, v_b) \\ &\leq \frac{1}{2} + \frac{1}{\text{poly}(l)}. \end{aligned}$$

Im Folgenden wird nun ein neues Verschlüsselungssystem von Ronald Cramer und Victor Shoup vorgestellt, das auf dem Modifizierten *Diffie Hellman Decision Problem* basiert und sich als beweisbar sicher gegen *adaptive chosen ciphertext attack* erweist [CrSh98]. Zum Zeitpunkt des Erscheinens war es das erste Verschlüsselungssystem, das dieses strenge Sicherheitskriterium ohne das *random oracle* Modell erfüllte und demzufolge von großer Wichtigkeit.

4.2.2 Das Verschlüsselungssystem von R. Cramer, V. Shoup

Sei G eine Gruppe mit $|G| := q$, q große Primzahl. Da die Gruppe eine Primzahlordnung besitzt, folgt, dass sie zyklisch ist, was ein wesentlicher Faktor für die Funktionsweise des Algorithmus ist. Für die Klartextnachrichten m_i gelte $m_i \in G$. Sei $HASH := (H_j)_{j \in J}$ eine Familie von universal-oneway Hashfunktionen mit

$$H_j : \{0, 1\}^* \rightarrow \mathbb{Z}_q$$

also ordnet jede Funktion H_j einem Bitstring ein Element aus \mathbb{Z}_q zu.

Schlüsselerzeugung Seien $g_1, g_2 \in_R G$, $x_1, x_2, y_1, y_2, z \in_R \mathbb{Z}_q$. Man berechnet nun $c, d, h \in G$ folgendermaßen

$$c = g_1^{x_1} g_2^{x_2}, \quad d = g_1^{y_1} g_2^{y_2}, \quad h = g_1^z.$$

Sei nun $H_j \in_R HASH$. Dann erhält man als public key $pk = (g_1, g_2, c, d, h, H_j)$ und als private key $sk = (pk, x_1, x_2, y_1, y_2, z)$.

Verschlüsselung Eine Nachricht $m \in G$ wird gemäß folgendem Algorithmus verschlüsselt:

1. Wähle $r \in_R \mathbb{Z}_q$.
2. Berechne

$$u_1 = g_1^r, \quad u_2 = g_2^r, \quad e = h^r m, \quad \alpha = H_j(u_1, u_2, e), \quad v = c^r d^{r\alpha}.$$

3. Gib als Schlüsseltext (u_1, u_2, e, v) aus.

Entschlüsselung Sei ein Schlüsseltext (u_1, u_2, e, v) gegeben. Die Entschlüsselung wird durch folgenden Algorithmus durchgeführt:

1. Berechne $\alpha = H_j(u_1, u_2, e)$.
2. Teste, ob $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$.
3. Falls ja, gib als Klartextnachricht $m = e/u_1^z$ aus, ansonsten lehne den Schlüsseltext ab.

Was nun neben der Sicherheit gezeigt werden muss, ist, dass die Entschlüsselung der Verschlüsselung einer Nachricht m wieder die ursprüngliche Nachricht ergibt, also $ent(sk, ver(pk, m)) = m$. Es gilt

$$u_1^{x_1} u_2^{x_2} = g_1^{rx_1} g_2^{rx_2} = c^r.$$

Weiter gilt

$$u_1^{y_1} u_2^{y_2} = d^r \text{ und } u_1^z = h^r.$$

Insgesamt folgt

$$u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = u_1^{x_1} (u_1^{y_1})^\alpha u_2^{x_2} (u_2^{y_2})^\alpha = c^r (d^r)^\alpha = v.$$

Also akzeptiert der Algorithmus die Verschlüsselung und gibt als Klartext $\frac{e}{h^r} = m$ aus.

Da sich das System als sicher gegen *adaptive chosen ciphertext attack* erweist, hat es keinen Sinn, sich aktive Angriffe gegen das System auszudenken, da damit die Sicherheit gegen alle Arten aktiver Angriffe mit eingeschlossen ist.

4.3 Höhere Reste (*Higher Residues*)

Wie im vorigen Kapitel anhand des *Square Root Problem* beschrieben, ist das Ziehen einer Wurzel in \mathbb{Z}_n schwer. Eine naheliegende Vermutung besteht darin, dass auch das Ziehen höherer, allgemein e -ter, Wurzel schwer ist. Tatsächlich ist bis heute kein polynomieller Algorithmus bekannt, der zu einer gegebenen Zahl eine e -te Wurzel berechnet, sofern die Faktorisierung von n unbekannt ist.

4.3.1 Einführung

Der Begriff eines höheren Restes wurde bereits in der zahlentheoretischen Einführung definiert. Im Folgenden werden einige Eigenschaften der höheren Reste vorgestellt; insbesondere wird auf ihre Gruppenstruktur und die Anzahl der vorhandenen Wurzeln eingegangen. Zum Abschluss wird eine auf höheren Resten basierende Funktion betrachtet, die sich als grundlegend für die Betrachtungen der folgenden Abschnitte erweist. In den folgenden Definitionen seien Zahlen $r \in \mathbb{N} \setminus \{1\}$ und $n = pq$ mit p, q prim vorausgesetzt. Desweiteren sei \mathbb{Z}_n^r die Menge aller r -ten Reste modulo n .

Lemma 4.3. *Die Menge \mathbb{Z}_n^r bildet eine Untergruppe von \mathbb{Z}_n^* .*

Beweis. Die Assoziativität vererbt sich aus der übergeordneten Gruppe.

Um die Abgeschlossenheit zu zeigen, betrachtet man zwei Elemente $z_1, z_2 \in \mathbb{Z}_n^r$, d.h. es existieren $x_1, x_2 \in \mathbb{Z}_n^*$ mit $z_i \equiv x_i^r \pmod{n}$ für $i = 1, 2$. Es folgt $z_1 z_2 \equiv x_1^r x_2^r \equiv (x_1 x_2)^r \pmod{n}$. Wegen $x_1 x_2 \in \mathbb{Z}_n^*$ folgt die Abgeschlossenheit der Verknüpfung.

Das neutrale Element ist offensichtlich ein Element der Menge \mathbb{Z}_n^r , da gilt: $1^r \equiv 1 \pmod{n}$.

Das Inverse eines Elementes $z \equiv x^r \pmod n$ ist gegeben durch $(x^{-1})^r \pmod n$ mit $x^{-1} \in \mathbb{Z}_n^*$, da nach Voraussetzung $x \in \mathbb{Z}_n^*$ gilt. Somit ist das Inverse ein Element von \mathbb{Z}_n^r . \square

Lemma 4.4. *Für fest gewählte Zahlen r und $n = pq$ besitzt jedes Element $z \in \mathbb{Z}_n^r$ die gleiche Anzahl an r -ten Wurzeln.*

Beweis. Seien ζ_1, \dots, ζ_k die paarweise verschiedenen r -ten Einheitswurzeln. Wegen $1 \in \mathbb{Z}_n^r$ existiert zumindest eine solche Einheitswurzel, nämlich die Zahl 1 selbst. Sei nun $z \in \mathbb{Z}_n^r$, d.h. es existiert ein $x \in \mathbb{Z}_n^*$ mit $z \equiv x^r \pmod n$. Wegen der Gruppenstruktur von \mathbb{Z}_n^* sind die Elemente $x\zeta_i$ alle paarweise verschieden für verschiedene i . Es gilt nun $(x\zeta_i)^r \equiv x^r \zeta_i^r \equiv z \pmod n$, folglich hat man k paarweise verschiedene r -te Wurzeln von z gefunden. Angenommen, es gäbe noch eine weitere Wurzel x' von z , die nicht von der betrachteten Form ist, so müsste gelten $z \equiv x'^r \pmod n$, also auch $x^r \equiv x'^r \pmod n \Leftrightarrow (x'x^{-1})^r \equiv 1 \pmod n$. Somit wäre $x'x^{-1}$ eine r -te Einheitswurzel, also von der Form $x'x^{-1} \equiv \zeta_j \pmod n$. Nun würde aber gelten: $x' \equiv x\zeta_j \pmod n$, was unserer Annahme widerspricht. \square

Lemma 4.5. *Sind die Zahlen r und $\varphi(n)$ teilerfremd, so ist jede Zahl $z \in \mathbb{Z}_n^*$ ein r -ter Rest modulo n . Eine r -te Wurzel ist durch $z^A \pmod n$ gegeben, wobei A der Gleichung $Ar - B\varphi(n) = 1$ genügt.*

Beweis. Da nach Voraussetzung $\text{ggT}(r, \varphi(n)) = 1$ gilt, liefert der erweiterte euklidische Algorithmus natürliche Zahlen A und B , die der Gleichung $Ar - B\varphi(n) = 1$ genügen. Nun gilt:

$$(z^A)^r \equiv z^{B\varphi(n)+1} \equiv (z^{\varphi(n)})^B z \equiv z \pmod n.$$

Folglich ist z^A eine r -te Wurzel von z . \square

Nach dieser kurzen Betrachtung der höheren Reste wird nun zuerst der allgemeine Begriff einer Restklasse bzw. Nebenklasse eingeführt, der eine zentrale Rolle in den nächsten Abschnitten spielen wird.

Definition 4.6 (Restklasse). *Sei G eine abelsche Gruppe, U eine Untergruppe von G . Dann wird für ein $a \in G$ die Menge $a \cdot U := \{a \cdot g \mid g \in U\}$ als Restklasse von a bezeichnet.*

Satz 4.7. *Die Menge aller Restklassen bildet wieder eine Gruppe, die sogenannte Faktorgruppe, wobei die Verknüpfung zweier Restklassen durch die Gruppenverknüpfung zweier beliebiger Elemente der betrachteten Restklassen durchgeführt wird. Diese Verknüpfung ist wohldefiniert.*

Ein Beweis dazu ist in jedem Algebrabuch zu finden, z.B. in [Lamp91]. Desweiteren gilt der folgende

Satz 4.8. *Zwei Restklassen R_1, R_2 sind entweder disjunkt oder gleich, d.h. es gilt $R_1 \cap R_2 = \emptyset$ oder $R_1 = R_2$.*

Beweis. Ein Beweis findet sich z.B. in [Lamp91]. \square

Nach dieser allgemeinen Betrachtung über Restklassen wendet man sich nun bestimmten Restklassen zu. Die Gruppe \mathbb{Z}_n^* tritt an die Stelle der Gruppe G , die Gruppe der r -ten Reste \mathbb{Z}_n^r wird als Untergruppe U benutzt, d.h. im folgenden haben Restklassen R eines Elementes $g \in \mathbb{Z}_n^*$ die Form $g \cdot \mathbb{Z}_n^r$, d.h. für ein Element $w \in \mathbb{Z}_n^*$ gilt: $w \in R \Leftrightarrow \exists z \in \mathbb{Z}_n^r : w \equiv gz \pmod{n} \equiv gx^r \pmod{n}$ für eine r -te Wurzel x von z , wobei hier wieder die übliche Repräsentantenschreibweise von \mathbb{Z}_n verwendet wurde. Erste Anwendungen höherer Reste nebst den dazu eng verbundenen Restklassen in der Kryptographie sind in [Bena87] zu finden, wo auch die im folgenden vorgestellte Notation eingeführt wurde.

Definition 4.9 (c-te Restklasse). *Seien $r, n, y \in \mathbb{Z}$ gegeben. Ist ein Element $w \in \mathbb{Z}_n^*$ darstellbar als $w \equiv y^c z \pmod{n}$ für ein $z \in \mathbb{Z}_n^r$ (d.h. $w \in y^c \cdot \mathbb{Z}_n^r$), so nennt man das Element w von c -ter Restklasse bzgl. r, n, y . Die Menge aller Elemente aus \mathbb{Z}_n^* von c -ter Restklasse bzgl. r, n, y wird mit $RC[c]_{(r,n,y)}$ bezeichnet bzw. mit $RC[c]$ sofern r, n, y aus dem Kontext ersichtlich sind.*

Offensichtlich gilt für alle $y \in \mathbb{Z}_n^*$ die Gleichung $RC[0]_{(r,n,y)} = \mathbb{Z}_n^r$.

Definition 4.10 (Norm). *Sei ein Tripel (r, n, y) wie oben gegeben. Die Norm von (r, n, y) , i.Z. $|(r, n, y)|$ ist die kleinste natürliche Zahl m mit $y^m \in \mathbb{Z}$. Sollte eine solche Zahl m nicht existieren, definiere $|(r, n, y)| := \infty$.*

Für $y \notin \mathbb{Z}_n^*$ gilt $|(r, n, y)| = \infty$, da folglich auch für alle Elemente y^i gilt: $y^i \notin \mathbb{Z}_n^*$. Für $y \in \mathbb{Z}_n^*$ liefert die Zahl r offensichtlich eine obere Schranke für die Norm von (r, n, y) .

Wie bereits in Satz 4.7 erwähnt wurde, sind zwei Restklassen entweder gleich oder disjunkt. Dies gilt insbesondere für die betrachteten Restklassen $RC[c]$. Es bleibt die Frage bestehen, wie man möglichst effizient testen kann, ob zwei gegebene Restklassen $RC[c_1]$ und $RC[c_2]$ gleich sind. Mit dieser Frage beschäftigt sich das folgende Lemma.

Lemma 4.11. *Sei (r, n, y) mit $y \in \mathbb{Z}_n^*$ gegeben und sei $m := |(r, n, y)|$. Dann gilt: $RC[c_1] = RC[c_2]$ genau dann wenn $c_1 \equiv c_2 \pmod{m}$ gilt.*

Beweis. Ein Beweis ist in [Bena87] zu finden. □

Eine unmittelbare Konsequenz des Lemmas besteht darin, dass es genau $m = |(r, n, y)|$ paarweise verschiedene Restklassen gibt, die durch

$$\mathbb{Z}_n^r = RC[0], RC[1], \dots, RC[m-1]$$

gegeben sind. Eine weitere Frage besteht darin, zu gegebenen Elementen w_1, w_2 zu entscheiden, ob sie der selben Restklasse angehören. Ein Kriterium dafür liefert folgendes Lemma.

Lemma 4.12. *Sei (r, n, y) mit $y \in \mathbb{Z}_n^*$ gegeben. Zwei Elemente $w_1, w_2 \in \mathbb{Z}_n^*$ liegen in der selben Restklasse genau dann wenn $w_1 w_2^{-1} \in \mathbb{Z}_n^r$ gilt.*

Beweis. Der Beweis findet sich wiederum in [Bena87]. □

Wie bereits erwähnt, entspricht die Anzahl der paarweise verschiedenen Restklassen gerade der Norm von (r, n, y) . Um die maximal mögliche Anzahl paarweise verschiedener Restklasse zu erhalten, muss folglich die Bedingung $|(r, n, y)| = r$ gestellt werden. Dies führt zu folgender Definition.

Definition 4.13 (konsonant). Ein Tripel (r, n, y) heißt konsonant, wenn gilt $|(r, n, y)| = r$. Für prime r spricht man von *primer Konsonanz*. Gilt weiterhin $r|\varphi(r)$ und $ggT(r, \frac{\varphi(n)}{r}) = 1$, so heißt das Tripel *perfekt konsonant*.

In Benaloh's Dissertation [Bena87] wurden verschiedene Eigenschaften konsonanter Tripel bewiesen, wobei ich mich auf einen wichtigen Satz beschränken werde.

Satz 4.14. Ist (r, n, y) ein perfekt konsonantes Tripel, so kann jedes $w \in \mathbb{Z}_n^*$ dargestellt werden als $w \equiv y^c z$ für genau ein $0 \leq c < r$ und genau ein $z \in \mathbb{Z}_n^r$.

Die eindeutige Zahl c des obigen Satzes wird im folgenden $[w]_{(r,n,y)}$ (bzw. einfach $[w]$) geschrieben werden. Die Berechnung der Zahl $[w]$ entspricht offensichtlich der Berechnung des diskreten Logarithmus in der Faktorgruppe $\mathbb{Z}_n^*/\mathbb{Z}_n^r$.

Nach dieser ausführlicheren Einführung in die Theorie der Restklassen kann man nun den Bezug zur eigentlichen Kryptographie herstellen. Die nun folgende, auf dieser Darstellung basierende Funktion fungiert als Grundlage vieler Kryptosysteme, deren Sicherheit auf Annahmen mit höheren Resten beruht. Sei ein Tripel (r, n, y) mit $y \in \mathbb{Z}_n^*$ gegeben, dann wird eine Funktion $\epsilon_{(r,n,y)}$ (bzw. einfach ϵ) definiert durch

$$\epsilon : \mathbb{Z}_r \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$$

$$(c, x) \rightarrow y^c x^r \pmod n.$$

Die Funktion ϵ wird bisweilen als Restklassenfunktion bezeichnet. Anhand der vorgestellten Notation kann man nun zu den eigentlichen Annahmen kommen. Die Funktion ϵ wurde in [Bena87] eingeführt und zur Realisierung von Wahlprotokollen benutzt. Auf die genaue Anwendung der Funktion ϵ bzw. deren Modifikationen wird bei der Darstellung der darauf aufbauenden Systeme in den Kapiteln 4.3.3, 4.3.4 und 4.3.5 näher eingegangen. Zur Funktion ϵ gilt das folgende Lemma:

Lemma 4.15. Definiert man auf $\mathbb{Z}_r \times \mathbb{Z}_n^*$ eine Verknüpfung \oplus durch

$$(c_1, x_1) \oplus (c_2, x_2) = (c_1 + c_2 \pmod r, x_1 x_2 \pmod n)$$

so wird die Menge $\mathbb{Z}_r \times \mathbb{Z}_n^*$ zu einer abelschen Gruppe und ϵ zu einem Gruppenhomomorphismus.

Beweis. Im folgenden wird nur die Homomorphieeigenschaft gezeigt. Ein Beweis, dass $\mathbb{Z}_r \times \mathbb{Z}_n^*$ Gruppe ist, ist in [Bena87] zu finden. Seien nun $(c_1, x_1), (c_2, x_2) \in \mathbb{Z}_r \times \mathbb{Z}_n^*$. Dann gilt:

$$\begin{aligned} \epsilon((c_1, x_1) \oplus (c_2, x_2)) &= \epsilon(c_1 + c_2 \pmod r, x_1 x_2 \pmod n) \\ &= y^{c_1 + c_2 \pmod r} (x_1 x_2)^r \pmod n \\ &= (y^{c_1} x_1^r \pmod n)(y^{c_2} x_2^r \pmod n) \\ &= \epsilon(c_1, x_1)\epsilon(c_2, x_2). \end{aligned}$$

□

Der nächste Abschnitt beschäftigt sich mit neuen, auf höheren Resten aufbauenden Annahmen, die in den später dargestellten Systemen die Basis für die Sicherheitsbetrachtungen bilden.

4.3.2 Definition der Annahmen

Man beginnt mit der Definition der *Higher Residuosity Assumption*. Sie behauptet, dass das Ziehen von r -ten Wurzeln in \mathbb{Z}_n schwer ist.

Definition 4.16 (Higher Residuosity Assumption).

Zahlentheoretisch: Seien $z \in \mathbb{Z}_n^r$, $r \in \mathbb{Z}_{\varphi(n)}$, $n = pq$ mit p, q prim gegeben. Gesucht: $x \in \mathbb{Z}_n^*$ mit $z \equiv x^r \pmod{n}$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(x^r \equiv z \pmod{n} \\
 \wedge x \in \mathbb{Z}_n^* \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n := pq; \\
 r \in_R \mathbb{Z}_{\varphi(n)}; \\
 z \in_R \mathbb{Z}_n^r; \\
 x \leftarrow A(l, n, r, z) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Offensichtlich stellt dieses Problem eine Verallgemeinerung des RSA-Problems dar. Obwohl dieses Problem in der Mathematik in der elementaren und algorithmischen Zahlentheorie erforscht wurde und wird, wurde es für $r > 2$ erst verhältnismäßig spät in der Kryptographie betrachtet. Analog zu dem Berechnungsproblem wird das dazugehörige Entscheidungsproblem definiert.

Definition 4.17 (Decisional Higher Residuosity Assumption).

Zahlentheoretisch: Seien $z \in \mathbb{Z}_n^*$, $r \in \mathbb{Z}_{\varphi(n)}$, $n = pq$ mit p, q prim gegeben. Entscheide, ob gilt $z \in \mathbb{Z}_n^r$.

Kryptographisch: Seien

$$\begin{aligned}
 P_{A,1}(l) := P(b = 1) &:: \\
 p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n := pq; \\
 r \in_R \mathbb{Z}_{\varphi(n)}; \\
 z \in_R \mathbb{Z}_n^r; \\
 b \leftarrow A(l, n, r, z).
 \end{aligned}$$

und

$$\begin{aligned}
 P_{A,2}(l) := P(b = 1) &:: \\
 p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n := pq; \\
 r \in_R \mathbb{Z}_{\varphi(n)}; \\
 z \in_R \mathbb{Z}_n^*; \\
 b \leftarrow A(l, n, r, z).
 \end{aligned}$$

gegeben. Die Definition ist dann \forall prob. poly. Algo. A :

$$|P_{A,1}(l) - P_{A,2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

Diese Definition bilden das kanonische “Decision Problem” der ersten Definition, was inzwischen bei vielen der hier vorgestellten Annahmen üblich ist. Dieses Problem fällt noch stärker in den Bereich der elementaren Zahlentheorie als das ursprüngliche Problem, da sich Mathematiker naturgemäß mehr Gedanken um Lösbarkeit als um Effizienz machen. Das Problem wurde demzufolge schon seit langem untersucht, jedoch erweist sich auch hier das Problem im heuristischen Sinne als schwer, sofern die Faktorisierung von n unbekannt ist. Bei bekannter Faktorisierung von n läßt sich das Problem sehr einfach lösen, wie in Kapitel 5 gezeigt wird.

Einen Spezialfall der *Decisional Higher Residuosity Assumption* stellt die folgende Annahme dar, auf dem die Sicherheit des ursprünglichen Benaloh-Systems basiert [Bena87]:

Definition 4.18 (Prime Residuosity Assumption).

Zahlentheoretisch: Seien $z \in \mathbb{Z}_n^*$, $r \in \mathbb{P} \cap \mathbb{Z}_{\varphi(n)}$ und $n = pq$ mit p, q prim gegeben. Entscheide, ob gilt: $z \in \mathbb{Z}_n^r$?

Kryptographisch: Seien

$$\begin{aligned} P_{A,1}(l) := P(b = 1) : \\ p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\ n := pq; \\ r \in_R \mathbb{P} \cap \mathbb{Z}_{\varphi(n)}; \\ z \in_R \mathbb{Z}_n^r; \\ b \leftarrow A(l, n, r, z). \end{aligned}$$

und

$$\begin{aligned} P_{A,2}(l) := P(b = 1) : \\ p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\ n := pq; \\ r \in_R \mathbb{P} \cap \mathbb{Z}_{\varphi(n)}; \\ z \in_R \mathbb{Z}_n^*; \\ b \leftarrow A(l, n, r, z). \end{aligned}$$

gegeben. Die Definition ist dann \forall prob. poly. Algo. A :

$$|P_{A,1}(l) - P_{A,2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

Im Jahre 1997 wurde in [OkUc97] ein neues, auf höheren Resten aufgebautes System vorgestellt, das in Abschnitt 4.3.4 dargestellt wird. Die Sicherheit des Systems basiert auf einer neuen Annahme, der *p-Subgroup Assumption*, die auf der *modifizierten* Faktorisierungsannahme aufbaut.

Definition 4.19 (p-Subgroup Assumption).

Zahlentheoretisch: Sei $n = p^2q$ für p, q prim gegeben. Weiterhin seien $r, r' \in \mathbb{Z}_n$ und $g \in \mathbb{Z}_n^*$

gegeben, so dass gilt: $\text{ord}(g^{p-1}) = p \bmod p^2$. Desweiteren sei $h = g^n \bmod n$. Unterscheide die beiden Elemente $h^r \bmod n$ und $gh^{r'} \bmod n$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(b' = b) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= p^2 q; \\
 g &\in_R \mathbb{Z}_n^* \text{ mit } \text{ord}(g^{p-1}) = p \bmod p^2; \\
 h &:= g^n \bmod n; \\
 r, r' &\in_R \mathbb{Z}_n; \\
 h_0 &:= h^r \bmod n; \\
 h_1 &:= gh^{r'} \bmod n; \\
 b &\in_R \{0, 1\}; \\
 b' &\leftarrow A(l, n, g, h, h_b) \\
 &\leq \frac{1}{2} + \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Im Jahre 1999 wurde von Pascal Paillier in [Pail99] ein Verschlüsselungssystem in verschiedenen Varianten vorgestellt, die die Einführung mehrerer abgewandelter Annahmen nötig machte, um die Systeme beweisbar sicher zu halten. Der Autor beginnt mit der Einführung einer neuen Annahme, der *Decisional Composite Residuosity Assumption*:

Definition 4.20 (Decisional Composite Residuosity Assumption $CR(P^2 Q^2)$).

Zahlentheoretisch: Seien $z \in \mathbb{Z}_n^*$, $n = pq$ sei von unbekannter Faktorisierung, p, q prim. Entscheide, ob gilt: $z \in \mathbb{Z}_{n^2}^n$.

Kryptographisch: Seien

$$\begin{aligned}
 P_{A,1}(l) &:= P(b = 1) :: \\
 p, q &\in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 z &\in_R \mathbb{Z}_{n^2}^n; \\
 b &\leftarrow A(l, n, z).
 \end{aligned}$$

und

$$\begin{aligned}
 P_{A,2}(l) &:= P(b = 1) :: \\
 p, q &\in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 z &\in_R \mathbb{Z}_{n^2}^*; \\
 b &\leftarrow A(l, n, z).
 \end{aligned}$$

gegeben. Die Definition ist dann $\forall \text{prob. poly. Algo. } A :$

$$|P_{A,1}(l) - P_{A,2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

Das Problem wird im folgenden mit $CR[n]$ bezeichnet. Der Autor behauptet in [Pail99], dass auch dieses Problem schwer ist und baut auf dieser Behauptung sein Kryptosystem auf. Obwohl dieses neue Problem die zugrundeliegende *Decisional Higher Residuosity Assumption* durch die Abhängigkeit von Exponent und Modulus anschaulich stark einschränkt, ist auch diese Annahme bis heute nicht gebrochen worden und man nimmt an, dass dies auch in Zukunft nicht geschehen wird (zumindest Paillier hofft darauf).

In seinem Artikel stellt Pascal Paillier eine weitere interessante neue Annahme vor, die sich in Kapitel 5 als grundlegend zur Betrachtung der Komplexität einzelner Probleme erweist, die auf höheren Resten aufbauen. Ihre Definition basiert auf einer Modifikation f_g der bereits eingeführten Restklassenfunktion ϵ . Hierbei wird der Modulus n durch n^2 ersetzt. An die Stelle des Exponenten r tritt die Zahl n .

Sei $n = pq$ und $g \in \mathbb{Z}_{n^2}^*$. Dann ist durch g folgende Funktion f_g definiert:

$$f_g : \mathbb{Z}_n \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_{n^2}^*$$

$$(c, x) \rightarrow g^c x^n \pmod{n^2}.$$

Die Funktion f_g stellt sich als bijektiv heraus, sofern die Ordnung von g ein von 0 verschiedenes Vielfaches von n ist. Ein Beweis dazu kann in [Pail99] gefunden werden.

Im folgende bezeichne $B_\alpha \subset \mathbb{Z}_{n^2}^*$ die Menge aller Elemente der Ordnung $n \cdot \alpha$. Für $\lambda = kgV(p-1, q-1)$ definiere

$$B := \bigcup_{i=1}^{\lambda} B_i.$$

Wie bereits in der Einleitung dieses Kapitels erwähnt wurde, spielt der Begriff der Restklasse bei dieser Funktion eine wichtige Rolle. Da jedes Element gemäß unseren Vorbetrachtungen nur in genau einer Restklasse vorkommen kann, liegt es nahe, sich näher mit dieser Klasse zu beschäftigen. Man definiert zuerst den Begriff einer n -ten Restklasse, der mittels der Funktion f_g analog zum weiter oben eingeführten Begriff einer c -ten Restklasse bzgl. der Funktion ϵ eingeführt wird.

Definition 4.21 (n-te Restklasse). Sei $g \in B$, $w \in \mathbb{Z}_{n^2}^*$. Dann ist die n -te Restklasse $[w]_g$ von w zur Basis g die durch die Bijektivität eindeutig bestimmte Zahl $c \in \mathbb{Z}_n$ für die $x \in \mathbb{Z}_n^*$ existiert, so dass gilt

$$f_g(c, x) = w.$$

Während man in [Bena87] noch ein perfekt konsonantes Tripel (r, n, y) brauchte, um eine eindeutige Darstellung zu erhalten, ist dies durch die Bijektivität der Funktion f_g bereits gegeben. Anhand dieser Definition kann man nun ein weiteres Hilfsproblem definieren:

Definition 4.22 (n-th Residuosity Class Assumption $Class(P^2 Q^2, g)$).

Zahlentheoretisch: Seien $g \in B$, $w \in \mathbb{Z}_{n^2}^*$ gegeben. Berechne die n -te Restklasse von w zur Basis g .

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(\exists x \in \mathbb{Z}_n^* : f_g(c, x) = w) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &g \in_R B; \\
 &w \in_R \mathbb{Z}_{n^2}^*; \\
 &c \leftarrow A(l, n, g, w) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Die Komplexität dieses Problems ist nicht mehr offensichtlich, da man hier nicht bereits intuitiv sagen kann, ob es sich hierbei um ein schwieriges Problem handelt oder nicht. Man beachte desweiteren, dass die Komplexität von der gegebenen Basis g abzuhängen scheint. Man kann jedoch zeigen, dass die Komplexität dieses Problems *unabhängig* von der gegebenen Basis g ist, was zur Folge hat, dass das gestellte Problem zu einem Problem vereinfacht werden kann, dessen Komplexität nur noch von n abhängt. Diese Behauptung wird in folgenden Lemma bewiesen:

Lemma 4.23. *Class(n, g) ist random-self-reducible über $b \in B$, d.h. das Problem ist entweder für beliebige g effizient lösbar oder für gar kein g .*

Beweis. Seien $g_1, g_2 \in B$ und $w \in \mathbb{Z}_{n^2}^*$ gegeben. Angenommen man besitzt ein Orakel A , dass $[w]_{g_1}$ berechnen kann. Wie bereits erwähnt, entspricht die Berechnung von $[w]_{g_1}$ der Berechnung des diskreten Logarithmus in der Faktorgruppe, sei \log_{g_1} dieser Logarithmus, also $[w]_{g_1} = \log_{g_1}(w)$. Aus dem Logarithmengesetz $\log_{g_1}(w) = \log_{g_2}(w) \cdot \log_{g_1}(g_2)$ folgt somit die Gleichung

$$[w]_{g_1} \equiv [w]_{g_2} [g_2]_{g_1} \pmod{n}.$$

Es folgt $[g_1]_{g_2} \equiv [g_2]_{g_1}^{-1} \pmod{n}$, d.h. $[g_1]_{g_2}$ ist invertierbar modulo n . Das Orakel A liefert nun auf Eingabe von g_2 und w die Werte von $[g_2]_{g_1}$ und $[w]_{g_1}$. Damit gilt

$$[w]_{g_2} \equiv [w]_{g_1} [g_2]_{g_1}^{-1} \pmod{n}.$$

und folglich ist $[w]_{g_2}$ mit einer nicht vernachlässigbaren Wahrscheinlichkeit berechenbar, was den Beweis vervollständigt. □

Unter Benutzung des bewiesenen Lemmas kann das Problem in ein äquivalentes Problem umformuliert werden, das nur noch in Abhängigkeit von n definiert ist:

Definition 4.24 (Composite Residuosity Class Assumption $Class(P^2 Q^2)$).

Zahlentheoretisch: Sei $w \in \mathbb{Z}_{n^2}^*$ gegeben. Berechne die n -te Restklasse von w zu einer selbst-, jedoch unabhängig von w , gewählten Basis g .

Kryptographisch: $\forall \text{prob. poly. Algo. } A_1, A_2 :$

$$\begin{aligned}
 P(\exists x \in \mathbb{Z}_n^* : f_g(c, x) = w) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, loc) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \mathbb{Z}_{n^2}^*; \\
 &c \leftarrow A_2(l, n, w, g, loc) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

In Kapitel 5 wird das Problem mit den anderen vorgestellten Problemen verglichen und demzufolge auf seine Komplexität untersucht. Analog zu den vorigen Problemen definiert man die Decision-Variante des Problems:

Definition 4.25 (Decisional Composite Residuosity Class Assumption $D\text{-Class}(P^2 Q^2)$).

Zahlentheoretisch: Seien $w \in \mathbb{Z}_{n^2}^*$, $g \in B$ und $x \in \mathbb{Z}_n$ gegeben. Entscheide, ob gilt: $x = [w]_g$?
Kryptographisch: Seien

$$\begin{aligned}
 P_{A_1, A_2, 1}(l) &:= P(b = 1) :: \\
 &p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, loc) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \mathbb{Z}_{n^2}^*; \\
 &c = [w]_g; \\
 &b \leftarrow A_2(l, n, g, w, c, loc).
 \end{aligned}$$

und

$$\begin{aligned}
 P_{A_1, A_2, 2}(l) &:= P(b = 1) :: \\
 &p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, loc) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \mathbb{Z}_{n^2}^*; \\
 &c \in_R \mathbb{Z}_n; \\
 &b \leftarrow A(l, n, g, w, c, loc).
 \end{aligned}$$

gegeben. Die Definition ist dann $\forall \text{prob. poly. Algo. } A_1, A_2 :$

$$|P_{A_1, A_2, 1}(l) - P_{A_1, A_2, 2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

Ich merke bereits jetzt an, dass dieses Problem in der Fachliteratur oft mit demselben Namen wie ein weiter oben vorgestelltes Problem bezeichnet wird: als *Decisional Composite Residuosity Assumption*. Der Grund liegt in der Äquivalenz der beiden Probleme, die in Kapitel 5 gezeigt

wird. In dieser Arbeit werden die Namen dieser (per Definition unterschiedlichen) Probleme jedoch strikt unterschieden. In [Pail99] wird noch eine weitere Annahme mit ihrem Decision-Problem vorgestellt, die sich als eine Kombination der obigen Annahme und dem *Discrete Logarithm Problem* erweist:

Definition 4.26 (Partial Discrete Logarithm Problem $PDL(P^2 Q^2, g)$).

Zahlentheoretisch: Seien $g \in B$, $w \in \langle g \rangle$ gegeben. Berechne $[w]_g$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A_1, A_2 :$

$$\begin{aligned}
 P(\exists x \in \mathbb{Z}_n^* : f_g(c, x) = w) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, \text{loc}) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \langle g \rangle; \\
 &c \leftarrow A_2(l, n, g, w, \text{loc}) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Analog wird die Decision-Variante definiert:

Definition 4.27 (Decisional Partial Discrete Logarithm Problem $D\text{-}PDL(P^2 Q^2, g)$).

Zahlentheoretisch: Seien $g \in B$, $w \in \langle g \rangle$ und $x \in \mathbb{Z}_n$ gegeben. Entscheide, ob gilt: $[w]_g = x$.

Kryptographisch: Seien

$$\begin{aligned}
 P_{A_1, A_2, 1}(l) := P(b = 1) &:: \\
 &p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, \text{loc}) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \langle g \rangle; \\
 &c = [w]_g; \\
 &b \leftarrow A_2(l, n, g, w, c, \text{loc}).
 \end{aligned}$$

und

$$\begin{aligned}
 P_{A_1, A_2, 2}(l) := P(b = 1) &:: \\
 &p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 &n := pq; \\
 &(g, \text{loc}) \leftarrow A_1(l, n) \text{ mit } g \in B; \\
 &w \in_R \langle g \rangle; \\
 &c \in_R \mathbb{Z}_n; \\
 &b \leftarrow A_2(l, n, g, w, c, \text{loc}).
 \end{aligned}$$

gegeben. Die Definition ist dann $\forall \text{prob. poly. Algo. } A_1, A_2 :$

$$|P_{A_1, A_2, 1}(l) - P_{A_1, A_2, 2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

In [Pail99] wird ein Verschlüsselungssystem vorgestellt, das auf der oben eingeführten *Decisional Composite Residuosity Assumption* basiert. Desweiteren wird eine Modifikation dieses Systems vorgestellt, die auf dem *Decisional Partial Discrete Logarithm Problem* aufbaut mit dem Ziel, eine höhere Effizienz bei Nachrichtenentschlüsselung zu erhalten.

Das erste der beiden Verschlüsselungssysteme wird in Abschnitt (4.3.5) vorgestellt. Im nächsten Abschnitt wird das Verschlüsselungssystem von Naccache und Stern vorgestellt, das im Großen und Ganzen dem ursprünglichen Benaloh-System entspricht.

4.3.3 Das Verschlüsselungssystem von D. Naccache und J.Stern

Das im folgenden vorgestellte System stammt aus [NaSt98]. Die einzelnen Algorithmen des Systems werden zuerst informell erklärt, die konkrete Realisierung des Systems wird im Anschluss zusammengefasst dargestellt.

Zur Schlüsselgenerierung wird ein üblicher RSA-Modulus n erzeugt zusammen mit einer ungeraden, B -glatten Zahl σ , so dass gilt: $\sigma | \varphi(n)$ und $ggT(\sigma, \frac{\varphi(n)}{\sigma}) = 1$. Wähle nun ein $g \in \mathbb{Z}_n$, wobei die Ordnung von g ein großes Vielfaches von σ sein muss. Der öffentliche Schlüssel besteht dann aus (n, g, σ) , der geheime besteht aus (p, q) .

Zur Verschlüsselung wird die Restklassenfunktion ϵ benutzt. Eine Nachricht m wird als $c \equiv x^\sigma g^m \pmod n$ verschlüsselt für ein $x \in_R \mathbb{Z}_n$.

Die Entschlüsselung eines gegebenen Schlüsseltextes wird mittels des chinesischen Restsatzes und dem Pohlig-Hellman Algorithmus [PoHe78] durchgeführt. Seien p_i die Primfaktoren von σ für $1 \leq i \leq k$. Der Algorithmus berechnet nun die Werte $m_i := m \pmod{p_i}$, um sie dann wieder unter Zuhilfenahme des chinesischen Restsatzes zusammensetzen. Um anhand eines gegebenen Schlüsseltextes $c = x^\sigma g^m \pmod n$ die besagten m_i zu errechnen, berechnet man

$$c_i := c \frac{\varphi(n)}{p_i} \equiv x \frac{\sigma \varphi(n)}{p_i} g \frac{m \varphi(n)}{p_i} \equiv 1 \cdot g \frac{(m_i + y_i p_i) \varphi(n)}{p_i} \equiv g \frac{m_i \varphi(n)}{p_i} g^{y_i \varphi(n)} \equiv g \frac{m_i \varphi(n)}{p_i} \pmod n.$$

Indem nun das Ergebnis mit allen möglichen Potenzen $g \frac{j \varphi(n)}{p_i}$ für $1 \leq j < p_i$ verglichen wird, erhält man das gesuchte m_i .

Kommen wir nun zur genauen Vorstellung des Systems:

Schlüsselgenerierung:

1. Wähle k paarweise verschiedene, kleine, ungerade Primzahlen für ein gerades k .
2. Setze $u := \prod_{i=1}^{k/2} p_i$ und $v := \prod_{i=k/2+1}^k p_i$ sowie $\sigma = uv = \prod_{i=1}^k p_i$.
3. Wähle zwei große Primzahlen a, b , so dass sowohl $p := 2au + 1$ als auch $q := 2bv + 1$ prim sind.
4. Setze $n := pq$.
5. Wähle $g \in_R \mathbb{Z}_n$ und teste ob für alle $i \leq k$ gilt: $g \frac{\varphi(n)}{p_i} \not\equiv 1 \pmod n$. Sollte dies nicht gelten, wähle ein neues g .

Wählt man z.B. für die p_i die ersten k ungeraden Primzahlen, so beträgt die Wahrscheinlichkeit, in Schritt 5 ein passendes g zu wählen in etwa $\frac{1}{lnk}$ [NaSt98].

Verschlüsselung: Eine gegebene Nachricht $m \in \mathbb{Z}_\sigma$ wird verschlüsselt durch

1. Wähle $x \in_R \mathbb{Z}_n$.
2. Setze $c = x^\sigma g^m \bmod n$.

Der Schlüsseltext besteht aus dem Element c .

Entschlüsselung: Um einen gegebenen Schlüsseltext c wieder zu entschlüsseln, wendet man den folgenden Algorithmus an. Er entspricht der bereits zu Anfang des Kapitels durchgeführten Vorüberlegung.

1. for $i = 1$ to k do {
2. $c_i := c^{\frac{\varphi(n)}{p_i}} \bmod n$.
3. for $j = 0$ to $p_i - 1$ {
4. if $c_i = g^{\frac{j\varphi(n)}{p_i}} \bmod n$ setze $m_j := j$. } }
5. Berechne m als Ergebnis des chinesischen Restsatzes angewandt auf die Zahlen m_i, p_i .

4.3.4 Das Verschlüsselungssystem von T. Okamoto und S. Uchiyama

Das System im folgenden vorgestellte System stammt aus [OkUc97]. Es erweist sich als semantisch sicher unter der bereits vorgestellten p -Subgroup Assumption. Zur Darstellung des Systems sind noch einige Vorüberlegungen nötig. Man beginnt mit der Einführung des Begriffes einer p -Sylow-Gruppe:

Definition 4.28 (p-Sylow-Gruppe). Sei p eine ungerade Primzahl. Dann ist die p -Sylow-Gruppe von $\mathbb{Z}_{p^2}^*$ definiert durch

$$\Gamma = \{x \in \mathbb{Z}_{p^2}^* \mid x \equiv 1 \pmod{p}\}$$

Die Gruppe $\mathbb{Z}_{p^2}^*$ stellt bekanntermaßen eine zyklische Gruppe der Ordnung $p(p-1)$ dar, die genau p Elemente x_i besitzt mit $x_i \equiv 1 \pmod{p}$; somit gilt $|\Gamma| = p$. Folglich ist auf der Gruppe Γ die folgende Funktion L_p wohldefiniert:

$$\begin{aligned} L_p : \Gamma &\rightarrow \mathbb{Z}_p \\ x &\rightarrow \frac{x-1}{p} \end{aligned}$$

Die Funktion L besitzt die folgende Eigenschaft.

Lemma 4.29. Seien $a, b \in \Gamma$, dann gilt:

$$L_p(ab) = L_p(a) + L_p(b) \pmod{p}.$$

Desweiteren ist L ein Isomorphismus.

Beweis. Offentlichtlich gilt

$$\frac{ab-1}{p} = \frac{(a-1)}{p}(b-1) + \frac{(a-1)}{p} + \frac{(b-1)}{p} = L_p(a)(b-1) + L_p(a) + L_p(b).$$

Es gilt weiter

$$L_p(ab) = \frac{ab-1 \bmod p^2}{p} = \frac{ab-1}{p} \bmod p.$$

Wegen $b-1 \equiv 0 \pmod p$ folgt somit, dass L_p einen Homomorphismus darstellt. Die Bijektivität der Funktion L_p ist offensichtlich, was den Beweis vervollständigt. □

Aufgrund der homomorphen Eigenschaft wird die Funktion L_p in diesem Zusammenhang auch als Logarithmusfunktion bezeichnet. Die Funktion L_p wird im Verschlüsselungssystem die entscheidende Rolle bei der Entschlüsselung einnehmen. Die Verschlüsselung wird mittels der bereits vorgestellten Funktion ϵ vorgenommen.

Schlüsselgenerierung:

1. Wähle zwei Primzahlen $p, q \in_R$ Menge der l -Bit Primzahlen und setze $n = p^2q$.
2. Wähle $g \in_R \mathbb{Z}_n^*$, so dass für $g_p := g^{p-1} \bmod p^2$ gilt: $\text{ord}(g_p) = p$.
3. Setze $h := g^n \bmod n$.

Der öffentliche Schlüssel besteht dann aus (n, g, h, k) , der geheime aus (p, q) .

Verschlüsselung: Eine gegebene Nachricht $m \in \mathbb{Z}_n$ wird wie folgt verschlüsselt:

1. Wähle $r \in_R \mathbb{Z}_n$.
2. Setze $c = g^m h^r \bmod n$.

Der Schlüsseltext besteht dann aus dem Element c .

Entschlüsselung: Die Entschlüsselung wird anhand der Logarithmusfunktion L_p durchgeführt:

1. Setze $c_p := c^{p-1} \bmod p^2$.
2. Berechne die ursprüngliche Nachricht m durch $m = \frac{L_p(c_p)}{L_p(g_p)} \bmod p$.

4.3.5 Das Verschlüsselungssystem von Pascal Paillier

In [Pail99] stellt der Autor neben den bereits eingeführten Annahmen ein public-key Verschlüsselungssystem vor, das sich als semantisch sicher unter der *Decisional Composite Residuosity Assumption* erweist. Es wird im folgenden vorgestellt. Zur Darstellung des Systems benötigt man noch einige Vorüberlegungen. Man beginnt mit dem folgenden Lemma.

Lemma 4.30. *Es gilt $[w]_g = 0$ genau dann, wenn w ein n -ter Rest modulo n^2 ist. Desweiteren gilt:*

$$\forall w_1, w_2 \in \mathbb{Z}_{n^2}^* [w_1 w_2]_g = [w_1]_g + [w_2]_g \pmod n$$

d.h. die Funktion $w \rightarrow [w]_g$ ist ein Homomorphismus von $(\mathbb{Z}_{n^2}^, \cdot)$ nach $(\mathbb{Z}_n^*, +)$ für jedes $g \in B$.*

Beweis. Man zeigt zuerst die erste Behauptung.

“ \Rightarrow ” Sei $[w]_g = 0$.

$$\Rightarrow f_g(0, y) = w \text{ für ein passendes } y.$$

$$\Rightarrow w \equiv g^0 \cdot y^n \equiv y^n \pmod{n^2}.$$

$$\Rightarrow w \text{ ist } n\text{-ter Rest modulo } n^2.$$

“ \Leftarrow ” Sei w ein n -ter Rest modulo n^2 .

$$\Rightarrow \exists y : w \equiv y^n \pmod{n^2}.$$

$$\Rightarrow w = f_g(0, y).$$

$$\Rightarrow [w]_g = 0.$$

Um die zweite Behauptung zu zeigen, rechnet man einfach die Homomorphie basierend auf den Eigenschaften von $f_g(x, y)$ nach. Man hat folgende Gleichungen:

$$1. w_1 w_2 = f_g([w_1 w_2]_g, y_0) = g^{[w_1 w_2]_g} y_0^n \pmod{n^2}$$

$$2. w_1 = f_g([w_1]_g, y_1) = g^{[w_1]_g} y_1^n \pmod{n^2}$$

$$3. w_2 = f_g([w_2]_g, y_2) = g^{[w_2]_g} y_2^n \pmod{n^2}$$

Aus 1) und 2) folgt nun

$$w_1 w_2 = g^{[w_1]_g} y_1^n \pmod{n^2} \cdot g^{[w_2]_g} y_2^n \pmod{n^2} = g^{[w_1]_g + [w_2]_g} (y_1 y_2)^n \pmod{n^2}$$

Aus der Eindeutigkeit der Lösung folgt nun direkt

$$g^{[w_1 w_2]_g} = g^{[w_1]_g + [w_2]_g} \pmod{n^2}$$

$$\Rightarrow [w_1 w_2]_g = [w_1]_g + [w_2]_g \pmod{\varphi(n^2)}$$

$$\Rightarrow [w_1 w_2]_g = [w_1]_g + [w_2]_g \pmod{n\varphi(n)}.$$

Da die Ordnung von g jedoch als ein Vielfaches von n festgelegt wurde, folgt nun die behauptete Gleichung

$$[w_1 w_2]_g = [w_1]_g + [w_2]_g \pmod{n}.$$

□

Ähnlich wie beim im vorigen Abschnitt vorgestellten Verschlüsselungssystem betrachtet man nun die Menge

$$S_n = \{u < n^2 \mid u \equiv 1 \pmod{n}\}$$

wobei hier im Gegensatz zum System von Okamoto und Uchijama der Modulus n verwendet wird. Offensichtlich ist S_n eine multiplikative Untergruppe von \mathbb{Z}_{n^2} , auf der die folgende Funktion wohldefiniert ist:

$$\begin{aligned} L_n : S_n &\rightarrow \mathbb{N} \\ u &\rightarrow \frac{u-1}{n} \end{aligned}$$

Man beweist zuerst das folgende Lemma:

Lemma 4.31. *Sei $\lambda = kgV(p-1, q-1)$. Dann gilt für alle $w \in \mathbb{Z}_{n^2}^*$: $L_n(w^\lambda \pmod{n^2}) = \lambda[w]_{1+n} \pmod{n}$, d.h. L_n entspricht der Logarithmusfunktion in der Faktorgruppe zur Basis $1+n$.*

Beweis. Da $1+n \in B$, gibt es ein eindeutig bestimmtes Paar $(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$, so dass gilt: $w \equiv (1+n)^{ab} \pmod{n^2}$. Für a muss nach Definition gelten: $a = [w]_{1+n}$. Dann gilt:

$$w^\lambda \equiv (1+n)^{a\lambda b^{n\lambda}} \equiv (1+n)^{a\lambda} \equiv \sum_{i=0}^{a\lambda} \binom{a\lambda}{i} n^i \equiv 1 + a\lambda n \pmod{n^2}$$

Es folgt nun unmittelbar

$$L_n(w^\lambda \pmod{n^2}) \equiv L_n(1 + a\lambda n \pmod{n^2}) \equiv \lambda a \pmod{n} \equiv \lambda[w]_{1+n} \pmod{n}$$

□

Da $[g]_{1+n}$ wegen $[g]_{1+n} = [1+n]_g^{-1}$ invertierbar modulo n ist, folgt aus dem zuletzt bewiesenen Lemma, dass auch $L_n(g^\lambda \pmod{n^2})$ modulo n invertierbar ist. Kennt man nun die Primfaktoren von n , so kann man offensichtlich $\lambda = kgV(p-1, q-1)$ berechnen und man erhält somit:

$$\frac{L_n(w^\lambda \pmod{n^2})}{L_n(g^\lambda \pmod{n^2})} = \frac{\lambda[w]_{1+n}}{\lambda[g]_{1+n}} = \frac{[w]_{1+n}}{[g]_{1+n}} = [w]_g \pmod{n}.$$

Dies ist genau die Eigenschaft, die im Verschlüsselungssystem von Paillier eine Entschlüsselung unter Benutzung der Primfaktoren p und q möglich macht. Wir kommen nun zum eigentlichen System.

Schlüsselgenerierung:

1. Wähle zwei Primzahlen $p, q \in_R$ Menge der l -Bit Primzahlen und setze $n = pq$.
2. Wähle $g \in_R B$. B steht hierbei für den im vorigen Abschnitt eingeführten Mengendurchschnitt derjenigen Elemente, die als Ordnung ein Vielfaches von n besitzen. Da es viele solche Elemente gibt, ist dies effizient möglich, indem für ein zufällig gewähltes g die Bedingung

$$ggT(L_n(g^\lambda \pmod{n^2}), n) = 1$$

getestet wird.

Der öffentliche Schlüssel besteht dann aus (n, g) , der geheime aus (p, q) .

Verschlüsselung: Eine gegebene Nachricht $m \in \mathbb{Z}_n$ wird wie folgt verschlüsselt:

1. Wähle $r \in_R \mathbb{Z}_n$.
2. Setze $c = g^m \cdot r^n \bmod n^2$.

Der Schlüsseltext besteht aus dem Element c .

Entschlüsselung: Zur Entschlüsselung eines Schlüsseltextes c führe die folgenden Schritte aus:

1. Berechne $\lambda = kgV(p-1, q-1)$.
2. Setze $m = \frac{L_n(c^\lambda \bmod n^2)}{L_n(g^\lambda \bmod n^2)} \bmod n$, was gemäß dem zuletzt bewiesenen Lemmas die ursprüngliche Nachricht m liefert.

4.3.6 Die Systeme in der Übersicht

Die wohl erste Anwendung von höheren Resten in der Kryptographie ist wie bereits erwähnt in [Bena87] zu finden. Josh Benaloh benutzte die Verschlüsselungsfunktion ϵ , um ein Wahlprotokoll zu konstruieren, dessen Sicherheit er aufbauend auf der *Prime Residuosity Assumption* beweisen konnte. Die drei vorgestellten Systeme lehnen sich mehr oder minder stark an die Arbeit von J. Benaloh an.

Das System von Naccache und Stern [NaSt98] nimmt sehr stark Bezug auf [Bena87], insbesondere ihre Entschlüsselungsmethode anhand des Pohlig-Hellman-Algorithmus wurde bereits von Benaloh benutzt. Die Neuerung, die das System lieferte war eine erheblich verbesserte Bandbreite der möglichen Nachrichten [NaSt98]. Nichtsdestotrotz bietet das System einen relativ ineffizienten Entschlüsselungsalgorithmus, zumindest im Vergleich mit den beiden übrigen Systemen.

Das System von Okamoto und Uchiyama [OkUc97] wurde in etwa zur gleichen Zeit wie das System in [NaSt98] entworfen, beide allerdings unabhängig voneinander. Dieses System nimmt wesentlich weniger Bezug auf das "Ursystem" von Benaloh. Neuerungen sind insbesondere der auf der modifizierten Faktorisierungsannahme beruhende Modulus $n = p^2q$ sowie ein neuartiger, effizienter Entschlüsselungsalgorithmus, der auf einer Trapdoor im Diskreten Log. basiert. Im Vergleich der beiden Systeme ist es meiner Meinung nach dieser Entschlüsselungsalgorithmus, der den Ausschlag dafür gibt, das System von Okamoto und Uchiyama dem System von Naccache und Stern vorzuziehen, da er sich als wesentlich eleganter und effizienter als der Ansatz mittels des Pohlig-Hellman-Algorithmus erweist.

Das System von Paillier [Pail99] wurde etwa ein Jahr nach den beiden übrigen Systemen entworfen. Hier wurde der Modulus durch n^2 ersetzt. Im Gegensatz zu den "glatten" Resten aus \mathbb{Z}_{pq}^* in [NaSt98] bzw. den primen Resten aus $\mathbb{Z}_{p^2q}^*$ in [OkUc97] beschäftigt sich dieses System mit zusammengesetzten Resten aus $\mathbb{Z}_{p^2q^2}^*$. Ähnlich wie in [OkUc97] wurde eine Trapdoor im Diskreten Log. gefunden, die für zusammengesetzte Reste anwendbar ist und als effizienter Entschlüsselungsalgorithmus des Systems benutzt wurde. Das System lehnt sich in der Grundidee an [OkUc97] an, erweist sich jedoch im Endeffekt als neuartiges und effizient einsetzbares Verschlüsselungssystem.

4.4 Die Starke RSA Annahme (*Strong RSA Assumption*)

4.4.1 Definitionen

Die Starke RSA Annahme wurde von Nico Baric und Birgit Pfitzmann in [BaPf97] im Jahre 1997 eingeführt.

Definition 4.32 (Strong RSA Assumption).

Zahlentheoretisch: Gegeben:

1. $n \in \mathbb{N}$ mit $n = pq$; p, q prim.

2. $c \in \mathbb{Z}_n$.

Gesucht: $(m, e) \in (\mathbb{Z}_n, \mathbb{P})$ mit $m^e \equiv c \pmod n$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(m^e \equiv c \pmod n \\
 \wedge e \in \mathbb{P} \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 & \quad n := pq; \\
 & \quad c \in_R \mathbb{Z}_n^*; \\
 & \quad (m, e) \leftarrow A(l, n, c)) \\
 & \leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Der Unterschied zur RSA Annahme besteht darin, dass man dem Angreifer die Wahl von e überlässt, wobei e als Primzahl zu wählen ist. Offensichtlich ist diese Annahme leichter zu brechen als das ursprüngliche RSA Problem, und es stellt sich die Frage, ob es nun mittels dieser Vereinfachung möglich ist, das Problem in polynomieller Laufzeit zu lösen. Auch hier scheint dies nicht der Fall zu sein. In [BaPf97] werden anschaulich Gründe dargelegt, die die Grundlage dafür bilden, auch dieses Problem als schwer anzusehen.

Im Gegensatz zum RSA Problem könnte die Schwäche dieses Problems darin liegen, dass man die frei wählbaren Komponenten m und e in Abhängigkeit voneinander wählt bzw. sich auf einen Exponenten beschränkt, für den das RSA Problem leichter zu lösen ist; ein solcher Exponent ist bis heute jedoch nicht gefunden worden.

In [FuOk97] wurde die Definition auf beliebige Exponenten verallgemeinert:

Definition 4.33 (Strong RSA Assumption 2).

Zahlentheoretisch: Gegeben:

1. $n \in \mathbb{N}$ mit $n = pq$; p, q prim;

2. $c \in \mathbb{Z}_n$;

Gesucht: $(m, e) \in (\mathbb{Z}_n, \mathbb{N} \setminus 1)$ mit $m^e \equiv c \pmod n$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(m^e \equiv c \pmod n) &:: p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 c &\in_R \mathbb{Z}_n^*; \\
 (m, e) &\leftarrow A(l, n, c) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Hier wird darauf verzichtet, dass e als Primzahl gewählt werden muss. Diese Verallgemeinerung führt jedoch nicht zu einer geringeren Sicherheit, sofern der Exponent e faktorisierbar bleibt bzw. solange man zumindest einen Faktor von e finden kann:

Ist q ein Primfaktor von e und m eine e -te Wurzel von c , so ist $m^{(e/q)}$ eine q -te Wurzel von c .

Solange man, wie in der Praxis üblich, den Exponent in der Grössenordnung von 2^{20} Bit wählt, kann man beide Definitionen als äquivalent ansehen. Lediglich wenn man Exponenten in der Grössenordnung von n zulässt, ist unklar, ob es sich noch immer um eine Äquivalenz handelt.

Da man auch von diesen Annahmen annimmt, dass sie nicht polynomiell lösbar sind, entwickelten Ronald Cramer und Victor Shoup ein Signatursystem, das auf der (zweiten) Starken RSA Annahme aufbaut [CrSh99]; es wird hier im Folgenden vorgestellt.

4.4.2 Das Signatursystem von R. Cramer, V. Shoup

Im folgenden seien l, l' zwei Sicherheitsparameter des vorgestellten Systems mit $l + 1 < l'$. Desweiteren bezeichne H eine kollisionsresistente Hashfunktion, deren Ausgabe $H(a)$ im Bereich $0 \leq H(a) \leq 2^l$ liegt.

Schlüsselerzeugung:

1. Wähle zwei l' -Bit Primzahlen p und q , für die gelten soll: $p = 2p' + 1$, $q = 2q' + 1$ mit p', q' prim. Setze $n = pq$.
2. Wähle $h, x \in_R QR_n$.
3. Wähle $e' \in_R$ Menge der l -bit Primzahlen.

Der public key besteht dann aus $pk = (n, h, x, e')$, der secret key aus $sk = (p, q)$.

Erzeugung einer Signatur: Sei eine Nachricht m gegeben. Wähle eine zufällige $(l + 1)$ -bit Primzahl $e \neq e'$ und $y' \in_R QR_n$.

1. Die Zahl x' genüge der Gleichung $(y')^{e'} = x'h^{H(m)}$.
2. Die Gleichung $y^e = x'h^{H(x')}$ wird nun nach y aufgelöst, indem mit Hilfe des secret keys eine RSA-Entschlüsselung durchgeführt wird.
3. Die Signatur besteht dann aus (e, y, y') .

Verifizierung der Signatur: Sei eine Signatur (e, y, y') gegeben.

1. Es wird überprüft, ob e eine ungerade $(l + 1)$ -Bit Zahl ist mit $e \neq e'$.
2. $x' = (y')^{e'} h^{-H(m)}$ wird berechnet.
3. Überprüfe, ob gilt: $x = y^e h^{-H(x')}$.

Unter der Voraussetzung, dass das Starke RSA Problem schwer ist und dass die gewählte Hashfunktion kollisionsresistent ist, wird in [CrSh99] bewiesen, dass dieses Signatursystem sicher ist gegen *adaptive chosen message attack*. Aus diesem Grund macht es auch keinen Sinn, sich Angriffe gegen das System zu überlegen; statt dessen müsste man versuchen, die Annahme zu brechen.

Hier erkennt man jedoch eine Besonderheit des vorgestellten Systems: Selbst wenn das Starke RSA Problem gebrochen werden könnte, reicht dies nicht zwangsweise zum Brechen des oben vorgestellten Systems. Man kann jedoch folgendes zeigen:

Satz 4.34. *Ein Algorithmus A , der die Strong RSA Assumption für $(l + 1)$ -Bit Exponenten bricht, kann dazu benutzt werden, das oben vorgestellte System zu brechen.*

Beweis. Folgender Algorithmus erzeugt ohne Kenntnis des *private keys* eine gültige Signatur:

- Wähle $y' \in_R QR_n$ und berechne $x' := h^{-H(m)}(y')^{e'}$.
- Wende den Algorithmus A auf $xh^{H(x')}$ an. Man erhält man eine $(l + 1)$ -Bit Primzahl e'' und $y \in \mathbb{Z}_n$ mit $y^{e''} = xh^{H(x')}$.

Damit bildet (e'', y, y') eine gültige Signatur. □

Offensichtlich stellt jedoch die Einschränkung auf $(l + 1)$ -Bit Exponenten eine sehr starke Eingrenzung des Problems dar. Das System kann unter Umständen jedoch auch von einem Algorithmus gebrochen werden, der die *Strong RSA Assumption* nicht nur explizit für $(l + 1)$ -Bit Exponenten bricht, sofern er den folgenden Anforderungen genügt:

1. Der Algorithmus berechnet eine zusammengesetzte Zahl e der Bitlänge l' mit $l' \geq l + 1$ und $y \in \mathbb{Z}_n$ mit $y^e = xh^{H(x')}$.
2. Die Zahl e besitzt berechenbare Faktoren $f_i, i \in I$. Also $e = (\prod_{i \in I} f_i) \cdot r$ für ein passendes r . Es sei $A := \prod_{i \in I} f_i$.
3. Es gibt eine Teilmenge $A' \subset A$ mit $f := e / (\prod_{i \in A'} f_i)$, f ist ungerade und hat Bitlänge $l + 1$.

Unter diesen Anforderungen kann man nun wieder analog mit f statt e'' eine gefälschte Signatur erzeugen. Der Unterschied liegt lediglich darin, dass f im Gegensatz zu e'' nicht zwangsweise prim sein muss. Dies wird jedoch bei der Verifizierung der Signatur nicht geprüft, womit man eine gültige Signatur erhält.

4.5 Die Dependent RSA Annahme (*Dependent RSA Assumption*)

Das Dependent RSA-Problem wurde 1999 von David Pointcheval in [Poin99] eingeführt. Mit Hilfe dieser Annahme konstruierte er ein neues Public-Key Verschlüsselungssystem, was in Abschnitt (4.5.2) vorgestellt und näher erläutert wird.

4.5.1 Definitionen

Unter der Computational Dependent RSA Annahme versteht man das folgende Problem:

Definition 4.35 (Computational Dependent RSA Assumption *C-DRSA*).

Zahlentheoretisch: Gegeben:

1. $n \in \mathbb{N}$ mit $n = pq$, p, q prim, $e \in \mathbb{N}$.
2. $a^e \in \mathbb{Z}_n^*$.

Gesucht: $(a + 1)^e \bmod n$.

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(d \equiv c \bmod n \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 e &\in_R \mathbb{Z}_{\varphi(n)}^*; \\
 a &\in_R \mathbb{Z}_n^*; \\
 b &:= a^e \bmod n; \\
 c &:= (a + 1)^e \bmod n; \\
 d &\leftarrow A(l, n, e, b) \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Im Folgenden wird die Annahme mit *C-DRSA* bezeichnet. Analog zum Diffie Hellman Problem gibt es auch zu diesem Problem ein Entscheidungsproblem:

Definition 4.36 (Decisional Dependent RSA Assumption *D-DRSA*).

Zahlentheoretisch: Seien $n = pq$ mit p, q prim und $e \in \mathbb{Z}_{\varphi(n)}^$ gegeben. Entscheide, ob für ein Tupel (b, c) ein $a \in \mathbb{Z}_n^*$ existiert mit $b \equiv a^e \bmod n$ und $c \equiv (a + 1)^e \bmod n$?*

Kryptographisch: Seien

$$\begin{aligned}
 P_{A,1}(l) &:= P(b = 1 \quad &:: \\
 p, q &\in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 a &\in_R \mathbb{Z}_n^*; \\
 e &\in_R \mathbb{Z}_{\varphi(n)}^*; \\
 c &:= (a + 1)^e \bmod n; \\
 b &\leftarrow A(l, n, a, c)
 \end{aligned}$$

und

$$\begin{aligned}
 P_{A,2}(l) &:= P(b = 1) : \\
 & p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 & n := pq; \\
 & a \in_R \mathbb{Z}_n^*; \\
 & e \in_R \mathbb{Z}_{\varphi(n)}^*; \\
 & c \in_R \mathbb{Z}_n^*; \\
 & b \leftarrow A(l, n, a, c)
 \end{aligned}$$

gegeben. Die Definition ist dann \forall prob. poly. Algo. A :

$$|P_{A,1}(l) - P_{A,2}(l)| \leq \frac{1}{\text{poly}(l)}.$$

Diese Annahme wird mit D -DRSA bezeichnet. Im weiteren Verlauf wird, wenn sich auf die Dependent RSA Annahme bezogen wird, immer die Computational Dependent RSA Annahme gemeint sein.

Anders als bei der Starken RSA Annahme ist der Bezug dieses Problems zum ursprünglichen RSA Problem nicht mehr offensichtlich. Es gibt jedoch auch hier intuitive Gründe, warum man diese Annahme als schwer betrachtet.

In Kapitel 2 wurde erwähnt, dass es unterschiedliche Definitionen von Sicherheit gegen aktive Angriffe gibt, die sich als äquivalent erweisen. Gemäß der non-malleability-Definition soll es nicht möglich sein, anhand eines gegebenen Schlüsseltextes c auf einen “dazu in Beziehung stehenden” Schlüsseltext c' zu schließen. So soll es z.B. schwer sein, aus $\text{ver}(pk, m)$ auf $\text{ver}(pk, m + 1)$ zu schließen; ebenso hofft man, dass es schwer sein wird, von m^e auf $(m + 1)^e$ zu schließen unter der Voraussetzung, dass man das Ziehen e -ter Wurzeln als schwer erachtet. Hierbei handelt es sich zugegebenermaßen lediglich um Wunschdenken, da die korrekte Definition von non-malleability nicht auf diesen Fall zutrifft.

Im Gegensatz zur Starken RSA Annahme, für die nicht bekannt ist, ob sie äquivalent zum RSA Problem ist, kann die Äquivalenz der Dependent RSA Annahme zur RSA Annahme mit festem Exponenten bewiesen werden, was in dieser Arbeit erstmals gezeigt wurde. Der Äquivalenzbeweis zu RSA mit allgemeinem Exponenten konnte bis jetzt noch nicht durchgeführt werden, es existiert jedoch ein in der Praxis effizient einsetzbarer Algorithmus, so dass man beide Probleme von einem praxisorientierten Standpunkt aus als äquivalent ansehen kann. Auf weitere Ergebnisse wird in Kapitel 5 näher eingegangen.

4.5.2 Das Verschlüsselungssystem von D. Pointcheval

Im Artikel [Poin99] stellt David Pointcheval neben den Definitionen obiger Annahmen ein Verschlüsselungssystem vor, das der Definition der semantischen Sicherheit von [GoMi84] genügt. Eine vorgestellte Erweiterung des Verfahrens erweist sich im *Random Oracle* Modell sogar als sicher gegen *adaptive chosen ciphertext attack*. Das System wird hier im Sinne der vollständigen Betrachtung der Dependent RSA-Annahme vorgestellt, wobei ich mich hier auf den einfachen Entwurf des Verfahrens beschränke. Die Sicherheit dieses einfacheren Entwurfes baut auf der *Decisional Dependent RSA Assumption* auf. Da im erweiterten Verfahren die Sicherheit gegen *adaptive chosen ciphertext attack* gegeben ist, ist es nicht sinnvoll nach Möglichkeiten zum

Brechen des Systems zu suchen; stattdessen wird gezeigt, wie das erste System durch einen passiven Angriff gebrochen werden kann unter der Annahme, dass ein Algorithmus existiert, der das Decisional Dependent RSA Problem bricht.

Das DRSA Verschlüsselungssystem: Initialisierung: Folgende Werte werden vom Benutzer bestimmt:

- $n = pq$, p, q große Primzahlen.
- e , der öffentliche Exponent, teilerfremd zu $\varphi(n)$.
- Public Key: (n, e) .
- Secret Key: $d \equiv e^{-1} \pmod{\varphi(n)}$.

Verschlüsselung einer Nachricht $m \in \{0, \dots, n-1\}$:

- Wähle $k \in_R \mathbb{Z}_n^*$.
- Berechne $A \equiv k^e \pmod{n}$ und $B = m \cdot (k+1)^e \pmod{n}$.
- Der Schlüsseltext ist dann $C = (A, B)$.

Entschlüsselung einer Nachricht $C = (A, B)$:

- Setze $k \equiv A^d \pmod{n}$.
- $m \equiv B / (k+1)^e \pmod{n}$.

Satz 4.37. *Wenn ein Angreiferalgorithmus X existiert, der die Decisional Dependent RSA Assumption mit einem nicht vernachlässigbaren Vorteil ϵ bricht, so kann man einen Algorithmus Y konstruieren, der X als Blackbox benutzt und das oben vorgestellte System bricht.*

Beweis. Sei ein Verschlüsselungsurakel \mathfrak{D} gegeben. Der Angreifer wählt sich zwei Nachrichten $m_0 \neq m_1$ aus und übergibt sie dem Orakel \mathfrak{D} . Das Orakel wählt $b \in_R \{0, 1\}$ zufällig und verschlüsselt die Nachricht m_b . Der dabei erzeugte Schlüsseltext $(A, B) = (k^e \pmod{n}, m_b \cdot (k+1)^e \pmod{n})$ wird dem Angreifer übergeben. Man definiert $T_0 := (A, B/m_0)$ und $T_1 := (A, B/m_1)$. Unser Blackboxalgorithmus X kann diese beiden Möglichkeiten effektiv mit Wahrscheinlichkeit ϵ unterscheiden, da die beiden Tupel T_0 und T_1 unterschiedlichen Distributionen der *Decisional Dependent RSA Assumption* angehören. Somit besitzt Y mindestens eine Erfolgswahrscheinlichkeit ϵ , wobei ϵ als nicht vernachlässigbar vorausgesetzt wurde, was den Beweis vervollständigt. □

Satz 4.38. *Hat man sogar einen Algorithmus Y , der die Computational Dependent RSA Assumption mit einer nicht vernachlässigbaren Wahrscheinlichkeit ϵ bricht, so kann aus einem beliebigen Schlüsseltext (A, B) die verwendete Nachricht noch darüber hinaus mit einer nicht vernachlässigbaren Wahrscheinlichkeit wieder rekonstruiert werden.*

Beweis. Sei ein Angreifer gegeben, der Zugriff auf den Algorithmus Y hat. Der Angreifer bekommt einen Schlüsseltext $(A, B) = (k^e \bmod n, m \cdot (k+1)^e \bmod n)$. Er wendet nun Algorithmus Y auf A an und erhält $(k+1)^e \bmod n$ mit einer Wahrscheinlichkeit von ϵ . Der Angreifer berechnet sich nun das Inverse c von $(k+1)^e \bmod n$. Der Wert von c entspricht dann $((k+1)^e)^{-1}$. Durch

$$B \cdot c \equiv m \cdot (k+1)^e \cdot c \equiv m \cdot (k+1)^e \cdot (k+1)^{-e} \equiv m \pmod{n}$$

erhält der Angreiferalgorithmus somit m . Die Berechnung ist korrekt für ein richtig berechnetes Element $(k+1)^e \bmod n$, somit wird die Nachricht m mit der nicht vernachlässigbaren Wahrscheinlichkeit ϵ richtig berechnet, was die Behauptung liefert. \square

4.6 Die ϕ -Hiding Assumption

Die ϕ -Hiding Assumption wurde von C. Cachin, S. Micali und M. Stadler im Jahre 1998 in [CaMS98] eingeführt. Aufbauend auf dieser Annahme wurde in diesem Paper weiterhin ein *Private Information Retrieval* System eingeführt, das in (4.6.2) vorgestellt wird.

4.6.1 Definition und Notation

Bevor ich zur Einführung der eigentlichen Annahme komme, wird erst die in besagtem Paper benutzte Notation vorgestellt, um die Definitionen und ein eventuelles Nachlesen der Beweise zu vereinfachen:

- H_a bezeichne die Menge aller natürlichen Zahlen, die das Produkt zweier a -bit Primzahlen sind.
- Man sagt: Eine zusammengesetzte Zahl m " ϕ -versteckt" eine Primzahl p wenn gilt: $p|\phi(m)$.
- $H^b(m)$ sei die Menge der b -bit Primzahlen, die durch m ϕ -versteckt werden.
- $\overline{H^b(m)}$ sei die Komplementmenge von $H^b(m)$ in den b -bit Primzahlen, d.h. alle b -bit Primzahlen, die nicht zu $H^b(m)$ gehören.
- H_b^a sei die Menge aller $m \in H_a$, die eine b -bit Primzahl ϕ -verstecken.

Nun kann man sich der eigentlichen Annahme zuwenden. Unter der ϕ -Hiding Assumption versteht man das folgende Problem:

Definition 4.39 (ϕ -Hiding Assumption ϕHA).

Zahlentheoretisch: Gegeben:

1. $n \in H_{kf}^k$.
2. eine k -bit Primzahl p .

Entscheide, ob gilt: m ϕ -versteckt p ?

Kryptographisch: $\exists f > 0 \forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(b = b') &:: m \in_R H_{kf}^k; \\
 & p_0 \in_R H_k(m); \\
 & p_1 \in_R \overline{H}^k(m); \\
 & b \in_R \{0, 1\}; \\
 & b' \leftarrow A(m, p_b) \\
 & \leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Die Annahme wird mit ϕHA abgekürzt. Im Paper wird eine weitere Annahme vorgestellt, die jedoch nichtzahlentheoretischer Art ist und demzufolge nur der Vollständigkeit halber aufgeführt wird:

Definition 4.40 (ϕ -Sampling Assumption). *Es existiert l_0 , so dass für alle $k > l_0$ gilt: Es existiert ein Sample-Algorithmus S , der auf Eingabe einer beliebigen k -bit Primzahlen p eine zufällige k^f -bit Zahl $m \in H_{kf}^k$ berechnet, die p ϕ -versteckt und sie zusammen mit ihrer Faktorisierung ausgibt.*

Die ϕ -Sampling Assumption wird mit ϕSA abgekürzt. Beide Annahmen zusammen werden als ϕ -Assumption, oder kurz ϕA , bezeichnet. Aufbauend auf diesem Zusammenschluss der beiden obigen Annahmen konnte ein *Private Information Retrieval System* konstruiert werden, das im folgenden vorgestellt wird.

4.6.2 Das Private Information Retrieval System von C. Cachin, S. Micali, M. Stadler

Im Artikel [CaMS98] wird ein *Private Information Retrieval System* beschrieben, dass auf der im letzten Abschnitt eingeführten ϕ -Annahme aufbaut. Zu einem solchen System gehören ein Datenbankalgorithmus D , ein Benutzeralgorithmus Q (der sogenannte Anfragengenerator, engl. *query generator*) sowie ein Algorithmus R , der die erhaltenen Antworten verarbeitet (engl. *query restrictor*). Zur Definition der besagten Algorithmen werden zwei weitere Algorithmen verwendet: ein Primzahlzahltest T sowie ein Generator P für Primzahlsequenzen. Als Primzahltest kann ein beliebiger deterministischer oder auch probabilistischer Test benutzt werden, demzufolge wird auf die Darstellung eines Algorithmus verzichtet. Die weiteren Algorithmen werden im folgenden im Detail vorgestellt:

Allgemeine Eingabe des Systems:

- $n \in \mathbb{N}$.
- eine n -bit Sequenz B .
- $i \in \{1, \dots, n\}$.
- ein in unärer Darstellung gegebener Sicherheitparameter k mit $k > (\log n)^2$.

Generator P für Primzahlsequenzen:

Eingabe:

- $a \in \{1, \dots, n\}$, eine Sequenz von k^3 k -bit Strings $Y = (y_0, \dots, y_{k^3-1})$, der Sicherheitsparameter k .

Berechnung:

1. $j := 0$.
2. $\sigma_{aj} := \bar{a} \circ \bar{j}$, wobei \bar{a} die $(\log n)$ -bit Darstellung von a und \bar{j} die $(k - \log n)$ -bit Darstellung von j ist. Die Verknüpfung \circ steht für die Konkatenation der beiden Strings.
3. $z_j := \sum_{l=0}^{k^3-1} y_l \sigma_{aj}^l$, wobei alle Strings y_l und σ_{aj} als Elemente von $GF(2^k)$ interpretiert und somit auch alle Berechnungen in diesem Körper ausgeführt werden.
4. Gilt nun $T(z_j) = 1$ oder $j = 2^{k-\log n}$ gib $p_a := z_j$ und terminiere. Ansonsten setze $j := j+1$ und fahre mit Schritt 2 fort.

Ausgabe:

- Eine k -bit Zahl p_a , die mit sehr großer Wahrscheinlichkeit eine Primzahl ist. Da P für feste Y und k ein deterministischer Algorithmus ist, wird eine Sequenz von (möglichen) Primzahlen p_1, \dots, p_n mit $a = 1, \dots, n$ erzeugt.

Der Anfragengenerator Q :

Eingabe:

- die Zahl $n, i \in \{1, \dots, n\}$, der Sicherheitsparameter k .

Berechnung:

1. Wähle $y_0, \dots, y_{k^3-1} \in_R \{0, 1\}^k$. Setze $Y := (y_0, \dots, y_{k^3-1})$.
2. $p_i := P(i, Y, 1^k)$.
3. Wähle $m \in H_{k^f}^k$, d.h. eine Zahl m , die die Zahl p_i ϕ -versteckt. Dabei sei s ihre Faktorisierung und damit das im folgenden benutzte Geheimnis des Users.
4. Wähle $x \in \mathbb{Z}_m^*$.
5. Gib als Anfrage $q = (m, x, Y)$ aus, sowie dem User das geheime s .

Ausgabe:

- Eine Anfrage $q = (m, x, Y)$ und eine geheime Zahl s , die die Faktorisierung von m darstellt. Hierbei ist m eine k^f -bit zusammengesetzte Zahl, $x \in \mathbb{Z}_m^*$ sowie Y eine Sequenz mit k^3 Elementen, die jeweils aus k -bit Strings bestehen.

Datenbankalgorithmus D :

Eingabe:

- die n -bit Sequenz B , eine von $Q(n, i, 1^k)$ erzeugte Anfrage $q = (m, x, Y)$, der Sicherheitsparameter k .

Berechnung:

1. $x_0 := x$.
2. Für $j = 1, \dots, n$ berechne: {
3. $p_j := P(j, Y, 1^k)$.
4. $e_j := p_j^{B_j}$.
5. $x_j := x_{j-1}^{e_j} \bmod m$. }
6. Gib als Antwort $r = x_n$ aus.

Ausgabe:

- $r \in \mathbb{Z}_m^*$.

Response Retriever R :

Eingabe:

- n, i , eine Ausgabe $((m, x, Y), s)$ des Algorithmus $Q(n, i, 1^k)$ sowie eine Ausgabe $r \in \mathbb{Z}_m^*$ des Algorithmus $D(B, (m, x, Y), 1^k)$ sowie den Sicherheitsparameter k .

Berechnung:

1. Besitzt die Zahl r mindestens eine p_i -te Wurzel modulo m , gib 1 aus, ansonsten 0.

Ausgabe:

- ein Bit b , für das gelten soll $b = B_i$.

Die Korrektheit des Systems wurde in [CaMS98] gezeigt. Der Beweis der Vertraulichkeit des Systems baut auf der ϕ -Assumption auf und kann ebenfalls in besagtem Paper nachgelesen werden.

4.7 Das Nullstellenproblem (*Root Finding Problem*)

Ein neues Problem, das auf der Faktorisierungsannahme aufbaut und sich die Eigenschaften von Polynomen vom Grad k in $\mathbb{Z}_n[X]$ zunutze macht, ist das *Root Finding Problem* $RFP(k)$ [ScEi96]. Es basiert auf der Annahme, dass das Berechnen einer bzw. aller Nullstellen eines Polynomes im Polynomring $\mathbb{Z}_n[X]$ schwer sein soll.

4.7.1 Definitionen

Definition 4.41 (Root Finding Problem $RFP(k)$).

Zahlentheoretisch: Sei $n = pq$ mit p, q prim gegeben. Sei weiterhin $f(x) \in \mathbb{Z}_n[X]$ mit $\deg(f) = k$ mit $k \geq 1$. f zerfalle vollständig in Linearfaktoren. Berechne eine Nullstelle von f .

Kryptographisch: $\forall \text{prob. poly. Algo. } A :$

$$\begin{aligned}
 P(f(x) \equiv 0 \pmod n \quad &:: \quad p, q \in_R \text{ Menge der } l\text{-bit Primzahlen;} \\
 n &:= pq; \\
 f \in_R \mathbb{Z}_n[X] \text{ mit } \deg(f) = k, \text{ so dass } f &\text{ vollständig zerfällt;} \\
 x \leftarrow A(l, n, f) & \\
 &\leq \frac{1}{\text{poly}(l)}.
 \end{aligned}$$

Das Problem wird mit $RFP(k)$ bezeichnet. Im Speziellen bezeichnet $RFP(\geq k)$ das Nullstellenproblem für alle Polynome f vom Grad $\deg(f) \geq k$. Mit RFP wird schliesslich $RFP(\geq 1)$ bezeichnet.

Offensichtlich ist die Forderung, eine Nullstelle von f berechnen zu lassen, im komplexitätstheoretischen Sinn äquivalent zu dem Problem *alle* Nullstellen von f berechnen zu lassen. Kann man eine Nullstelle α von $f(x) = a_d X^d + a_{d-1} X^{d-1} + \dots + a_1 X + a_0$ berechnen, so ist $(x - \alpha)$ ein Teiler von f und man erhält

$$f(x) = (x - \alpha) \cdot g(x) \text{ mit } g(x) \in O(X^{d-1})$$

Nun sucht man wiederum eine Nullstelle von g . Da der Grad des zu untersuchenden Polynoms in jedem Schritt um eins dekrementiert wird, erhält man nach genau m Schritten alle Nullstellen des Polynoms f .

In der Kryptographie hingegen fällt die Erfolgswahrscheinlichkeit exponentiell im Grad des betrachteten Polynoms. Kann man zu einem gegebenen Polynom vom Grad d jeweils eine Nullstelle mit Wahrscheinlichkeit $0 < \epsilon < 1$ berechnen, so erhält man die vollständige Faktorisierung des Polynoms mit Wahrscheinlichkeit ϵ^{d-1} , also exponentiell fallend in d .

Aufbauend auf dem *Root Finding Problem* wurden von Schwenk und Eisfeld ein Verschlüsselungs- und ein Signatursystem entworfen, die im Folgenden vorgestellt werden.

4.7.2 Das Verschlüsselungssystem von J. Schwenk, J. Eisfeld

Die Verschlüsselung einer Nachricht m läuft in diesem System prinzipiell analog ab zu dem bereits 1980 vorgestellten Rabin-System [Rabi80]:

Initialisierung:

1. Wähle $n = pq$ mit p, q große Primzahlen
2. Der public key ist n , der private key ist (p, q) .

Verschlüsselung:

1. Spalte die Nachricht m in k Blöcke $\overline{m_1} || \dots || \overline{m_k}$ auf.

2. Versehe die einzelnen Blöcke mit Redundanz mit dem Ziel, sie erkennen und ordnen zu können. Man erhält als Ergebnis m_1, \dots, m_k .
3. Interpretiere m_1, \dots, m_k als Zahlen kleiner oder gleich n .
4. Berechne das Polynom $f(x) := (x - m_1) \cdots (x - m_k) \bmod n = x^k + a_{k-1}x^{k-1} + \cdots + a_1x + a_0$.
5. Der Schlüsseltext lautet dann $c := (a_{k-1}, \dots, a_1, a_0)$.

Entschlüsselung:

1. Rekonstruiere das Polynom $f(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_1x + a_0$ anhand des Schlüsseltextes $c = (a_{k-1}, \dots, a_1, a_0)$.
2. Berechne die Wurzeln v_1, \dots, v_k von $f_p(x) := f(x) \bmod p$ über $GF(p)$ sowie die Wurzeln w_1, \dots, w_k von $f_q(x) := f(x) \bmod q$ über $GF(q)$. Ein effizienter Algorithmus ist in [BeOr81] zu finden.
3. Berechne mittels des erweiterten euklidischen Algorithmus zwei Zahlen λ und μ mit $\lambda p + \mu q = 1$. Anhand dieser Gleichung lassen sich nun mittels des chinesischen Restsatzes die k^2 Nullstellen $z_{ij} = w_i \lambda p + v_j \mu q \bmod n$ von $f(x)$ in \mathbb{Z}_n bestimmen.
4. Durch die bei der Verschlüsselung zugefügten Redundanz lassen sich mit hoher Wahrscheinlichkeit genau k Nullstellen herausfinden, die der geforderten Redundanz entsprechen. Durch eine entsprechende Anordnung der gefundenen k Nullstellen und durch die Entfernung der Redundanz erhält man die ursprüngliche Nachricht.

4.7.3 Das Signatursystem von J. Schwenk, J. Einfeld

Eine nicht ganz offensichtliche Anwendung des *Root Finding Problems* stellt die Konstruktion eines Signatursystems dar. Um dies zu erreichen, werden die Blöcke einer Nachricht als Koeffizienten eines Polynoms aufgefasst. Die Signatur einer solchen Nachricht besteht aus einer oder auch mehreren Wurzeln des betrachteten Polynoms.

Initialisierung:

1. Wähle $n = pq$ mit p, q große Primzahlen
2. Der public key ist n , der private key ist (p, q) .

Signaturerzeugung:

1. Splitte die Nachricht m in $k - 1$ Blöcke: a_{k-1}, \dots, a_1 .
2. Interpretiere diese Blöcke als Zahlen kleiner oder gleich n .
3. Wähle $\bar{a}_0 \in_R \{0, \dots, n - 1\}$. Versehe \bar{a}_0 mit Redundanz und benutze das Ergebnis a_0 als konstanten Koeffizient des Polynoms.
4. Berechne eine oder mehrere Wurzeln r_1, \dots, r_t mit $1 \leq t \leq k$ des Polynoms $f(x) := x^k + a_{k-1}x^{k-1} + \cdots + a_1x + a_0 \bmod n$, indem die Wurzeln modulo p bzw. modulo q einzeln berechnet werden. Sollte $f(x)$ keine bzw. nicht genügend Wurzeln besitzen, so wiederhole Schritt 3.

5. Die Signatur besteht aus $s := \text{Sig}(m) := (a_0, r_1, \dots, r_t)$. Übermittele also (m, s) .

Verifikation:

1. Rekonstruiere das Polynom $f(x) := x^k + a_{k-1}x^{k-1} + \dots + a_1x + a_0$ aus der Nachricht m sowie $s = (a_0, r_1, \dots, r_t)$.
2. Für $i = 1, \dots, t$ berechne $f(r_i)$ und überprüfe, ob gilt: $f(r_i) \equiv 0 \pmod{n}$. In diesem Fall akzeptiere die Signatur, andernfalls weise sie zurück.

Kapitel 5

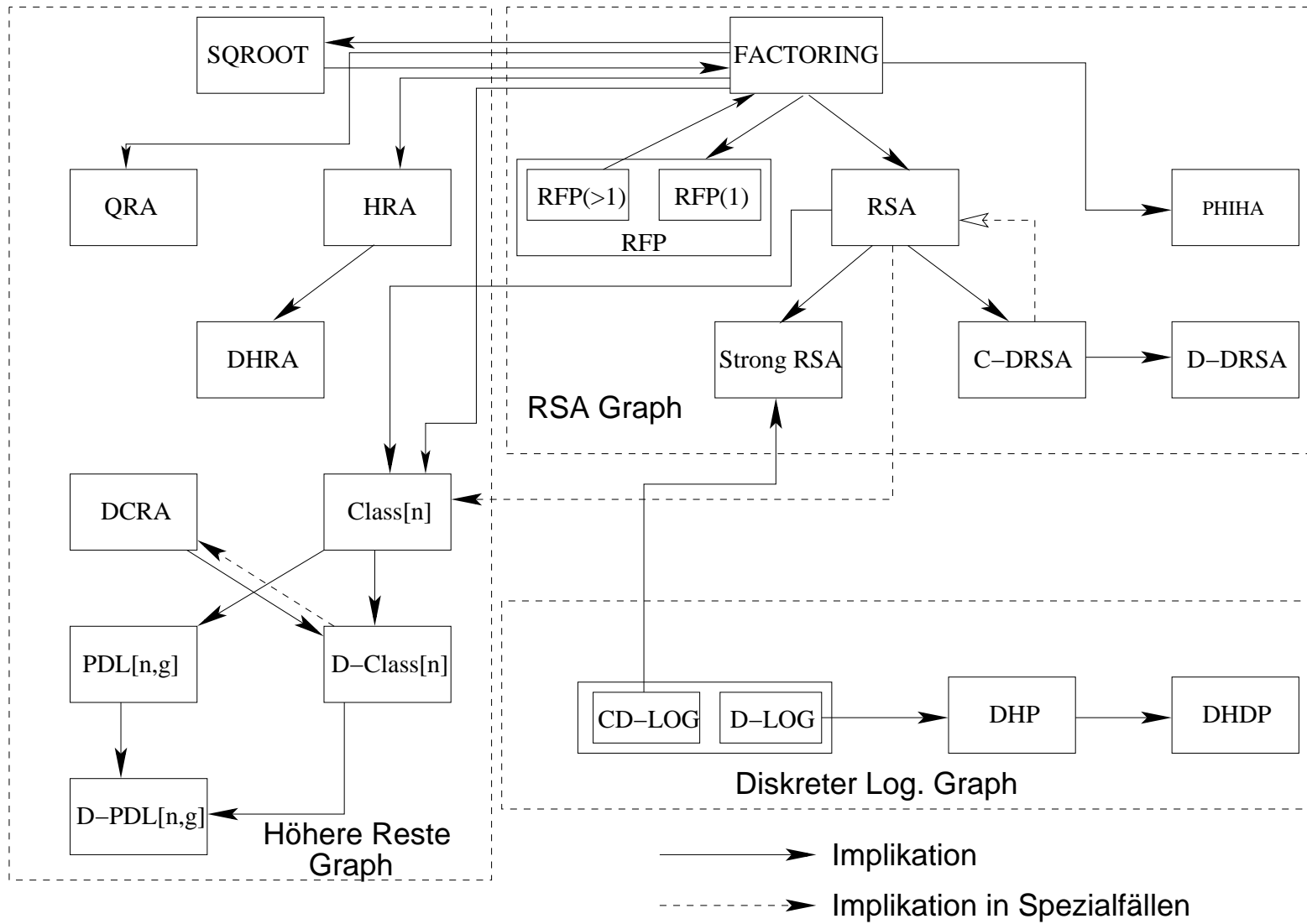
Die verschiedenen Annahmen im Vergleich

5.1 Einleitung

Dieses Kapitel ist der Komplexität der einzelnen Annahmen gewidmet. Die Annahmen werden miteinander verglichen und, sofern möglich, durch eine polynomielle Reduktion aufeinander zurückgeführt mit dem Ziel, ein Problem als mindestens so schwer wie ein anderes zu klassifizieren. Eine auf der Hand liegende Anwendungsmöglichkeit einer solchen Reduktion liegt im Vergleich zweier Kryptosysteme, deren Sicherheitsbeweise auf den zwei betrachteten Annahmen beruhen. Ist das erste Problem gemäß der hierarchischen Struktur leichter als das zweite, so kann man diesen Sachverhalt auf die Sicherheitsbetrachtung der Systeme übertragen, d.h. das erste System ist leichter zu brechen als das zweite. Das Ziel dieses Kapitels besteht nun darin, eine hierarchische Struktur zwischen den einzelnen Annahmen zu entwickeln, was solche Betrachtungen für möglichst viele Kryptosysteme möglich machen soll. Da in diesem Kapitel viele dieser Beweise durchgeführt werden, wird zuerst anhand einer Grafik ein allgemeiner Überblick bereitgestellt, der dem Leser als Leitfaden beim Lesen dieses Kapitels helfen soll. In der Grafik werden außerdem die Probleme in mehrere Klassen eingeteilt, so dass zuerst die Beweise und Reduktionen innerhalb dieser Klassen durchgeführt werden und ich erst am Schluss zu den klassenübergreifenden Reduktionen kommen werde. Diese Reihenfolge soll es dem Leser erleichtern, die Beweise und vor allem die internen Zusammenhänge der Annahmen besser zu verstehen, indem zuerst ein Gebiet, also eine Klasse, vollständig abgehandelt wird.

5.2 Übersicht

Wie in der Einleitung angekündigt, folgt nun eine Übersicht über die durchgeführten Reduktionen; sie spiegelt den Ablauf des folgenden Kapitels wieder. Die durchgezogenen Pfeile entsprechen den gezeigten Implikationen. Eine Implikation der Art *Problem 1* \rightarrow *Problem 2* bedeutet, dass aus der effizienten Lösbarkeit von *Problem 1* die effiziente Lösbarkeit von *Problem 2* folgt. Die gestrichelten Linien entsprechen Implikationen, die nur in bestimmten Spezialfällen gültig sind. Die genaue Beschaffenheit dieser Implikationen sowie die dafür notwendigen Voraussetzungen werden in den zugehörigen Beweisen aufgeführt.



5.3 Der RSA Graph

In diesem Abschnitt werden die auf dem RSA Problem basierenden Annahmen sowie die Beziehung zu dem zum RSA Problem übergeordneten Faktorisierungsproblem näher betrachtet.

5.3.1 Integer Factorisation Problem \Rightarrow RSA Problem

Satz 5.1. *Sollte ein polynomieller Algorithmus A existieren, der die Faktorisierungsannahme bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das RSA Problem in polynomieller Zeit bricht.*

Beweis. Seien n, e, c wie im RSA-Problem beschrieben. Durch Ausführung des Algorithmus A kann man die Faktorisierung von $n = pq$ als bekannt voraussetzen, was eine effiziente Berechnung von $\varphi(n) = (p-1)(q-1)$ zulässt. Nun kann man mit der normalen RSA-Entschlüsselung fortfahren, da sich der geheime RSA Schlüssel d bei bekanntem Wert von $\varphi(n)$ leicht berechnen lässt. □

Ob auch die umgekehrte Richtung des Beweises gilt, ist nicht bekannt.

5.3.2 Integer Factorisation Problem \Rightarrow Root Finding Problem

Satz 5.2. *Sollte ein polynomieller Algorithmus A existieren, der die Faktorisierungsannahme bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das Root Finding Problem in polynomieller Zeit bricht.*

Beweis. Sei ein Algorithmus A gegeben, der die Faktorisierungsannahme für $n = pq$ bricht, d.h. man erhält die beiden Primfaktoren p und q . Sei nun ein Polynom $f(x) \in \mathbb{Z}_n[X]$ gegeben. Nun kann man zuerst die Nullstellen in den Körpern $GF(p)$ und $GF(q)$ suchen. Dies ist in polynomieller Zeit möglich [BeOr81]. Mittels der Isomorphie $GF(p) \times GF(q) \simeq \mathbb{Z}_n$ lassen sich die Nullstellen in den beiden Körpern zu den gesuchten Nullstellen in \mathbb{Z}_n zusammensetzen. □

5.3.3 RSA Problem \Rightarrow Strong RSA Problem

Angenommen man hätte einen Algorithmus A , der das RSA Problem mit Vorteil ϵ bricht. Wählt man nun ein $e \in_R \mathbb{Z}_{\varphi(n)}^*$, so erhält man durch Anwendung des Algorithmus A eine e -te Wurzel aus einer Zahl m mit Wahrscheinlichkeit ϵ . Man gibt nun das Tupel (e, m) als Lösung des *Strong RSA Problems* aus. Offensichtlich hat man durch diesen Algorithmus einen Vorteil von ϵ , was den Beweis vervollständigt.

5.3.4 RSA Problem \Rightarrow Computational Dependent RSA Problem

Satz 5.3. *Sei X ein polynomieller Algorithmus, der das RSA Problem bricht, dann kann man mittels X einen Algorithmus Y konstruieren, der die Computational Dependent RSA Assumption in polynomieller Zeit bricht.*

Beweis. Angenommen wir haben einen effizienten Algorithmus X , der uns das RSA Problem löst, d.h. bei gegebenen $n = pq$ mit p, q prim, $y \in \mathbb{Z}_n^*$ und $e < n$ mit $\text{ggT}(e, (p-1)(q-1)) = 1$ berechnet der Algorithmus mit einer nicht vernachlässigbaren Wahrscheinlichkeit ϵ ein Element $x \in \mathbb{Z}_n$, für das gilt: $x^e \equiv y \pmod{n}$. Unser Angreiferalgorithmus Y auf die C-DRSA läuft nun folgendermaßen:

1. Auf das gegebene Element $a^e \in \mathbb{Z}_n^*$ wendet Y den Algorithmus X an und erhält als Ergebnis mit Wahrscheinlichkeit ϵ den Wert a .
2. Der Algorithmus berechnet aus dem nun gegebenen Element a das Element $(a+1)$.
3. Das Element $(a+1)$ wird z.B. durch Square-and-multiply Algorithmen mit e potenziert und man erhält das gesuchte $(a+1)^e$.

Der so konstruierte Algorithmus Y besitzt offensichtlich eine Erfolgswahrscheinlichkeit von ϵ , was die Behauptung liefert. □

5.3.5 Computational Dependent RSA Problem \Rightarrow RSA Problem ?

Vorbemerkung: Die oben behauptete Implikation konnte für das allgemeine RSA Problem bis jetzt nicht vollständig bewiesen werden. Es wurde jedoch versucht, Methoden zu entwickeln, die in der Praxis in den meisten Fällen erfolgreich sind und diese dann im mathematischen Sinn zu untersuchen. Es reicht demzufolge nur zu folgendem

Satz 5.4. *Sollte ein polynomieller Algorithmus X existieren, der die Computational Dependent RSA Assumption bricht, so kann man für jedes feste e einen Algorithmus Y konstruieren, der das RSA Problem für feste Exponenten e in polynomieller Zeit bricht.*

Man beginnt mit einer kurzen Vorüberlegung, die als Motivation für den späteren Beweis dienen soll. Es muss gezeigt werden, dass die Berechnung der e -ten Wurzel von m^e effizient möglich ist unter der Voraussetzung, dass neben m^e auch die Werte $(m+1)^e, (m+2)^e, \dots, (m+j)^e$ bekannt ist, wobei j nur so groß gewählt werden darf, dass eine Berechnung der j verschiedenen Werte in akzeptabler Zeit möglich und noch mit einer nicht vernachlässigbaren Wahrscheinlichkeit korrekt ist. Für die nun folgenden theoretischen Überlegungen beschränkt man sich vorerst auf *einen* berechneten Wert, der Angreifer besitzt im Moment als nur die Werte e, m^e und $(m+1)^e$ mit dem Ziel, m zu berechnen. Im folgenden sei $\alpha := m^e \pmod{n}$ und $\beta := (m+1)^e \pmod{n}$. Eine Möglichkeit, dies zu erreichen, ist wie folgt gegeben:

Behauptung: Für alle $e \in \mathbb{N}$ existieren Polynome Q_1, Q_2 in α, β mit

$$m = \frac{Q_1(\alpha, \beta)}{Q_2(\alpha, \beta)}$$

Beispiel:

Sei $e = 3$. Man erhält somit

$$\begin{aligned} \alpha &= m^3 \pmod{n}; \\ \beta &= (m+1)^3 \pmod{n} = m^3 + 3m^2 + 3m + 1 \pmod{n} = \alpha + 3m^2 + 3m + 1 \pmod{n}. \end{aligned}$$

Dann gilt

$$\begin{aligned} m(\beta - \alpha) - 3\alpha &= 3m^2 + m \pmod{n}; \\ \beta - \alpha &= (3m^2 + m) + 2m + 1 \pmod{n} \\ &= m(\beta - \alpha + 2) - 3\alpha + 1 \pmod{n}. \end{aligned}$$

Es folgt:

$$m = \frac{2\alpha + \beta - 1}{\beta - \alpha + 2} \pmod{n}.$$

Es ist jedoch kein Algorithmus bekannt, der die Koeffizienten der beiden Polynome für größere e effizient berechnen kann. Diese Möglichkeit ist somit in der Praxis uninteressant und wird nicht weiter verfolgt. Das dabei verwendete Prinzip der Polynome diene allerdings als Motivation für den nun folgenden, eigentlichen Beweis.

Beweis. Wende zu Beginn den Algorithmus X auf das Element α an. Man erhält das Element β mit einer nicht vernachlässigbaren Wahrscheinlichkeit. Definiere Polynome Q_1, Q_2 durch $Q_1(x) = x^e - \alpha$ und $Q_2 = (x + 1)^e - \beta$.

Betrachte nun die beiden Gleichungen:

$$x^e - \alpha \equiv 0 \pmod{n}$$

und

$$(x + 1)^e - \beta \equiv 0 \pmod{n}.$$

Da m beide Gleichungen trivialerweise erfüllt, ist $(x - m)$ ein gemeinsamer Teiler beider Polynome. Man hätte gerne, dass $(x - m)$ immer das Ergebnis des euklidischen Algorithmus der beiden Polynome Q_1 und Q_2 ist, da dies für kleine Exponenten effizient berechnet werden kann. Diese "Hoffnung" wird nun im folgenden untersucht.

Zuerst muss gesagt werden, dass der Begriff des größten gemeinsamen Teilers im obigen Satz mit Absicht vermieden wurde. Da \mathbb{Z}_n einen Ring mit Nullteilern bildet, stellt somit der Polynomring $\mathbb{Z}_n[X]$ keinen faktoriellen Ring dar, was den Begriff des ggT's in diesem Ring sinnlos macht.

Was man jedoch tun kann, ist den euklidischen Algorithmus wie folgt zu modifizieren, so dass er auch im Falle eines auftretenden Nullteilers zu einem Abschluss kommt:

Betrachtet man die Division mit Rest der beiden Polynome $P(X) = \sum_{i=0}^k a_i X^i$ und $Q(X) = \sum_{i=0}^l b_i X^i$ so muss die Zahl b_l invertiert werden. Diese Invertierung wird durch den erweiterten euklidischen Algorithmus in \mathbb{Z} durchgeführt. Hierbei erkennt man auch, ob eine Invertierung im Ring \mathbb{Z}_n überhaupt möglich ist. Sofern ein Inverses existiert, führe die Polynomdivision durch und fahre mit dem nächsten Schritt fort.

Nun gibt es zwei Möglichkeiten:

1. Es tritt im i -ten Schritt ein Leitkoeffizient $b_l^{(i)}$ auf, der nicht invertierbar ist.
2. Jeder betrachtete Leitkoeffizient $b_l^{(i)}$ ist invertierbar. In diesem Fall erhält man einen gemeinsamen Teiler der beiden ursprünglichen Polynome.

Im ersten Fall muss somit gelten: $ggT(b_l^{(i)}, n) \neq 1$. Da aber $n = pq$ gilt, muss somit gelten $b_l^{(i)} = kp$ für $1 \leq k \leq q-1$ oder $b_l^{(i)} = kq$ für $1 \leq k \leq p-1$. Folglich muss gelten $ggT(b_l^{(i)}, n) = p$ oder $ggT(b_l^{(i)}, n) = q$. Damit ist die Zahl n faktorisiert, was eine Berechnung von $\varphi(n)$ und somit eine übliche RSA-Entschlüsselung möglich macht, die als Ergebnis das gesuchte m liefert.

Im zweiten Fall erhält man ein Polynom $Q(X)$, das einen gemeinsamen Teiler der beiden ursprünglichen Polynome bildet. Sollte dieses Polynom linear sein, also $Q(X) = aX + b$ so muss nach obiger Überlegung $m = -\frac{b}{a}$ gelten, und man ist fertig.

Es bleibt die Frage, ob das aus dem Algorithmus resultierende Polynom linear sein muss. Um dies näher zu untersuchen, betrachtet man die Gleichungen $Q_1 = 0$ und $Q_2 = 0$ im Körper $K := \mathbb{Z}_p$ mit dem Ziel, die Körperaxiome sowie Satz (2.15) anwenden und das Ergebnis mittels des chinesischen Restsatzes wieder zusammensetzen zu können. Man betrachtet somit:

$$x^e - \alpha \equiv 0 \pmod{p};$$

$$(x+1)^e - \beta \equiv 0 \pmod{p}.$$

In einem Körper haben beide Polynome wegen Satz (2.15) höchstens e Nullstellen, wobei m eine gemeinsame ist. Nun betrachtet man den gemeinsamen Zerfällungskörper der beiden Polynome: $K[\alpha_1] \dots [\alpha_{e-1}][\beta_1] \dots [\beta_{e-1}]$, wobei die α_i Nullstellen des Polynoms Q_1 und die β_i Nullstellen des Polynoms Q_2 sind. Desweiteren adjungiert man noch eine e -te primitive Einheitswurzel ζ : $K[\alpha_1] \dots [\alpha_{e-1}][\beta_1] \dots [\beta_{e-1}][\zeta]$.

In diesem Körper zerfallen nun beide Polynome vollständig in Linearfaktoren. Die betrachteten Polynome haben beide eine bestimmte Form, sie stellen sogenannte Kreisteilungspolynome dar. Da man zu jeder Gleichung eine Lösung hat (nämlich m), kann man mittels der e -ten primitiven Einheitswurzel ζ alle Lösungen der beiden Gleichungen angeben:

Lemma 5.5. *Die erste Gleichung hat die Lösungen $m\zeta^i$ für $0 \leq i \leq e-1$. Die zweite Gleichung hat die Lösungen $(m+1)\zeta^i - 1$ für $0 \leq i \leq e-1$. Jede Lösung der Gleichungen hat diese Gestalt.*

Beweis. Zur ersten Gleichung:

Es gilt:

$$(m\zeta^i)^e - \alpha = m^e \zeta^{ie} - \alpha = m^e (\zeta^e)^i - \alpha = m^e 1^i - \alpha = m^e - \alpha = 0$$

d.h. alle $m\zeta^i$ sind Lösungen der ersten Gleichung.

Desweiteren gilt: $m\zeta^i \neq m\zeta^j$ für $i \neq j$, da sonst gelten müsste: $\zeta^{i-j} = 1$ für oBdA $i > j \Rightarrow \zeta$ wäre keine primitive e -te Einheitswurzel, da gilt: $i-j < n$.

Demzufolge hat man e paarweise verschiedene Lösungen der ersten Gleichung. Da alle Rechnungen in einem Körper stattfinden, kann obige Gleichung gemäß dem Fundamentalsatz der Algebra höchstens e Lösungen haben, was die Behauptung liefert.

Zur zweiten Gleichung:

Es gilt:

$$((m+1)\zeta^i - 1) + 1)^e - \beta = (m+1)^e \zeta^{ie} - \beta = (m+1)^e - \beta = 0$$

Es gilt weiter: $(m+1)\zeta^i - 1 \neq (m+1)\zeta^j - 1$, sonst würde gelten $m = p-1$ oder $\zeta^{i-j} = 1$. Also wäre ζ wieder keine Primitivwurzel (außer im vernachlässigbaren Fall $m = p-1$).

Somit erhält man wieder e paarweise verschiedene Lösungen. Da es auch hier wieder maximal e Lösungen geben kann, ist das Lemma bewiesen.

□

Es stellt sich nun die Frage, ob es gemeinsame, von m verschiedene, Nullstellen gibt. Dazu müsste gelten: $m\zeta^i = (m+1)\zeta^j - 1 \Leftrightarrow m(\zeta^i - \zeta^j) = \zeta^j - 1 \Leftrightarrow m = \frac{\zeta^j - 1}{\zeta^i - \zeta^j}$ für $i \neq j$. Für $i = j$ erhält man direkt $\zeta^j = 1 \Rightarrow j = 0$, d.h. die Nullstelle m . Aber wann gilt nun $m = \frac{\zeta^j - 1}{\zeta^i - \zeta^j} \in K$?

Nach obiger Überlegung durchlaufen die Variablen i und j genau e verschiedene Werte. Folglich erhält man für $i \neq j$ höchstens $e \cdot (e-1)$ verschiedene Nachrichten m , auf die obige Bedingung zutreffen könnte. Für die Wahrscheinlichkeit P_1 , einen linearen ggT zu erhalten, gilt somit in Abhängigkeit von l und p :

$$P_1(l, p) \geq \frac{p - (e \cdot (e-1))}{p} \geq 1 - \frac{e^2}{p}.$$

Da man eine Wahrscheinlichkeit P_1 benötigt, die nur noch von l abhängt, wird noch über alle in Frage kommenden Primzahlen gemittelt. Sei in der folgenden Formel \mathbb{P}_l die Menge aller l -bit Primzahlen, p_1 sei o.b.d.A. die kleinste l -bit Primzahl. Dann gilt für die Wahrscheinlichkeit P_1

$$P_1(l) = \sum_{p_i \in \mathbb{P}_l} \left(1 - \frac{e^2}{p_i}\right) \cdot \frac{1}{|\mathbb{P}_l|} \geq |\mathbb{P}_l| \cdot \left(1 - \frac{e^2}{p_1}\right) \cdot \frac{1}{|\mathbb{P}_l|} = 1 - \frac{e^2}{p_1}.$$

Betrachtet man nun einen festen Exponenten e , so gilt $P_1(l) \rightarrow 1$ für $l \rightarrow \infty$, was den Beweis vervollständigt.

□

Betrachtet man die asymptotische Wahrscheinlichkeit von $P_1(l)$ im vorigen Beweis, so erkennt man, dass man sich nicht zwangsläufig auf ein festes e beschränken muss, sondern auch Exponenten e zulassen, die in polynomieller Abhängigkeit von l gewählt wurden. Dies wird in dem folgenden Lemma festgehalten.

Lemma 5.6. *Sollte ein polynomieller Algorithmus X existieren, der die Computational Dependent RSA Assumption bricht, so kann man einen Algorithmus Y konstruieren, der das RSA Problem in polynomieller Zeit bricht, sofern e im Bildbereich eines Polynoms pol liegt, d.h. die Einschränkung $e \in \text{Im}(pol)$ für ein Polynom pol gestellt wird.*

Beweis. Wegen der Einschränkung an den Exponenten e existiert ein Polynom pol , so dass für die möglichen Werte $E(l)$ für e gilt: $E(l) \in O(pol(l)) = O(l^k)$ für ein passendes k . Der Beweis selbst verläuft analog zu dem weiter oben vorgeführten. Die abschließende Erfolgswahrscheinlichkeit P_2 beträgt hierbei

$$P_2(l) = 1 - \frac{pol(l)^2}{p} \geq 1 - \frac{l^{2k}}{p} \geq 1 - \frac{l^{2k}}{2^{l-1}} \rightarrow 1 \text{ bei } l \rightarrow \infty.$$

□

Die Korrektheit des konstruierten Algorithmus in fast allen Fällen kann man für die Praxis noch wie folgt verstärken. Bis jetzt wurden nur die beiden Werte a^e und $(a+1)^e$ betrachtet. Wie bereits am Anfang des Abschnittes erwähnt, ist die Berechnung der Werte $\beta_j := (a+j)^e$ möglich, sofern die Zahl j in einer vernünftigen Größenordnung gewählt wird und die verschiedenen Werte noch mit einer nicht vernachlässigbaren Wahrscheinlichkeit berechnet werden können. Sollte der

Algorithmus einen nichtlinearen ggT als Ergebnis liefern, so wird der Algorithmus erneut mit den beiden Polynomen f und $g_2 := (x+2)^e - \beta_2$ durchgeführt. Sollte nach diesem Schritt wieder ein Polynom vom Grad > 1 erhalten werden, wird der Algorithmus sukzessiv weiter angewendet, bis man das gewünschte lineare Polynom erhält. Offensichtlich erhält man durch diese Modifikation eine (leicht) verbesserte Chance, das gesuchte lineare Polynom zu berechnen.

5.4 Der Diskreter Log. Graph

Analog zu Abschnitt (5.3) werden hier die vom *discrete logarithm problem* abgeleiteten Diffie Hellman Probleme betrachtet. Die Beweise dieses Abschnittes sind alle offensichtlich und dienen wiederum nur der Vollständigkeit.

5.4.1 *Discrete Logarithm Problem* \Rightarrow *Diffie Hellman Problem*

Satz 5.7. *Sollte ein polynomieller Algorithmus X existieren, der das Discrete Logarithm Problem bricht, so kann man mittels X einen Algorithmus Y konstruieren, der das Diffie Hellman Problem in polynomieller Zeit bricht.*

Beweis. Seien die Voraussetzungen wie im *Diffie Hellman Problem*. Da man das *Discrete Logarithm Problem* anhand des Algorithmus X mit einer signifikanten Wahrscheinlichkeit ϵ lösen kann, kann man aus $\alpha^b \bmod p$ und α effizient den Wert von b bestimmen. Durch Square-and-multiply Algorithmen kann nun effizient $\alpha^{ab} \bmod p = (\alpha^a)^b \bmod p$ berechnet werden, wobei die Erfolgswahrscheinlichkeit offensichtlich ϵ beträgt, also wurde das *Diffie Hellman Problem* gebrochen. □

5.4.2 *Diffie Hellman Problem* \Rightarrow *Diffie Hellman Decision Problem*

Dieser Beweis folgt unmittelbar aus der Tatsache, dass das zu einem Problem definierte Entscheidungsproblem immer leichter ist als die zugrundeliegende Annahme. In diesem speziellen Fall kann der Angreifer aus g^a und g^b den Wert von g^{ab} berechnen, da er das *Diffie Hellman Problem* brechen kann. Da er nun den Wert von g^{ab} kennt, kann er somit die vorgelegten Distributionen unterscheiden, also das *Diffie Hellman Decision Problem* lösen. Mit der umgekehrten Richtung haben sich z.B. U. Maurer und S. Wolf beschäftigt [MaWo99]. Ein Beweis wurde jedoch noch nicht gefunden. In [Joux00] konnte gezeigt werden, dass es Gruppen auf speziellen elliptischen Kurven gibt, auf denen das *Diffie Hellman Problem* leicht ist, das *Discrete Logarithm Problem* jedoch (anscheinend) nicht, was wiederum gegen eine Äquivalenz spricht.

5.5 Der Höhere Reste Graph

Dieser Abschnitt beherbergt die bis jetzt schwierigsten Beweise, sofern man von dem Äquivalenzbeweis von RSA und C-DRSA absieht. Im Gegensatz zu den meisten oben vorgestellten Beweisen, sind die hier bewiesenen Beziehungen der einzelnen Annahmen zum Teil nicht mehr offensichtlich, was übersichtliche, gut strukturierte Beweise nötig macht.

5.5.1 *Higher Residuosity Problem* \Rightarrow *Decisional Higher Residuosity Problem*

Der Beweis dieser Behauptung folgt wiederum direkt aus der Tatsache, dass die kanonische Decisionvariante eines jeden Problems einfacher zu lösen ist als das ursprüngliche Problem, also dass die Berechnung einer Lösung immer schwieriger ist, als zu entscheiden, ob eine Lösung existiert. Damit stellt dieser "Beweis" einen der einfachsten Teile dieses Abschnittes dar.

5.5.2 $Class[P^2Q^2] \Rightarrow D-Class[P^2Q^2]$

Es handelt sich wiederum um einen Vergleich eines Problems mit seiner kanonischen Decisionannahme. Aus den dazu bereits mehrfach durchgeführten Betrachtungen folgt wiederum die behauptete Implikation.

5.5.3 $D-Class[P^2Q^2] \Leftrightarrow CR[P^2Q^2]$?

Für zufällig gewählte $x \in \mathbb{Z}_n$ konnte die Hinrichtung nicht gezeigt werden. Es gilt lediglich der folgende Satz:

Satz 5.8. *Sollte ein polynomieller Algorithmus A existieren, der $D-Class[P^2Q^2]$ für den festen Wert $x = 0$ mit Vorteil β bricht, so kann man mittels A einen Algorithmus A' konstruieren, der die $CR[P^2Q^2]$ in polynomieller Zeit bricht.*

Beweis. Es sei ein Algorithmus A gegeben, der $D-Class[P^2Q^2]$ löst.

1. Man wählt nun ein $g \in B$; eine sinnvolle Wahl ist $g = 1 + n$, die auch im folgenden benutzt wird.
2. Setze $x = 0$.
3. Übergebe das Tripel (g, w, x) dem Algorithmus A .
Man betrachtet in diesem Fall also die Funktion $\epsilon_g(0, y) = y^n \bmod n^2$. Das Orakel liefert uns nun einen booleschen Wert, je nachdem, ob ein solches y existiert oder nicht. Dies geschieht mit einem Vorteil β gegenüber der trivialen Ratestrategie. Dieser boolesche Wert besagt, ob das betrachtete w ein n -ter Rest modulo n^2 ist oder nicht, was die Lösung von $CR[P^2Q^2]$ liefert.
4. Gib die Antwort des Orakels als Antwort von $CR[P^2Q^2]$ aus.

Der Algorithmus erfüllt die an ihn gestellten Anforderungen und besitzt offensichtlich den Vorteil β gegenüber der trivialen Ratestrategie. □

Satz 5.9. *Sollte ein polynomieller Algorithmus A existieren, der $CR[P^2Q^2]$ mit Vorteil β bricht, so kann man mittels A einen Algorithmus A' konstruieren, der $D-Class[P^2Q^2]$ in polynomieller Zeit bricht.*

Beweis. Um den Beweis durchführen zu können, wird ein im System von Paillier (4.3.5) benutzter Sachverhalt benutzt, nämlich dass gilt: w ist n -ter Rest modulo n^2 genau dann wenn $[w]_g = 0$ gilt. Sei nun also ein Orakel gegeben, das $CR[P^2Q^2]$ löst.

1. Lasse das Orakel $wg^{-x} \bmod n^2$ lösen.

Fall 1: Erhält man als Antwort, dass wg^{-x} ein n -ter Rest modulo n^2 ist, so folgt aus dem oben zitierten Satz $[wg^{-x}]_g = 0$. Aus der Homomorphieeigenschaft folgt $0 = [wg^{-x}]_g = [w]_g + [g^{-x}]_g \bmod n$.

Nun gilt aber $\epsilon_g(-x, 1) = g^{-x} \cdot 1^n \bmod n^2 = g^{-x} \bmod n^2 \Rightarrow [g^{-x}]_g = -x$. Durch Einsetzen erhält man somit $0 = [w]_g - x \bmod n \Leftrightarrow [w]_g = x \bmod n$.

Wählt man nun x von Anfang an mit der Einschränkung $0 \leq x \leq n-1$, so erhält man die gewünschte Gleichung $[w]_g = x$.

Fall 2: Ist wg^{-x} kein n -ter Rest modulo n^2 , so muss gelten $[w]_g \neq x$. Ansonsten gäbe es ein y mit $w = g^x y^n \bmod n^2 \Rightarrow wg^{-x} = y^n \bmod n^2$, d.h. wg^{-x} wäre entgegen unserer Voraussetzung ein n -ter Rest modulo n^2 . Widerspruch.

2. Im Fall (1) gib $[w]_g = x$ aus, im Fall (2) gib aus "Keine Lösung".

Da der Algorithmus den gestellten Anforderungen genügt und der Vorteil β sich offensichtlich auf den Algorithmus A' vererbt, folgt die Behauptung. □

5.5.4 $Class[P^2Q^2] \Rightarrow PDL[P^2Q^2, g]$

Es gilt der folgende Satz:

Satz 5.10. *Sollte ein polynomieller Algorithmus A existieren, der $Class[P^2Q^2]$ mit Vorteil β bricht, so kann man mittels A einen Algorithmus A' konstruieren, der $PDL[P^2Q^2, g]$ in polynomieller Zeit bricht.*

Beweis. Der Algorithmus A erhält als Eingabe ein $g \in B$ sowie ein $w \in_R \langle g \rangle$, folglich gilt $w \equiv g^z \bmod n^2$ für ein passendes z . Offensichtlich ist das Element z wegen $\epsilon(z, 1) \equiv g^z 1^n \equiv g^z \bmod n^2$ die gesuchte Restklasse von w . Um den Algorithmus A anwenden zu können, benötigt man jedoch ein zufälliges Element aus $\mathbb{Z}_{n^2}^*$, was man durch Randomisieren wie folgt erreicht. Wähle zuerst $y \in_R \mathbb{Z}_n^*$ und setze

$$\bar{w} := wy^n \bmod n^2,$$

also $\bar{w} \equiv g^z y^n \bmod n^2$. Das Element \bar{w} ist ein zufälliges Element aus $\mathbb{Z}_{n^2}^*$, da wegen der zufälligen Wahl von w das Element z zufällig ist und y zufällig gewählt wurde. Somit ist der Algorithmus A auf \bar{w} anwendbar, was uns den Wert von $[\bar{w}]_g$ mit einem Vorteil β liefert. Aus der Definition von \bar{w} folgt direkt $[\bar{w}]_g = z$. Gib $[\bar{w}]_g$ als Restklasse von w aus.

Der so konstruierte Algorithmus besitzt offensichtlich ebenfalls den Vorteil β , was den Beweis vervollständigt. □

5.5.5 $D-Class[P^2Q^2] \Rightarrow D-PDL[P^2Q^2, g]$

Es gilt der folgende Satz:

Satz 5.11. *Sollte ein polynomieller Algorithmus A existieren, der $D\text{-Class}[P^2Q^2]$ mit Vorteil β bricht, so kann man mittels A einen Algorithmus A' konstruieren, der $D\text{-PDL}[P^2Q^2, g]$ in polynomieller Zeit bricht.*

Beweis. Der Algorithmus A erhält als Eingabe ein $g \in B$ sowie ein $w \in_R \langle g \rangle$, folglich gilt $w \equiv g^z \pmod{n^2}$ für ein passendes z , sowie ein $x \in \mathbb{Z}_n$. Offensichtlich ist das Element z wegen $\epsilon(z, 1) \equiv g^z 1^n \equiv g^z \pmod{n^2}$ die gesuchte Restklasse von w . Man muss nun entscheiden, ob x die Restklasse des Elementes w ist. Analog zu dem Beweis in (5.5.4) muss man wieder randomisieren, um den Algorithmus A anwenden zu können. Wähle wieder zuerst $y \in_R \mathbb{Z}_n^*$ und setze

$$\bar{w} := wy^n \pmod{n^2},$$

also $\bar{w} \equiv g^z y^n \pmod{n^2}$. Dieses Element \bar{w} stellt gemäß den Überlegungen aus (5.5.4) ein zufälliges Element aus $\mathbb{Z}_{n^2}^*$ dar. Somit ist der Algorithmus A auf \bar{w} und x anwendbar und man erhält mit Vorteil β eine Entscheidung, ob x der Restklasse von \bar{w} entspricht. Da w und \bar{w} der gleichen Restklasse angehören, liefert dies die ebenso die Entscheidung, ob x die Restklasse von w ist, was den Beweis vervollständigt. □

5.6 Graphverbindungsbeispiele

Die Beweise in diesem Abschnitt stellen die Zusammenhänge zwischen den oben eingeführten Bäumen dar. Im Gegensatz zu vielen Beweisen, die innerhalb einer Baumstruktur ablaufen, sind solche Verbindungsbeispiele oft nur schwer zu führen. Dies liegt daran, dass den betrachteten Problemen eine gemeinsame Grundlage fehlt, die innerhalb der Bäume oft für die Beweise genutzt wurde. Also muss erst eine gemeinsame Basis für die Probleme geschaffen werden, was oft nur durch Abstraktion der Probleme, durch abstrakt geführte Beweise oder durch den Vergleich mit neu definierten Hilfsproblemen möglich ist.

5.6.1 Integer Factorisation Problem \Leftrightarrow Square Root Problem

Obwohl die behauptete Äquivalenz seit langem bekannt ist, sollte sie dem Leser, sofern er sie zum ersten Mal sieht, zu denken geben. Bis jetzt wurde das Faktorisierungsproblem stets als *das* Referenzproblem angegeben - ein Problem, auf das sich nahezu alle anderen Probleme reduzieren lassen und das offensichtlich schwieriger als jedes andere Problem ist. Umso erstaunlicher ist die Äquivalenz zum *Square Root Problem*, das bei rein intuitiver Betrachtung als ein wesentlich leichteres Problem erscheint. Anhand des unten vorgestellten Beweises wird das Problem jedoch de facto als ein äußerst schwieriges Problem klassifiziert.

Im Beweis erweisen sich trotz der "intuitiven Überlegenheit" des Faktorisierungsproblems beide Richtungen als schwierig.

Satz 5.12. *Sollte ein polynomieller Algorithmus A existieren, der das Integer Factorisation Problem bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das Square Root Problem in polynomieller Zeit bricht.*

Beweis. Die Wurzeln einer Zahl a können, sofern die Zahl a Wurzeln modulo n besitzt, effizient berechnet werden unter der Voraussetzung, dass die Faktorisierung von n bekannt ist [BeOr81].

Da der angeführte Algorithmus die gesuchten Lösungen in polynomieller Zeit findet, reduziert sich das *Square Root Problem* polynomial auf das *Integer Factorisation Problem*. \square

Satz 5.13. *Sollte ein polynomieller Algorithmus A existieren, der das Square Root Problem mit Vorteil ϵ bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das Integer Factorisation Problem in polynomieller Zeit bricht.*

Beweis. Anhand des gegebenen Algorithmus A konstruiert man sich einen randomisierten Algorithmus A' wie folgt:

1. Wähle eine natürliche Zahl x mit $\text{ggT}(x, n) = 1$.
2. Berechne $a = x^2 \bmod n$.
3. Wende Algorithmus A auf (a, n) an. Man erhält eine Wurzel y von a modulo n mit einem Vorteil ϵ gegenüber der trivialen Ratestategie.
4. Teste, ob $y \equiv \pm x \pmod n$. Falls nein, starte den Algorithmus beginnend bei Schritt 1 neu.
5. $y \not\equiv x \pm \pmod n \Rightarrow (\text{ggT}(x - y, n) = p) \vee (\text{ggT}(x - y, n) = q)$. Man erhält durch die Berechnung des ggT's also einen Faktor von n und hat somit n faktorisiert.

Es bleibt zu zeigen, dass der obige Algorithmus wirklich als Ausgabe einen Faktor von n liefert und desweiteren, wieviele x der Algorithmus im Schnitt für die Berechnung wählen muss.

Lemma 5.14. *Seien x, y, n natürliche Zahlen. Gilt nun $x^2 \equiv y^2 \pmod n$, aber $x \not\equiv \pm y \pmod n$ so ist $d := \text{ggT}(x - y, n)$ ein nichttrivialer Faktor von n .*

Beweis. Man weiß nach Voraussetzung

1. $n \mid x^2 - y^2$
2. $(n \nmid (x - y)) \wedge (n \nmid (x + y))$

Aus 1. folgt nun aber $n \mid (x - y)(x + y)$, aber n teilt keinen der beiden Faktoren, d.h. der zu berechnende ggT ist nichttrivial. Also muss gelten $\text{ggT}(x - y, n) = p$ oder $\text{ggT}(x - y, n) = q$. \square

Also ist anhand dieses Lemmas die Korrektheit unseres Faktorisierungsalgorithmus bewiesen. Man betrachtet nun die Effizienz des Algorithmus. Seine Effizienz ist offensichtlich proportional zur Anzahl der verschiedenen Zahlen, die man im ersten Schritt des Algorithmus wählen muss. Es genügt folglich, sich die durchschnittliche Anzahl von Schleifendurchläufen in einem Programmablauf zu betrachten:

Da n die Form $n = pq$ mit p, q prim hat, besitzt a vier Wurzeln modulo n , d.h. man erhält in Punkt 3. des Algorithmus eine Wurzel von x , für die entweder gilt $y \equiv \pm x \pmod n$ oder $y \equiv \pm z \pmod n$ für ein passendes z . Der Algorithmus terminiert in diesem Durchlauf genau dann, wenn das berechnete y von der zweiten Form ist. Die Wahrscheinlichkeit dafür beträgt offensichtlich $\frac{1}{2}$. Betrachtet man nun die Wahrscheinlichkeit, dass der Algorithmus nach k Durchläufen noch immer nicht terminiert ist, so erhält man eine Wahrscheinlichkeit von $\frac{1}{2^k}$, d.h. die Wahrscheinlichkeit fällt exponentiell in der Anzahl der Schleifendurchläufe, was dem Algorithmus

auf jeden Fall zu einer sehr geringen Laufzeit verhilft. Die Anzahl der erwarteten Durchläufe, bis ein Faktor gefunden wurde, beträgt bei einer Erfolgswahrscheinlichkeit von $\frac{1}{2}$ genau 2 Durchläufe.

Folglich handelt es sich um eine Polynom-Zeit-Reduktion, wobei sich die nicht vernachlässigbare Erfolgswahrscheinlichkeit ϵ des Algorithmus A aufgrund obiger Überlegung in eine nicht vernachlässigbare Erfolgswahrscheinlichkeit des Algorithmus A' überträgt, was den Beweis vervollständigt. \square

5.6.2 Integer Factorisation Problem \Rightarrow Higher Residuosity Problem

Es wurde bereits bei der Einführung der *Higher Residuosity Assumption* erwähnt, dass das Problem sehr leicht zu lösen ist, sofern die Faktorisierung von n bekannt ist. Dabei wird das folgende Kriterium aus der elementaren Zahlentheorie benutzt:

Satz 5.15. *Seien $e, n \in \mathbb{N}$, $c \in \mathbb{Z}_n$, $d := \text{ggT}(e, \varphi(n))$, dann sind für zyklische \mathbb{Z}_n^* äquivalent:*

1. c ist e -ter Potenzrest modulo n .
2. Es ist $c^{\varphi(n)/d} \equiv 1 \pmod{n}$.

Beweis. Ein Beweis ist in nahezu jedem Buch über elementare Zahlentheorie zu finden, siehe z.B. [Bund98]. \square

Der Satz kann in unserem Fall nicht unmittelbar ungewendet werden, da für $n = pq$ die Gruppe \mathbb{Z}_n^* gemäß Satz (2.24) nicht zyklisch ist. Kann man allerdings die Faktorisierung von n effektiv berechnen, so kann man obigen Satz auf die Gruppen $GF(p)$ und $GF(q)$ anwenden, da ihre Ordnungen $p - 1$ bzw. $q - 1$ nun bekannt sind. Man erhält folgenden Satz.

Satz 5.16. *Sollte ein polynomieller Algorithmus A existieren, der das Integer Factorisation Problem mit einer nicht vernachlässigbaren Wahrscheinlichkeit bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das Higher Residuosity Problem in polynomieller Zeit bricht.*

Beweis. Der Algorithmus A' wendet zu Beginn den Algorithmus A auf die Zahl n an und erhält somit die Primfaktoren p und q mit einer nicht vernachlässigbaren Wahrscheinlichkeit. Nun wendet er obigen Satz auf die Eingaben (e, p, c) bzw. (e, q, c) an. Dies ist möglich, da $\varphi(p) = p - 1$ bzw. $\varphi(q) = q - 1$ anhand der inzwischen bekannten Faktorisierung berechnet werden können. Als Ausgabe erhält er zwei Bits b_p und b_q mit $b_p = 1$ genau dann wenn c ein e -ter Potenzrest modulo p ist, analog modulo q . Die Zahl c ist genau dann ein e -ter Potenzrest modulo n , wenn sie sowohl e -ter Potenzrest modulo p als auch e -ter Potenzrest modulo q ist. Der Algorithmus A' gibt nun aus, dass c ein e -ter Potenzrest modulo n ist, genau dann wenn $b_p \wedge b_q = 1$ gilt. Die Erfolgswahrscheinlichkeit von A' vererbt sich offensichtlich von der Erfolgswahrscheinlichkeit von A , was den Beweis vervollständigt. \square

5.6.3 Integer Factorisation Problem \Rightarrow Decisional Composite Residuosity Problem

Satz 5.17. *Sollte ein polynomieller Algorithmus A existieren, der das Faktorisierungsproblem mit Vorteil ϵ bricht, so kann man mittels A einen Algorithmus A' konstruieren, der die Decisional Composite Residuosity Assumption in polynomieller Zeit bricht.*

Beweis. Seien $n = pq$ sowie $z \in \mathbb{Z}_n^*$ gegeben. Wendet man den gegebenen Algorithmus A auf die Zahl n an, so erhält man die Primfaktoren p und q mit Vorteil ϵ über der trivialen Ratestrategie. Allgemein gilt $\varphi(n^2) = n\varphi(n)$, was somit effizient berechnet werden kann; setze desweiteren $d := ggT(e, \varphi(n^2))$. Gemäß dem Satz des letzten Abschnittes ist z genau dann ein n -ter Rest modulo n^2 , wenn gilt: $z^{\varphi(n^2)/d} \equiv 1 \pmod{n}$. Da man $\varphi(n^2)$ bereits berechnet hat, kann auch d berechnet und das Kriterium nachgeprüft werden, was den Beweis vervollständigt. \square

5.6.4 RSA Problem \Rightarrow CR[P^2Q^2]?

Satz 5.18. *Sollte ein polynomieller Algorithmus A existieren, der das RSA Problem mit Vorteil ϵ für den festen Exponenten $e = n$ bricht, so kann man mittels A einen Algorithmus A' konstruieren, der Class[P^2Q^2] in polynomieller Zeit bricht.*

Beweis. Wie bereits in Kapitel 4 bereits beschrieben wurde, bildet Class[P^2Q^2] eine Verallgemeinerung des Class[P^2Q^2, g]-Problems, die darauf beruht, dass sich alle Instanzen des Class[P^2Q^2, g]-Problems bzgl. g als äquivalent erweisen, d.h. dass ihre Berechnung entweder für alle g oder für kein g möglich ist.

Folglich genügt es, sich auf den Fall

$$\text{RSA} \Rightarrow \text{Class}[P^2Q^2, 1 + P^2Q^2]$$

zu beschränken.

Sei nun ein Algorithmus A gegeben, welcher das RSA Problem löst. Es gilt nun $w = (1 + n)^x \cdot y^n \pmod{n^2}$ für $x \in \mathbb{Z}_n$, $y \in \mathbb{Z}_n^*$ passend. Damit gilt weiter: $w \equiv y^n \pmod{n}$. Da wegen $n = pq$ gelten $ggT(n, (p-1)(q-1)) = 1$ muss, ist der Algorithmus A auf das Element w anwendbar. Man erhält auf diese Weise $y \pmod{n}$ mit einem Vorteil ϵ gegenüber der trivialen Ratestrategie. Da y nach Definition aus \mathbb{Z}_n^* gewählt werden muss, ist somit auch der Wert von $y^n \pmod{n^2}$ bekannt und es gilt:

$$\frac{w}{y^n} \equiv (1 + n)^x \equiv 1 + xn \pmod{n^2}.$$

was zu der Gleichung $x = [w]_{1+n}$ führt. \square

5.6.5 Composite Discrete Logarithm Problem \Rightarrow Strong RSA Problem

Die Idee bei diesem Beweis ist, zu dem gegebenen Element $c \in \mathbb{Z}_n$ zuerst ein Element $g \in \mathbb{Z}_n$ mit $c \in \langle g \rangle$ zu wählen und dann anhand eines Orakels für das *Composite Discrete Logarithm Problem* den gesuchten Exponenten e zu bestimmen. Da die Gruppe \mathbb{Z}_n für $n = pq$ nicht zyklisch ist (vgl. Satz 2.24) bereitet es Schwierigkeiten, ein geeignetes $g \neq c$ zu finden, das der Anforderung $c \in \langle g \rangle$ genügt, da es keine Generatoren gibt. Folglich wird zuerst untersucht, mit welcher Wahrscheinlichkeit ein solches g gefunden werden kann. In den folgenden Rechnungen geht man der Einfachheit halber davon aus, dass p und q sogenannte *safe primes* sind, d.h. das sie sich darstellen lassen als $p-1 = 2p'$ bzw. $q-1 = 2q'$ mit p', q' prim. Die Verwendung solcher *safe primes* ist in der Praxis üblich.

Lemma 5.19. *Sei $n = pq$ mit p, q prim gegeben. Dann ist es effizient möglich, zu einem Element $c \in \mathbb{Z}_n$ ein Element $g \in \mathbb{Z}_n^*$ mit $g \neq c$ zu bestimmen, so dass gilt: $c \in \langle g \rangle$.*

Beweis. Man wählt zuerst ein $g \in \mathbb{Z}_n^*$ zufällig. Es sei $g_p := g \bmod p$ und $g_q := g \bmod q$. Da die Gruppe \mathbb{Z}_p^* $\varphi(\varphi(p)) = \varphi(p-1) = \varphi(2p') = p' - 1 \approx \frac{\mathbb{Z}_p^*}{2}$ Generatoren besitzt, (d.h. in etwa jedes zweite Element ist ein Generator) kann man o.B.d.A davon ausgehen, dass g_p ein Generator von \mathbb{Z}_p^* ist. Die Aussage gilt analog für g_q in \mathbb{Z}_q^* . Man erhält somit, dass für ein zufällig gewähltes Element g in etwa mit der Wahrscheinlichkeit $\frac{1}{4}$ gilt: g_p ist Generator von \mathbb{Z}_p und g_q ist Generator von \mathbb{Z}_q . Iteriert man die Versuche l mal, ein Element g mit dieser Eigenschaft zu wählen, so besitzt man eine Erfolgswahrscheinlichkeit von $1 - (\frac{3}{4})^l$. Folglich kann man im folgenden o.B.d.A. davon ausgehen, dass man ein Element g mit obiger Eigenschaft gewählt hat.

Da nun g_p, g_q Generatoren von \mathbb{Z}_p^* bzw. \mathbb{Z}_q^* sind, existieren $x \in \mathbb{Z}_{p-1}$ bzw. $y \in \mathbb{Z}_{q-1}$ mit $c \equiv g_p^x \bmod p$ und $c \equiv g_q^y \bmod q$.

Das Ziel ist zu zeigen, dass ein $z \in \mathbb{Z}_{\varphi(n)}$ existiert mit $c \equiv g^z \bmod n$. Dies gibt es genau dann, wenn gilt $c \equiv g_p^z \bmod p$ und $c \equiv g_q^z \bmod q$. Zusammen ergibt sich nun, dass ein solches z existiert, wenn gilt $z \equiv x \bmod p-1$ und $z \equiv y \bmod q-1$. Für gerade x, y ist diese Bedingung auf jeden Fall erfüllt. Sei $x = 2x'$ und $x = 2y'$, dann liefert die Kürzungsregel die Kongruenzen $\frac{z}{2} \equiv x' \bmod p'$ und $\frac{z}{2} \equiv y' \bmod q'$. Wegen $ggT(p', q') = 1$ ist der chinesische Restsatz anwendbar, was die Existenz der Lösung z liefert. Der Fall x, y gerade tritt offensichtlich mit Wahrscheinlichkeit $\frac{1}{4}$ auf. Zusammen mit der Erfolgswahrscheinlichkeit der Suche nach einem geeigneten g erhält man als endgültige Wahrscheinlichkeit, ein Element g zu finden mit $c \in \langle g \rangle$, mindestens $\frac{1}{16}$. Durch l -fache Iteration erreicht man wiederum eine Erfolgswahrscheinlichkeit von $1 - (\frac{15}{16})^l$, was den Beweis vervollständigt. □

Kommen wir nun zum eigentlichen Äquivalenzbeweis. Unter Benutzung des soeben bewiesenen Lemmas erweist sich dieser als sehr einfach.

Satz 5.20. *Sollte ein polynomieller Algorithmus A existieren, der das Composite Discrete Logarithm Problem mit Vorteil ϵ bricht, so kann man mittels A einen Algorithmus A' konstruieren, der das Strong RSA Problem in polynomieller Zeit bricht.*

Beweis. Gemäß dem soeben bewiesenen Lemmas ist es effizient möglich, zu einem gegebenen Element $c \in \mathbb{Z}_n$ ein Element $g \in \mathbb{Z}_n^*$ mit $g \neq c$ zu bestimmen, so dass $c \in \langle g \rangle$ gilt. Wendet man nun den Algorithmus A auf das Tupel (n, g, c) an, so erhält man den Exponenten e mit einer nicht vernachlässigbaren Wahrscheinlichkeit ϵ . Als Ausgabe der *Strong RSA Assumption* gibt man das Tupel (g, e) aus. □

Kapitel 6

Orakel-Bitsicherheiten der einzelnen Annahmen

6.1 Einleitung

Nachdem die einzelnen Annahmen im letzten Kapitel miteinander verglichen wurden, beschäftigt sich dieses Kapitel mit der Sicherheit jeder Annahme an sich. Da alle vorgestellten Annahmen als schwer angesehen werden, kann man nicht erwarten, einen Algorithmus zu finden, der eine der Annahmen bricht, ohne dass ihm irgendwelche Zusatzinformationen zur Verfügung stehen. Es bleibt zu klären, was genau man unter Zusatzinformationen versteht. Im Folgenden werden Zusatzinformationen stets als bekannte oder vorhersagbare Bits eines Geheimnisses angesehen. Man nimmt also z.B. an, man hätte ein Orakel zur Hand, das das i -te Bit eines Geheimnisses bestimmen kann. Es stellt sich nun die Frage, welches Geheimnis man durch das Orakel angreifen will:

1. Den private key eines Verschlüsselungssystem. Eine mögliche Frage hierzu wäre: Welche und wieviele Bits des RSA-private keys d brauche ich, um d vollständig zu rekonstruieren? Diese Art der Bitsicherheit wird im nächsten Kapitel behandelt.
2. Die unverschlüsselte Nachricht m ; d.h. es interessiert die Frage: Welche und wieviele Bits von m muss ich voraussagen können, um aus der Verschlüsselung $E(m)$ die ursprüngliche Nachricht zu rekonstruieren, also die zugrundeliegende Annahme zu brechen?

Dieses Kapitel beschäftigt sich mit der in Punkt 2 beschriebenen Bitsicherheit.

Auf den ersten Blick mag es dem Leser vielleicht nicht einleuchten, warum man sich überhaupt mit der Bitsicherheit beschäftigt. Wenn man anhand eines Bits die ursprüngliche Nachricht rekonstruieren könnte und dieses Bit noch mit einer nicht vernachlässigbaren Wahrscheinlichkeit voraussagen könnte, so hätte man die zugrundeliegende Annahme gebrochen, was jedoch im Widerspruch zu unseren Vorüberlegungen über schwere Probleme stehen würde. Der kritische Punkt liegt darin, dass ein solches Bit, das die Rekonstruktion der ganzen Nachricht ermöglicht, nicht vorhersagbar sein darf, also dass ein Orakel in der Praxis nicht existieren kann. Ein wichtiger Grund, sich dennoch damit zu beschäftigen, liegt darin, dass man sich zum Brechen einer Annahme auf Algorithmen beschränken kann, die nur Aussagen über das entsprechende Bit liefern, was eine gründlichere Untersuchung der Annahmen ermöglicht. Desweiteren steht die Bitsicherheit

in einem engen Zusammenhang mit der semantischen Sicherheit von Kryptosystemen. Dieser Zusammenhang stellte den Auslöser der Bitsicherheitsbetrachtungen dar.

In der Praxis sind die Bitsicherheitsbetrachtungen aus Punkt 1 von großer Bedeutung. Besonders bei Verschlüsselungssystemen kann es vorkommen, dass dem Angreifer ein Teil der Nachricht bekannt ist oder er durch sogenannte Timing-Angriffe einmaligen Zugriff auf bestimmte Bits des geheimen Schlüssels bekommt, was ihm nicht helfen sollte, die Nachricht zu entschlüsseln. Auf diese Art der Bitsicherheit wird im nächsten Kapitel näher eingegangen.

Da Bitsicherheitsbeweise zu den schwierigsten Beweisen in der Kryptographie zählen, die im Allgemeinen nur durch abstrakte Konzepte und hauptsächlich durch wahrscheinlichkeitstheoretische Betrachtungen durchführbar sind, muss man sich zuerst der Definition der auftretenden Begriffe widmen. Der folgende Abschnitt gibt einen Überblick über die in diesem Kapitel benutzte Terminologie. Um für die folgenden Beweise eine größtmögliche Übersicht bereitzustellen, werden auch die bereits im Grundlagen-Abschnitt eingeführten Begriffe wiederholt, sofern ihre Kenntnis in den den Beweisen vonnöten ist. In diesem Kapitel wird der Beweis der Bitsicherheit eines jeden Bits in der RSA Annahme genau durchgeführt, um dem Leser ein Gefühl dafür zu geben, wie man solche Beweise angeht und als wie komplex sie sich letztendlich erweisen. Abschliessend werden noch die bekannten Resultate der Bitsicherheit der *discrete logarithm assumption* aufgeführt.

6.2 Notation und Definitionen

Wie in der Kryptographie üblich, wird im Folgenden als Berechnungsmodell das Modell einer probabilistischen polynomiellen Turingmaschine benutzt, die in Zeit $poly(l)$ läuft, wobei l die Länge der Eingabe bezeichnet, in der Kurzform: ppTm. Zu einem binären String y bezeichne $\|y\|$ die Länge des Strings; auf Zahlen angewandt entspricht $\|\cdot\|$ der Länge der Zahl als binärer String. Ist S eine Menge, so wird durch $|S|$ ihre Mächtigkeit bezeichnet. Eine weiterer wichtiger Begriff ist die Erfolgswahrscheinlichkeit eines Orakels:

Definition 6.1 (Erfolgswahrscheinlichkeit eines Orakels). Sei f eine beliebige Funktion, b sei eine in polynomieller Zeit berechenbare boolesche Funktion. Ein $\epsilon(l)$ -Orakel für b unter f ist eine ppTm D , für die gilt:

$$P(D(f(x)) = b(x) :: x \in_R \{0, 1\}^l) \geq \frac{1 + \epsilon(l)}{2}.$$

Man interessiert sich offensichtlich nur für den Fall $\epsilon(l) > 0$, da man sich ansonsten auf die triviale Ratestrategie beschränken könnte. Sollte kein $\epsilon(l)$ -Orakel existieren, so heißt b $\epsilon(l)$ -sicher für f , ist b sogar $\epsilon(l)$ -sicher für alle nicht vernachlässigbaren $\epsilon(l)$, so heisst b sicher für f .

Für $m, z \in \mathbb{Z}, m > 0$ definiert man $[z]_m$ durch $z \bmod m$, wobei hier wie immer die kleinste nichtnegative Zahl gemeint ist, die die entsprechende Restklasse repräsentiert. Desweiteren setzt man

- $abs_m(z) = \min\{[z]_m, m - [z]_m\}$
- Gilt für ein $\delta \in [0, 1)$: $abs_m(z) \leq \delta m$, so wird z als δ -klein (modulo m) bezeichnet.
- Eine Zahl x heißt δ -determiniert modulo m , wenn x als $y + z$ geschrieben werden kann, wobei y effizient berechnet werden kann und z δ -klein ist.

- Die RSA-Verschlüsselung wird durch $E_n(x, e)$ abgekürzt, d.h. $E_n(x, e)$ entspricht $[x^e]_n$ für $n = pq$ mit p, q prim und $ggT(e, (p-1)(q-1)) = 1$.
- Für $z \in \mathbb{Z}$, $0 \leq i < \|z\|$ entspricht $bit_i(z)$ dem i -ten Bit der binären Darstellung von z , also $bit_i(z) = \lfloor \frac{z}{2^i} \rfloor \bmod 2$. Speziell definiert man $lsb(z) := bit_0(z)$. Man spricht von dem sogenannten *least significant bit*.
- Die Funktion $half_n(x)$ wird durch $half_n(x) = \begin{cases} 0 & \text{falls } x \leq \frac{n-1}{2} \\ 1 & \text{falls } x > \frac{n-1}{2} \end{cases}$ definiert.
- Um ganze Blöcke von Bits zu erfassen, definiert man $B_i^j(z)$ als die Bits $i, i+1, \dots, j$ in der binären Darstellung von z .

Ein Problem entsteht bei der Definition der $\epsilon(l)$ -Sicherheit der höherwertigsten Bits. Da die uniforme Wahrscheinlichkeitsverteilung in \mathbb{Z}_n nicht der uniformen Verteilung in $\{0, 1\}^l$ entspricht, sind die Bits in $[z]_n$ nicht einheitlich verteilt. Als Beispiel betrachtet man das höchstwertige Bit: Es sei α so gegeben, dass gilt: $2^\alpha \leq n \leq 2^{\alpha+1}$. Dann haben genau 2^α Elemente aus \mathbb{Z} als höchstes Bit eine 0 in ihrer Binärdarstellung, aber nur $n - 2^\alpha$ Elemente eine 1. Somit gilt z.B. für $n = 2^\alpha + 1$: $P(bit_\alpha(z) = 0 :: z \in_R \mathbb{Z}_n) = \frac{2^\alpha}{2^\alpha + 1} \approx 1$ für grosse α . Das Beispiel zeigt, dass die höherwertigen Bits von m schon durch eine triviale Ratestrategie mit einer erhöhten Wahrscheinlichkeit vorausgesagt werden können, was eine neue Definition der $\epsilon(l)$ -Sicherheit erforderlich macht. Diese Eigenschaft nennt man den *bias* des i -ten Bits. In Kapitel (6.3.5) wird gezeigt, was für die $O(\log l)$ höherwertigsten Bits von m beachtet werden muss, und es muss eine neue Sicherheitsdefinition für diese Bits eingeführt werden. Zuletzt wird noch definiert, wann zwei Wahrscheinlichkeitsverteilungen unterscheidbar sind:

Definition 6.2 (Polynomielle Unterscheidbarkeit von Wahrscheinlichkeitsverteilungen). Seien D, D' zwei Wahrscheinlichkeitsverteilungen des selben Wahrscheinlichkeitsraum S . Dann heissen D, D' polynomiell unterscheidbar, wenn es eine pptm T gibt, für die

$$P_{y \in_D S}[T(y) = 1] - P_{y' \in_{D'} S}[T(y') = 1]$$

nicht vernachlässigbar ist.

6.3 Bitsicherheit der RSA-Funktion

6.3.1 Übersicht

Die Bitsicherheit von RSA wird bereits seit langem untersucht. Dennoch war bis vor kurzem nur wenig über die genaue Sicherheit des Problems bekannt. Die ersten Bitsicherheitsbeweise zu RSA, die sich lediglich mit den niederwertigsten Bits befassten, wurden von S. Goldwasser, S. Micali, P. Tong [GoMT82] und M. Ben-Or, B. Chor, A. Shamir [?] durchgeführt.

Die Sicherheit des niederwertigsten Bits wurde fortan stetig verbessert, siehe Abschnitt (6.3.2), um schließlich im Beweis der vollständigen Sicherheit zu enden [ACGS88].

Bis vor kurzem gab es jedoch noch keine zufriedenstellende Abschätzung für die inneren Bits der Nachricht m . Man wusste nur, dass sie nicht mit einer Wahrscheinlichkeit $\geq \frac{3}{4}$ vorhersagbar sind, was jedoch noch eine grosse Lücke zu dem erhofften $\leq \frac{1}{2} + \frac{1}{poly(l)}$ -Resultat für die inneren

Bits, bzw. einer analogen Schranke für die *biased* Bits, offengelassen hat.

In [HaNa98] fanden J. Hastad und M. Näslund schließlich einen Beweis, der diese Schranke lieferte, d.h. man weiß nun:

Theorem 3. *Jedes einzelne Bit eines Schlüsseltextes c einer RSA-Verschlüsselung ist sicher, sofern das RSA Problem schwer ist.*

Im Folgenden wird diese Behauptung hergeleitet und bewiesen. Der Beweis wird in zwei Schritten durchgeführt. Man zeigt:

1. Die exakte Nachricht m ist in polynomieller Zeit rekonstruierbar, sofern man ein lsb-Orakel zur Verfügung hat, bzw. die Nachricht m ist mit einer nicht vernachlässigbaren Wahrscheinlichkeit in polynomieller Zeit rekonstruierbar, sofern ein Orakel existiert, das eine nicht vernachlässigbare Erfolgswahrscheinlichkeit besitzt. (6.3.2)
2. Aus einem Orakel für das entsprechende i -te Bit lässt sich in polynomieller Zeit ein lsb-Orakel konstruieren. Dabei wird wieder unterschieden zwischen:
 - der Sicherheit der $O(\log l)$ niederwertigsten Bits. (6.3.3)
 - der Sicherheit der "mittleren" RSA-Bits, genauer der verbliebenen Bits ohne die $O(\log l)$ höherwertigsten Bits. (6.3.4)
 - der Sicherheit der $O(\log l)$ höherwertigsten Bits. (6.3.5)

Das Kapitel (6.3.6) beschäftigt sich mit der simultanen Sicherheit von Bitblöcken einer RSA-Nachricht, was die Betrachtungen des RSA-Problems vervollständigt.

6.3.2 Die Sicherheit des niederwertigsten Bits

Der Sicherheit des niederwertigsten RSA-Bits wurde bereits seit geraumer Zeit eine grosse Aufmerksamkeit gewidmet. Das erste Resultat erreichten hierbei Goldwasser, Micali und Tong [GoMT82] im Jahre 1982; sie zeigten die $(1 - o(1))$ -Sicherheit des niederwertigsten Bits. Damit war klar, dass es bei endlicher Bitlänge von n kein Orakel für das lsb geben kann, das dieses fehlerlos vorhersagen kann.

Von nun an war das Ziel, die $\epsilon(l)$ -Sicherheit zu zeigen für jedes nicht vernachlässigbare $\epsilon(\cdot)$, was einer vollständigen Sicherheit des betrachteten Bits entsprechen würde. Ein Jahr später wurde die $(\frac{1}{2} + o(1))$ -Sicherheit durch Einführung einer neuen ggT-Berechnungsmöglichkeit von Ben-Or, Chor und Shamir gezeigt [BeCS83]. Durch Verfeinerung dieser Technik zeigte Vazirani [VaVa83] eine 0.464-Sicherheit, Goldreich ein Jahr später, im Jahre 1984, eine 0.45-Sicherheit [Gold84]. Der endgültige Beweis der $\epsilon(l)$ -Sicherheit wurde von Chor und Goldreich im Jahre 1985 gezeigt [ChGo85] (s.a. [ACGS88]). Ein neuartiger, einfacher Beweis der $\epsilon(l)$ -Sicherheit wurde von Fischlin und Schnorr im Jahre 1997 vorgestellt [FiSc97]. Anhand obiger Resultate gilt somit:

Satz 6.3. *Das niederwertigste RSA-Bit ist $\epsilon(l)$ -sicher für jedes nicht vernachlässigbare $\epsilon(l)$, d.h. das lsb ist sicher.*

Im Folgenden wird nun zuerst auf intuitiver Basis erklärt, wie man die zugrundeliegende Nachricht m unter Benutzung eines perfekten lsb-Orakels rekonstruieren könnte. Wie bereits

oben erwähnt, kann ein solches Orakel nicht existieren, der Beweis erleichtert jedoch das Verständnis der späteren Beweise.

Anschließend wendet man sich der wesentlichen Aussage zu, dass man m mit einer nicht vernachlässigbaren Wahrscheinlichkeit rekonstruieren kann, sofern ein $\epsilon(l)$ -Orakel für das niederwertigste Bit existiert mit einem nicht vernachlässigbaren $\epsilon(l)$. Man beginnt mit folgendem Satz:

Satz 6.4. *Sollte ein Orakel Q_0 für das niederwertigste RSA-Bit existieren, das das betreffende Bit anhand eines Schlüsseltextes $c \equiv m^e \pmod n$ mit Wahrscheinlichkeit 1 voraussagen kann, so kann man mit Hilfe des Orakels Q_0 die Nachricht m in polynomieller Zeit rekonstruieren.*

Beweisidee:

Man zeigt zuerst folgenden Hilfssatz:

Lemma 6.5. *Es gilt:*

$$\text{half}_n(m) = Q_0(c \cdot 2^e \pmod n)$$

Beweis. Man unterscheidet die beiden Fälle

1. $\text{half}_n(m) = 0$, d.h. $0 \leq m \leq \frac{n-1}{2}$
2. $\text{half}_n(m) = 1$, d.h. $\frac{n-1}{2} < m \leq n-1$

Im ersten Fall gilt somit $2m < n$. Folglich ist $(2m) \pmod n$ gerade, d.h. $\text{lsb}((2m) \pmod n) = 0$ und es gilt

$$0 = \text{lsb}(2m) = Q_0((2m)^e \pmod n) = Q_0(m^e \pmod n \cdot 2^e \pmod n) = Q_0(c \cdot 2^e \pmod n)$$

Es folgt die Behauptung.

Im zweiten Fall gilt $n < 2m < 2n$. Also wird $2m \pmod n$ repräsentiert durch $2m - n$. Da $2m$ gerade und n ungerade ist, folgt

$$1 = \text{lsb}(2m - n) = \text{lsb}(2m \pmod n) = Q_0((2m)^e \pmod n) = Q_0(m^e \cdot 2^e \pmod n) = Q_0(c \cdot 2^e \pmod n)$$

was die Behauptung auch im zweiten Fall liefert. □

Obige Art der Berechnung wird Sampletechnik oder binäre Suche genannt.

Der eigentliche Beweis läuft nun folgendermaßen ab: Gemäß obigem Lemma kann man ein Intervall bestimmen, in dem das gesuchte m liegt. Durch Iteration soll das Intervall verkleinert werden, bis es nur noch ein Element enthält. Dieses Element ist dann das gesuchte m .

Nach dieser einführenden Erklärung beschäftigt man sich nun mit dem eigentlichen Hauptsatz dieses Abschnittes:

Satz 6.6. *Angenommen, es existiert ein lsb -Orakel O_1 mit einer Erfolgswahrscheinlichkeit $\geq \frac{1+\epsilon(l)}{2}$ für ein nicht vernachlässigbares $\epsilon(l)$, so kann die Nachricht m mit einer nicht vernachlässigbaren Wahrscheinlichkeit rekonstruiert werden. Nimmt man der Einfachheit halber an, dass ϵ^{-1} und l Zweierpotenzen mit $l \geq 2^9$ sind sowie das Orakel eine erwartete Laufzeit T besitzt, so beträgt die erwartete Laufzeit der RSA-Inversion $O(l^2 \epsilon^{-2} T + l^2 \epsilon^{-6})$, also polynomielle Laufzeit in l .*

Dieser Satz wurde durch Konstruktion eines geeigneten Algorithmus bewiesen. Dieser im folgenden angeführte Algorithmus sowie der Beweis seiner Korrektheit stammt aus [FiSc97]. Ich beginne mit einer informellen Beschreibung des später angeführten Algorithmus.

Der Algorithmus basiert auf der bereits im letzten Lemma angedeuteten Sampletechnik, mittels Iteration den “Standpunkt” des gesuchten m immer weiter einzuschränken. Im Gegensatz zu obigem Beweis wird hier jedoch ein probabilistischer Algorithmus konstruiert:

1. Man wählt zuerst $a, b \in_R \mathbb{Z}_n^*$.
2. Man bestimmt $lsb([am]_n)$ und $lsb([bm]_n)$ und “errät” ungefähre Abschätzungen der Position von $[am]_n$ und $[bm]_n$.
3. Für $a_t := a2^{-t} \bmod n$ iteriert man rationale Approximationen $u_t n$ mit $|[a_t m]_n - u_t n| \leq \frac{\epsilon n}{4 \cdot 2^t}$ für $t = 1, \dots, l$.
4. Durch die finale Approximation $u_l n$ für $[a_l m]_n$ erhält man $m = a_l^{-1} \cdot [u_l n + \frac{1}{2}] \bmod n$.

Die eigentlich Iteration in Schritt 3 wird durch die Bestimmung von $lsb(a_t m)$ mittels des Orakel O_1 durchgeführt. Da jedoch das Orakel O_1 nur mit einer bestimmten Wahrscheinlichkeit das niederwertigste Bit korrekt bestimmen kann, können somit Fehlinformationen auftreten. Aus diesem Grund reicht es nicht aus, das Orakel nur einmal zu befragen, sondern man muss mehrere Anfragen durchführen und sich dann anhand des Durchschnitts orientieren. Offensichtlich genügt es jedoch nicht, dem Orakel den selben Wert mehrfach zu geben, da es in diesem Fall unter Umständen immer ein falsches Ergebnis liefert. Der Trick liegt nun darin, sich Werte zu konstruieren, die zwar von dem ursprünglichen Wert verschieden sind, aber durch Umformungen wieder in eine Gestalt gebracht werden können, in der die Aussagen des Orakels von Nutzen sind. Man spricht von einer sogenannten Mehrheitsentscheidung (*majority decision*), die nun formal eingeführt werden soll:

Um eine Mehrheitsentscheidung für das Bit $lsb(a_t m)$ durchzuführen, werden Multiplikatoren $a_t + ia_{t-1} + b$ benutzt, die paarweise unabhängig sein sollen, wobei i für die i -te Durchführung einer Orakelanfrage stehen soll. Die Addition der drei Komponenten wird über dem Ring \mathbb{Z} durchgeführt. Der Algorithmus bestimmt nun eine Zahl $w_{t,i}$, die der Gleichung

$$(a_t + ia_{t-1} + b)m = [a_t m]_n + i[a_{t-1} m]_n + [bm]_n - w_{t,i}n$$

genügt. In diesem Fall heißt $w_{t,i}$ korrekt. Die Additionen werden hier in \mathbb{Z} durchgeführt. Man bestimmt nun im i -ten Durchgang $lsb(a_t m)$ für beide Seiten der Gleichung folgendermaßen: Man schätzt das niederwertigste Bit der linken Seite mittels des Orakels O_1 und benutzt nun, dass die rechte Seite der Gleichung linear in $lsb(a_t m)$ ist.

Zur Mehrheitsentscheidung werden nun $h = \min\{2^t, 2l\}\epsilon^{-2}$ Messungen durchgeführt, wobei h und die im Algorithmus eingeführte Menge A_h so gewählt wurden, dass sie einen Mittelweg zwischen möglichst geringer Fehlerwahrscheinlichkeit und Effizienz liefert. Man kann nun den eigentlichen Algorithmus formulieren:

RSA-Inversionsalgorithmus

1. Eingabe: $c \equiv m^e \bmod n$, e , n .
 $t := 0$, wähle zufällige Zahlen $a, b \in_R \mathbb{Z}_n^* \subset [0, n)$,
wähle rationale Zahlen $u \in \frac{\epsilon^3}{4}[0, 4\epsilon^{-3})$, $v \in \frac{\epsilon}{4}[0, 4\epsilon^{-1})$, die den Gleichungen

$$|[am]_n - un| \leq \frac{\epsilon^3}{8}n, \quad |[bm]_n - vn| \leq \frac{\epsilon^3}{8}n$$

genügen. Bestimme mittels O_1 $lsb(am)$ und $lsb(bm)$. Setze $a_0 := a$ und $b_0 := b$.

2. WHILE $t < n$ DO

$t := t + 1$, $a_t := \frac{1}{2}a_{t-1} \bmod n$, $u_t := \frac{1}{2}(u_{t-1} + lsb(a_{t-1}m))$,
 $h := \min\{2^t, 2l\}e^{-2}$.

$A_h := \{i \mid |1 + 2i| \leq h\}$, $w_{t,i} := \lfloor u_t + iu_{t-1} + v \rfloor$ für alle $i \in A_h$.

Mehrheitentscheidung:

$z := \#\{i \in A_h \mid O_1(E_n((a_t + ia_{t-1} + b)m)) = i \cdot lsb(a_{t-1}m) + lsb(bm) - w_{t,i}n \bmod 2\}$

$lsb(a_t m) := 0$ if $z \geq \frac{h}{2}$ und 1 sonst.

END WHILE

3. Ausgabe: $m := a_n^{-1} \cdot \lfloor u_l n + \frac{1}{2} \rfloor \bmod n$

Es bleibt, die Korrektheit des Algorithmus zu zeigen. Weitere wichtige Fragen sind die Erfolgswahrscheinlichkeit sowie die erwartete Laufzeit des Algorithmus. Auf diese Fragen wird nun im folgenden eingegangen.

Korrektheit. Die Korrektheit folgt eigentlich bereits aus der zugrundeliegenden Sampletechnik. Wird $lsb(a_t m)$ in jedem Schritt korrekt bestimmt, so wird in jedem Schleifendurchlauf die Approximation $u_t n$ zu $[a_t m]_n$ um den Faktor 2 verbessert. Für $a_t := \frac{1}{2}a_{t-1} \bmod n$ gilt bereits gemäß unserer Vorüberlegung $[a_t m]_n = \frac{1}{2}[a_{t-1} m]_n$ falls $[a_t m]_n$ gerade ist sowie $[a_t m]_n = \frac{1}{2}([a_{t-1} m]_n + n)$ für ungerade $[a_{t-1} m]_n$. Folglich gilt

$$[a_t m]_n - u_t n = [a_t m]_n - \frac{1}{2}(u_{t-1} + lsb(a_{t-1}m))n = \frac{1}{2}([a_{t-1} m]_n - u_{t-1}n)$$

Durch diese Halbierungen des Abstandes der Approximation muss man folglich bei der zugrundeliegenden geordneten Menge \mathbb{Z}_n in $\log n = l$ Schritten zum Ziel kommen, was die Korrektheit des Algorithmus beweist. □

Erfolgswahrscheinlichkeit. Das bei der *majority decision* berechnete $w_{t,i}$ heißt per Definition genau dann korrekt, wenn gilt $0 \leq [a_t m]_n + i[a_{t-1} m]_n + [bm]_n - w_{t,i}n < n$. Da n als ungerade vorausgesetzt wird, gilt $-w_{t,i}n \equiv w_{t,i} \bmod 2$ und ein korrektes $w_{t,i}$ genügt somit der Gleichung

$$lsb((a_t + ia_{t-1} + b)x) \equiv lsb(a_t m) + i \cdot lsb(a_{t-1}m) + lsb(bm) + w_{t,i} \bmod 2$$

In der *majority decision* wird die linke Seite der Gleichung durch eine Befragung des Orakels ersetzt, also durch $O_1(E_n((a_t + ia_{t-1} + b)x))$, und man bestimmt $lsb(a_t m)$ so, dass die Gleichung in der Mehrheit aller Fälle $i \in A_h$ erfüllt ist. Der Algorithmus ist erfolgreich, wenn die geschätzten Approximationen in Schritt 1 genau genug und alle *majority decisions* für $lsb(a_t m)$ richtig waren. In diesem Fall erhält man $|[a_t m]_n - u_t n| \leq \frac{en}{4 \cdot 2^t} < \frac{1}{2}$ und folglich gilt: $a_t m = \lfloor u_t n + \frac{1}{2} \rfloor \bmod n$, was eine richtige Ausgabe liefert. Es bleibt somit nur noch zu prüfen, mit welcher Wahrscheinlichkeit ein korrektes $w_{t,i}$ bestimmt wird und mit welcher Wahrscheinlichkeit die *majority decision* ein richtiges Ergebnis liefert.

Fehlerwahrscheinlichkeit bei der Bestimmung von $w_{t,i}$:

Sei $w'_{t,i} = u_t + iu_{t-1} + v$, so dass gilt $w_{t,i} = \lfloor w'_{t,i} \rfloor$. Mittels obiger Rekursion erhält man die Formel

$$[a_j m]_n - u_j n = 2^{-j}([am]_n - un) \forall j \leq t$$

die leicht per Induktion bewiesen werden kann. Nach Konstruktion von h gilt weiterhin:

$$2^{-t}\epsilon^2|1+2i| \leq 1$$

für alle $i \in A_h$. Zusammen erhält man somit

$$|[a_t m]_n - i[a_{t-1} m]_n + [b m]_n - w'_{t,i} n| \leq \frac{\epsilon}{8}(2^{-t}\epsilon^2|1+2i| + 1)n \leq \frac{\epsilon}{4}n$$

Folglich ist $w_{t,i}$ genau dann korrekt, wenn es keine natürliche Zahl zwischen $w'_{t,i}n$ und $[a_t m]_n + i[a_{t-1} m]_n + [b m]_n$ gibt. Somit ist $w_{t,i}$ höchstens mit einer Wahrscheinlichkeit von $\frac{\epsilon}{4}$ nicht korrekt. \square

Fehlerwahrscheinlichkeit der majority decision. Die Multiplikatoren $(\frac{1}{2} + i)a + b$ sind paarweise unabhängig für $|i| < \frac{1}{2}\min\{p, q\}$, da für die Matrix $\begin{bmatrix} 1 & \frac{1}{2} + i \\ 1 & \frac{1}{2} + j \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$ der \mathbb{Z}_n -linear Transformationen gilt: $\det \begin{bmatrix} 1 & \frac{1}{2} + i \\ 1 & \frac{1}{2} + j \end{bmatrix} = \frac{1}{2} + j - (\frac{1}{2} + i) = j - i \neq 0 \pmod n$ für alle $i \neq j$. Somit folgt wegen $(a, b) \in_R (\mathbb{Z}_n^*)^2$ die paarweise Unabhängigkeit der betrachteten Multiplikatoren. Mit derselben Argumentation zeigt man nun auch, dass die Fehler der $w_{t,i}$ bei $i \in A_h$ paarweise unabhängig sind, sofern man in der richtigen Alternative ist. Der i -te Durchlauf der *majority decision* ist genau dann korrekt, wenn gilt

$$O_1(E_n((a_t + ia_{t-1} + b)x)) = \text{lsb}(a_t m) + i \cdot \text{lsb}(a_{t-1} m) + \text{lsb}(b m) + w_{t,i} \pmod 2$$

Dies ist der Fall, wenn das Orakel richtig "rät" und $w_{t,i}$ korrekt ist. Um die Fehlerwahrscheinlichkeit des i -ten Durchlaufes zu berechnen, führt man sogenannte Indikatorvariablen X_i ein, die nur die Werte 0 bzw 1 einnehmen können mit

$$E[X_i] = E[O_1(E_n((a_t + ia_{t-1} + b)m)) \neq \text{lsb}((a_t + ia_{t-1} + b)m)] + E[w_{t,i} \text{ ist nicht korrekt}]$$

so dass die Variablen X_i nach der überlegten paarweisen Unabhängigkeit selbst paarweise unabhängig sind für $i \in A_h$. Somit gilt zusammen mit dem Resultat über die Fehlerwahrscheinlichkeit für die Bestimmung von $w_{t,i}$

$$E[X_i] \leq \frac{1}{2} - \frac{3}{4}\epsilon \text{ und } \text{Var}[X_i] \leq \frac{1}{4}$$

Eine *majority decision* wurde mit dem richtigen Ergebnis durchgeführt, wenn die Mehrheit der h Durchläufe das richtige Ergebnis liefert. Eine *majority decision* kann nur dann das falsche Ergebnis liefern, wenn gilt: $\frac{1}{h} \sum_i X_i - \frac{1}{h} \sum_i E[X_i] \geq \frac{3}{4}\epsilon$. Auf die h paarweise unabhängigen Variablen X_i für $i \in A_h$ wird nun die Chebyshevsche Ungleichung angewandt:

$$P\left[\left|\frac{1}{h} \sum_i X_i - \frac{1}{h} \sum_i E[X_i]\right| \geq \frac{3}{4}\epsilon\right] \leq \sum_i \text{Var}[X_i] (h \frac{3}{4}\epsilon)^{-2} = h \cdot \text{Var}[X_i] \frac{16}{9\epsilon^2 h^2} \leq h \cdot \frac{1}{4} \cdot \frac{16}{9\epsilon^2 h^2} = \frac{4}{9h\epsilon^2}$$

Da h durch $h = \min\{2^t, 2l\}\epsilon^{-2}$ definiert ist, liefert die *majority decision* mit der Wahrscheinlichkeit $\frac{4}{2^t \cdot 9}$ für $t \leq 1 + \log l$ bzw. mit $\frac{2}{9l}$ für $t \geq 1 + \log l$ ein falsches Ergebnis. Folglich gilt für die Fehlerwahrscheinlichkeit der Mehrheitsentscheidung für $t = 1, \dots, l$:

$$P(\text{Fehlentscheidung}) = \sum_{t \geq 1} \frac{4}{2^t \cdot 9} + \frac{2l - 2 \log l}{9l} = \frac{4}{9} \sum_{t \geq 1} \left(\frac{1}{2}\right)^t + \frac{2}{9} - \frac{2 \log l}{9l} \leq \frac{4}{9} + \frac{2}{9} = \frac{2}{3}$$

□

Laufzeitanalyse. Es bleibt noch die erwartete Laufzeit des Algorithmus zu untersuchen. Um den Beweis zu vervollständigen, muss eine polynomieller Schranke für die Laufzeit aufgezeigt werden. Wie bereits in der Einführung kurz angesprochen, wird im Folgenden eine Laufzeit von $O(l^2 \epsilon^{-2} T + l^2 \epsilon^{-6})$ bewiesen. Dabei bezeichnet man $l^2 \epsilon^{-2} T$ als den von den Orakelanfragen induzierten Kostenterm, der Term $l^2 \epsilon^{-6}$ wird *additional overhead* genannt. Beide Terme werden nun separat untersucht:

Das Orakel wird im Laufe des Algorithmus nach $E_n((a_t + ia_{t-1} + b)x)$ für $t = 1, \dots, l$ für jedes $i \in A_h$ befragt. Da die Orakelanfrage offensichtlich nur von a, b abhängt, nicht jedoch von $u, v, \text{lsb}(am), \text{lsb}(bm)$, kann man a und b fest wählen und alle Möglichkeiten für $u, v, \text{lsb}(am), \text{lsb}(bm)$ durchprobieren. Da der Algorithmus wie oben bewiesen eine Erfolgswahrscheinlichkeit von $\geq \frac{1}{3}$ besitzt und das Orakel in jedem Schleifendurchlauf höchstens $h \leq 2l \epsilon^{-2}$ mal befragt wird, erhält man insgesamt für die Laufzeit höchstens $3 \cdot 2l^2 \epsilon^{-2} T$, wobei T für die Laufzeit des Orakels steht.

Man kommt nun zum *additional overhead*:

Jede *majority decision* trägt zum *additional overhead* höchstens $2l \epsilon^{-2}$ Schritte bei, die anhand der gegebenen Orakelantworten durchgeführt werden. Der Algorithmus selbst braucht die exakte rationale Zahl $u_t + v$ nicht zu berechnen und berechnet somit $w_{t,i} = \lfloor u_t + v + iu_{t-1} \rfloor$ anhand von $\log(l \cdot \epsilon^{-1}) + O(1)$ Bits von $u_t + v$ und iu_{t-1} . Der *additional overhead* setzt sich also höchstens aus dem Produkt der folgenden Faktoren zusammen:

1. Anzahl der Quadrupel $(u, v, \text{lsb}(am), \text{lsb}(bm))$: $4^2 \epsilon^{-4} 2^2$.
2. Anzahl der Schleifendurchläufe : l .
3. Anzahl der Schritte der *majority decision* : $2l \epsilon^{-2}$.
4. Das Inverse der Erfolgswahrscheinlichkeit : 3 .

Man erhält als *additional overhead* somit höchstens $3 \cdot 2^7 l^2 \epsilon^{-6}$. Insgesamt erhält man eine erwartete Laufzeit von $2l^2 \epsilon^{-2} (2T + 2^7 \epsilon^{-4}) \in \text{poly}(l)$.

□

In [FiSc97] werden weiterführende Methoden gezeigt, die die erwartete Laufzeit anhand einer sogenannten *subsample majority decision* auf $O(l^2 \epsilon^{-2} T + l^2 \epsilon^{-4} \log(l \epsilon^{-1}))$ herunterdrücken.

6.3.3 Die Sicherheit der $O(\log l)$ niederwertigsten Bits

Die Sicherheit der $O(\log l)$ niederwertigsten Bits kann auf die gleiche Art und Weise wie die Sicherheit des niederwertigsten Bits gezeigt werden. Es handelt sich hierbei nur um eine einfache Modifikation des obigen Beweises, so dass darauf verzichtet wird, die einzelnen Schritte näher zu erläutern. Eine kurze Herleitung der Modifikationen sowie der abschließende Beweis sind z.B. in [FiSc97] zu finden. Ein alternativer, bereits im Jahre 1986 durchgeführter Beweis zu der betrachteten Bitsicherheit, kann in [Pera86] nachgelesen werden.

6.3.4 Die Sicherheit der mittleren RSA-Bits

In diesem Abschnitt wird die Bitstelle i durch die Ungleichungen

$$9\log\epsilon(l)^{-1} + 2\log l + 67 \leq i \leq l - 16\log\epsilon(l)^{-1} - 3\log l - 109$$

eingeschränkt. Der Leser sollte sich durch diese auf den ersten Blick sehr merkwürdige Abschätzung nicht verunsichern lassen, die Kernaussage besagt lediglich, dass sowohl in Richtung der höherwertigen Bits als auch in Richtung der niederwertigen Bits eine logarithmische Anzahl von nicht zu beachtenden Bitstellen liegt. Die genau Art der Abschätzung wird in der folgenden Beweisen benötigt, ist aber im Grunde nicht weiter von Interesse.

Man beginnt wieder mit einigen einführenden Definitionen:

Definition 6.7. Ein Intervall J ist eine Menge von aufeinanderfolgenden Werten $J = \{[u]_n, [u+1]_n, \dots, [v]_n\}$ in \mathbb{Z}_n . Die Länge des Intervalls wird durch $\#J$ bezeichnet. Ist J ein Intervall und $z \in \mathbb{Z}$, so wird die Translation von J um z definiert als $J + z := \{[y + z]_n | y \in J\}$.

Für eine Wahrscheinlichkeitsverteilung D auf $J \subset \mathbb{Z}_n$ sei $P_D^{\mathfrak{D}}(J)$ die erwartete Anzahl der 1-Antworten, die das Orakel \mathfrak{D} auf D gibt, formal:

$$P_D^{\mathfrak{D}}(J) := E_{z \in_D J}[\mathfrak{D}(E_n(z))] = P_{z \in_D J}[\mathfrak{D}(E_n(z)) = 1]$$

Ist D die uniforme Wahrscheinlichkeitsverteilung, so wird sie im Folgenden in der Notation weggelassen. Desweiteren definiert man für $J_1, J_2 \subset \mathbb{Z}_n$: $\Delta^{\mathfrak{D}}(J_1, J_2) := |P^{\mathfrak{D}}(J_1) - P^{\mathfrak{D}}(J_2)|$.

Um nun zu zeigen, dass jedes innere RSA-Bit i sicher ist, zeigt man, dass man mittels eines Orakels O_i für das i -te Bit ein *lsb*-Orakel konstruieren kann. Die Konstruktion basiert auf der Theorie der Unterscheidbarkeit von Wahrscheinlichkeitsverteilungen auf Intervallen; genauer bedeutet dies, dass man J_1 und J_2 unterscheiden kann, wenn die Bedingung $\Delta^{\mathfrak{D}}(J_1, J_2) \geq \epsilon(l)$ für ein nicht vernachlässigbares $\epsilon(l)$ gilt. Diese Eigenschaft könnte man dann wie folgt ausnutzen: Angenommen man hätte ein Orakel für das i -te Bit, so dass für ein nicht zu kleines Intervall J gilt:

$$\Delta^{\mathfrak{D}}(J, J + \frac{n+1}{2}) \geq \epsilon(l)$$

d.h. dass das Orakel o.B.d.A. für einen gegebenen Schlüsseltext $c \equiv x^e \pmod n$ (bzw. kurz $E_n(x)$) mit größerer Wahrscheinlichkeit "1" antwortet für $x \in J$ als für $x \in J + \frac{n+1}{2}$. Befragt man nun dieses Orakel nach $E_n([2^{-1}x]_n)$, so erhält man unter Benutzung der weiter oben vorgestellten Sampletechnik

$$[2^{-1}x]_n = \frac{x - \text{lsb}(x)}{2} + \text{lsb}(x)[2^{-1}]_n = \frac{x - \text{lsb}(x)}{2} + \text{lsb}(x)\frac{n+1}{2}$$

Falls nun gilt $\frac{x - \text{lsb}(x)}{2} \in J$, so folgt $[2^{-1}x]_n \in J + \text{lsb}(x)\frac{n+1}{2}$. Da sich das Orakel nach Voraussetzung auf den beiden Intervallen unterschiedlich verhält, ist es somit möglich, das niederwertigste Bit von x zu bestimmen. Dabei bleiben jedoch zwei Fragen offen:

1. Es ist unklar, wie man J wählen muss, damit $\frac{x - \text{lsb}(x)}{2} \in J$ gilt
2. Die Existenz eines solchen Intervalls J ist noch nicht bewiesen.

Anhand dieser im groben vorgestellten Intuition wurde in [HaNa98] die Sicherheit der inneren Bits von RSA bewiesen. Der folgende Beweis wurde bis auf erklärende Kommentare dem besagten Paper entnommen.

Die eigentliche Inversion einer mittels RSA verschlüsselten Nachricht verläuft nun informell beschrieben auf folgende Art und Weise:

Man wählt $a \in \mathbb{Z}_n$ beliebig aber fest. Hat man nun eine Menge R von zufälligen Werten, d.h. $R = \{r_j | j \in I\} \subset \mathbb{Z}_n$, wobei I eine beliebige Indexmenge bezeichnet, so kann man das Orakel nach den i -ten Bits von

$$R' := \{E_n^{-1}[(r_j + a) \cdot x]_n | r_j \in R\}$$

befragen. Gilt nun $\text{bit}_i(x) = 0$, so induziert R' eine Wahrscheinlichkeitsverteilung D_0 auf \mathbb{Z}_n , ansonsten eine Verteilung D_1 . Sind diese Verteilungen in polynomieller Zeit unterscheidbar, so kann man durch genügend Zufallswerte r_j die Bits von x entscheiden.

Die beiden Verteilungen D_0, D_1 betrachtet man auf zwei Teilmengen von \mathbb{Z}_n , von denen man hofft, das sie zur Unterscheidung des niederwertigsten Bits beitragen können; Beispiele sind z.B. die weiter oben erwähnten Teilmengen J und $J + \frac{n+1}{2}$. Um darauf die vorgestellte Sampletechnik anwenden zu können, muss man zumindest die ungefähre Position der Samplepunkte wissen. Damit befasst sich das folgende Lemma.

Lemma 6.8. *Seien $m(l) \in \text{poly}(l)$, $d_I(l), d_Y(l) \in O(\log l)$. Dann ist es möglich, zu einem gegebenen Schlüsseltext $E_n(x)$ und $r, s \in_U \mathbb{Z}_n$ deterministisch in polynomieller Zeit eine Liste von $m(l)$ Werten der Form $E_n(r_j x)$ zu erzeugen, so dass $[r_j x]_n$ einheitlich verteilt und die Werte in $\{[r_j x]_n\}$ paarweise unabhängig sind. Weiter generiert man eine Menge von $2^{4+2(d_I(l)+d_Y(l))} m(l)^2$ Paaren von Listen $\{L^I, L^J\}$, so dass L^I aus $m(l)$ Werten aus $\mathbb{Z}_{2^{i+1}}$ und L^J aus $m(l)$ Werten aus \mathbb{Z} besteht.*

Für mindestens ein $(L', L'') \in \{(L^I, L^Y)\}$, für jedes $j = 1, \dots, m(l)$, für ein z_j mit $[z_j]_n = [r_j x]_l$ gilt

$$|z_j - L''_j| \leq \frac{n}{2^{d_Y(l)}}$$

und

$$\text{abs}_{2^{i+1}}(z_j - L'_j) \leq 2^{i+1-d_I(l)}.$$

Beweis. Der Beweis dazu ist in [HaNa98] zu finden. □

Offensichtlich liefert die erste Gleichung uns, dass jedes $[r_j x]_n$ $\frac{n}{2^{d_Y(l)}}$ -determiniert ist.

Man kommt nun zum eigentlichen Problem, zwei Teilmengen von \mathbb{Z}_n anhand der Orakelausgaben zu unterscheiden. Dazu muss man sich zuerst überlegen, wie sich das Orakel in beiden Fällen verhalten soll. Man betrachtet zuerst folgendes

Lemma 6.9. *Sei $J \subset \mathbb{Z}_n$, wobei $\frac{\#J}{n}$ nicht vernachlässigbar sei, so dass man in deterministisch polynomieller Zeit testen kann, ob ein gegebenes $x \in \mathbb{Z}_n$ Element von J ist. Dann kann man für jedes nicht vernachlässigbare $\epsilon'(l)$ und $K(l) \in \text{poly}(l)$ in probabilistisch polynomieller Zeit einen Wert \tilde{p} berechnen, so dass gilt:*

$$P(|P^{\mathcal{D}}(J) - \tilde{p}| \geq \epsilon'(l)) \leq \frac{1}{K(l)}$$

Beweis. Sei $m'(l) = \epsilon'(l)^{-2} \ln(4l \cdot K(l))$ und setze $m(l) = 4\lambda(J)^{-1}m'(l)$. Wähle zufällige und unabhängige $x_1, \dots, x_{m(l)} \in \mathbb{Z}_n$. Für jedes x_j mit $x_j \in J$ befrage das Orakel nach $E_n(x_j)$ und berechne \tilde{p} als den Anteil aller 1-Antworten, die das Orakel ausgibt. Zwei Anwendungen von *Chernoff* vervollständigen nun das Lemma: Die erste beschränkt die Wahrscheinlichkeit, dass $\#(\{x_j\} \cap J)$ klein ist, die zweite beschränkt die Wahrscheinlichkeit, dass sich \tilde{p} zu stark von dem erwarteten Wert $P^{\mathcal{D}}(J)$ unterscheidet. \square

Das folgende Lemma beschäftigt sich nun mit der Konstruktion eines lsb-Orakels \mathcal{D}' aus einem Orakel \mathcal{D} des i -ten Bits.

Lemma 6.10. *Sei \mathcal{D} ein i -Bit Orakel, so dass für ein Intervall J gilt: $\Delta^{\mathcal{D}}(J, J + \frac{n+1}{2}) \geq \epsilon'(l)$, wobei $\lambda(J), \epsilon'(l)$ nicht vernachlässigbar sind. Dann kann man in polynomieller Zeit ein Orakel \mathcal{D}' konstruieren, so dass für alle $\frac{\lambda(J)\epsilon'(l)}{512l}$ -determinierten $[ax]_n$ gilt: \mathcal{D}' bestimmt lsb($[ax]_n$) mit Wahrscheinlichkeit $\geq 1 - \frac{1}{2l}$.*

Ich beginne mit einer informellen Beschreibung des Beweises. Unter Benutzung von Lemma 6.8 erhält man eine Menge von zufälligen, paarweise verschiedenen Zahlen der Form $\{[r_j x]_n\}$, für die man ihre ungefähre Position in \mathbb{Z}_n kennt, d.h. man kennt für alle j ein L_j so dass gilt: $\text{abs}(r_j x - L_j)$ ist klein.

Angenommen es gilt nun $\text{lsb}([ax]_n) = 0$. Ist diese Hypothese richtig, so kann man, da $[ax]_n$ $\frac{\lambda(J)\epsilon'(l)}{512l}$ -determiniert ist und da man gute Approximationen der Zahlen $[r_j x]_n$ besitzt, mit Sicherheit sagen, ob die Zahlen $[2^{-1}ax + r_j x]_n = [(2^{-1}a + r_j)x]_n$ sich im Intervall J befinden oder nicht. Im ersten Fall befragt man das Orakel nach diesem Wert, im zweiten Fall verwirft man die Zahl $r_j x$ und fährt mit dem nächsten $r_{j'} x$ fort. Ist unsere Hypothese hingegen falsch, also $\text{lsb}([ax]_n) = 1$, so verfährt man analog zur obigen Beschreibung auf dem Intervall $J + \frac{n+1}{2}$. In jeden Fall kann man anhand der Antworten des Orakels die beiden Fälle und somit die beiden Intervalls unterscheiden.

Wir sind nun auf den formalen Beweis vorbereitet:

Beweis. Anhand Lemma 6.9 kann man sich Approximationen \tilde{p}_0, \tilde{p}_1 zu $P^{\mathcal{D}}(J), P^{\mathcal{D}}(J + \frac{n+1}{2})$ berechnen mit maximalem Abstand $\leq \frac{\epsilon'(l)}{4}$. Dies kann mit Wahrscheinlichkeit $\geq 1 - \frac{1}{4l}$ erreicht werden und man nimmt der Einfachheit halber $\tilde{p}_1 > \tilde{p}_0$ an.

Desweiteren kann man mittels Lemma 6.8 eine Menge R' konstruieren mit

$$m(l) = 512\lambda(J)^{-1}l\epsilon'(l)^{-2}$$

paarweise unabhängigen, einheitlich verteilten Werten der Form $r_j x$, wobei bekannt ist, dass jedes $[r_j x]_n$ innerhalb von $2^{-d(l)}n$ für $d(l) = 9 + \log \epsilon'(l)^{-1} + \log \lambda(J)^{-1} + \log l$ liegt. Offensichtlich gibt es nur polynomiell viele Möglichkeiten in l , die dieser Einschränkung genügen. Somit kann man einfach alle Möglichkeiten durchprobieren, also für jede Möglichkeit die folgenden Schritte durchführen. Im weiteren Beweis beschränkt man sich der Einfachheit halber auf den richtigen Wert.

Betrachte nun die Menge

$$R = \{[2^{-1}a + r_j)x]_n \mid [r_j x]_n \in R'\}$$

Angenommen es gilt $lsb([ax]_n) = 0$, dann kann man zu jedem j ein a_j berechnen, so dass $[a_j - (2^{-1}a + r_j)x]_n \frac{\lambda(J)\epsilon'(l)}{256l}$ -klein ist. Ist nun $a_j \in J$, so entscheidet man, dass gilt: $[(2^{-1}a + r_j)x]_n \in J$, im anderen Fall, dass gilt $[(2^{-1}a + r_j)x]_n \notin J$ und man entfernt es aus R .

Die Frage ist nun, mit welcher Wahrscheinlichkeit der letzte durchgeführte Schritt falsch ist. Dazu benötigt man folgende Definition.

Definition 6.11 (missklassifiziert). *Ist $lsb([ax]_n) = 0$ und $[(2^{-1}a + r_j)x]_n \in J$, aber es gilt $[(2^{-1}a + r_j)x]_n \notin R$ (bzw. andersherum), so nennt man $(2^{-1}a + r_j)m$ missklassifiziert. Die selbe Notation wird im Fall $lsb([ax]_n) = 1$ und J ersetzt durch $J + \frac{n+1}{2}$ benutzt.*

Es gilt weiterhin das folgende Lemma.

Lemma 6.12. *Nicht zu viele Punkte sind missklassifiziert, genauer: Die erwartete Anzahl der missklassifizierten Punkte wird noch oben durch*

$$\frac{m(l)\epsilon'(l)\lambda(J)}{64l}$$

beschränkt.

Beweis. Da die betrachteten Punkte nach Voraussetzung $\frac{\epsilon'(l)\lambda(J)}{256l}$ -determiniert sind, können die missklassifizierten Punkte nur solche Punkte sein, deren Abstand vom Rand des Intervalls höchstens $\frac{\epsilon'(l)\lambda(J)}{256l}$ beträgt. Da die Punkte einheitlich verteilt sind, beträgt die erwartete Anzahl solcher Punkte $\frac{m(l)\epsilon'(l)\lambda(J)}{64l}$. □

Befrage nun das Orakel \mathfrak{D} über alle Punkte aus R . Falls die Anzahl der 1-Antworten mindestens

$$\frac{m(l)\lambda(J)(\tilde{p}_0 + \tilde{p}_1)}{2}$$

betragen, nimm an, dass gilt: $lsb([ax]_n) = 1$, ansonsten nimm an $lsb([ax]_n) = 0$.

Mit welcher Wahrscheinlichkeit führt dieses Vorgehen nun zu einem richtigen Ergebnis?

Sei o.B.d.A. $lsb([ax]_n) = 0$. Angenommen, alle Punkte sind korrekt klassifiziert worden, d.h. es gibt keine missklassifizierten Punkte. In diesem Fall sind alle Punkte einheitlich verteilt und paarweise unabhängig. Die erwartete Anzahl der Punkte in R mit der Antwort "1" ist $P^{\mathfrak{D}}(J)\lambda(J)m(l)$, die Varianz dieser Anzahl ist höchstens $P^{\mathfrak{D}}(J)\lambda(J)m(l)$. Die Wahrscheinlichkeit, dass mehr als $\lambda(J)m(l)(P^{\mathfrak{D}}(J) + \frac{\epsilon'(l)}{8})$ Punkte in die Menge R mit Antwort "1" genommen werden, ist wiederum durch die Chebyshevsche Ungleichung beschränkt durch

$$\frac{64\lambda(J)P^{\mathfrak{D}}(J)m(l)}{\epsilon'(l)^2\lambda(J)^2m(l)^2} \leq \frac{64}{\epsilon'(l)\lambda(J)m(l)} \leq \frac{1}{8l}$$

wobei die letzte Ungleichung aus der Definition von $m(l)$ folgt. Solange also nicht mehr als $\frac{\lambda(J)\epsilon'(l)m(l)}{8}$ Punkte missklassifiziert sind, ist die Anzahl der "1"-Antworten des Orakels durch $\lambda(J)m(l)(P^{\mathfrak{D}}(J) + \frac{\epsilon'(l)}{4})$ beschränkt. Wegen der Annahme $\tilde{p}_0 \geq P^{\mathfrak{D}}(J) - \frac{\epsilon'(l)}{4}$ und $\tilde{p}_1 \geq P^{\mathfrak{D}}(J + \frac{n+1}{2}) - \frac{\epsilon'(l)}{4} \geq P^{\mathfrak{D}}(J) + \frac{3\epsilon'(l)}{4}$ gilt somit $P^{\mathfrak{D}}(J) + \frac{\epsilon'(l)}{4} \leq \frac{\tilde{p}_0 + \tilde{p}_1}{2}$ und der Algorithmus liefert die richtige Ausgabe.

Laut Lemma 6.12 ist die Wahrscheinlichkeit, mehr als $\frac{\lambda(J)m(l)\epsilon'(l)}{8}$ missklassifizierte Punkte zu haben durch $\frac{1}{8l}$ nach oben beschränkt, womit die Aussage des zu beweisenden Lemmas 6.10 folgt. \square

Zusammenfassend kann man also sagen, dass es möglich ist, aus einem Orakel für das i -te Bit ein Orakel für das niederwertigste Bit zu konstruieren, sofern ein passendes, konstruierbares Intervall J existiert, das die obigen Bedingungen erfüllt. Dieser Sachverhalt wird in folgendem finalen Lemma festgehalten.

Lemma 6.13. *Falls ein i -Bit Orakel O_1 gegeben ist, so dass für ein Intervall J gilt $\Delta^D(J, J + \frac{n+1}{2}) \geq \epsilon'(l)$, wobei $\lambda(J), \epsilon'(l)$ nicht vernachlässigbar sind, so kann man in polynomieller Zeit die zugrundeliegende Nachricht m mit einer Wahrscheinlichkeit von $\frac{1}{2}$ rekonstruieren.*

Dies ist klar nach dem oben bewiesenen Lemma und den Betrachtungen der Sicherheit des niederwertigsten RSA-Bits.

Die kritische Frage der Existenz und der Konstruierbarkeit des Intervalls J bleibt jedoch bestehen. Diese Frage erweist sich als zu komplex, um sie in dieser Arbeit in Gänze zu behandeln. Aus diesem Grund beschränke ich mich darauf, die Ergebnisse aus [HaNa98] aufzuzeigen:

- Entweder es existiert ein konstruierbares Intervall J , das den Anforderungen von Lemma 6.10 genügt oder
- es ist möglich mittels eines wesentlich komplizierteren Algorithmus die Intervallsuche auf eine "Tupel"-Suche im zweidimensionalen Raum zu erweitern. In diesem Fall wird das betrachtete Intervall durch ein Rechteck, eine sogenannte Box, mit ähnlichen Eigenschaften ersetzt. Eine solche Box kann, sofern das gewünschte Intervall J nicht existiert, immer konstruiert werden.

Diese beiden Fälle komplettieren den Beweis der Sicherheit der inneren RSA-Bits.

6.3.5 Die Sicherheit der $O(\log l)$ höherwertigsten Bits

Wie in Abschnitt (6.2) bereits vorweg genommen wurde, ist es möglich, die $O(\log l)$ höherwertigsten Bits mit einer Wahrscheinlichkeit vorherzusagen, die signifikant größer als $\frac{1}{2}$ ist. Demzufolge muss zuerst neu definiert werden, was man unter $\epsilon(l)$ -Sicherheit des i -ten Bits versteht, sofern das i -te Bit eines der höherwertigsten Bits ist. Schrift und Shamir gaben die korrekte Definition der Unvorhersagbarkeit von sogenannten *biased functions* [ScSh91].

Definition 6.14. *Sei p ein nicht-konstantes Prädikat. Ein Orakel \mathcal{D} sagt p mit Vorteil $\epsilon(l)$ voraus, wenn gilt*

$$|P[\mathcal{D}(E_n(m)) = 1 | p(m) = 1] - P[\mathcal{D}(E_n(m)) = 1 | p(m) = 0]| \geq \epsilon(l)$$

Ein Prädikat heißt $\epsilon(l)$ -sicher, wenn keine ppTm existiert mit Vorteil $\epsilon(l)$ und es heißt unvorhersagbar, wenn es $\epsilon(l)$ -sicher ist für alle nicht vernachlässigbaren $\epsilon(l)$.

Um den Beweis führen zu können, ist es notwendig, das in (6.3.6) gezeigte Ergebnis bzgl. der simultanen Bitsicherheit der RSA-Annahme vorwegzunehmen. Dort wird gezeigt, dass Blöcke von $O(\log l)$ Bits simultan sicher sind, d.h. dass sie polynomiell nicht von einer gleichlangen Zufallsbitfolge unterschieden werden können. Dies gilt insbesondere für die $t(l) \in O(\log l)$ niederwertigsten Bits. Offensichtlich folgt daraus weiter, dass es unmöglich ist, diese Bits mit

einer nicht vernachlässigbaren Wahrscheinlichkeit vorauszusagen. Im folgenden wird gezeigt, wie man mit Hilfe eines $\epsilon(l)$ -Orakels für das i -te Bit, wobei i nach Voraussetzung der Bedingung $i = l - O(\log l)$ genügen soll, einen Algorithmus konstruieren kann, der für ein $t(l) \in O(\log l)$ die Bits $B_0^{t(l)-1}(x)$ mit einer Wahrscheinlichkeit $2^{-t(l)} + \epsilon'(l)$ für ein nicht vernachlässigbares $\epsilon'(l)$ voraussagen kann. Da die triviale Ratewahrscheinlichkeit für die $t(l)$ Bits genau $2^{-t(l)}$ beträgt, hätte man somit einen Widerspruch zu dem in (6.3.6) gezeigten Resultat, was den Sicherheitsbeweis der höherwertigsten Bits vervollständigt.

Man beginnt mit einer Vorüberlegung zum Beweis, der den *bias* der betrachteten Bits außer Acht lässt. Das Orakel \mathfrak{D} befragt man nach $\text{bit}_i(E_n^{-1}[2^{-t}x]_n)$, wobei $t = l - i + t_0$ mit $t_0 \in \theta(\log l)$ zu wählen ist. Damit ist $t \in O(\log l)$, wie gewünscht. Gemäß der bereits mehrfach benutzten Sampletechnik gilt für $[2^{-t}x]_n$:

$$[2^{-t}x]_n = \frac{x - B_0^{t-1}(x)}{2^t} + B_0^{t-1}(x)[2^{-t}]_n$$

Der Term $\frac{x - B_0^{t-1}(x)}{2^t}$ ist klein, genauer:

$$\frac{x - B_0^{t-1}(x)}{2^t} \leq \frac{n}{2^t} \leq 2^{i-t_0}.$$

Folglich gilt mit Wahrscheinlichkeit $1 - 2^{-t_0}$: $\text{bit}_i(B_0^{t-1}(x)[2^{-t}]_n) = \text{bit}_i([2^{-t}x]_n)$. Dies bedeutet also, dass aus den a priori vorhandenen 2^t Möglichkeiten für $B_0^{t-1}(x)$ unter Berücksichtigung der Orakelinformation nur noch ungefähr 2^{t-1} Möglichkeiten übrigbleiben. Somit hätte man einen Algorithmus, der eine um den Faktor 2 größere Erfolgswahrscheinlichkeit verspricht als eine triviale Ratestrategie. Kommen wir nun zum formalen Beweis.

Der folgende Algorithmus basiert auf der oben angeführten Vorüberlegung, wobei noch der *bias* des i -ten Bits einbezogen wurde. Im Anschluss an die Beschreibung des Algorithmus wird seine Erfolgswahrscheinlichkeit untersucht.

Eingabe: $E_n(x)$, $|n| = l$.

Ausgabe: $B_0^{t(l)-1}(x)$ für ein $t(l) = l - i + t_0(l) \in O(\log l)$.

1. $b \leftarrow \mathfrak{D}(E_n([2^{-t(l)}x]_n))$.
2. $J \leftarrow \{j \mid 0 \leq j < 2^{t(l)} \wedge \exists z : 0 \leq z \leq 2^{i-t_0(l)} \text{ so dass gilt : } \text{bit}_i([j2^{-t(l)} + z]_n) = b\}$.
3. Wähle $j \in_U J$.
4. Gebe j zurück.

Da in allen Fällen gelten muss $t(l) \in O(\log l)$ mit $t_0(l) \geq 1$, ist die Laufzeit des Algorithmus aus folgenden Grund immer polynomiell:

Für jedes j , $0 \leq j < 2^{t(l)}$ braucht man nur den Fall $z = 0$ und $z = 2^{i-t_0(l)}$ zu betrachten, um die Menge J zu bestimmen. Das nun folgende Lemma untersucht die Erfolgswahrscheinlichkeit des Algorithmus.

Lemma 6.15. *Angenommen, das Orakel \mathfrak{D} erfüllt die Anforderungen der Definition für biased functions und der bias des i -ten Bits ist durch $\beta_i(n)$, $\beta_i(n) \leq 1 - \delta(l)$ nach oben beschränkt, wobei $\delta(l)$ nicht vernachlässigbar sein soll. Dann gibt der obige Algorithmus für $t(l) = l - i + t_0(l)$ mit $t_0(l) \geq \log_e(l)^{-1} + \log \delta(l)^{-1} + 3$ die Bits $B_0^{t(l)-1}(x)$ mit einer Wahrscheinlichkeit $\geq 2^{-t(l)}(1 + \frac{\epsilon(l)}{2})$ richtig aus.*

Ein Beweis hierzu ist in [HaNa98] zu finden. Somit ist die Sicherheit aller Bits eines RSA-Schlüsseltextes gezeigt, was den Beweis für Theorem 3 liefert.

6.3.6 Simultane Bitsicherheit

Bereits im vorherigen Abschnitt wurde die simultane Bitsicherheit der RSA Annahme benutzt. Dies wird in folgenden Theorem festgehalten:

Theorem 4. *Sei $d(l) \in O(\log l)$. Dann ist jeder Block von $d(l)$ konsekutiven Bits von m simultan sicher oder RSA kann in polynomieller Zeit gebrochen werden.*

Da für den Beweis wieder eine grössere Vorarbeit geleistet sowie weitere Notation eingeführt werden müsste, wird auf die Darstellung des Beweises verzichtet; er ist in [HaNa98] zu finden.

6.3.7 Bitsicherheit der abgeleiteten Annahmen

Unter Berücksichtigung der oben bewiesenen Sicherheit des RSA-Problems ist es nun ein Leichtes, die Bitsicherheit der abgeleiteten Annahmen zu bestimmen.

Satz 6.16 (Bitsicherheit der Strong RSA Assumption). *Jedes Bit einer Nachricht m ist bzgl. der Strong RSA Assumption $\epsilon(l)$ -sicher für alle nicht vernachlässigbaren $\epsilon(l)$, sofern einmalig ein festes e gewählt wird, oder RSA kann in polynomieller Zeit gebrochen werden.*

Dies folgt unmittelbar aus der Tatsache, dass sich die *Strong RSA Assumption* polynomiell auf das RSA Problem zurückführen lässt. Der Angreifer wählt einmalig ein e fest und lässt sich anhand des Orakels Bits der Nachricht m ausgeben. Somit folgt aus der Sicherheit der RSA-Bits die Sicherheit der *Strong RSA* Bits unter der Voraussetzung, dass der Exponent e nur einmalig und fest gewählt wird und sich die Ausgabe des Orakels auf die Bits der Nachricht m beschränkt. Negiert man obige Aussage, erhält man die gewünschte, folgende Aussage: Wäre nun also ein Bit der *Strong RSA Assumption* nicht sicher, so könnte man aufgrund der bisherigen Überlegungen RSA invertieren, was insbesondere die *Strong RSA Assumption* brechen würde.

Satz 6.17 (Bitsicherheit der Dependent RSA Assumption). *Jedes Bit der im Dependent RSA Problem gesuchten Nachricht $(m+1)^e$ ist für feste e bzgl. der Dependent RSA Assumption $\epsilon(l)$ -sicher für alle nicht vernachlässigbaren $\epsilon(l)$, oder RSA kann in polynomieller Zeit gebrochen werden.*

Der Beweis dieses Satzes beruht auf der in Kapitel 5 bewiesenen Äquivalenz der beiden Probleme für feste Exponenten e . Wäre nun ein Bit bzgl. der *Dependent RSA Assumption* nicht sicher, so könnte man die *Dependent RSA Assumption* brechen und folglich auch die RSA Annahme, was den gewünschten Widerspruch liefert.

6.4 Bitsicherheit der Diskreter Log. Annahme

Wie bereits in der Einleitung angekündigt, wird sich beim Diskreten Log. darauf geschränkt, die bekannten Ergebnisse aufzuzeigen. Sei p eine l -Bit Primzahl. Dann gilt:

- Gilt $p - 1 = p'2^k$ für ein ungerades k , so können die k niederwertigsten Bits durch den Pohlig-Hellman-Algorithmus effizient berechnet werden. Da p als ungerade vorausgesetzt wurde, ist somit das niederwertigste Bit immer berechenbar.

- Die an die k niederwertigsten Bits folgenden $O(\log l)$ Bits sind sicher [Pera86].
- Die höherwertigsten $O(\log l)$ Bits sind sicher [LoWi88].
- Unter der Voraussetzung, dass die Exponentiation modulo p eine Einwegfunktion darstellt, wurde in [PaSu98] gezeigt, dass fast alle Bits simultan sicher sind.

Betrachtet man den diskreten Logarithmus modulo einer zusammengesetzten Zahl $n = pq$, so weiss man:

- Alle Bits der diskreten Logarithmus Annahme sind sicher [HaSS82].
- $\frac{l}{2}$ -Bits sind simultan sicher [HaSS82].

Kapitel 7

Partielle Bitsicherheit der einzelnen Annahmen

7.1 Einleitung

In diesem Kapitel wird die Sicherheit der Annahmen gegenüber *einmaligem* Gewinn partieller Informationen untersucht, d.h. im Gegensatz zum vorigen Kapitel hat der Angreifer kein Orakel zur Hand, das er beliebig oft benutzen kann. Er besitzt lediglich eine einmalige Information über ein Geheimnis, das ihm zum Brechen der betrachteten Annahme genügen muss. Diese Art der Sicherheit ist für die Praxis von sehr großer Bedeutung, da es anhand von sogenannten Timing-Angriffen oftmals möglich ist, eine bestimmte Anzahl von Bits des geheimen Schlüssels zu erfahren. Desweiteren erweist es sich in der Praxis oft als einfach, einmalig eine größere Menge an partiellen Informationen zu beschaffen, was hauptsächlich durch die Unvorsichtigkeit der Benutzer begünstigt wird. Im Gegensatz dazu ist die Orakelbitsicherheit einer Annahme stärker der Theorie zugeordnet, da die Existenz eines Orakels für ein Hardcorebit wohl sofort publik gemacht werden würde. Als Folge davon würde die zugrundeliegende Annahme in Zukunft gemieden, was wiederum eine Bedrohung in der Praxis ausschließt. Im Folgenden werden die Faktorisierungsannahme sowie die RSA-Annahme auf diese Art der Bitsicherheit untersucht.

7.2 Bitsicherheit der Faktorisierungsannahme

Bei der Faktorisierungsannahme bezieht sich der Gewinn partieller Informationen auf die Bits der geheimen Primzahlen p und q , d.h. wie viele Bits von p und q muss der Angreifer zu Verfügung haben, um p und q vollständig zu rekonstruieren, d.h. n zu faktorisieren?

Das erste Resultat hierzu wurde im Jahre 1985 von R. L. Rivest und A. Shamir veröffentlicht [RiSh85]. Das bis heute beste Resultat stammt von D. Coppersmith [Copp97]:

Theorem 5 (Coppersmith). *Kennt man zu einer gegebenen Zahl $n = pq$ die $(\frac{1}{4}\log n)$ höherwertigsten Bits von p , dann ist es in polynomieller Zeit möglich, die Faktoren p und q zu rekonstruieren, d.h. die Zahl n zu faktorisieren.*

Es stellt sich bei diesem Problem jedoch die Frage, inwiefern ein solcher Gewinn an partiellen Informationen in der Praxis wirklich möglich ist. Wie bereits in der Einleitung erwähnt ist es oftmals möglich, einzelne Bits des geheimen Schlüssels zu erhalten. Ein Beispiel liefern z.B. Systeme, die per Konstruktion einzelne Bits der Primfaktoren verraten, um z.B. den "Eigentümer"

des berechneten Modulus n festzulegen. Vanstone und Zuccherato haben z.B. in [VaZu95] ein sogenanntes *identitätsbasierendes* System vorgeschlagen, bei dem der Modulus n von der Identität des Benutzers abhängt, z.B. indem in die höherwertigsten Bits von n der Name des Schlüsselerzeugers hineinkodiert wurde. Dieses Kryptosystem erfüllt die Voraussetzungen für Theorem 5 und wurde somit gebrochen.

Obwohl die Bitsicherheit dieses Problems von eher geringer praktischer Bedeutung erscheint, entsteht jedoch eine wichtige Anwendung bei der im nächsten Abschnitt behandelten Bitsicherheit des RSA-Problems. Im Folgenden beschränke ich mich jedoch darauf, die zugrundeliegenden Ideen zu beschreiben.

Der Beweis von Theorem 5 wird durch Nullstellenbestimmung von *bivariaten* ganzzahligen Gleichungen bestimmt. Es wird eine Möglichkeit vorgestellt, wie man polynomielle Gleichungen in zwei Variablen der Form $p(x, y) = 0$ über \mathbb{Z} löst unter der Voraussetzung, dass die Lösung durch Konstanten X und Y beschränkt ist, wobei X und Y von den Koeffizienten und dem Grad des Polynoms p abhängen. Der Algorithmus selbst benutzt sogenannte Gitterbasismethoden [LeLL82] und ist in [Copp97] zu finden. Hat man nun eine Zahl $n = pq$ gegeben und besitzt man die x höherwertigsten Bits P_0 von p , so berechnet man sich zuerst die höherwertigsten Bits Q_0 von q , indem man n durch $P_0 \cdot 2^x$ teilt. Kann man nun die Gleichung $(P_0 + x)(Q_0 + y) - n = 0$ lösen, so hat man n vollständig faktorisiert. Copperfield hat gezeigt, dass zum Lösen dieser Gleichung die $\frac{1}{4} \log n$ höherwertigsten Bits von p genügen, was zum behaupteten Theorem führt.

Die Konstruktion sowie der Beweis des Algorithmus verlangen ein grundlegendes Verständnis der linearen Algebra, insbesondere der Matrizenrechnung sowie der Gitterbasismethoden, so dass ich dem interessierten Leser empfehle, den Artikel [LeLL82] vor dem eigentlichen Artikel [Copp97] zu lesen. Ein (einfacher) Teil des Beweises wird in Abschnitt (7.3.2) durchgeführt.

7.3 Bitsicherheit der RSA-Annahme

In diesem Abschnitt wird die Frage gestellt, wieviele Bits des geheimen RSA-Schlüssels d ein Angreifer benötigt, um ihn vollständig zu rekonstruieren. Das neueste (und im Großen und Ganzen einzige) Resultat stammt aus [BoDF98]. Die folgenden Feststellungen wurden dem besagten Paper entnommen. Ich beginne mit einer kurzen Zusammenfassung der gezeigten Ergebnisse.

7.3.1 Überblick

Die in [BoDF98] gezeigten Resultate können in zwei Theoremen zusammengefasst werden. Dabei wird unterschieden, ob es sich bei dem öffentlichen Exponenten e um einen eher kleinen oder einen großen Wert handelt. Im Folgenden bezeichne $n = pq$ einen üblichen RSA-Modulus, man nimmt weiter an, dass die beiden Primfaktoren der Ungleichung $\frac{\sqrt{n}}{2} < q < p < 2\sqrt{n}$ genügen. Diese Bedingung ist sinnvoll, da in der Praxis zwei Primfaktoren von ungefähr derselben Länge gewählt werden müssen. Kommen wir nun zu den beiden Theoremen.

Theorem 6. *Sei $n = pq$ ein üblicher l -Bit RSA-Modulus. Sei $1 \leq e, d \leq \varphi(n)$ mit $ed \equiv 1 \pmod{\varphi(n)}$. Dann gibt es einen Algorithmus, der bei Eingabe der $\frac{n}{4}$ niederwertigsten Bits von d den Rest von d in polynomieller Zeit in l und e bestimmt.*

Folglich kann für kleine e wie z.B. $e = 3$ ein Angriff in einer akzeptablen Zeit durchgeführt werden. Für größere e wie z.B. den oft benutzten Standardexponenten $e = 65537$ ist eine solche Attacke immer noch sinnvoll, braucht aber wesentlich länger.

Theorem 7. Sei $n = pq$ ein l -Bit RSA-Modulus. Sei $1 \leq e, d \leq \varphi(n)$ mit $ed = 1 \pmod{\varphi(n)}$.

1. Angenommen e ist eine Primzahl im Intervall $[2^t, \dots, 2^{t+1}]$ mit $\frac{l}{4} \leq t \leq \frac{l}{2}$. Dann gibt es einen Algorithmus, der bei Eingabe der t höherwertigsten Bits von d den Rest von d in polynomieller Zeit in l berechnet.
2. Ist $e \in [2^t, \dots, 2^{t+1}]$ das Produkt von höchstens r paarweise verschiedenen Primzahlen mit $\frac{l}{4} \leq t \leq \frac{l}{2}$ und ist die Faktorisierung von e sowie die t höherwertigsten Bits von d bekannt, so gibt es einen Algorithmus, der den Rest von d in polynomieller Zeit in l und 2^r berechnet.
3. Ist die Faktorisierung von e unbekannt, erhält man ein schwächeres Ergebnis. Angenommen es gilt $e \in [2^t, \dots, 2^{t+1}]$ mit $t \in [0, \dots, \frac{n}{2}]$, desweiteren gelte $d > \epsilon n$ für ein $\epsilon \in \mathbb{R}$ mit $\epsilon > 0$. Dann gibt es einen Algorithmus, der bei der Eingabe der $l - t$ höherwertigsten Bits von d den Rest von d in polynomieller Zeit in l und $\frac{1}{\epsilon}$ berechnet.

Offensichtlich unterscheiden sich die beiden Theoreme hauptsächlich in der Position der gegebenen Bits. Während in Theorem 6 die niederwertigsten Bits gefordert werden, beschäftigt sich Theorem 7 mit gegebenen höherwertigeren Bits. Im zweiten Theorem wird behauptet, dass bereits die Hälfte der höherwertigsten Bits von d zur Rekonstruktion ausreicht, sofern e als Primzahl gewählt wurde. Liegt e in der Nähe von $n^{0.25}$, so ist bereits ein Viertel der Bits von d ausreichend. Das gleiche Resultat erhält man in dem Fall, dass e keine Primzahl ist, sofern die Faktorisierung von e bekannt ist und sich e nur aus einer geringen Anzahl von Primfaktoren zusammensetzt. Der letzte Teil von Theorem 7 beschäftigt sich mit dem Fall, dass die Faktorisierung von e unbekannt ist für $e < \sqrt{n}$. Für einen solchen Angriff ist mindestens die Hälfte der höherwertigsten Bits von d vonnöten, ihre Anzahl steigt mit kleiner werdendem Exponenten e .

7.3.2 Notation und ein bekanntes Theorem

Wie üblich beginnt man mit einer Auflistung der im Beweis verwendeten Terminologie. Man nimmt an, dass p und q zwei unterschiedliche Primzahlen in der Nähe von \sqrt{n} sind mit o.B.d.A. $p > q$, genauer soll gelten

$$4 < \frac{\sqrt{n}}{2} < q < p < 2\sqrt{n}$$

Aus dieser Gleichung folgt mit $n = pq$ die Ungleichung $p + q < 3\sqrt{n}$. Im Folgenden setzt man

$$s := p + q.$$

Unter der Voraussetzung $p > q$ folgt nun unmittelbar

$$p = \frac{1}{2}(s + \sqrt{s^2 - 4n}).$$

Desweiteren folgt aus der ersten Ungleichungskette

$$\frac{n}{2} < n - 4\sqrt{n} < \varphi(n) < n.$$

Seien nun $1 \leq e, d \leq \varphi(n)$ die entsprechenden RSA-Exponenten. Somit gilt $ed \equiv 1 \pmod{\varphi(n)}$. Im folgenden bezeichne k die eindeutig bestimmte Zahl mit

$$ed - k\varphi(n) = ed - k(p-1)(q-1) = ed - k(pq - (p+q) + 1) = ed - (n - s + 1) = 1. \quad (7.1)$$

Da nach Voraussetzung $\varphi(n) > d$ gilt, folgt $k < e$.

Nach dieser kurzen Einführung in die im folgenden verwendete Notation kommen wir zu einem Satz aus der Computeralgebra, der sich als grundlegend für die folgenden Betrachtungen erweist.

Theorem 8 (Coppersmith [Copp96]). *Sei $f(x, y)$ ein Polynom in zwei Variablen über \mathbb{Z} von maximalem Grad δ in jeder Variablen und die Koeffizienten von f seien paarweise prim. Seien weiter X und Y obere bzw. untere Schranken der gesuchten Lösungen x_0, y_0 . Definiere $\tilde{f}(x, y) := f(Xx, Yy)$ und sei D der Absolutbetrag des betragsmäßig größten Koeffizienten von \tilde{f} . Gilt nun $XY < D^{\frac{2}{3\delta}}$, dann ist es in polynomieller Zeit in $(\log D, 2^\delta)$ möglich, alle Paare (x_0, y_0) mit $f(x_0, y_0) = 0$ zu finden, die der Einschränkung $|x_0| < X, |y_0| < Y$ genügen.*

Ein Beweis dazu ist in [Copp96] zu finden.

Aus dem Theorem folgt nun unmittelbar das folgende

Korollar 7.1. *Sei $n = pq$ ein l -Bit RSA-Modulus. Sei $r \geq 2^{l/4}$ sowie $p_0 := p \bmod r$ mit $ggT(p_0, r) = 1$ gegeben. Dann ist es möglich, die Zahl n in polynomieller Zeit in l zu faktorisieren.*

Beweis. Anhand des gegebenen p_0 kann man $q_0 := q \bmod r = \frac{n}{p_0} \bmod r$ berechnen. Man sucht nun nach einer Lösung (x_0, y_0) von $f(x, y) = (rx + p_0)(ry + q_0) - n$ mit $0 \leq x < X = \frac{2^{l/2+1}}{r}$ sowie $0 \leq y < Y = \frac{2^{l/2+1}}{r}$. Der größte gemeinsame Teiler der Koeffizienten des Polynoms $f(x, y)$ ist r , d.h. um Theorem 8 anwenden zu können, definiert man sich ein neues Polynom $g(x, y) = \frac{f(x, y)}{r}$. Der größte Koeffizient von $\tilde{g}(x, y) = g(Xx, Yy)$ beträgt mindestens $\frac{2^{l+2}}{r}$. Um das Theorem anwenden zu können brauchen wir

$$2^{l+2} < r^4 \Leftrightarrow 2^{\frac{l+2}{3}} < r^{\frac{4}{3}} \Leftrightarrow XY = \frac{2^{l+2}}{r^2} < \left(\frac{2^{l+2}}{r}\right)^{2/3}.$$

Durch eine vollständige Suche über die beiden höherwertigsten Bits von x_0 und y_0 erhält man $r \geq 2^{l/4}$, was den Beweis vervollständigt. \square

Man sieht, dass es sich um einen Teil des in Abschnitt (7.2) angesprochenen Sicherheitsbeweises des Faktorisierungsproblems handelt, bei dem der schwierige Teil als Theorem ohne Beweis vorausgesetzt ist.

7.3.3 Beweis der Theoreme 6 und 7

Wir sind nun bereit, unter Zuhilfenahme des bereits bewiesenen Korollars das Theorem 6 zu beweisen:

Beweis zu Theorem 6. Angenommen, die $\frac{n}{4}$ niederwertigsten Bits von d sind bekannt, d.h. man kennt $d_0 := d \bmod 2^{l/4}$. Gemäß Gleichung 7.1 gilt nun

$$ed_0 = 1 + k(n - s + 1) \bmod 2^{l/4}.$$

Nun werden alle möglichen k im Intervall $[0, \dots, e]$ durchgetestet. Für jedes k löst der Angreiferalgorithmus die obige Gleichung nach $s \bmod 2^{l/4}$ auf. Anhand des berechneten Wertes für s kann man nun $p \bmod 2^{l/4}$ bestimmen, indem man die quadratische Gleichung

$$p^2 - sp + n = 0 \bmod 2^{l/4}$$

nach p auflöst. Auf den berechneten Wert p wird nun Korollar 7.1 angewandt, um die Zahl n zu faktorisieren. Wegen der Größe des zugrundeliegenden Intervalls braucht man höchstens e Versuche, um den richtigen Wert von $p \bmod 2^{l/4}$ herauszufinden und demzufolge die Faktorisierung von n zu berechnen. Die Laufzeit ist somit linear in e . □

In den ersten beiden Unterpunkten von Theorem 7 beschränkt man sich auf einen public key e , der zwischen $2^{l/4}$ und $2^{l/2}$ liegt. Der Angreifer erhält im Gegensatz zum vorigen Theorem die *höherwertigsten* Bits des secret key d . Der Angreifer baut seinen Angriff auf der bereits eingeführten Gleichung 7.1

$$ed - k(n - s + 1) = 1$$

auf. Unser Augenmerk richten wir auf die Berechnung von k . Da k im Intervall $[0, \dots, e]$ liegt, ist eine vollständige Suche wegen der Größe von e nicht durchführbar. Folglich muss man die gegebene Zusatzinformation, die höherwertigsten Bits von d , benutzen. Es stellt sich heraus, dass für $e < \sqrt{n}$ bereits die *loge* höherwertigsten Bits von d ausreichen, um den Wert von k zu berechnen. Dies wird in folgendem Satz festgehalten.

Satz 7.2. *Unter Benutzung der bereits eingeführten Notation sei $t \in [0, \dots, \frac{n}{2}]$ mit $2^t < e < 2^{t+1}$. Kennt man nun die t höherwertigsten Bits von d , so ist es in polynomieller Zeit möglich, den Wert von k berechnen, der der Gleichung 4 genügt, bis auf einen konstanten additiven Fehler.*

Um das Theorem beweisen zu können, beweist man zuerst das folgende Lemma.

Lemma 7.3. *Angenommen, ein Wert d_0 ist gegeben, der den folgenden Bedingungen genügt:*

1. $|e(d - d_0)| < c_1 n$.
2. $ed_0 < c_2 n^{3/2}$.

Dann liegt der eindeutige Wert von k , der der Gleichung $ed - k\varphi(n) = 1$ genügt, innerhalb des Intervalls $[\tilde{k} - \Delta, \tilde{k} + \Delta]$ mit $\tilde{k} = (ed_0 - 1)/n$ und $\Delta = 8c_2 + 2c_1$.

Beweis. Sei $\tilde{k} = (ed_0 - 1)/n$. Dann gilt

$$|\tilde{k} - k| = \left| (ed_0 - 1) \left(\frac{1}{\varphi(n)} - \frac{1}{n} \right) + \frac{e(d - d_0)}{\varphi(n)} \right| < c_2 n^{3/2} \left(\frac{n - \varphi(n)}{\varphi(n)n} \right) + c_1 \frac{n}{\varphi(n)}.$$

Da nun gilt $n - \varphi(n) = n - (p - 1)(q - 1) = n - pq + (p + q) - 1 < 4\sqrt{n}$ und $\varphi(n) > \frac{n}{2}$ folgt

$$|\tilde{k} - k| < 8c_2 + 2c_1.$$

Somit liegt k innerhalb des Intervalls $[\tilde{k} - \Delta, \tilde{k} + \Delta]$. □

Fahren wir nun mit dem Beweis zu Satz 7.2 fort.

Beweis von Satz 7.2. Sind die t höherwertigsten Bits von d gegeben, so kann man eine Zahl d_0 konstruieren, die der Bedingung $|d - d_0| < 2^{l-t}$ genügt. Anhand von Lemma 7.3 kann man nun k berechnen. Die erste Vorbedingung des Lemmas ist wegen der Einschränkung an e mit $c_1 = 2$ erfüllt. Da desweiteren $d_0 < n$ trivialerweise gilt, ist auch die zweite Voraussetzung mit $c_2 = 2$ eingehalten. Die Länge des betrachteten Intervalls beträgt somit $2 \cdot \Delta = 2(8 \cdot 2 + 2 \cdot 2) = 40$. \square

Wir sind nun in der Lage, Theorem 7 (1) auf folgende Art zu beweisen. Mittels Satz 7.2 ist es möglich, den Wert von k zu bestimmen, der der Gleichung 7.1 genügt. Betrachtet man nun die Gleichung 7.1 modulo e , so entfällt der Term ed und s bleibt als einzige Unbekannte übrig, die man errechnen kann. Man erhält somit $s \bmod e$, was zur Faktorisierung von n ausreicht. Kommen wir nun zum formalen Beweis.

Beweis von Theorem 7. Die Annahmen aus Theorem 7 (1) genügen den Vorbedingungen von Satz 7.2, somit kann man die Position von k auf ein Intervall konstanter Größe beschränken. Für alle möglichen Werte k dieses Intervalls führe die folgenden Schritte aus:

1. Berechne $s \equiv n+1-k^{-1} \pmod{e}$. Das Inverse zu k existiert modulo e , da gilt $ggT(e, k) = 1$.
2. Bestimme $p \bmod e$, indem man die eine Nullstelle x_0 der quadratischen Gleichung

$$x^2 - sx + n \equiv 0 \pmod{e}$$

berechnet. Diese ist effizient möglich, da es sich bei dem Modulus um eine Primzahl handelt [BeOr81]. Da s durch $s = p + q$ definiert ist, folgt $x_0 \equiv p \pmod{e}$.

3. Benutze nun Korollar 7.1, um den Wert von p anhand $p \bmod e$ zu bestimmen. Die Voraussetzungen des Korollars sind wegen $e \geq 2^{l/4}$ erfüllt.

Wegen der konstanten Größe des Intervalls wird nun nach einer konstanten Anzahl von Versuchen der Wert von k gefunden und somit die Zahl n faktorisiert.

Bei dem Beweis fällt auf, dass die Voraussetzung, dass e als Primzahl gewählt werden muss, nicht zwangsläufig notwendig ist. Für den Beweis genügt es, dass man in der Lage ist, die Gleichung in Punkt 2 des Algorithmus zu lösen. Dafür genügt bereits die Faktorisierung von e , um die Gleichung auf Gruppen mit Primzahlordnung zu reduzieren und die Ergebnisse mittels des chinesischen Restsatzes wieder zusammensetzen. Dabei tritt jedoch das Problem auf, dass eine quadratische Gleichung modulo einer zusammengesetzten Zahl viele Nullstellen haben kann, genauer sind für r unterschiedliche Primfaktoren 2^r Nullstellen zu betrachten. Diese müssen alle durchprobiert werden; folglich muss man die Anzahl der Primfaktoren von e auf ein sinnvolles Maß beschränken, damit man polynomielle Laufzeit in l und 2^r erhält. Wir haben soeben Theorem 7 (2) bewiesen.

Kommen wir nun zum letzten Teil von Theorem 7.

Sei $e \in [2^t, \dots, 2^{t+1}]$ mit $0 \leq t \leq \frac{n}{2}$, wobei die Faktorisierung von e unbekannt sei. Desweiteren sei $k > \epsilon \cdot e$ für ein $\epsilon > 0$. Sind nun die $n - t$ höherwertigsten Bits von d gegeben, so kann man eine Zahl d_0 konstruieren, die der Ungleichung $0 \leq d - d_0 < 2^t$ genügt. Da $e < 2^{n/2}$ gilt, kann man d_0 und Satz 7.2 benutzen, um die Position von k auf ein Intervall konstanter Länge einzuschränken. Für alle möglichen Werte für k führe die folgenden Schritte aus:

1. Berechne $d_1 \equiv e^{-1} \pmod k$. Dies ist möglich, da e und k teilerfremd sind. Betrachtet man die Gleichung $ed - k\varphi(n) = 1$ modulo k , so folgt nun $d_1 \equiv d \pmod k$.
2. Nach Voraussetzung gilt $k > \epsilon \cdot e > \epsilon \cdot 2^t$. An dieser Stelle des Algorithmus sind bereits die $d \pmod k$ sowie die $n - t$ höherwertigsten Bits von d bekannt. Die restlichen Bits werden durch eine vollständige Suche bestimmt, genauer schreibe

$$d = kd_2 + d_1$$

Dann gilt $d_2 \equiv \frac{d_0}{k} + \frac{d-d_0}{k} - \frac{d_1}{k}$. Die einzige Unbekannte bildet der Term $v := (d - d_0)/k$. Wegen $k > \epsilon \cdot 2^t$ gilt nun $v < \frac{1}{\epsilon}$. Um v zu finden, werden nun alle möglichen Werte im Intervall $[0, \dots, \frac{1}{\epsilon}]$ durchgetestet.

3. Sobald die Werte für v und k gefunden sind, ist der geheime Schlüssel d rekonstruiert.

Offensichtlich wird der vorgestellte Angriff für kleine k immer langsamer, da in diesem Fall ϵ sehr klein ist und somit in Punkt 2 ein großes Intervall durchsucht werden muss. Abstrahiert man nun den Wert von k als gewöhnliche Zufallszahl im Intervall $[1, \dots, e]$, so gilt z.B. in 90% aller Fälle $k > \frac{e}{10}$ für ein $e \in [2^t, \dots, 2^{t+1}]$. Man kann somit heuristisch sagen, dass der Algorithmus für fast alle e zum Erfolg führt.

□

Literaturverzeichnis

- [ACGS88] Werner Alexi, Benny Chor, Oded Goldreich and Claus P. Schnorr. RSA and Rabin functions: Certain Parts are as hard as the whole. *SIAM Journal on Computing* 17/2 (1988) 194-209.
- [ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. *Crypto 2000*, LNCS 1880, Springer-Verlag, Berlin 2000, 255-270.
- [BaPf97] Niko Barić and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. *Eurocrypt '97*, LNCS 1233, Springer-Verlag, Berlin 1997, 480-494.
- [BDPR98] Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Crypto '98*, LNCS 1462, Springer-Verlag, Berlin 1998, 26-45.
- [BeCS83] Michael Ben-Or, Benny Chor and Adi Shamir. On the cryptographic security of single RSA bits. In *Proceeding of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC) 1983*, ACM, New York 1983, 421-430.
- [Bena87] Josh Cohen Benaloh. Verifiable Secret-Ballot Elections. Dissertation, Yale University, September 1987.
- [Bena94] Josh Cohen Benaloh. Dense Probabilistic Encryption. *First Annual Workshop on Selected Areas in Cryptography (SAC '94)*, Kingston, Ontario, May 1994, 120-128.
- [BeOr81] Michael Ben-Or. Probabilistic Algorithms in Finite Fields. *Proc. IEEE* (1981), 394-398.
- [BeRo93] Mihir Bellare and Phillip Rogaway. Random Oracles are practical: A paradigm for designing efficient protocols. *1st ACM Conference on Computer and Communications Security*, ACM Press, Fairfax, Virginia, November 1993.
- [BIMi82] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences Of Pseudo Random Bits. *23rd Symposium on Foundations of Computer Science (FOCS) 1982*, IEEE Computer Society, 1982, 112-117.
- [BoDF98] Dan Boneh, Glenn Durfee and Yair Frankel. An Attack on RSA Given a Small Fraction of the Private Key Bits. *Asiacrypt '98*, LNCS 1514, Springer-Verlag, Berlin 1998, 25-34.

- [BoDH99] Dan Boneh, Glenn Durfee and Nick Howgrave-Graham. Factoring $n = p'q$ for Large r . Crypto '99, LNCS 1666, Springer-Verlag, Berlin 1999, 326-337.
- [BoVe96] Dan Boneh and Ramarathnam Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. Crypto '96, LNCS 1109, Springer-Verlag, Berlin 1996, 129-142.
- [BoVe98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 59-71.
- [Bund98] Peter Bundschuh. Einführung in die Zahlentheorie. 4. Auflage, Springer-Verlag, Berlin 1998.
- [CaMi98] Jan Camenisch and Markus Michels. A Group Signature Scheme with Improved Efficiency. Asiacrypt '98, LNCS 1514, Springer-Verlag, Berlin 1998, 160-174.
- [CaMS98] Christian Cachin, Silvio Micali and Markus Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. Eurocrypt '99, LNCS 1592, Springer-Verlag, Berlin 1999, 402-414.
- [CFPR96] Don Coppersmith, Matthew Franklin, Jacques Patarin and Michael Reiter. Low-exponent RSA with related messages. Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin, 1996, 1-9.
- [ChGo85] Benny Chor and Oded Goldreich. RSA/Rabin least significant bits are $\frac{1}{2} + \frac{1}{poly(logn)}$ secure (Extended Abstract). Crypto '84, LNCS 196, Springer-Verlag, Berlin 1985, 303-313.
- [Copp96] Don Coppersmith. Finding a small root of a univariate modular equation. Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 155-165.
- [Copp97] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 178-189.
- [CrSh98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. Crypto '98, LNCS 1462, Springer-Verlag, Berlin, 13-25.
- [CrSh99] Ronald Cramer and Victor Shoup. Signature Schemes Based on the Strong RSA Assumption. Theory of Cryptography Library 99-01, January 1999, <http://philby.ucsd.edu/cryptolib.html>.
- [DiHe76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory 22/6 (1976) 644-654.
- [DoDN91] Dany Dolev, Cynthia Dwork and Moni Naor. Non-Malleable Cryptography. 23rd Symposium on Theory of Computing (STOC) 1991, ACM, New York 1991, 542-552.
- [ElGa85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, 31/4 (1985) 469-472.

- [FiSh87] Amos Fiat and Adi Shamir. How to prove Yourself: Practical Solutions to Identification and Signature Problems. *Crypto '86*, LNCS 263, Springer-Verlag, Berlin 1987, 186-194.
- [FiSc97] Roger Fischlin and Claus P. Schnorr. Stronger Security Proofs for RSA and Rabin bits. *Eurocrypt '97*, LNCS 1233, Springer-Verlag, Berlin 1997, 267-279.
- [FuOk97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. *Crypto '97*, LNCS 1294, Springer-Verlag, Berlin 1997, 16-30.
- [Gold84] Oded Goldreich. On the Number of Close-to-Equal Pairs of Bits in a String (with Implications on the Security of RSA's L.S.B.). *Eurocrypt '84*, LNCS 209, Springer-Verlag, Berlin 1985, 127-141.
- [GoMi84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences* 28 (1984) 270-299.
- [GoMR88] Shafi Goldwasser, Silvio Micali and Ronald L. Rivest. A Digital Signature Scheme Secure Against Chosen-Message Attacks. *SIAM Journal on Computing* 17/2 (1988) 281-308.
- [GoMT82] Shafi Goldwasser, Silvio Micali and Po Tong. Why and How to Establish a Private Code on a Public Network. *23rd Symposium on Foundations of Computer Science (FOCS) 1982*, IEEE Computer Science, 1982, 134-144.
- [HaNa98] Johan Hastad, Mats Näslund: The Security of all RSA and Discrete Log Bits, *Theory of Cryptography Library*, Record 99-19, available at <http://philby.ucsd.edu/cryptolib>, 1998.
- [HaSS82] Johan Hastad, Avital W. Schrift and A. Shamir. The discrete logarithm modulo a composite hides $O(n)$ bits. *Journal of Computer Science*, Nov. 3-5 1982, Chicago, Illinois, 1982.
- [KnMe99] Lars R. Knudsen and Willi Meier. Cryptanalysis of an Identifikation Scheme Based on the Permuted Perceptron Problem. *Eurocrypt '99*, LNCS 1592, Springer-Verlag, Berlin 1999, 363-374.
- [Koch96] Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. *Crypto '96*, LNCS 1109, Springer-Verlag, Berlin 1996, 104-113.
- [Joux00] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *4th Algorithmic Number Theory Symposium (ANTS)*, LNCS 1838, Springer-Verlag, Berlin 2000, 385-393.
- [Lamp91] Erich Lamprecht. *Einführung in die lineare Algebra*. Birkhäuser Verlag, Berlin, 1991.
- [LeLL82] A. K. Lenstra, H. W. Lenstra Jr. and L. Lovasz. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen* 261 (1982), 515-534.
- [LoWi88] Douglas L. Long and Avi Wigderson. The discrete logarithm hides $O(\log n)$ bits. *SIAM Journal on Computing* (1988) 363-372.

- [MaWo99] Ueli Maurer and Stefan Wolf. The Relationship Between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. *SIAM Journal on Computing* 28/5 (1999) 1689-1721.
- [McEl79] Robert J. McEliece. The Theory of Information and Coding. A mathematical Framework for Communication. *Encyklopädia of Mathematics and its Applications Vol. 3*, Addison-Wesley Publishing Company, Massachusetts, 1979.
- [McSa81] Robert J. McEliece and D.V. Sarwate. On Sharing Secrets and Reed-Solomon Codes. *Communications of the ACM* 24/9 (1981), 583-584.
- [MiRS88] Silvio Micali, Charles Rackoff and Bob Sloan. The Notion of Security for Probabilistic Cryptosystems. *SIAM Journal on Computing* 17/2 (1988) 412-426.
- [NaSt98] David Naccache and Jacques Stern. A New Public Key Cryptosystem Based on Higher Residues. 5th ACM Conference on Computer and Communication Security, San Francisco, California, November 1998, ACM Press, New York 1998, 59-66.
- [NaYu89] Moni Naor and Moti Yung. Universal One-way Hash Functions and their Cryptographic Applications. 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 33-43.
- [OkUc97] Tatsuaki Okamoto and Shigenori Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 308-318.
- [Pail99] Pascal Paillier. Public Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt '99, LNCS 1592, Springer-Verlag, Berlin 1999, 223-238.
- [PaSu98] Sarvar Patel and Ganapathy S. Sundaram. An Efficient Discrete Log Pseudo Random Generator. Crypto'98, LNCS 1462, Springer-Verlag, Berlin 1998, 304-317.
- [Pera86] Rene Peralta. Simultaneous Security of Bits of the Discrete Log. Eurocrypt '85, LNCS 219, Springer-Verlag Berlin 1986, 62-72.
- [Pfit96] Birgit Pfitzmann. Digital Signature Schemes - General Framework and Fail-Stop Signatures. LNCS 1100, Springer-Verlag, Berlin 1996.
- [Pfit98] Birgit Pfitzmann. Höhere kryptographische Protokolle. Skript zur Vorlesung im Sommersemester 98, Universität des Saarlandes, Saarbrücken, Juli 1998, <http://www-krypt.cs.uni-sb.de>.
- [Pfit99] Birgit Pfitzmann. Sicherheit. Folienskript zur Vorlesung Im Wintersemester 97/98, Universität des Saarlandes, Saarbrücken, Februar 1998, updates Wintersemester 98/99 und 99/00, <http://www-krypt.cs.uni-sb.de>.
- [PoHe78] S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, vol. IT-24-1, 1978, 106-110.
- [Poin95] David Pointcheval. A New Identifikation Scheme Based on the Perceptrons Problem. Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 319-328.

- [Poin99] David Pointcheval. New Public Key Cryptosystems based on the Dependent-RSA Problems. Eurocrypt '99, LNCS 1592, Springer-Verlag, Berlin 1999, 239-254.
- [Rabi80] Michael O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Massachusetts Institute of Technology, Laboratory for Computer Science, MIT/LCS/TR-212, January 1979.
- [RaSi91] Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. Crypto '91, Santa Barbara, CA, 1991, 433-444.
- [RiSA78] Ronald L. Rivest, Adi Shamir and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21/2 (1978), reprinted: 26/1 (1983) 96-99.
- [RiSh85] Ronald L. Rivest and Adi Shamir. An efficient factoring based on partial information. Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 31-44.
- [ScAl86] Claus P. Schnorr and Werner Alexi. RSA-bits are $0.5 + \epsilon$ secure. Eurocrypt '84, LNCS 209, Springer-Verlag, Berlin 1985, 113-126.
- [ScEi96] Jörg Schwenk and Jörg Eisfeld. Public Key Encryption and Signature Schemes Based on Polynomials over \mathbb{Z}_n . Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 60-71.
- [Shan49] C. E. Shannon. Communication Theory of Secrecy Systems. The Bell System Technical Journal 28/4 (1949), 656-715.
- [Schn98] Claus P. Schnorr. Security of almost all discrete log bits. Electronic Colloquium on Computational Complexity, report TR98-033, 1998, available online from <http://www.eccc.uni-trier.de/eccc/>.
- [ScSh91] Avital W. Schrift and Adi Shamir. On the Universality of the Next Bit Test. Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 394-408.
- [VaVa83] Umesh V. Vazirani and Vijay V. Vazirani. RSA bits are $.732 + \epsilon$ secure. Crypto '83, Plenum Press, New York 1984, 369-375.
- [VaZu95] Scott A. Vanstone and Robert J. Zuccherato. Short RSA Keys and Their Generation. Journal of Cryptology 8/2 (1995), 101-114.

Anhang A

Tabellarische Übersicht der Annahmen

Da man in Anbetracht der großen Anzahl der in diesem Paper vorgestellten Probleme leicht den Überblick verlieren kann, scheint es sinnvoll, sie in einer Tabelle übersichtlich zusammenzufassen. Besonders wichtig sind hierbei die jeweiligen Kurzformen der einzelnen Probleme, die zum Teil in den Beweisen in Kapitel 5 sowie in der Fachliteratur benutzt werden. Aus diesem Grund empfiehlt es sich, die Tabelle als Referenz zu Kapitel 5 zu betrachten, da sie das Verständnis der Beweise zum Teil erheblich erleichtert.

Deutscher Name des Problems	Originalname	Bezeichnung
Faktorisierungsproblem	<i>Integer Factorisation Problem</i>	Factoring
Modifiziertes Faktorisierungsproblem	<i>Modified Factorisation Problem</i>	M-Factoring
Diskreter Log. Problem	<i>Discrete Logarithm Problem</i>	D-Log
Zusammengesetzter Diskreter Log.	<i>Composite Discrete Logarithm Problem</i>	CD-Log
RSA Problem	<i>RSA Problem</i>	RSA
Diffie Hellman Problem	<i>Diffie Hellman Problem</i>	DHP
Wurzel Problem	<i>Square Root Problem</i>	SQROOT
Quadratische Reste Problem	<i>Quadratic Residuosity Assumption</i>	QRA
Diffie Hellman Entscheidungsproblem	<i>Diffie Hellman Decision Problem</i>	DHDP
-	<i>Higher Residuosity Assumption</i>	HRA
-	<i>Decisional Higher Residuosity Assumption</i>	DHRA
-	<i>Decisional Composite Residuosity Assumption</i>	DCRA
-	<i>Composite Residuosity Class Problem</i>	Class[$P^2 Q^2$]
-	<i>Decision Composite Residuosity Class Problem</i>	D- Class[$P^2 Q^2$]
-	<i>Partial Discrete Logarithm Problem</i>	PDL[$P^2 Q^2, g$]
-	<i>Decisional Partial Discrete Logarithm Problem</i>	DPDL[$P^2 Q^2, g$]
Starke RSA Annahme	<i>Strong RSA Assumption</i>	Strong RSA
-	<i>Computational Dependent RSA Assumption</i>	C-DRSA
-	<i>Decisional Dependent RSA Assumption</i>	D-DRSA
-	<i>ϕ-Hiding Assumption</i>	ϕ HA
Nullstellenproblem	<i>Root Finding Problem</i>	RFP

Abschließend wird noch eine Übersicht in tabellarischer Form gegeben, welche Systeme zum heutigen Zeitpunkt mit den neuen Annahmen realisiert wurden.

Problem	Verschlüsselungs-systeme	Signatur-systeme	Sonstige Systeme
DHDP	1. [CrSh98]	-	-
Höhere Reste	1. [Bena94] 2. [OkUc97] 3. [NaSt98] 4. [Pail99]	-	1. Wahlprotokolle: [Bena87] 2. Verifiable Secret Sharing: [Bena94]
Strong RSA	-	1. [CrSh99]	1. One-way Akkumulatoren: [BaPf97] 2. Gruppensignaturen: [CaMi98],[ACJT00] 3. Commitments: [FuOk97]
C-DRSA	1. [Poin99]	-	-
ϕ HA	-	-	1. Private Information Retrieval: [CaMS98]
RFP	1. [ScEi96]	1. [ScEi96]	-

Index

- Algorithmen
 - Probabilistische, 12
- Diffie Hellman
 - Decision Problem, 29
 - Problem, 26
- Discrete Logarithm Problem, 23
 - Composite, 24
 - Decisional Partial, 42
 - Generalised, 23
 - Partial, 42
- Einheit, 4
- Element
 - Erzeugnis, 4
 - irreduzibles, 5
 - Ordnung, 4
 - Prim-, 6
- Ereignis, 8
 - komplementäres, 8
- Erwartungswert, 9
- Faktorisierungsproblem, 21
 - modifiziertes, 22
- Gruppe, 3
 - abelsche, 3
 - Generator, 3
 - Ordnung, 4
 - p-Sylow, 44
 - Unter-, 4
 - zyklische, 4
- Hashfunktionen, 12
- Higher Residuosity Assumption, 36
- Integritätsbereich, 5
- Jacobi-Symbol, 8
- Körper, 5
 - kongruent, 6
- Legendre-Symbol, 7
- Nullteiler, 4
- Phi-Funktion, Eulersche, 7
- Polynom, 5
 - ring, 5
 - Grad, 5
- Public-key Verschlüsselung, 9
- Random Oracle Modell, 19
- Residuosity Assumption
 - Decisional Composite, 38
 - Decisional Higher, 36
 - Prime, 37
 - Quadratic, 27
- Residuosity Class Assumption
 - Composite, 40
 - Decisional Composite, 41
 - n-th, 39
- Rest
 - höherer, 8
 - quadratischer, 7
- Restklasse, 33
- Restsatz, chinesischer, 7
- Restsystem, 6
- Ring, 4
 - kommutativer, 4
 - nullteilerfreier, 4
 - unitärer, 4
 - ZPE-, 6
- Root Finding Problem, 59
- RSA Assumption, 24
 - Computational Dependent, 52
 - Decisional Dependent, 52
 - Dependent, 52
 - Strong, 49
- Satz von

- Euler, 7
- Lagrange, 4
- Sicherheit
 - gegen adaptive chosen ciphertext attack, 16
 - gegen adaptive chosen message attack, 18
 - non-malleability, 18
 - semantische, 15
- Signaturssysteme, 11
- Square Root Problem, 26
- Subset Sum Problem, 27

- Teiler, 6
- Tschebychevsche Ungleichung, 9

- Varianz, 9

- Wahrscheinlichkeit, 8
 - bedingte, 8
 - vernachlässigbare, 12
- Wahrscheinlichkeitsverteilung, 8
 - uniforme, 8

- Zufallsvariablen, 9