Die Bedeutung von kryptographischen Falltürfunktionen für Algorithmisches Lernen

Diplomarbeit an der Universität Ulm
Fakultät für Informatik

vorgelegt von:

Guido Kehrle

1. Gutachter: *Priv. Doz. Dr. J. Köbler*
2. Gutachter: *Prof. Dr. U. Schöning*

1996

Zusammenfassung

Diese Arbeit beschäftigt sich mit den Zusammenhängen von Kryptographie und Erlernbarkeitstheorie. Hierbei werden vorwiegend die Auswirkungen von kryptographischen Annahmen auf das effiziente Erlernen von Konzeptklassen untersucht.

In neuerer Zeit wurden einige Zusammenhänge zwischen Kryptographie und Algorithmischem Lernen aufgedeckt. Die wichtigsten dieser Veröffentlichungen werden untersucht und verständlich dargestellt. Um die zum Verständnis dieser Arbeit benötigten Grundlagen auf ein verträgliches Maß zu reduzieren, werden die Fachkenntnisse eines vordiplomierten Informatikers vorausgesetzt.

Es werden zunächst die komplexitätstheoretischen Grundlagen erläutert. Im Anschluß an eine kurze Einführung in die Kryptographie und die Erlernbarkeitstheorie weisen wir die nichteffiziente Erlernbarkeit von Konzeptklassen auf der Grundlage kryptographischer Annahmen nach. Wir sehen dabei, daß diese Resultate Auswirkungen auf die Approximierbarkeit von Optimierungsproblemen haben. Danach zeigen wir - ebenfalls auf der Grundlage von kryptographischen Annahmen - daß viele Klassen auch durch eine erweiterte Erlernbarkeitsdefinition nicht effizient erlernbar werden. Schließlich weisen wir noch nach, daß sich durch die Verwendung dieser erweiterten Erlernbarkeitsdefinition beim effizienten Erlernen von einigen Konzeptklassen, deren Erlernbarkeitsstatus noch offen ist, kein Vorteil ergibt. Abschließend untersuchen wir, welche Schlüsse sich von Erlernbarkeitsresultaten auf kryptographische Systeme ziehen lassen.

Inhaltsverzeichnis

1

Einleitung

1.1 Problemstellung

Das Gebiet des Algorithmischen Lernens, wie wir es heute kennen, hat sich erst vor etwas mehr als einem Jahrzehnt entwickelt. Der Grundstein dazu wurde 1984 durch eine Veröffentlichung von Valiant [Val84] gelegt. Valiant führt darin die Erlernbarkeitsdefinition *PAC* ein. Während zum Beispiel *kCNF* und *kDNF*[1] Formeln effizient nach dieser Definition erlernbar sind, konnte Valiant nachweisen, daß unter plausiblen Annahmen die Klasse der Booleschen Formeln nicht effizient erlernbar ist.

Die moderne Kryptographie gründet sich zu einem Großteil auf die Veröffentlichung von W. Diffie und M. E. Hellmann. [DH76]. Sie stellten 1976 in dieser Veröffentlichung das erste Public-Key Verfahren vor. Seither sind viele weitere Public-Key Verfahren entwickelt und untersucht worden.

Eine der grundlegenden Fragen ist, inwieweit ein Computer Lernfähigkeit besitzen kann. Daß der Lernfähigkeit von Computern im Sinne der *PAC*-Erlernbarkeitsdefinition Grenzen gesetzt sind, zeigte schon Valiant. Dabei stellt sich die Frage, welche Lernprobleme durch Computer noch in Polynomialzeit bewältigt werden können.

Man kann positive Erlernbarkeitsresultate durch Angabe eines das Problem bewältigenden Lernalgorithmus erhalten. Bei negativen Resultaten ist der Nachweis der nicht effizienten Erlernbarkeit ungleich aufwendiger. In diesem Fall muß ein Gegenbeweis angetreten werden. Kearns und Valiant [KV89] haben 1989 die Möglichkeit erkannt, kryptographische Annahmen zu verwenden, um die nicht effiziente Berechenbarkeit von Lernproblemen nachzuweisen. Diese und andere Zusammenhänge

[1] *kCNF* und *kDNF* sind Boolesche Formeln in konjunktiver beziehungsweise disjunktiver Normalform, wobei jeder Term nur k Literale enthalten darf

von Kryptographie und Erlernbarkeitstheorie werden wir in dieser Arbeit untersuchen.

1.2 Aufbau der Arbeit

Wir erläutern zu Beginn der Arbeit die komplexitätstheoretischen Grundlagen, die wir insbesondere für die Erlernbarkeitstheorie benötigen.

In Kapitel 3 zeigen wir die für diese Arbeit notwendigen Grundlagen der Kryptographie auf. Zu diesem Zweck behandeln wir auch die zahlentheoretischen Grundlagen der beschriebenen Verfahren.

Wir führen dann die Grundlagen der Erlernbarkeitstheorie in Kapitel 4 ein. Dabei gehen wir insbesondere auf die vielfältigen Grunddefinitionen für Erlernbarkeit ein, die in der Literatur verwendet werden.

Nach einer kurzen Einführung in die Zusammenhänge zwischen Kryptographie und Algorithmischem Lernen in Kapitel 5 weisen wir in Kapitel 6 zunächst die nicht-effiziente Erlernbarkeit von Konzeptklassen auf der Grundlage kryptographischer Annahmen nach. Dabei gehen wir auch auf die Tatsache ein, daß diese Resultate Auswirkungen auf die Approximierbarkeit von Optimierungsproblemen haben.

Im darauffolgenden Kapitel weisen wir ebenfalls auf der Grundlage kryptographischer Annahmen nach, daß viele Klassen auch durch die Erweiterung der Erlernbarkeitsdefinition um Mitgliedschaftsfragen nicht effizient erlernbar werden.

Schließlich führen wir in Kapitel 8 vor, daß man durch Erweiterung des Erlernbarkeitsmodells um Mitgliedschaftsfragen beim effizienten Erlernen von einigen Konzeptklassen, deren Erlernbarkeitsstatus noch ungelöst ist, keinen Vorteil erhält. Das heißt, diese Konzeptklassen sind entweder überhaupt nicht effizient erlernbar oder sie sind bereits im normalen Erlernbarkeitsmodell ohne Mitgliedschaftsfragen effizient erlernbar. Im Gegensatz dazu ist die Klasse der deterministischen endlichen Automaten C_{DFA}, wenn man plausible kryptographische Annahmen voraussetzt, nur unter Verwendung von Mitgliedschaftsfragen effizient erlernbar.

Abschließend werden wir in Kapitel 9 zeigen, daß man auch Erlernbarkeitsresultate auf kryptographische Systeme anwenden kann.

2

Komplexitätstheoretische Grundlagen

Wir werden in diesem Kapitel nur kurz auf die komplexitätstheoretischen Grundlagen eingehen, die in dieser Arbeit benötigt werden. Für weiterführende Informationen sind die Bücher [HU90], [Pap94], [Sch95] und [Lee90] zu empfehlen.

Die Komplexitätstheorie beschäftigt sich mit dem Resourcenverbrauch von Turingmaschinen bei der Lösung von Problemen. Im allgemeinen sind dies Entscheidungsprobleme, wir werden jedoch auch einige funktionale Probleme betrachten. Die Einteilung nach Zeit- und Speicherverbrauch hat sich weitgehend durchgesetzt, obwohl auch andere Komplexitätsmaße möglich sind. Trotzdem ist die Einteilung nach Zeit- und Speicherverbrauch auch aus praktischer Sicht naheliegend. Man benötigt zum Beispiel in Real-Time Systemen möglichst schnelle Algorithmen oder man muß neuere speicherintensive Programme auf älteren Rechnern mit wenig Speicherkapazität lauffähig halten. Moderne Compiler bieten auch die Möglichkeit einen zeitoptimierten Code zu erzeugen.

Die Komplexitätsklassen stehen durch ihren unterschiedlichen Zeit- und Speicherverbrauch in Beziehung zueinander. Besonders interessant sind dabei Inklusionsbeziehungen. Sie ermöglichen die Einordnung von Erlernbarkeits- und Nichterlernbarkeitsresultaten.

In dieser Arbeit werden wir häufiger auf komplexitätstheoretische Aussagen stoßen. Zum Beispiel sind Erlernbarkeitsklassen, die wir in der Erlernbarkeitstheorie benötigen, im weiteren Sinn auch Komplexitätsklassen für Entscheidungsprobleme. Der Lernalgorithmus versucht so gesehen, eine spezielle Sprache aus einer Komplexitätsklasse zu erlernen. Da die Sprache durch eine Turingmaschine erkannt wird, versucht der Lernalgorithmus das Verhalten dieser Turingmaschine bis auf einen kleinen

Fehler nachzubilden. Viele der Erlernbarkeitsklassen sind daher aus der Komplexitätstheorie bekannt. Das Wissen um die Beziehungen zwischen diesen Klassen gibt uns die Möglichkeit Erlernbarkeitsresultate auf einfachere und Nichterlernbarkeitsresultate auf komplexere Repräsentationsklassen beziehungsweise Komplexitätsklassen zu übertragen. Dafür verwendet man die Begriffe Reduktion und Vollständigkeit aus der Komplexitätstheorie. Diese Vorgehensweise wird insbesondere in Kapitel 7 verwendet.

Wir benötigen die Komplexitätstheorie auch in der Kryptographie. Um zum Beispiel den Begriff Falltürfunktion zu formalisieren ist es notwendig, die Zeitkomplexität der Falltürfunktion und deren Umkehrfunktion zu vergleichen.

2.1 Einführung

In der Komplexitätstheorie betrachten wir die Zeit- und Speicherkomplexität von Turingmaschinen in Abhängigkeit von der Eingabegröße. Wir betrachten die worst-case Komplexität einer Mehrband-Turingmaschine. Um von konstanten Laufzeitvorteilen zu abstrahieren, benützen wir hierbei die O-Notation. Wir werden nur bei probabilistischen Turingmaschinen[1] leicht von dieser Vorgehensweise abweichen und die schlechteste zu erwartende Laufzeit für eine beliebige Eingabe betrachten. Um auch Speicherkomplexitäten betrachten zu können, die unter der linearen Schranke liegen, wird das Eingabeband als schreibgeschützt angesehen. Ohne diese Vorgehensweise müßte man das Eingabeband während des Einlesens als bearbeitet ansehen und somit könnte der Speicherverbrauch nicht unter $O(n)$ fallen. Gerade in diesem Bereich liegen aber die Klassen, deren nicht effiziente[2] Erlernbarkeit wir nachweisen werden.

Abgesehen von Komplexitätsklassen, die durch den Resourcenverbrauch von Turingmaschinen definiert werden, werden wir auch auf Schaltkreisen beruhende Klassen betrachten. Der Zeitverbrauch entspricht in diesem Fall der Schaltkreistiefe. Der Speicherverbrauch für die Auswertung des Schaltkreises ist auf jeden Fall nach oben durch die maximalen Breite und die Tiefe des Schaltkreises beschränkt, es ist aber nicht ohne weiteres möglich eine direkte Entsprechung zu finden.

Wir unterscheiden zwischen Entscheidungsproblemen und funktionalen Problemen. Bei Entscheidungsproblemen muß die Turingmaschine beziehungsweise der Schaltkreis innerhalb der gegebenen Resourceneinschränkung entscheiden, ob ein Wort in einer Sprache liegt oder nicht. Der Speicherverbrauch einer Turingmaschine ist dann

[1] Eine probabilistische Turingmaschine kann auf einen Zufallsbitgenerator zugreifen. Dies benötigt man zum Beispiel bei einem probabilistischen Public-Key Verfahren

[2] Wir werden in dieser Arbeit häufig das Wort effizient statt polynomialzeitlich verwenden.

die Anzahl der besuchten Speicherzellen und der Zeitverbrauch die Anzahl der ausgeführten Überführungsfunktionen.

Bei funktionalen Problemen muß die Turingmaschine zusätzlich auf dem Ausgabeband den korrekt berechneten Funktionswert ablegen. Ein Schaltkreis wird mit ausreichend vielen Ausgängen versehen um das Ergebnis liefern zu können.

Eine Komplexitätsklasse wird also definiert, indem man Sprachen mit ähnlicher Speicher- bzw. Zeitkomplexität zusammenfaßt. Zum Beispiel beschreibt die Komplexitätsklasse P alle in Polynomialzeit entscheidbaren Sprachen.

Um diese Zusammenhänge formal exakt festlegen zu können, benötigen wir die Definitionen $\mathit{TIME}(f(n))$ und $\mathit{SPACE}(g(n))$. Eine Mehrband-Turingmaschine liegt in $\mathit{TIME}(f(n))$ genau dann, wenn sie für jede Eingabe der Größe n weniger als $f(n)$ Rechenschritte benötigt, und sie liegt in $\mathit{SPACE}(g(n))$, wenn sie weniger als $g(n)$ Speicherzellen, abgesehen von Eingabeband, benutzt. Die formalen Definitionen folgen:

Definition 2.1 $\mathit{TIME}(f(n))$
$\mathit{TIME}(f(n))$ beschreibt die Klasse aller Sprachen S, für die es eine deterministische Mehrband-Turingmaschine gibt, die S erkennt und für jede Eingabe x weniger als $f(|x|)$ Schritte benötigt.

Definition 2.2 $\mathit{SPACE}(\mathbf{g}(\mathbf{n}))$
$\mathit{SPACE}(g(n))$ beschreibt die Klasse aller Sprachen S, für die es eine deterministische Mehrband-Turingmaschine gibt, die S erkennt und für jede Eingabe x weniger als $g(|x|)$ Speicherzellen benötigt. Dabei bleibt das schreibgeschützte Eingabeband unberücksichtigt.

Bei Funktionalen Problemen wird auch das Ausgabeband gesondert behandelt, da die Ergebnisse häufig eine in der Eingabegröße lineare Länge besitzen. Das Ausgabeband ist dann lesegeschützt und wird nicht zum Speicherverbrauch hinzugezählt.

2.2 Komplexitätsklassen und ihre Beziehungen

Für *NTIME* und *NSPACE* gilt dieselbe Definition, jedoch wird statt einer *deterministischen* eine *nichtdeterministische* Mehrband-Turingmaschine verwendet. Da deterministische in nichtdeterministischen Mehrband-Turingmaschinen enthalten sind, ist *NTIME* beziehungsweise *NSPACE* eine Obermenge von *TIME* beziehungsweise *SPACE*.

Die Komplexitätsklasse P oder $PTIME$ beschreibt die Menge aller Sprachen, die in polynomieller Zeit entschieden werden können, also:

$$P = PTIME = \bigcup_{p \text{ ist Polynom}} TIME(p(n))$$

Die Klasse $PSPACE$ beschreibt die Menge aller Sprachen, die unter Verwendung von polynomiell viel Speicher entschieden werden können.

$$PSPACE = \bigcup_{p \text{ ist Polynom}} SPACE(p(n))$$

Für die Klassen NP, $NPTIME$, $NPSPACE$ gelten die Definitionen entsprechend für eine nichtdeterministische Turingmaschine, also mit $NTIME$ und $NSPACE$.

Es ist klar, daß P in NP enthalten ist, da man jede deterministische Turingmaschine auch mit einer nichtdeterministischen Turingmaschine simulieren kann. Die wahrscheinlich wichtigste Frage in der Informatik ist jedoch, ob P eine echte Teilmenge von NP ist, beziehungsweise ob $P = NP$ oder $P \neq NP$.

Dieses sogenannte *P-NP-Problem* wird als Voraussetzung für einige ältere Nichterlernbarkeitsresultate verwendet. Diese Resultate sind praktisch als sicher anzusehen. Für die in dieser Arbeit bewiesenen Resultate werden wir ausnahmslos stärkere kryptographische Annahmen verwenden.

Ein Beispiel für eine Klasse mit geringerem als linearem Speicherverbrauch, ist die Komplexitätsklasse $L = SPACE(O(log(n)))$. Sie beschreibt die Menge aller Sprachen, die durch eine Turingmaschine mit nur logarithmischem Speicherverbrauch entscheidbar sind. Es ist bekannt, daß L eine Teilmenge von P ist, das heißt, daß jede Sprache mit maximal logarithmischem Speicherverbrauch auch in Polynomialzeit entscheidbar ist.

Wir werden besonders im nächsten Abschnitt intensiv auf probabilistische Komplexitätsklassen eingehen, darum geben wir hier noch die Definition der probabilistischen Entscheidbarkeitsklasse PP an.

Definition 2.3 Komplexitätsklasse PP
Eine Sprache S ist in der Komplexitätsklasse PP, wenn es eine nichtdeterministische polynomialzeitbeschränkte Turingmaschine T gibt, so daß für alle Eingaben x gilt: Genau wenn $x \in L$, dann akzeptiert die Turingmaschine T bei Eingabe von x in über der Hälfte der Berechnungen.

Man kann sich überlegen, daß die Klasse NP in der Klasse PP enthalten ist. Angenommen eine Sprache L liegt in NP, dann existiert eine nichtdeterministische Maschine N, die L erkennt. Wir müssen N nur um einen neuen Startzustand ergänzen, der entweder auf den ursprünglichen Startzustand übergeht oder auf einen Zustand, der die gleiche Anzahl an Schritten ausführt, jedoch immer akzeptiert. Genau dann, wenn die Eingabe in der Sprache S liegt, akzeptieren dann mehr als die Hälfte der Berechnungen.

Zusammenfassend gilt also:

$$L \subseteq P \subseteq NP \subseteq PP$$

Wir haben in diesem Abschnitt verschiedene Komplexitätsklassen von Entscheidungsproblemen betrachtet. Wir werden aber häufig funktionale Problemstellungen betrachten. Deshalb werden wir im nächsten Abschnitt die wichtigsten funktionalen Komplexitätsklassen behandeln.

2.3 Berechenbarkeit von Funktionen

Bei funktionalen Komplexitätsklassen betrachten wir die Zeit- und Speicherkomplexität verschiedener Turingmaschinen. Als Eingabe erhalten wir eine binär codierte Zahl aus dem Definitionsbereich der Funktion. Die Turingmaschine soll dann mit dem korrekten Funktionswert auf dem Ausgabeband halten. Im Unterschied zu Entscheidbarkeitsproblemen erhalten wir also nicht eine reine ja/nein Antwort, sondern das Ergebnis einer Funktion.

Die Komplexität einer Funktion ist demnach der Zeit- beziehungsweise Speicherverbrauch des bestmöglichen Algorithmus, der die Funktion berechnet.

In der Erlernbarkeitstheorie und der Kryptographie werden häufig probabilistische Turingmaschinen verwendet. Es reicht dann aus, wenn der Algorithmus im Mittel innerhalb der vorausgesetzten Zeit- und Speichergrenzen erfolgreich ist. Wir werden dies noch formalisieren.

Definition 2.4 Probabilistische Turingmaschine (funktional)
Eine probabilistische Turingmaschine ist eine Turingmaschine mit einem zusätzlichem Leseband. Auf diesem Band sind fair gezogene Zufallsbits gespeichert, die im Laufe der Berechnung die Arbeitsweise der Turingmaschine beeinflussen können.

Man kann sich eine solche Turingmaschine auch mit Zugriff auf ein Zufallsorakel vorstellen, das bei Aufruf das Ergebnis eines fairen Münzwurfs zurückgibt. Wir definieren nun, wann eine Funktion durch eine probabilistische Turingmaschine effizient berechenbar ist.

Definition 2.5 f effizient berechenbar durch eine probabilistische Turingmaschine
Eine probabilistische Turingmaschine berechnet eine Funktion f effizient genau dann, wenn sie für jede Eingabe x mit einer Wahrscheinlichkeit von mindestens $\frac{1}{p(|x|)}$, für ein beliebiges aber festes positives Polynom $p(|x|)$, den Funktionswert $f(x)$ ausgibt und in zu erwartender Polynomialzeit hält.

Wenn das Ergebnis der Turingmaschine überprüfbar ist, kann man durch Test des Ergebnisses und Mehrfachausführung der probabilistischen Turingmaschine in Polynomialzeit die Fehlerwahrscheinlichkeit auf einen beliebig kleinen exponentiellen Anteil reduzieren.

Durch eine Reduktion ist es möglich, aus der Berechenbarkeit einer Funktion durch eine probabilistische Turingmaschine auf die Berechenbarkeit von anderen Funktionen zu schließen. Wir werden diese Reduktion in Kapitel 6 und 7 mehrfach für den Beweisschluß benötigen.

Definition 2.6 effizient probabilistisch polynomialzeitreduzierbar (funktional) $\leq_{\mathbf{epp}}$
Seien f und g Funktionen und G ein Orakel, das g berechnet. Die Berechnung der Funktion f ist effizient probabilistisch polynomialzeitreduzierbar auf die Berechnung der Funktion g, falls es eine probabilistische Turingmaschine F gibt, die als Unterroutine das Orakel G benützen kann und f mit einer Wahrscheinlichkeit von über $\frac{1}{p(n)}$ für ein beliebiges aber festes Polynom $p(n)$ berechnet.

Wenn in der vorangegangenen Definition g durch eine effiziente probabilistische Turingmaschine berechenbar ist, das Ergebnis von g überprüfbar ist und $f \leq_{epp} g$ gilt, existiert eine probabilistische Turingmaschine T_F, die f effizient berechnet. Denn dann können wir die Fehlerwahrscheinlichkeit von g soweit reduzieren, daß bei polynomiell vielen Anfragen praktisch kein Fehler auftreten kann. Wenn wir nun diese Turingmaschine anstelle des Orakels G einsetzen, hält auch die probabilistische Maschine F in zu erwartender Polynomialzeit und gibt ein korrektes Ergebnis aus. Also ist dann auch f durch eine effiziente Turingmaschine berechenbar. Es wird sich später in dieser Arbeit immer um überprüfbare Ergebnisse handeln.

Für die Umkehrrichtung gilt entsprechend, wenn f nicht durch eine effiziente probabilistische Turingmaschine berechenbar ist, g überprüfbar ist und $f \leq_{epp} g$ gilt, dann kann auch g nicht durch eine effiziente probabilistische Turingmaschine berechenbar sein.

2.4 Funktionale Komplexitätsklassen

In diesem Abschnitt betrachten wir verschiedene funktionale Komplexitätsklassen und ihre Beziehungen untereinander.

Zuerst führen wir den NC^1-Schaltkreis ein, der in den Kapiteln 6 und 7 eine entscheidende Rolle spielen wird.

Um Schaltkreise formalisieren zu können benötigen wir noch ein paar Begriffe. Ein Schaltkreis besteht aus einer endlichen Anzahl von Schaltelementen, den Gattern. Jedes Gatter besitzt einen Ausgang, eine beliebige Anzahl von Eingängen und führt eine logische Operation aus, wie zum Beispiel *AND* und *OR*.

Üblicherweise kann der Ausgang mit Eingängen von beliebigen anderen Gattern zyklenfrei verbunden werden, man spricht von einem unbeschränkten *fan-out*. Bei einem unbeschränkten *fan-in* kann jedes Gatter beliebig viele Eingänge gleichzeitig mit *AND* beziehungsweise *OR* verknüpfen, bei *fan-in 2* besitzt jedes Gatter nur 2 Eingänge. Bei den hier behandelten Schaltkreisen kann man sich den Schaltkreis schichtweise aufgeteilt vorstellen, so daß die Gatter einer Ebene nur Ausgangsinformationen der vorhergehenden Ebene benützen dürfen. Im Allgemeinen ist diese einschränkende Annahme jedoch nicht anwendbar.

Der wesentliche Unterschied zu Booleschen Formeln ist, daß eine einmal berechnete Teilinformation an beliebigen Stellen verwendet werden kann. Eine Boolesche Formel ist quasi ein Schaltkreis mit fan-out 1. Eine Boolesche Formel benötigt also im allgemeinen mehr Verknüpfungen als ein Schaltkreis.

Wir definieren nun die Komplexitätsklasse NC^1, die eine Teilklasse der Komplexitätsklasse NC ist.

Definition 2.7 Komplexitätsklasse NC^1
Jeder Schaltkreis aus der NC^1-Schaltkreisfamilie besitzt fan-in 2 *und einen unbeschränkten fan-out. Seine Tiefe ist logarithmisch in n beschränkt. Er besitzt n Eingänge und $p(n)$ Ausgänge, wobei $p(n)$ eine polynomiell in n beschränkte Funktion ist. Es sind AND, OR und NAND, NOR Gatter zugelassen. Die funktionale Komplexitätsklasse NC^1 beschreibt dann die Menge der Funktionen, die durch NC^1-Schaltkreisfamilien berechenbar sind.*

Es könnten auch andere Gatter verwendet werden. Wichtig ist dabei nur, daß alle $(2^2)^2 = 16$ Verknüpfungsmöglichkeiten in konstanter Tiefe durch die verwendeten Gatter erzeugt werden können.

Aus der obigen Definition folgt auch, daß die Anzahl der Gatter in einem NC^1-Schaltkreis polynomiell beschränkt sind. Wir betrachten hierzu einen der $q(n)$

Ausgänge. Da der *fan-in* auf 2 beschränkt ist, kann die Information nur aus maximal 2 Ausgängen der vorhergehenden Ebene stammen, dadurch erhält man von einer Ebene zur nächsten jeweils maximal eine Verdopplung der beteiligten Gatter. Da es nur logarithmisch viele Ebenen gibt, kann sich jeder Ausgang also nur logarithmisch oft verzweigen: Daher können wir über der Anzahl der Gatter auf den $O(\log(n))$ vielen Ebenen summieren. Wenn wir also über die maximale Verzweigunsrate ausgehend von der obersten Schicht summieren, erhalten wir:

$$G = \sum_{i=0}^{k \cdot \log(n)} q(n) \cdot 2^i = q(n) \cdot O(n^k)$$

Es können also nicht mehr als polynomiell viele Gatter an der Berechnung des Ergebnisses beteiligt sein.

Die Nichterlernbarkeitsresultate aus den Kapiteln 6 und 7 werden zuerst für diese Klasse gezeigt. Es ist daher interessant auch die nächstkleinere Klasse AC^0 zu betrachten, für die diese Resultate nicht gelten werden.

Wir definieren nun die Komplexitätsklasse AC^0, die eine Teilklasse der Komplexitätsklasse AC ist.

Definition 2.8 AC^0**-Schaltkreis**
Jeder Schaltkreis aus der Schaltkreisfamilie AC^0 besitzt einen unbeschränkten fan-in und einen unbeschränkten fan-out. Seine Tiefe ist konstant zur Eingabegröße n und die Gatteranzahl ist polynomiell beschränkt. Er besitzt n Eingänge und $p(n)$ Ausgänge, wobei $p(n)$ eine polynomiell beschränkte Funktion ist. Es sind AND, OR und NAND, NOR Gatter zugelassen. Die funktionale Klasse AC^0 beschreibt dann die Menge der Funktionen, die durch AC^0-Schaltkreise berechenbar sind.

Die Klassen AC^0 und NC^1 unterscheiden sich also nur in den Eigenschaften *fan-in* und Tiefe. Da man ein *AND* beziehungsweise *OR* Gatter mit n Eingängen durch einen *fan-in* 2 Schaltkreis der Tiefe $\log(n)$ nachbilden kann, ist AC^0 eine Teilklasse von NC^1.

Die Klasse NC^1 kann im uniformen Modell auch durch eine logarithmisch Speicherbeschränkte Turingmaschine ausgewertet werden. ([Pap94] Seite:395). Also liegt NC^1 in $L = SPACE(\log(n))$.

Zusammenfassend gilt demnach für Entscheidungsprobleme

$$AC^0 \subseteq NC^1 \subseteq L \subseteq P \subseteq NP \subseteq PP$$

und für funktionale Probleme.

$$AC^0 \subseteq NC^1 \subseteq L \subseteq P$$

3

Grundlagen der Kryptographie

Die Wurzeln der Kryptographie lassen sich bis etwa ins fünfte Jahrhundert vor Christus zurückverfolgen. Damals benutzten die Spartaner die Skytala, einen Stab der spiralförmig mit einem Papyrusstreifen umwickelt und dann längs des Stabes beschrieben wurde. Die Anzahl der Zeilen, die man auf dem Stab unterbringen kann, berechnet sich aus seinem Umfang geteilt durch den verwendeten Zeilenabstand und entspricht dem verwendeten Schlüssel in einem Kryptosystem.

Die Skytala mußte transportfähig und lesbar sein. Daher war die Anzahl der möglichen Schlüssel auf ein Minimum beschränkt. Die Sicherheit des Systems beschränkte sich damit im Wesentlichen auf das Wissen um das Verfahren selbst.

Heute geht man jedoch davon aus, daß das verwendete Verschlüsselungssystem einem potentiellen Kodebrecher bekannt ist. Dies ist mit Sicherheit eine notwendige Annahme, da uns die Geschichte lehrt, daß solche Geheimnisse nicht von Dauer sein können.

Es ist also erforderlich den Schlüsselraum drastisch zu vergrößern. Dies wurde auch schon früh erkannt und in späteren Systemen sind Worte oder sogar ganze Sätze als Schlüssel verwendet worden. Trotzdem waren diese Systeme, auf die wir hier jedoch nicht näher eingehen wollen, immer noch nicht besonders sicher. Das lag im Wesentlichen an der einfachen Berechnungsstruktur der Verfahren, was aber zu dieser Zeit aufgrund fehlender Rechenmaschinen kaum zu umgehen war.

Typische Verschlüsselungsfunktionen waren die Vertauschung der Buchstabenfolge, wie bei der Skytala, und die Zuordnung anderer Symbole für die verwendeten Zeichen des Klartextes. Hierbei bleiben aber einige Eigenschaften der Klartextsprache erhalten, wie zum Beispiel die Buchstabenhäufigkeit. Unter Zuhilfenahme dieser Eigenschaften lassen sich diese sogenannten klassischen Kryptosysteme leicht brechen.

Dieses Manko versuchte man dann durch Produktchiffren, die wiederholte Ausführung von einfachen Grundfunktionen, in den Griff zu bekommen. Der *DES* (Data Encryption Standard 1977) gehört zu diesen Systemen und wird auch heute noch benutzt.

Produktchiffren können zwar sehr sicher sein, lösen jedoch nicht die Hauptprobleme klassischer Kryptoverfahren. Wenn zwei Personen miteinander kommunizieren wollen, muß zuvor ein geheimer Schlüssel auf einem abhörsicheren Kommunikationsweg ausgetauscht werden. Zudem benötigt man für n Teilnehmer, die untereinander beliebig Nachrichten austauschen wollen n^2 Schlüssel.

Die Lösung dieser Probleme begann erst 1976 mit der Einführung der Public-Key Kryptographie von Whitfield Diffie und Martin Hellman [DH76]. Der erste Vertreter dieser Gattung war das *RSA* Verfahren (Rivest-Shamir-Adelman), das in diesem Abschnitt noch genauer behandelt wird.

Jeder Teilnehmer benötigt nur noch ein Schlüsselpaar, einen geheimen und einen öffentlichen Schlüssel. Ein Teilnehmer A, der eine Nachricht an einen Teilnehmer B schicken will, liest den öffentlichen Schlüssel e_B von B aus der frei zugänglichen Schlüsseldatei, in der die öffentlichen Schlüssel abgelegt sind, verschlüsselt seine Nachricht mit e_B und schickt sie an B. B kann dann mit seinem geheimen Schlüssel d_B die Nachricht entschlüsseln. Es ist also nicht mehr nötig, einen abhörsicheren Kanal zum Austausch der Schlüssel zu verwenden und die Anzahl der benötigten Schlüssel für n Teilnehmer ist n.

Bemerkenswert ist die Tatsache, daß die Verschlüsselungsfunktion dabei eindeutig durch den öffentlichen Schlüssel definiert und in Polynomialzeit berechenbar ist, wobei jedoch Ihre Umkehrfunktion nicht in Polynomialzeit berechenbar sein darf, da sonst jeder unter Zuhilfenahme des öffentlichen Schlüssels die Nachricht entschlüsseln könnte. Aus diesem Grund liegen allen Public-Key Systemen Falltürfunktionen zugrunde. Das sind Funktionen, deren Umkehrfunktion nur mit Hilfe einer zusätzlichen Information, Falltürinformation genannt, effizient berechenbar ist. Diese Falltürinformation entspricht dann dem geheimen Schlüssel.

Beim *RSA* Verfahren wird die modulare Potenzierung als Falltürfunktion verwendet. Wir werden im nächsten Abschnitt auf diese Funktionsklasse noch näher eingehen.

Leider existiert für das *RSA* Verfahren bisher noch kein Beweis, mit dem sich seine Sicherheit gegen Angriffe abschätzen läßt. Im Gegensatz zum Rabin Verfahren, von dem bewiesen wurde, daß es mindestens so schwer ist Blum-Integer zu faktorisieren[1], wie das Verfahren zu brechen. Diese Public-Key Verfahren werden ausführlich in

[1] Blum-Integer faktorisieren ist ein Problem aus der Zahlentheorie und wird in Abschnitt 3.1.1 vorgestellt.

Abschnitt 3.2 beschrieben, vorher liefert Abschnitt 3.1.1 noch den notwendigen zahlentheoretischen Hintergrund.

3.1 Grundlagen für Public-Key Verfahren

Die bei allen Kryptosystemen angestrebte Eigenschaft ist die Informationstheoretische Sicherheit.

Definition 3.1 Informationstheoretische Sicherheit
Ein Kryptosystem ist informationstheoretisch sicher genau dann, wenn für alle Klartexte m und alle Kryptotexte c gilt:

$$p(m) = p(m|c)$$

Dabei ist $p(m)$ die Wahrscheinlichkeit, daß in dem Kryptosystem der Klartext m verschlüsselt wird und $p(m|c)$ die Wahrscheinlichkeit, daß bei vorliegendem Kryptotext c der Klartext m verschlüsselt wurde.

Das bedeutet, daß ein Kryptoanalytiker durch das Vorlegen des Kryptotextes keine weitere Information über den verschlüsselten Klartext erhält. Diese sehr hohe Anforderung an ein Kryptosystem kann durch ein *one-time tape* erreicht werden. Hierbei wird ein zu verschlüsselnder Klartext in Form einer binären Zeichenkette kodiert, indem er mit einer speziell für diese einmalige Verschlüsselung generierten zufälligen binären Zeichenkette mit *XOR* verknüpft wird. Dabei ist zu beachten, daß auch die Länge eines solchen Kryptotextes Informationen über den Klartext preisgeben kann. Daher sollte man nur Nachrichten mit fester Länge verschicken.

Dieser Verschlüsselung besitzt jedoch zwei grundlegende Probleme. Erstens muß für jede einzelne Nachricht eine zufällige Zeichenkette erzeugt und auf einem abhörsicheren Weg übermittelt werden. Zweitens kann man in der Praxis im allgemeinen nur pseudozufällige Zeichenketten erzeugen. Der Kodebrecher hat dann die Möglichkeit sich die speziellen Eigenschaften des Zufallsgenerators zunutze zu machen. Daher beschränkt man sich auf schwächere Sicherheitsdefinitionen, die wir später noch einführen werden (Abschnitt 3.3) und benutzt Public-Key Verfahren.

Definition 3.2 Public-Key Verfahren
In einem Public-Key System erhält jeder Benutzer einen privaten und einen öffentlichen Schlüssel. Auf die öffentlichen Schlüssel kann jeder Teilnehmer zugreifen und auf die privaten nur der jeweilige Benutzer. Es gibt eine Verschlüsselungsfunktion

$E(M, K_p) = C$, die als Eingabe den öffentlichen Schlüssel des Adressaten erhält, und eine Entschlüsselungsfunktion $D(C, K_q) = M$, die als Eingabe den privaten Schlüssel benötigt.

Wie wir bereits wissen, benutzen Public-Key Verfahren Falltürfunktionen um Sicherheit gegen Kodebrecher zu gewährleisten. Die Falltürfunktionen beruhen auf Annahmen aus der Zahlentheorie, die wir jetzt vorstellen werden.

3.1.1 Zahlentheorie

In diesem Abschnitt wird der für die Kryptosysteme notwendige zahlentheoretische Hintergrund behandelt. Allgemeine Definitionen dieses Abschnitts findet man in [BS91] und [Sch96]. Im Anhang von [BSW91] und in [Kob94] sind alle für das Gebiet der Kryptographie interessanten zahlentheoretischen Grundlagen zusammengefaßt.

Grunddefinitionen

Wir bezeichnen mit $\mathbf{N} = \{1, 2, 3, \ldots\}$ die Menge der *natürlichen Zahlen* und mit $\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ die Menge der *ganzen Zahlen*.

Eine ganze Zahl $a \in \mathbf{Z}$ teilt eine ganze Zahl $b \in \mathbf{Z}$ - wir schreiben $a|b$ - wenn eine ganze Zahl $k \in \mathbf{Z}$ existiert, so daß gilt: $a \cdot k = b$. Dabei ist zu beachten, daß für jede ganze Zahl a gilt: $a|0 \Leftrightarrow a \cdot 0 = 0$. Außerdem gilt $a|b \Leftrightarrow \pm a| \pm b$.

Modulare Arithmetik

Besonders wichtig in der Kryptographie ist die Modulo-Rechnung. Es gilt $a \bmod n = b$ genau dann, wenn b die kleinste natürliche Zahl ist, für die gilt: $n|(a-b)$.

Für die Modulare Arithmetik benötigen wir noch die Begriffe *kongruent modulo* und *Äquivalenzklasse*.

In der Kryptographie wird jedoch nicht mit der Menge der *Natürlichen Zahlen* gerechnet, da diese Zahlen auf einem Computer mit endlichem Speicher nicht darstellbar sind. Man verwendet statt dessen *Restklassen*.

Zwei Zahlen $a, b \in \mathbf{N}$ sind *kongruent modulo* n ($a \equiv_n b$) genau dann, wenn $n|(a-b)$. Alternativ kann man die Kongruenz zweier Zahlen $a, b \in \mathbf{Z}$ überprüfen, indem man den Term $a \mod n = b \mod n$ auswertet.

Durch die *Äquivalenzrelation* $a \equiv_n b$ wird $\mathbf{Z}$ in n *Äquivalenzklassen* unterteilt $[a]_n = \{b \in \mathbf{Z} | a \equiv_n b\}$. Zum Beispiel ist $[0]_2$ die Menge der geraden Zahlen und $[1]_2$ die Menge der ungeraden Zahlen.

Für jede natürliche Zahl $n \in \mathbf{N}$ ist Z_n eine *Restklasse* wobei gilt:

$$Z_n = \{[0]_n, [1]_n, \ldots, [n-1]_n\}$$

Die Addition, Subtraktion und Multiplikation ist wie folgt definiert:

$$\begin{aligned} a \oplus b &:= (a+b) \mod n, \\ a \ominus b &:= (a-b) \mod n, \\ a \otimes b &:= (a \cdot b) \mod n \end{aligned}$$

Wie nachfolgende Tabelle zeigt bildet Z_n zusammen mit dem $\oplus$ Operator eine kommutative Gruppe.

abgeschlossen	$a, b \in Z_n \Rightarrow (a \oplus b) \in Z_n$
assoziativ	$a, b, c \in Z_n : (a \oplus b) \oplus c = a \oplus (b \oplus c)$
neutrales Element	$a \in Z_n : [0]_n \oplus a = a$
inverses Element	$a \oplus (n-a) = a + (n-a) \mod n = n \mod n = 0$
kommutativ	$a, b \in Z_n : a \oplus b = b \oplus a$

Um auch mit dem $\otimes$ Operator eine *kommutative Gruppe* zu erhalten, müssen wir den Wertebereich einschränken. Hierfür definieren wir die Gruppe:

$$Z_n^* = \{a \in Z_n | ggT(a, n) = 1\}$$

Die Eulersche φ-Funktion entspricht der Mächtigkeit der Menge Z_n^*.

$$\varphi(n) = |Z_n^*|$$

Es ist klar, daß die Gruppe Z_n^* assoziativ und kommutativ ist und $[1]_n$ als neutrales Element enthält (Z_0 bildet die Ausnahme und besitzt das neutrale Element $[0]_n$). Für zwei Zahlen $a, b \in Z_n^*$ gilt $ggT(a, n) = ggT(b, n) = 1$ und daraus folgt, daß a und b teilerfremd zu n sind. Dann ist aber auch das Produkt $a \cdot b$ teilerfremd zu n und somit $\text{ggT}(a \cdot b, n) = 1$ und $\text{ggT}(a \cdot b \mod n, n) = 1$. Also ist die *multiplikative Gruppe* auch abgeschlossen.

Um ein inverses Element von a in Z_n^* zu erhalten, berechnen wir mit dem Erweiterten Euklidschen Algorithmus (siehe auch [BSW91], Seite 115) eine Linearkombination $\lambda \cdot a + \mu \cdot n = 1$. Damit ergibt sich das inverse Element von a zu $\lambda \mod n$.

Solche multiplikativen Gruppen besitzen einige für die Kryptographie sehr nützliche Eigenschaften. Zum Beispiel ist es im Allgemeinen nur mit großem Aufwand (Exponentiell in der Eingabelänge) möglich von einer Zahl in Z_n^* zu erkennen, ob sie eine Wurzel modulo n besitzt. Für die natürlichen Zahlen hingegen existieren effiziente Algorithmen.

Modulare Potenzierung

Man kann eine beliebige Exponentialfunktion g^x durch aufeinanderfolgendes Multiplizieren und Quadrieren berechnen. Zum Beispiel gilt $13^{11} = 13^{1011_2} = (((13^2)^2) \cdot 13)^2 \cdot 13$. Wie man sieht, wird für die Transformation die Binärdarstellung des Exponenten verwendet. Wenn ein modulares Ergebnis verlangt ist, kann man in jedem Zwischenschritt den Modul bilden.

Wurzel ziehen modulo n

Wie schwer es ist eine Quadratwurzel modulo n zu berechnen, hängt von der Primfaktorzerlegung von n ab. Ist n eine Primzahl, so existieren effiziente Verfahren. Wenn n jedoch das Produkt zweier großer unbekannter Primzahlen p und q ist, dann ist es im allgemeinen nicht in Polynomialzeit (gemessen an der Länge von n) möglich eine Quadratwurzel modulo n zu bestimmen.

Wir werden im Abschnitt 6 einen Spezialfall dieses Problems benötigen. Dabei ist $n = p \cdot q$ eine Blum-Integer.

Definition 3.3 Blum-Integer
Eine Blum-Integer ist eine aus zwei Primzahlen p und q zusammengesetzte Zahl $n = p \cdot q$. Zusätzlich gilt für die zwei Primzahlen $p \equiv_4 3$ und $q \equiv_4 3$.

Annahme 3.1 Annahme der nicht effizienten Faktorisierbarkeit von Blum-Integern
Sei $n = p \cdot q$ eine Blum-Integer der Länge l. Sein $p(l)$ ein beliebiges aber festes Polynom. Dann gilt für jeden probabilistischen Algorithmus A, der als Eingabe nur n erhält, für unendlich viele n, daß

$$Pr[A(n) = p \; oder \; q] < \frac{1}{p(l)}.$$

Die Wahrscheinlichkeit wird über der Variable n, die gleichverteilt aus zwei Primzahlen erzeugt wird, und über dem Zufallsverhalten des Algorithmus A berechnet.

Man geht davon aus, daß eine Blum-Integer nicht effizient bezüglich ihrer Länge faktorisierbar ist. Diese Eigenschaft wird in vielen Kryptosystemen verwendet. Zum Beispiel beruht die Sicherheit des Rabin Verfahrens nachweisbar auf der Faktorisierbarkeit von Blum-Integern.

Die Wurzel aus einer Zahl a modulo einer Blum-Integer n läßt sich ermitteln, indem man

$$\pm a^{\frac{p+1}{4}} \mod p \,\text{und}\, \pm a^{\frac{q+1}{4}} \mod q$$

berechnet, und dann die Ergebnisse mit Hilfe des chinesischen Restsatzes zu vier Ergebnissen verbindet.

Der Chinesische Restsatz

Mit Hilfe des Chinesischen Restsatzes ist es möglich, für ein modulares lineares Kongruenzgleichungssystem unter bestimmten Voraussetzungen eine gemeinsame Lösung zu finden. Die genaue Vorgehensweise findet sich zum Beispiel in [Kob94] auf Seite 35 und in [BSW91] auf Seite 115.

Quadratische Reste

Ein Element aus der invertierbaren Restklasse $a \in Z_n^*$ ist ein *Quadratischer Rest* genau dann, wenn ein $b \in Z_n^*$ existiert für das gilt:

$$b^2 \equiv_n b \cdot b \equiv_n a$$

Falls dieses b nicht existiert, nennt man a einen *Quadratischen Nichtrest*.

Umgangssprachlich formuliert bedeutet dies also, daß eine Zahl $a \in Z_n^*$ genau dann ein quadratischer Rest ist, wenn eine Wurzel von a in Z_n^* existiert.

Die Menge QR_n beschreibt dann die Menge aller Quadratischen Reste modulo n und QNR_n die Menge aller Quadratischen Nichtreste.

Das *Legendre Symbol* $L(x,p)$ ist auf allen Primzahlen p und allen $x \in Z_p^*$ definiert. Es gilt:

$$L(x,p) \equiv_p \begin{cases} 1 & \text{wenn } x \in QR_p \\ -1 & \text{sonst} \end{cases}$$

Das Legendre Symbol läßt sich effizient durch die Formel

$$L(x,p) = x^{\frac{p-1}{2}}$$

berechnen.

Für den Spezialfall, daß n das Produkt zweier Primzahlen p und q ist, definiert man das Jakobisymbol $J(x,n) = L(x,p) \cdot L(x,q)$. Die vollständige Definition des Jakobisymbols werden wir in dieser Arbeit nicht benötigen.

Wir benötigen nun nur noch eine Untergruppe von Z_n^* um einige interessante Eigenschaften der Quadratischen Reste angeben zu können.

$$Z_n^*(+1) = \{x | x \in Z_n^* \text{ und } J(x,n) = 1\}$$

Es gelten nun folgende Eigenschaften, die wir in Kapitel 7 benötigen werden.

1. Das Jakobisymbol $J(x,n)$ ist für ein $x \in Z_n^*$ und ein $n \in Z$ effizient berechenbar.
2. Wenn p und q Primzahlen sind und $n = p \cdot q$, dann besitzen genau die Hälfte der Zahlen auf Z_n^* das Jakobisymbol 1 ($|Z_n^*(+1)| = |Z_n^*|/2$). Weiter sind genau ein Viertel der Zahlen aus Z_n^* Quadratische Reste ($|QR_n| = |Z_n^*|/4$).
3. Für alle $x \in Z_n^*$ gilt nach dem Satz von Euler-Fermat die Gleichung $x^{\varphi(n)} \equiv_n 1$.

Mit diesen Eigenschaften und dem folgenden Lemma kann man Quadratische Reste modulo einer zusammengesetzten Zahl erkennen.

Lemma 3.1 Quadratische Reste erkennen
Sei $n = p \cdot q$ aus zwei Primzahlen zusammengesetzte Zahl. Dann ist $a \in Z_n^*$ genau dann ein Quadratischer Rest modulo n, wenn er sowohl zur Basis p wie auch zu Basis q ein Quadratischer Rest ist.

Ohne diese Information ist es jedoch schwer, Quadratische Reste zu erkennen.

Für eine ausreichende Problemgröße k läßt sich also kein polynomiellgroßer Schaltkreis finden, der für einen polynomiellkleinen Anteil der zusammengesetzten $2k$-Bit Primzahlen n entscheidet, ob ein $x \in Z_n^*(+1)$ ein Quadratischer Rest modulo n ist. Es gibt Sätze, die dieses Resultat weiter verschärfen. (siehe auch [GM83]).

Wurzel ziehen modulo n

Für alle Blum-Integer $n = p \cdot q$ und alle Quadratischen Reste $a \in QR_n$ kann man die Kongruenzgleichung $x^2 \equiv_n a$ lösen. Man berechnet hierfür zuerst die Wurzeln von a modulo p und q. Hierfür verwendet man die Formeln:

$$x_{p1,2} \equiv_p \pm a^{\frac{p+1}{4}}$$
$$x_{q1,2} \equiv_q \pm a^{\frac{q+1}{4}}$$

Die Ergebnisse $x_{p1,2}$ und $x_{q1,2}$ lassen sich dann durch den Chinesischen Restsatz zu vier Resultaten modulo n verbinden. Genau eines dieser Ergebnisse ist ein quadratischer Rest.

3.1.2 Einweg und Falltürfunktionen

Obwohl man durch die Verwendung von Falltürfunktionen in Public-Key Verfahren viele Probleme lösen konnte, besteht immer die Möglichkeit eines gewählten Klartextangriffs. Interessanterweise wird genau diese Eigenschaft verwendet, um die Nichterlernbarkeit von Konzeptklassen (siehe auch Kapitel 6) nachzuweisen.

Eine *Einwegfunktion* ist eine Funktion, die effizient berechenbar ist, deren Umkehrfunktion jedoch nicht. Eine *Falltürfunktion* ist eine Einwegfunktion, die unter Verwendung einer Zusatzinformation effizient invertierbar wird. Es ist bis heute nicht klar, ob Einwegfunktionen überhaupt existieren. Trotzdem werden die Funktionen diskrete Potenzfunktion, Quadrieren modulo n, Erzeugen von Quadratischen Resten/Nichtresten unter Verwendung eines Quadratischen Nichtrestes modulo n und Produktbildung zweier Primzahlen, als Falltürfunktionen angesehen.

Es ist aufgrund der Quadratischen Reste Annahme (Abschnitt 6.1) im allgemeinen nicht möglich, für einer zusammengesetzten Zahl $n = p \cdot q$ und ein $x \in Z_n^*(+1)$ in Polynomialzeit zu entscheiden, ob x ein quadratischer Rest modulo n ist oder nicht.

Wenn wir einen Quadratischen Nichtrest $y \in QNR_n$ zur Verfügung haben, können wir durch Erzeugen eines $x \in Z_n^*$ ohne weiteres Quadratische Reste durch die Formel $x^2 \mod n$ und Quadratische Nichtreste durch die Formel $x^2 \cdot y \mod n$ erzeugen. Die Umkehrfunktion, also das Erkennen ob ein gegebenes $x \in Z_n^*$ ein Quadratischer Nichtrest ist oder nicht, ist laut der Quadratischen Reste Annahme nicht effizient möglich. Wir haben also eine Einwegfunktion konstruiert. Durch Hinzufügen der Falltürinformation (p, q) kann man jedoch Quadratische Reste erkennen, indem man die Gleichungen $x^{\frac{p-1}{2}} \equiv_p 1$ und $x^{\frac{q-1}{2}} \equiv_q 1$ überprüft. Diese Falltürfunktion wird im Verfahren benützt, das im Kapitel 3.3.1 erläutert wird.

Man kann ohne weiteres durch Verwendung der modularen Potenzfunktion die Gleichung $g^k \mod n$ effizient auswerten. Man nimmt an, daß die Umkehrfunktion sich nicht ohne die Verwendung der Faktorisierung von $n = p \cdot q$ effizient berechnen läßt. Diese Eigenschaft benützt das *RSA* Verschlüsselungsverfahren.

Wir benötigen hierbei häufig Primzahlen. Daß sich solche großen Primzahlen effizient erzeugen lassen, kann man zum Beispiel in [Sch96] nachlesen. Mögliche Verfahren sind der Solovay-Strassen und Rabin-Miller Test.

3.2 Einige Kryptosysteme

3.2.1 Public-Key Systeme

Das *RSA* System

Die Sicherheit des *RSA* Verfahrens beruht auf der modularen Potenzfunktion. Zu Erzeugung der Falltürinformation wird das Faktorisierungsproblem verwendet. Man wählt zuerst eine aus zwei großen Primzahlen p und q zusammengesetzte Zahl $n = p \cdot q$. Dann wählt man einen Verschlüsselungsexponenten $e \in Z_n^*$. Daraufhin löst man die Kongruenzgleichung

$$e \cdot d = 1 \mod \varphi(n) \Leftrightarrow e \cdot d = 1 \mod (p-1)(q-1)$$

Der Schlüsselerzeuger kann nun die Faktorisierung von n aus seinem Speicher löschen, da sie nicht mehr benötigt wird. Denn es gilt

$$\begin{aligned} (x^e)^d &\equiv_n x^{ed} \\ &\equiv_n x^{k\varphi(n)+1} \\ &\equiv_n \left(x^{\varphi(n)}\right)^k \cdot x \equiv_n x \end{aligned}$$

Der öffentliche Schlüssel besteht aus dem Tupel (e, n) und der private aus der Falltürinformation d. Das Verfahren benützt die Verschlüsselungsfunktion $E(x, (e, n)) = x^e \mod n$ und Entschlüsselungsfunktion $D(y, (d, n)) = x^d \mod n$.

Annahme 3.2 Annahme der Nichtinvertierbarkeit der *RSA* Verschlüsselungsfunktion

Sei $n = p \cdot q$ eine aus zwei Primzahlen der Länge l zusammengesetzte Zahl, e ein RSA-Verschlüsselungsexponent und x ein Klartext aus Z_n. Dann besagt die Annahme der Nichtinvertierbarkeit der RSA Verschlüsselungsfunktion, daß für jeden probabilistischen Algorithmus A und für unendlich viele l gilt:

$$Pr[A(RSA(x,n,e),n,e) = x] < \frac{1}{2} + \frac{1}{p(l)}$$

Die Wahrscheinlichkeit wird dabei über den gleichverteilt aus Z_n gezogenen Klartexten x und dem Zufallsverhalten des Algorithmus berechnet.

Das *RSA* Verfahren ist im Laufe der Zeit sehr genau untersucht worden und es sind inzwischen viele Fallstricke und deren Vermeidung bekannt. Diese Gefahrenquellen sollten bei einer Implementation des *RSA* Systems berücksichtigt werden. Einige Angriffsmöglichkeiten davon sind in [Sch96] auf Seite 470ff beschrieben. Zum Beispiel ist das *RSA* Verfahren nicht sicher gegen gewählte Kryptotextangriffe, wenn gewisse Sicherheitsmaßnahmen nicht eingehalten werden. Trotz dieser Schwächen gehört das *RSA* Verfahren zu den sichersten heute bekannten Kryptosystemen und wird unter anderem für Passwortdateien verwendet.

Das Rabin Verfahren

Das Rabin Verfahren beruht auf dem Problem, Wurzeln modulo einer Blum-Integer zu finden. Daher besitzt das Verfahren gegenüber dem *RSA* Verfahren den Vorteil, daß nachgewiesen werden konnte, daß es möglich wäre Blum-Integer zu faktorisieren, wenn man das Verfahren brechen könnte.

Wie schon in Abschnitt 3.1.1 beschrieben, kann man für jeden Quadratischen Rest $a \in QR_n$ modulo einer Blum-Integer n unter Verwendung der Faktorisierung von n genau 4 Quadratwurzeln berechnen, von denen eine ein Quadratischer Rest ist. Das Rabin Verfahren nutzt die Eigenschaft aus, daß man die Wurzeln nur mit Hilfe der Faktorisierung von n (nach heutigem Wissen) effizient berechnen kann. Es bleibt noch das Problem, aus den 4 möglichen Wurzeln die korrekte zu identifizieren.

Man wählt einen öffentlichen Schlüssel (b, n), wobei n eine Blum-Integer ist und $b \in Z_n$. Der Geheime Schlüssel ist die Faktorisierung von $n = p \cdot q$. Die Verschlüsselungsfunktion besteht dann aus den drei Informationen:

$$E(x,(b,n)) = x(x+b), (J(x + 2^{-1}b, n), x + 2^{-1}b \leq (n-1)/2 \mod n).$$

Die zwei Zusatzinformationen ermöglichen die genaue Identifizierung der Klartextnachricht beim Entschlüsseln und sie werden durch ein spezielles Verfahren mitverschlüsselt. Bei diesem Verfahren werden die zwei Zusatzinformationen des nachfolgenden Datenblocks im jeweils vorangegangenen Datenblock verschlüsselt. Dabei ist es notwendig vor dem ersten Datenblock einen Dummyblock zu verschlüsseln, der die Zusatzinformationen des ersten Datenblocks enthält und unabhängig von diesen selbst immer die selben Zusatzinformationen benötigt. Um einen Kryptotext c zu entschlüsseln, löst man die quadratische Kongruenzgleichung (siehe auch Abschnitt 3.1.1)

$$(x + 2^{-1}b)^2 \equiv_n (2^{-1}b)^2 + c$$

und identifiziert die korrekte Lösung durch die Zusatzinformationen (siehe auch [Kob94]).

Wir werden der Vollständigkeit halber noch den Satz über die Sicherheit des Rabinsystems angeben, da er in Kapitel 6 verwendet wird. Der Satz folgt direkt aus folgendem Lemma:

Lemma 3.2 Faktorisieren durch Lösen einer quadratischen Kongruenzgleichung
Wenn y und z zwei Lösungen der Kongruenzgleichung $x^2 \equiv_n a$ für ein beliebiges a sind und $y \not\equiv_n \pm z$, ist $ggT(y + z, n)$ ein Faktor von n.

Beweis:

Es gilt $y^2 \equiv_n z^2 \Leftrightarrow y^2 - z^2 \equiv_n 0$. Daher existiert ein t für das gilt:

$$y^2 - z^2 = (y + z)(y - z) = tm$$

Weder $y + z$ noch $y - z$ können durch n teilbar sein. Also gilt:

$$ggT(x + z, n) \notin \{1, m\}$$

und somit ist $y + z$ ein nichttrivialer Teiler von n. $\square$

Wir nehmen nun an, daß ein Algorithmus A existiert, der die Wurzel aus einer Zahl $x \in Z_n^*$ effizient berechnen kann (Dieser Algorithmus wäre also insbesondere in der Lage, das Rabinsystem zu brechen). Dann konstruieren wir einen Faktorisierungsalgorithmus, indem wir zufällige $y \in Z_n^*$ erzeugen und den Algorithmus A aus y^2 die

Wurzel z ziehen lassen. Wenn y ungleich $\pm z$ ist, können wir durch $ggT(x+y, n)$ einen nichttrivialen Faktor von n berechnen. Im anderen Fall beginnen wir von vorn. Nach k Schleifendurchläufen ist die Wahrscheinlichkeit n faktorisiert zu haben $1 - 2^{-k}$.

Es ist also mindestens so schwer, das Rabin Verfahren zu brechen, wie das Produkt zweier großer Primzahlen zu faktorisieren.

3.2.2 Probabilistische Public-Key Verfahren

In einem Probabilistischen Public-Key Verfahren können einzelne Klartextnachrichten zu verschiedenen Kryptotexten verschlüsselt werden. Welcher der Kryptotexte dabei von der probabilistischen Verschlüsselungsfunktion ausgegeben wird, hängt vom Zufallsverhalten des Verschlüsselungsalgorithmus ab. Da die Entschlüsselung der möglichen Kryptotexte wieder eindeutig auf den verschlüsselten Klartext zurückführen muß, sind die Kryptotexte länger als die Klartexte. Man verschlüsselt daher meist nur ein Bit zu einem Kryptotext der Länge n, wobei n der Sicherheitsparameter des Systems ist.

Eine Möglichkeit ein solches System zu realisieren ist die Verwendung der Quadratischen Reste Annahme ([GM83]). Man wählt eine aus zwei Primzahlen zusammengesetzte Zahl $n = p \cdot q$ und verwendet die Faktorisierung von n als geheimen Schlüssel. Als öffentlichen Schlüssel verwendet man einen quadratischen Nichtrest $y \in QNR_n$ und n. Ein 1-Bit verschlüsselt man zu einem Quadratischen Rest und ein 0-Bit zu einem Quadratischen Nichtrest aus der Menge $Z_n^*(+1)$. Die Entschlüsselung kann dann unter Zuhilfenahme des geheimen Schlüssels wie in Abschnitt 3.1.1 beschrieben erfolgen.

Auf dieser Grundlage ist es sogar möglich, ein gegen gewählte Kryptotextangriffe sicheres Public-Key Verfahren zu konstruieren (siehe auch Abschnitt 3.3.1).

3.3 Angriffsmöglichkeiten auf ein Kryptosystem

Folgende Angriffshierarchie beschreibt die üblichen Angriffe auf ein Kryptosystem. Im Folgenden sind immer Polynomialzeitalgorithmen gemessen am Sicherheitsparameter des Systems gemeint.

1.) Bekannter Kryptotext (BKr)
Der Angreifer kennt einige Kryptotexte c_i, die mit dem selben Schlüssel verschlüsselt worden sind. Die Aufgabe besteht darin, möglichst viele der

Kryptotexte zu entschlüsseln oder sogar den verwendeten Schlüssel zu bestimmen.

2.) Bekannter Klartext (BKl)
Der Angreifer besitzt mehrere Klartext- Kryptotextpaare. Er soll nach der Analyse einen neuen Kryptotext entschlüsseln können.

3.) Gewählter Klartext (GKl)
Der Angreifer besitzt nun die Möglichkeit selbst beliebige Klartexte zu verschlüsseln. Nach der Bearbeitungsphase soll er Nachrichten entschlüsseln können.

4.) Gewählter Kryptotext (GKr)
Bei einem gewählten Kryptotextangriff kann der Angreifer selbst gewählte Kryptotexte entschlüsseln. Nachdem er keinen Zugriff mehr auf den Entschlüsselungsalgorithmus besitzt, muß er dann einen Kryptotext entschlüsseln. Man nennt einen solchen Angriff auch Mitternachtsattacke, da der Kodebrecher für einen gewissen Zeitraum Zugriff auf die sonst unzugängliche Entschlüsselungsfunktion hat.

Häufig wird auch schon das Ermitteln einer beliebigen Teilinformation über den Klartext durch den Angreifer als gültiger Angriff gewertet. Es reicht dann zum Beispiel aus, daß ein gewählter Kryptotextangreifer das niederwertigste Bit des Klartextes ermitteln kann um die Unsicherheit des Verschlüsselungssystems nachzuweisen.

Diese Angriffsmöglichkeiten sind jedoch nicht die einzig möglichen. Auch die Verfälschung von Nachrichten oder das Abschicken einer Nachricht unter falscher Absenderadresse sind weitere Gefahrenquellen für Kryptosysteme. Beispielsweise kann auf das Pohlig-Hellman Schlüsselverteilungsverfahren ein Angriff durch Kommunikation geführt werden. Dabei schaltet sich ein Angreifer zwischen zwei Teilnehmer und erreicht auf diese Weise, daß beide Teilnehmer mit ihm einen Schlüssel vereinbaren. Wenn dann später Nachrichten verschickt werden, entschlüsselt der Angreifer diese und schickt sie verschlüsselt mit dem entsprechenden eigenen Schlüssel an die andere Seite weiter.

3.3.1 Ein gegen gewählte Kryptotextangriffe sicheres probabilistisches Public-Key Verfahren

Public-Key Systeme sind aufgrund ihrer Struktur von Natur aus den Angriffstypen eins bis drei ausgeliefert. Daher müssen sie gegen diese Angriffe sicher sein. Meistens sind sie aber gegen einen gewählten Kryptotextangriff nicht sicher. Jedoch haben Naor und Yung [NY90] nachgewiesen, daß man durch Kombination von einem Probabilistischen Public-Key Verfahren mit einem nicht-interaktiven zero-knowledge Beweissystem ein gegen gewählte Kryptotextangriffe sicheres Public-Key Verfahren erzeugen kann. Aufgrund der Quadratischen Reste Annahme läßt sich sowohl ein sicheres Probabilistisches Public-Key Verfahren realisieren ([GM83]), als auch die Existenz von nicht interaktiven zero-knowledge Beweisen für alle Sprachen $L \in NP$ nachweisen. Diese Annahmen werden zur Konstruktion des gegen gewählte Kryptotextangriffe sicheren probabilistischen Public-Key Verfahrens benötigt. Weitere gegen gewählte Kryptotextangriffe sichere Public-Key Verfahren für andere Falltürfunktionen sind in [FLS90] zu finden.

3.4 Zero-Knowledge Beweise

Die Idee hinter einem Zero-Knowledge Beweis ist, einer Partei B zu beweisen, daß man eine Information besitzt, ohne diese preiszugeben. Die Existenz solcher Beweise wird zur Konstruktion des im letzten Abschnitt beschriebenen gegen gewählte Klartextangriffe sicheren Public-Key Verfahrens benötigt. Es ist bekannt, daß für alle NP Sprachen Zero-Knowledge Beweise existieren, wenn die Quadratische Reste Annahme korrekt ist. Man findet einige solcher Beweisverfahren in [Sch96] und in den Veröffentlichungen von Blum, Feldmann und Micali [BFM88] und von De Santis, Micali und Persiano [SMP87] werden solche Verfahren beschrieben.

3.5 Unterschriftsschema

Ein Unterschriftsschema besteht aus drei Turingmaschinen: einem Schlüsselerzeuger G, einem Unterschriftenerzeuger SP und einem Überprüfer U. Der *Schlüsselerzeuger* ist eine probabilistische Turingmaschine und liefert in zu erwartender polynomieller Zeit ein Schlüsselpaar (PK, SK). Der *Unterschriftenerzeuger* ist ebenfalls eine probabilistische Turingmaschine, die in zu erwartender Laufzeit hält. Er erhält als Eingabe das Schlüsselpaar und eine Zeichenkette x und gibt eine Unterschrift y aus. Der *Überprüfer* ist eine *deterministische* Turingmaschine, erhält als Eingabe den öffentlichen Schlüssel PK, die Nachricht x und die Unterschrift y und gibt aus, ob die Unterschrift gültig ist. Unterschriftsschemata werden in Kapitel 7 (Definition 8.2) benötigt und dort genauer behandelt.

4

Grundlagen der Erlernbarkeitstheorie

Einen interessanten Überblick über den Stand der Erlernbarkeitstheorie hat Angluin im Jahr 1992 [Ang92] veröffentlicht. Wie werden uns diesem Thema jedoch von einem allgemeineren Standpunkt aus nähern.

Bevor wir überprüfen können was erlernbar ist, ist es notwendig den Begriff *Lernen* genauer zu untersuchen. Im Lexikon[1] stehen Definitionen, wie die Folgende:

Lernen, *zusammenfassende Bezeichnung für die Prozesse, die ein Individuum befähigen beziehungsweise Veranlassen, sein Verhalten zu ändern, sofern die Verhaltensänderung nicht durch organische Reifung, Ermüdung, Sinnesadaption oder Eingriffe von außen (...) sondern allein durch die frühere Begegnung mit der Situation hervorgerufen wird. ...*

Diese Definition des Begriffs Lernen ist nur für Lebewesen gedacht und zudem sehr ungenau. Im mathematisch logischen Sinn hängt das Verhalten eines Lebewesens nie *allein* von einer früheren Begegnung ab. Eine mehr technisch orientierte Definition findet sich in Meyers Taschenlexikon:

Lernen, *Sammelbezeichnung für durch Erfahrung entstandene Verhaltensänderungen beziehungsweise Verhaltensmöglichkeiten. Lernen kann also als Prozeß verstanden werden, der bestimmte Organismen, jedoch auch technische Anlagen (...) befähigt, auf Grund früherer Erfahrungen und durch organische Eingliederung weiterer Erfahrungen situationsangemessen zu reagieren.*

Der wesentliche Unterschied zwischen diesen Definitionen ist, daß die Erste versucht anzugeben, wie man an dem Verhalten eines Lebewesens Lernfähigkeit erkennen

[1] Das neue Fischerlexikon in Farbe

kann, während die Zweite sich mehr dem Begriff selbst zuwendet, ohne sich um die Nachweisbarkeit zu kümmern.

Daher ist die zweite Definition allgemeiner gehalten und erlaubt auch Maschinen Lernfähigkeit zu besitzen. Zudem sind einige weitere interessante Aspekte angesprochen. So reicht allein schon das Vorhandensein einer neuen *verbesserten* Verhaltensmöglichkeit aus um von Lernfähigkeit zu sprechen. Die *Verbesserung* wird durch den Begriff *situationsangemessen* umschrieben. Das bedeutet, es muß ein Maß geben, nach dem ein geändertes Verhalten bewertet wird.

Ein weiterer Aspekt in obiger Definition ist die Eingliederung des neu erworbenen Wissens in eine bereits vorhandene Repräsentation der Umwelt. Es wird also nicht nur eine neue Information dem bereits vorhandenen Wissen hinzugefügt, sondern es werden durch die Verwendung von bereits vorhandenem Wissen neue Verhaltensmöglichkeiten abgeleitet.

Obwohl die zweite Erklärung der Lernfähigkeit schon relativ gut den intuitiven Gedanken erfaßt, ist sie trotzdem für eine formale Definition noch zu ungenau. Insbesondere lassen sich auf dieser Grundlage keine Beweise führen. Tatsächlich gibt es bis jetzt noch keine mathematischen Erlernbarkeitsdefinitionen, die alle Aspekte der Lernfähigkeit erfassen.

Wir werden nun verschiedene Lernmethoden betrachten und vergleichen, die wir aus dem täglichen Leben kennen. Menschen lernen aus Büchern, durch Lehrer oder durch Evolution. Diese Lernmethoden finden ihre Entsprechung auch in der Erlernbarkeitstheorie. Lernen aus Büchern entspricht der üblichen *PAC*[2] Definition die wir in Abschnitt 4.1 behandeln werden. Es handelt sich dabei um eine rein passive Erlernbarkeitsdefinition, in der der Lernalgorithmus nur ein gewisses Pensum an zuvor festgelegtem Wissen verarbeiten darf. Lernen durch einen Lehrer beziehungsweise interaktives Lernen beinhaltet auch die Möglichkeit zusätzlich spezifische Fragen stellen zu dürfen. Zu diesem Erlernbarkeitstyp zählt zum Beispiel auch Erlernen mit Mitgliedschaftsfragen (Abschnitt 4.2.5).

Dem evolutionären Lernen widmet sich ein eigenes Gebiet, das *simulierte Evolution und genetische Algorithmen* behandelt. Schlecht an vorher zu definierende Bedingungen angepaßte Algorithmen werden ausgesondert, die Eigenschaften der besten Algorithmen werden dagegen entweder mutiert oder miteinander vermischt. Dafür gibt es verschiedenste Methoden. Dies ist jedoch nicht Thema dieser Arbeit und wir werden hier dieses Gebiet nicht näher behandeln.

Bei Neuronalen Netzen geht es meist um Bilderkennung. Zum Beispiel muß ein autonomer Roboter eine Türe erkennen, ehe er sie benützen kann. Die Bilddaten

[2]probably approximately correct

werden dabei entsprechend einem Punkteraster zerlegt und in einen geschichteten Schaltkreis eingespeist. Die Gatter des Schaltkreises gewichten die eingehenden Signale und geben bei Überschreitung eines Schwellenwertes ein Signal (logisch 1) aus. Die Lernfähigkeit wird häufig durch ein Gradientenabstiegsverfahren erreicht, das die Bewertungs- und Schwellenfaktoren angleicht. Das sicherlich bekannteste Verfahren ist „Backpropagation". In dieser Arbeit werden auch Ergebnisse für Erlernbarkeit bei Neuronalen Netzen angeschnitten (Kapitel 6).

Bevor man sich mit Algorithmischem Lernen befaßte war es für einen Programmierer notwendig, das erforderliche Wissen in den Algorithmus mit zu integrieren. Er war also gezwungen, alle Situationen vorauszusehen mit denen sein Algorithmus konfrontiert werden würde. Bei sehr komplexen Problemen, wie zum Beispiel Schachspielen, ist dies jedoch unmöglich.

Wir wenden uns nun unsrem Unterbereich der Erlernbarkeitstheorie zu und führen zunächst den Begriff der Konzeptklasse ein.

4.1 Konzeptklassen und ihre Repräsentation

Im weiteren Sinn ist ein Konzept ein Begriff. Angelehnt an die vorher angeführte Situation des Roboters haben wir zum Beispiel den Begriff *Tür*. Um nun eine Türe von anderen Objekten unterscheiden zu können benötigen wir zuerst eine Repräsentation der Objekte. Für den Menschen werden die Objekte durch seine Sinne dargestellt. In unserem Fall wären dies die Informationen, die die Sehzäpfchen des Auges an das Gehirn weitergeben. Die entsprechende Darstellung für einen Computer ist das in Pixel zerlegte Kamerabild eines Objekts.

Wenn wir uns der Einfachheit halber nur auf schwarzweiße Bilder beschränken, können wir die Rasterpunkte durch 0 und 1 Bits darstellen. Werden die Bits eines Bildes zeilenweise aneinandergehängt, so entsteht daraus eine binäre Zeichenkette die das Bild repräsentiert. Wir werden eine solche Zeichenkette *Beispiel* nennen. Ein Beispiel kann entweder positiv sein, wenn es den Begriff trifft oder negativ wenn der Begriff nicht getroffen ist. Es stellt also für unseren Roboter eine Tür dar, die positiv bewertet wird oder ein anderes Objekt, das die Bewertung negativ erhält. Diese Eigenschaft kann man durch ein zusätzliches Bit angeben. Im allgemeinen bedeutet das Bit 1, daß ein positives Beispiel und 0 daß ein negatives Beispiel vorliegt. Ein so bewertetes Beispiel nennen wir *Muster*.

Eine weitere wichtige Frage ist, wie komplex die Funktion ist, die positive von negativen Mustern unterscheidet. Bevor wir auf diese Frage näher eingehen können, müssen wir den Begriff Konzept formal definieren.

Konzept

Wie schon im letzten Abschnitt angedeutet, benutzen wir der Einfachheit halber als Grundmenge X die binären Zeichenketten $\{0,1\}^*$. Alle Objekte werden also durch binäre Zeichenketten beschrieben.

Ein Konzept entspricht dann einer Sprache auf der Grundmenge X. Es ist also eindeutig festgelegt, welche Elemente der Grundmenge in der Sprache liegen. Auf unser Beispiel bezogen sind alle Zeichenketten, die Kodierungen von Bildern einer Tür darstellen, in der Sprache.

Wir stellen zuerst eine allgemeingültige Möglichkeit vor ein solches Konzept zu repräsentieren. Wir betrachten die Grundmenge X als aufsteigend sortiert. Wir konstruieren nun einen Binärstring, der genau dann an der i-ten Stelle eine 1 enthält, wenn das i-te Element aus X im Konzept liegt. Die Länge dieser Repräsentation entspricht dann der Größe der Grundmenge. Wir erhalten also sehr große Repräsentationen, insbesondere sind Repräsentationen für unendliche Grundmengen nicht mehr handhabbar. Man vermutet jedoch im allgemeinen einfachere Zusammenhänge. Ein Lernalgorithmus sollte diese Zusammenhänge erkennen und eine quasi komprimierte Beschreibung des erlernten Konzepts ausgeben (siehe auch Occam-Algorithmus, Abschnitt 4.2.4).

Wir benötigen deshalb ähnlich wie im Kapitel 2 eine Repräsentation für jedes Konzept. Der Programmcode einer Turingmaschine ist zum Beispiel eine Repräsentation der durch die Turingmaschine erkannten Sprache. Ebenso beschreibt eine Boolesche Formel mit n Eingängen die Teilmenge der n-stelligen binären Zeichenketten, bei denen die Formel zu 1 ausgewertet wird. Wir werden diesen Zusammenhang noch formalisieren.

Es existieren leicht differierende Definitionen für Repräsentationen von Konzepten, wir verwenden eine an die Definition aus [AK91] angelehnte.

Die Menge der binären Zeichenketten X ist die Grundmenge oder Beispielmenge. Eine Repräsentation von Konzepten C ist eine Untermenge von $X \times X$. Ein Element $< u, x > \in X \times X$ besteht aus einer Konzeptbeschreibung, zum Beispiel einer Booleschen Formel, und einem Element der Beispielmenge, zum Beispiel der Variablenbelegung der Booleschen Formel. Ein Beispiel x ist ein Mitglied einer Konzeptbeschreibung u genau dann, wenn $< u, x > \in C$.

Wir definieren das durch u repräsentierte Konzept als

$$\kappa_C(u) = \{x | < u, x > \in C\}$$

Die Menge $\kappa_C(u)$ enthält also alle Beispiele, die zusammen mit u in der Repräsentationsklasse C liegen.

Wir werden eine Reihe von Booleschen Repräsentationsklassen behandeln. Das bedeutet sie liefern für jede Eingabe entweder die Ausgabe 0 oder 1. Wir betrachten zuerst die Repräsentationsklasse C_{BF} der Booleschen Formeln.

Zuerst überlegen wir uns, wie man geklammerte Booleschen Formeln, die durch die Variablen $X_1, X_2, \ldots, X_n$ aufgebaut und mit *AND*, *OR*, *NAND* und *NOR* verknüpft sind, durch binäre Zeichenketten kodieren kann. Wir verwenden zur Kodierung einen Präfixcode. Das einzige Problem tritt dabei bei der Kodierung der X_i auf, da zur eindeutigen Decodierung bekannt sein muß, wie viele Variablen benutzt werden. Wir nennen die Anzahl der Variablen n und kodieren nach folgender Tabelle.

n	$\wedge$	$\vee$	$\neg\wedge$	$\neg\vee$	$\neg$	(	)	X_i
$\underbrace{1\ldots1}_{n\ mal}0$	000	001	010	011	100	101	110	$111\ \underbrace{bin(i,n)}_{\lceil\log n\rceil\ stellig}$

Dabei ist $bin(i,n)$ die $\lceil n\rceil$-stellige binäre Kodierung der Integerzahl i.

Demnach kodieren wir zum Beispiel die Boolesche Formel

$$X_1 \wedge X_2 \vee (\neg X_3 \neg \wedge (X_4 \vee X_5))$$

zu

$$\underbrace{111110}_{n=5}\underbrace{111001}_{X_1}\underbrace{000}_{\wedge}\underbrace{111010}_{X_2}\underbrace{001}_{\vee}\underbrace{101}_{(}\underbrace{100}_{\neg}\underbrace{111011}_{X_3}\underbrace{010}_{\neg\wedge}\underbrace{101}_{(}\underbrace{111100}_{X_4}\underbrace{001}_{\vee}\underbrace{111001}_{X_1}\underbrace{110}_{)}\underbrace{110}_{)}$$

Ein Element $< u, x >$, in dem u die Kodierung einer Booleschen Formel und x eine Variablenbelegung ist, ist genau dann in C_{BF}, wenn u die Zahl n und die Formel ϕ kodiert, die über den Variablen $X_1, X_2, \ldots, X_n$ definiert ist, die Länge von x gleich n ist ($|x| = n$) und die Boolesche Formel ϕ wird durch Belegung der Variablen X_i durch x_i erfüllt ist.

Entsprechend besteht die Repräsentationsklasse C_{CNF} aus all den Elementen $\langle u, x\rangle \in C_{BF}$, bei denen u die Kodierung einer Boolesche Formel in konjunktiver Normalform ist.

Bei der Repräsentationsklasse C_{kCNF} ist die Größe der Terme auf k Variablen beschränkt und bei $C_{kTERM-CNF}$ ist die Anzahl der Terme auf k beschränkt.

Für die disjunktiven Normalformen gilt, daß $< u, x >$ in der Repräsentationsklasse C_{DNF} liegt, wenn $< u, x > \ \in C_{BF}$ und u eine Formel in disjunktiver Normalform repräsentiert. Weiter ist in der Repräsentationsklasse C_{kDNF} die Größe der Klauseln auf k Variablen beschränkt und bei $C_{kKLAUSEL-DNF}$ die Anzahl der Klauseln auf k beschränkt.

Die Repräsentationsklasse der determinisitischen endlichen Automaten beschreibt die Menge der regulären Sprachen [Sch94]. Auch *DFA's*, die auf dem Eingabealphabet $\{0, 1\}$ arbeiten, lassen sich binär kodieren. Man kodiert dafür die Zustände als binäre Zahlen und speichert für jeden Zustand den Folgezustand für die Eingaben 0 und 1. Ein weiteres Bit gibt an, ob ein Zustand akzeptierend ist oder nicht. Die Anzahl der Zustände wird ähnlich wie bei der Variablencodierung im Beispiel der Booleschen Formeln durch die Kodierung von n am Anfang und $bin(i, n)$ bei Verweisen erreicht. Der Zustand 0 ist immer der Startzustand. Zum Beispiel beschreibt die folgende Kodierung einen *DFA*, der nur Eingaben akzeptiert, die mit 1 beginnen.

1110	3 Zustände
0	Zustand 00 akzeptiert nicht
01	bei Eingabe von 0 Übergang auf Zustand 01
10	bei Eingabe von 1 Übergang auf Zustand 10
0	Zustand 01 akzeptiert nicht
01	bei Eingabe von 0 Übergang auf Zustand 01
01	bei Eingabe von 1 Übergang auf Zustand 01
1	Zustand 10 akzeptiert
10	bei Eingabe von 0 Übergang auf Zustand 10
10	bei Eingabe von 1 Übergang auf Zustand 10

Man kann sich überlegen, daß dabei eine polynomielle Beziehung zwischen der Anzahl der Zustände, die mindestens nötig sind um eine Sprache A durch einen deterministischen endlichen Automaten zu beschreiben und der Größe der kleinsten Repräsentation u, die die Sprache A erkennt, existiert.

Im Kapitel 2 haben wir die Komplexitätsklassen NC^1 und AC^0 kennengelernt. Beide durch Schaltkreise beschriebenen Komplexitätsklassen waren dabei mit einer polynomiell in der Anzahl der Eingänge beschränkten Anzahl von Ausgängen versehen. Um jedoch eine Konzeptrepräsentation zu beschreiben setzen wir die Anzahl der Ausgänge auf konstant 1.

Zur Kodierung eines Schaltkreises verschlüsseln wir zuerst die Anzahl der Gatter und der Eingänge. Dann geben wir für jedes Gatter den Typ und die Vorgänger an. Bei unbeschränktem fan-in benötigen wir noch eine Ende-Kodierung, die nach den Gattereingangsdefinitionen angehängt wird.

Anhand dieser Vorgehensweise lassen sich auch die Repräsentationsklassen Boolesche Schaltkreise C_{BSK} und Schaltkreise mit konstanter Tiefe C_{dTC} definieren.

Parametrisierte Repräsentationsklassen

Für die Erlernbarkeitsdefinitionen benötigen wir jedoch parametrisierte Repräsentationsklassen. Zuerst wird die Beispielmenge X parametrisiert. Als Parameter n nimmt man meist die Anzahl der Variablen, die Länge der Eingabezeichenkette oder die Anzahl der Eingänge eines Schaltkreises.

Effiziente Berechnung von Repräsentationsklassen

Ein Ergebnis von [Sch89] besagt, daß nicht in Polynomialzeit auswertbare Repräsentationsklassen auch nicht erlernbar sein können. Alle bisher vorgestellten Klassen sind in Polynomialzeit entsprechend ihrer Repräsentationslänge auswertbar. Damit sie also auch entsprechend n effizient auswertbar sind, muß die Repräsentationsgröße $|c|$ polynomiell auf n beschränkt sein. Die Klassen sind also gewissermaßen doppelt parametrisiert. Daher beschränkt man sich auf Klassen, welche Sprachen beschreiben, die in P auswertbar sind.

Formal bedeutet das, daß für eine Klasse C durch eine Turingmaschine in Polynomialzeit bei Eingabe von u und x die Zugehörigkeit von x zu $\kappa_C(u)$ ausgegeben werden muß. Man kann dabei auch von Turingmaschinen auf Schaltkreise übergehen, indem man c direkt in den Schaltkreis mit einbaut.

Ein Beispiel für eine parametrisierte Repräsentationsklasse C_n ist die Menge aller Booleschen Formeln über n Variablen mit maximaler Repräsentationsgröße $|c| < n^3$. Bei parametrisierten Repräsentationsklassen bestehen die positiven und negativen Beispiele aus n-stelligen binären Zeichenketten.

4.2 Einige grundlegende Erlernbarkeitsdefinitionen

In diesem Abschnitt überlegen wir uns, was wir darunter verstehen, daß eine der im letzten Abschnitt vorgestellten Klassen erlernbar ist. Wir nennen die Repräsentationsklasse, die wir erlernen wollen *Zielkonzeptklasse*. Ein Lernalgorithmus versucht nun ein *Zielkonzept*, das aus dieser Zielkonzeptklasse gewählt wird zu erlernen. Hierfür hat er Zugriff auf ein Orakel, das ihm nach einer unbekannten aber festen Verteilung generierte Beispiele einschließlich deren Zugehörigkeit zum zu erlernenden

Zielkonzept liefert. Nachdem der Lernalgorithmus genügend solcher Orakelaufrufe benutzt hat, gibt er eine Hypothese zurück, die eine möglichst exakte Annäherung an das Zielkonzept sein soll. Häufig muß diese Hypothese aus einem vorgegebenen Hypothesenraum gewählt werden. Je kleiner dieser Hypothesenraum ist, desto schwieriger hat es der Lernalgorithmus, das Zielkonzept zu erlernen.

Im Laufe der Entwicklung der Erlernbarkeitstheorie sind verschiedene Erlernbarkeitsdefinitionen eingeführt worden, um den intuitiven Begriff der Erlernbarkeit zu formalisieren. Im folgenden Abschnitt werden wir einige der am häufigsten verwendeten Erlernbarkeitsdefinitionen vorstellen und vergleichen.

4.2.1 Die *PAC*-Erlernbarkeitsdefinition

Im letzten Abschnitt haben wir bereits die Definitionen für Beispiel und Muster vorgestellt. Wir nennen eine Menge von Mustern eine *Probe*. Anhand einer solchen Probe kann dann ein Lernalgorithmus versuchen, ein Zielkonzept zu erlernen. Wir sagen eine Probe ist *konsistent* mit einer Hypothese h, wenn jedes der Muster in der Probe entsprechend der Hypothese bewertet ist. Die nun folgende Erlernbarkeitsdefinition *PAC*[3] ist sicherlich die gebräuchlichste ohne Lernzeiteinschränkung.

Definition 4.1 ***PAC*-Erlernbarkeit**
Ein Genauigkeitswert ϵ und ein Vertrauenswert δ sind gegeben. Es muß eine Probengröße m_0 in Abhängigkeit von ϵ und δ geben, so daß für Proben, die entsprechend einer unbekannten aber festen Verteilung gezogen werden und mehr als m_0 Muster enthalten gilt, daß der Algorithmus mit einer Wahrscheinlichkeit von über $1-\delta$ eine Hypothese h liefert, die getestet bei gleicher Verteilung auf dem Beispielraum nur in ϵ der Fälle einen Fehler macht.

Mathematisch ausgedrückt bedeutet dies:

$$\forall \delta, \epsilon\, \exists m_0 = m_0(\delta, \epsilon).\, \forall m \geq m_0, t, \mu\, .\mu^m\{s \in S(m,t) | er_\mu(L(s)) < \epsilon\} > 1 - \delta$$

Dabei ist $S(m,t)$ die Menge der Proben mit m Mustern, die konsistent mit der Zielhypothese t sind. Der Lernalgorithmus L gibt die Hypothese $L(s)$ aus. $er_\mu(h)$ gibt die Vorhersagefehlerwahrscheinlichkeit einer Hypothese h in Bezug auf das Zielkonzept t:

$$er_\mu(h) = \mu\{x \in X | t(x) \neq h(x)\}$$

[3]probably approximately correct

Damit ist $er_\mu(L(s))$ die Vorhersagefehlerwahrscheinlichkeit der ausgegebenen Hypothese $L(s)$ die entsprechend der Verteilung μ getestet wird. Somit ist $\mu^m\{s \in S(m,t) | er_\mu(L(s)) < \epsilon\}$ die Wahrscheinlichkeit, daß der Algorithmus bei Eingabe von Proben der Länge m, die entsprechend einer Verteilung μ gezogen werden, eine Hypothese ausgibt, die mit einer Wahrscheinlichkeit von höchstens ϵ einen Vorhersagefehler macht.

Man kann also dem Lernalgorithmus sagen, wie genau (Parameter ϵ) die ausgegebene Hypothese sein soll und mit welcher Zuverlässigkeit (Parameter δ) eine solchermaßen exakte Hypothese erzeugt werden muß. Aus diesen Parametern läßt sich dann eine Probengröße m_0 berechnen. Die Probe wird entsprechend einer unbekannten aber festen Verteilung erzeugt und dem Lernalgorithmus über das Orakel zur Verfügung gestellt. Der Lernalgorithmus muß dann mit Mindestwahrscheinlichkeit $1 - \delta$ eine Hypothese liefern. Diese Hypothese, darf - gemessen an der Beispielverteilung - in maximal ϵ der Fälle einen Fehler machen.

Diese Erlernbarkeitsdefinition besitzt noch keine Zeitbeschränkung. Daher sind die meisten Zielkonzeptklassen nach der Definition *PAC* erlernbar. Ein einfaches Beispiel für eine solche Klasse sind *Monome*.

Die Konzeptklasse der *Monome* besteht aus einer Konjunktion von Literalen[4]. Beispielsweise ist $x_1 \wedge x_3 \wedge \neg x_4$ ein Monom.

Für diese Klasse existiert ein einfacher Lernalgorithmus. Dieser Lernalgorithmus ist sogar *konsistent*, das bedeutet, daß er eine Hypothese generiert, die sämtliche Muster, die der Lernalgorithmus erhält, korrekt bewertet.

```
set U = {u_1, ¬u_1, ..., u_n, ¬u_n}
for i := 1 to m do
    if b_i = 1 then
        for j := 1 to n do
            if (x_i)_j = 1 then delete ¬u_j if present in U
            else delete u_j if present in U
L(s) := h_U (* Konjunktion der Literale in U *)
```

Dabei ist b_i die Bewertung des i-ten Beispiels x_i und $(x_i)_j$ das j-te Bit des i-ten Beispiels. Die Anfangshypothese des Lernalgorithmus ist konstant 0 und wird nur verändert, wenn positive Beispiele in der Probe auftauchen. In diesem Fall streicht der Lernalgorithmus die Literale, die dem Beispiel widersprechen.

Wenn wir zum Beispiel ein Monom über $n = 3$ Variablen erlernen und die Probe $P = \{(101, 1), (110, 0), (011, 1)\}$ ist, erhalten wir folgende Zwischenergebnisse für die

[4]ein Literal ist eine Variable oder deren Negation

Hypothese:

$$< u_1, \neg u_1, u_2, \neg u_2, u_3, \neg u_3 >, < u_1, \neg u_2, u_3 >, < u_1, \neg u_2, u_3 >, < u_3 >$$

Man sieht sofort, daß die generierte Hypothese $h_U = u_3$ mit den Mustern der Probe konsistent ist. Die Hypothese ist sogar die kleinste mit dieser Probe konsistente. Dies ist im allgemeinen nicht der Fall.

Man kann zeigen, daß es genügt, m polynomiell in $1/\epsilon$ und $1/\delta$ zu wählen (siehe Seite 17 [KuV94]). Der vorgestellte konsistente Algorithmus ist also sogar ein *pPAC* Lernalgorithmus (Abschnitt 4.2.3).

Der obige Lernalgorithmus erlernt Monome durch Monome. Das bedeutet, daß die ausgegebene Hypothese ebenfalls in Form eines Monoms zurückgegeben wird. Es gibt aber auch Lernalgorithmen, die einen größeren Hypothesenraum besitzen; man spricht dabei von repräsentationsunabhängigen Lernalgorithmen.

Der Monom Lernalgorithmus ist ein *PAC* Lernalgorithmus: Das bedeutet, er erlernt das Zielkonzept unabhängig von der Verteilung auf dem Beispielraum. Es gibt auch Erlernbarkeitsresultate, die nur für eine bestimmte Verteilung auf dem Beispielraum gültig sind.

Weiterhin gibt es auch alternative Erlernbarkeitsdefinitionen, bei denen nicht als Ergebnis eine Hypothese ausgegeben wird, sondern bei dem am Ende des Lernprozesses ein neues von einem Testorakel generiertes Beispiel bewertet werden muß. Dadurch entsteht jedoch bei hypothesenraumunabhängiger Erlernbarkeit kein Unterschied, da man den Lernalgorithmus mit festverdrahteter Probe selbst als Hypothesenrepresentation ansehen kann. Bei hypothesenraumabhängiger Erlernbarkeit folgt zumindest aus unserer Definition der *PAC*-Erlernbarkeit auch die Erlernbarkeit in der alternativen Definition.

Außerdem unterscheidet man noch one- und two-Button Modelle. Im two-Button Model gibt es zwei separate Orakel, ein negatives und ein positives. Entsprechend ergeben sich damit auch separate Verteilungen für die positiven und negativen Beispiele. Der Fehler darf weder für die positiven noch für die negativen Beispiele den Genauigkeitsparameter ϵ überschreiten.

Daß die Unterschiede zwischen one- und two-Button Modellen rein formalen Charakter haben, ergibt sich aus der Arbeit von Haussler, Kearns, Littlestone und Warmuth [HKL88]. In ihrer Veröffentlichung wird auch die Äquivalenz von verschiedenen anderen leicht abgewandelten polynomialzeitlichen Erlernbarkeitsdefinitionen im one- und two-Button Modell bewiesen.

Eine wichtigere Unterscheidung bei der Erlernbarkeit ist die zwischen *weak PAC* und *strong PAC*. Mit strong *PAC* ist unser Modell gemeint. Bei weak *PAC* entfällt der Parameter ϵ. Statt dessen muß die Hypothese nur um einen polynomiellen Teil besser sein als das bloße Erraten. Wir formalisieren diese neue Definition:

Definition 4.2 *wPAC* **(weak** *PAC***)**
Sei C eine Zielkonzeptklasse und H ein Hypothesenraum über der Beispielmenge X. Dann ist C schwach erlernbar durch H, wenn es einen Algorithmus A gibt, der Zugriff auf zwei Orakel (POS und NEG) besitzt (two-Button Modell) und als Eingabe den Zuverlässigkeitsparameter δ erhält sowie die folgenden Eigenschaften erfüllt. Der Algorithmus muß nun für jedes Zielkonzept, jede Verteilung auf den positiven beziehungsweise negativen Beispielen und für jeden Zuverlässigkeitsparameter δ mit einer Wahrscheinlichkeit von über $1 - \delta$ eine Hypothese $h \in H$ ausgeben, die auf den positiven wie negativen Beispielen weniger als $1/2 - 1/p(n)$ Fehler macht.

Diese Erlernbarkeitsdefinitionen werden häufig auch in einer leicht veränderten Form benutzt. Man läßt dann probabilistische Lernalgorithmen zu. Der Algorithmus kann dann für eine bestimmte Probe bei mehrfacher Ausführung verschiedene Hypothesen generieren. Diese Tatsache wird dann mit in die Wahrscheinlichkeitsbewertung einbezogen.

Aus der Definition folgt direkt, daß eine *PAC* erlernbare Klasse auch *wPAC* erlernbar ist. Und es gilt entsprechend, daß eine Repräsentationsklasse die nicht *PAC* erlernbar ist auch nicht *wPAC* erlernbar sein kann. Nichterlernbarkeit in *wPAC* ist also ein stärkeres Resultat als Nichterlernbarkeit in *PAC*. Bei einem Nichterlernbarkeitsresultat ist es auch wichtig, daß die Klasse für einen beliebigen Hypothesenraum nichterlernbar bleibt.

Nichterlernbarkeitsresultate für festgelegte Hypothesenräume werden nur dann interessant, wenn eine Klasse zwar durch sich selbst noch nicht erlernbar ist, jedoch durch einen größeren Hypothesenraum erlernbar wird. Es stellt sich dann die Frage, bis zu welcher Hypothesenraumgröße die Klasse nichterlernbar bleibt.

4.2.2 Potentielle Erlernbarkeit und *VC*-Dimension

In diesem Abschnitt erläutern wir die Zusammenhänge zwischen Erlernbarkeit, potentieller Erlernbarkeit und *VC*-Dimension.

Die potentielle Erlernbarkeit ist wie folgt definiert:

Definition 4.3 potentiell erlernbar
Für gegebene Genauigkeits- und Zuverlässigkeitsparameter ϵ und δ existiert eine positive ganze Zahl m_0 in Abhängigkeit von δ und ϵ, so daß für alle $m \geq m_0$, alle Verteilungen μ auf X und jedes Zielkonzept $t \in H$ gilt:

$$\mu^m\{s \in S(m,t)|H[s] \cap B_\epsilon = \emptyset\} > 1 - \delta$$

Dabei steht $S(m,t)$ für die Menge der Proben der Größe m, die entsprechend der Hypothese t generierte Muster enthält. $H[s]$ steht für die Menge der mit der Probe s konsistenten Hypothesen und B_ϵ steht für die Menge der Hypothesen, die bei Verteilung μ auf den Beispielen X entsprechend dem Zielkonzept t in mehr als ϵ der Fälle einen Fehler machen.

In Worten bedeutet dies:
Die Genauigkeit ϵ und das Vertrauen δ sind gegeben. Es muß eine Probengröße m_0 in Abhängigkeit von ϵ und δ geben, so daß für alle Proben die mehr als m_0 Muster enthalten gilt, daß die Wahrscheinlichkeit innerhalb der konsistenten Hypothesen nur günstige Hypothesen zu finden, größer als $1 - \delta$ ist.

Angenommen, eine Repräsentationsklasse ist potentiell erlernbar und wir kennen einen konsistenten Lernalgorithmus für diese Klasse. Dann liegt die Hypothese die der Algorithmus ausgibt in $H[s]$ und man kann aus $H[s] \cap B_\epsilon = \emptyset$ schließen, daß der Fehler der Hypothese kleiner als ϵ ist. Aus der potentiellen Erlernbarkeit und der Existenz eines konsistenten Lernalgorithmus folgt also die Erlernbarkeit in *PAC*.

Wozu *VC*-Dimension?

Die Definition der potentiellen Erlernbarkeit ist ziemlich komplex und schwer nachzuweisen. Man kann zwar zeigen, daß jede endliche Zielkonzeptklasse potentiell erlernbar ist, bei unendlichen Hypothesenräumen bleibt es jedoch unklar, ob sie potentiell erlernbar sind. Um auch die potentielle Erlernbarkeit von unendlichen Hypothesenräumen nachweisen zu können, führen wir die *VC*- (Vapnik-Chervonenkis [VC71]) Dimension ein. Sie ist ein Maß für die Mächtigkeit eines Hypothesenraumes. Ein mächtigerer Hypothesenraum wird also auch schwerer zu erlernen sein.

Durch aufwendige Beweise kann man dann nachweisen, daß ein Hypothesenraum genau dann potentiell erlernbar ist, wenn er eine endliche *VC*-Dimension besitzt.

Beispiel reelles Perceptron

Um die folgenden Definitionen zu veranschaulichen führen wir das reelle Perceptron (real perceptron) ein.

Definition 4.4 Reelles Perceptron P_n
Ein Reelles Perceptron P_n besitzt n reellwertige Eingänge und einen binären Ausgang. Die Beispielmenge ist also R^n. Die Eingänge werden durch reelle Konstanten α_i gewichtet und die Summe mit einem Schwellenwert ψ verglichen. Wenn der Schwellenwert überschritten wird, wird der Ausgang auf 1 gesetzt und ansonsten auf 0.

Ausdruckskraft von P_2

Ein Reelles Perceptron stellt also eine lineare Ungleichung dar. Im Fall P_2 kann das Reelle Perceptron den zweidimensionalen Beispielraum durch eine Geradenungleichung aufspalten. Man kann also eine Menge von Punkten im zweidimensionalen Raum durch ein P_2 Perceptron nur auf wenige Arten in positive und negative Beispiele trennen. Daher ist die Mächtigkeit bzw. Ausdruckskraft des P_2 gering, dies wird sich auch in der VC-Dimension widerspiegeln.

Um die VC-Dimension zu erläutern benötigen wir noch einige Grunddefinitionen.

Definition 4.5 Def $\Pi_H(x)$
Sei $A \subseteq X$ eine Menge von Beispielen und H ein Hypothesenraum. Dann ist $\Pi_H(A)$ die Anzahl der unterschiedlichen Klassifikationen der Beispiele in A, die durch Hypothesen aus H möglich sind.

Definition 4.6 Wachstumsfunktion $\Pi_H(m)$
Die Wachstumsfunktion $\Pi_H(m)$ ist definiert als die maximale Anzahl von Klassifikationen von A, $|A| = m$ in Bezug auf H.

$$\Pi_H(m) = max_{A,|A|=m} \Pi_H(A)$$

Bsp.: Wie verhält sich $\Pi_H(m)$ bei P_1

Das reelle Perceptron P_1 beschreibt die Gleichung $\alpha_1 x_1 \geq \Theta$. Man kann also auf der Zahlengerade einen Trennungspunkt wählen, von dem aus Beispiele in einer Richtung positiv bewertet werden und in der anderen negativ. Für jede strikt aufsteigend geordnete Probe

$$x_1 < x_2 < \cdots < x_n$$

gibt es daher genau $2m$ mögliche Klassifikationen in H und für jede Permutation einer geordneten Probe genauso viele. Falls zwei Beispiele aufeinanderfallen, reduziert sich die Anzahl der Bewertungsmöglichkeiten. Also gilt $\Pi_H(m) = 2m$.

Die maximale Anzahl an Klassifikationen für einen beliebig komplexen Hypothesenraum wäre natürlich 2^m. In diesem Fall könnte man jedes der Beispiele beliebig als positiv oder negativ klassifizieren. Diese Überlegung führt uns auf die nächste Definition:

Definition 4.7 H **erschlägt** A
Ein Hypothesenraum H erschlägt (shatters) eine Beispielmenge A der Größe m, wenn die maximale Anzahl an möglichen Klassifikationen (2^m) durch Klassifikationen aus H erreicht wird, das heißt

$$\Pi_H(A) = 2^m.$$

Mit dieser Definition sind wir nun in der Lage die *VC*-Dimension eines Hypothesenraumes einzuführen.

Definition 4.8 $VCdim(H) = max\{m : \Pi_H(m) = 2^m\}$
VCdim(H) ist also die maximale Größe einer Beispielmenge, die von H erschlagen wird.

VCdim(H) beschreibt also die Größe der Beispielmenge, bei der gerade noch alle möglichen Klassifikationen in H enthalten sind.

Wir verdeutlichen uns dies wieder anhand des reellen Perceptrons P_1. Zwei Beispiele lassen sich ohne weiteres auf alle 4 Arten bewerten, indem man den Trennungspunkt links oder in der Mitte der zwei Beispiele wählt und entweder die rechte oder die linke Seite positiv bewertet. Also gilt *VCdim*$(P_1) \geq 2$. Aber bereits bei drei Beispielen kann man das mittlere Beispiel nicht mehr von den äußeren trennen. Bei Probenlänge 3 gibt es also nur maximal 6 mögliche Klassifikationen und daher gilt

$$\mathit{VCdim}(P_1) = 2.$$

Für *VCdim*(P_2) ist es möglich, daß man 3 Punkte im zweidimensionalen Raum durch eine Gerade auf $2^3 = 8$ Möglichkeiten trennt. Es reicht dabei entsprechend der Definition aus, nur ein Beispiel zu finden. Es gilt also

$$VCdim(P_2) \geq 3$$

Wir überlegen uns nun, daß man bereits ab 4 Beispielen nicht mehr sämtliche Klassifikationsmöglichkeiten benützen kann. Es gibt nur zwei Fälle, entweder sind alle 4 Beispielpunkte Randpunkte ihrer konvexen Hülle oder ein Punkt liegt innerhalb. Aber in beiden Fällen gibt es immer Punkte, die nicht linear trennbar sind. Im ersten Fall sind dies die gegenüberliegenden Punkte und im zweiten der Innere von den Äußeren. Also gilt

$$VCdim(P_2) = 3$$

VC-Dimension von Monomen

Um die *VC*-Dimension von Monomen zu untersuchen benützen wir die folgende Überlegung. Ein endlicher Hypothesenraum H hat maximale *VC*-Dimension $\log|H|$[5], denn da es nur $|H|$ Hypothesen gibt, können für Beispielfolgen der Länge $m \geq \log|H|$ keine $2^m > 2^{\lg|H|}$ unterschiedliche Klassifikationen existieren.

Mit dem letzten Lemma läßt sich die *VC*-Dimension der Monomen (M_n) nach oben abschätzen.

In einem Monom M_n kann jede der n Variablen entweder negativ, positiv oder überhaupt nicht in der Formel vorkommen. Die Anzahl der Hypothesen ist also durch 3^n nach oben beschränkt. Mit der vorangegangenen Überlegung gilt dann:

$$VCdim(M_n) \leq n \cdot \log(3)$$

Um auch eine untere Schranke zu erhalten, wählen wir die n Einheitsvektoren e_i als Beispielfolge. Man erhält alle möglichen Klassifikationen, indem man jede 0 an der i-ten Stelle im Bewertungsvektor durch ein $\neg u_i$ erzwingt.

Es gilt also

$$n \leq VCdim(M_n) \leq n \cdot \log(3)$$

[5] log ist der Logarithmus zu Basis 2

VC-Dimension für beliebige Reelle Perceptronen P_n

Wir zeigen zuerst, daß keine Probe mit $n+2$ Beispielen durch P_n erschlagen werden kann. Danach werden wir nachweisen, daß die VC-Dimension eines Reellen Perceptrons mit n Eingängen größer gleich $n+1$ ist. Daraus folgt dann, daß VC-Dimension des reellen Neurons P_n gleich $n+1$ ist.

Sei $E_x = (x_1, \ldots, x_{n+2})$ eine Beispielfolge. Falls zwei Beispielvektoren x_i den gleichen Wert besitzen, können diese Beispiele nicht unterschieden werden. Daher reicht es aus, nur unterschiedliche Beispiele zu betrachten. Nach Radons Theorem existiert dann eine nichtleere Untermenge $S \subset E_x$, so daß $conv(S) \cap conv(E_x \backslash S) \neq \emptyset$ gilt[6]. Es existiert also eine Schnittmenge der konvexen Hüllen von S und $E_x \backslash S$.

Wir nehmen nun an, daß eine Hypothese für jede mögliche Aufteilung der Probe (der Länge $n+2$) in positive und negative Beispiele existiert. Wir wissen aber, daß es nach Radons Theorem eine Aufteilung gibt, deren konvexe Hüllen nicht leer sind. Die konvexen Hüllen liegen aber innerhalb der ebenfalls konvexen Halbräume mit denen wir die Beispiele ja trennen wollen. Daher ist die Schnittmenge der Halbräume auch nicht leer und somit ergibt sich ein Widerspruch. Daraus folgt

$$VCdim(P_n) \leq n+1.$$

Um zu zeigen daß auch $VCdim(P_n) \geq n+1$ gilt, betrachten wir die Beispielfolge $(0, e_1, \ldots, e_n)$. Für eine beliebige Aufteilung in positive S und negative Beispiele $E_x \backslash S$ wählen wir

$$\alpha_i = \begin{cases} 1, & \text{if } e_i \in S; \\ -1, & \text{if } e_i \notin S; \end{cases}$$

und

$$\theta = \begin{cases} -1/2, & \text{if } 0 \in S; \\ 1/2, & \text{if } 0 \notin S. \end{cases}$$

Dann wird die Untermenge $S \subset E_x$ durch die Hypothese $\omega = (\alpha_1, \ldots, \alpha_n, \theta)$ aus R_n positiv bewertet und $E_x \backslash S$ negativ. Also ist die VC-Dimension mindestens $n+1$. Damit folgt die Behauptung.

Die VC-Dimension eines Hypothesenraumes ist - wie wir bereits wissen - über die Wachstumsfunktion $\Pi_H(m)$ definiert. Es existiert jedoch ein weiterer interessanter

[6] conv(S) ist die Konvexe Hülle über den Vektoren aus der Menge S

Zusammenhang, der als Sauers Lemma bekannt ist. Wenn H eine *endliche VC-Dimension* d hat, läßt sich die Wachstumsfunktion durch ein Polynom vom Grad d abschätzen. Dieses Lemma wird unter anderem zum Beweis des folgenden fundamentalen Satzes benötigt.

Satz 4.1 Potentielle Erlernbarkeit entspricht einer endlichen *VC*-Dimension
Ein Hypothesenraum ist genau dann potentiell erlernbar, wenn er eine endliche VC-Dimension hat.

Wir wissen bereits, daß aus der potentiellen Erlernbarkeit und der Existenz eines konsistenten Lernalgorithmus die Erlernbarkeit folgt. Mit dem letzten Satz können wir diesen Zusammenhang auch auf folgende Weise formulieren:

Wenn ein Hypothesenraum eine endliche *VC*-Dimension besitzt und ein konsistenter Lernalgorithmus existiert, so ist der Hypothesenraum auch *PAC* erlernbar. Für die Umkehrrichtung gilt daher, wenn ein Hypothesenraum nicht *PAC*-erlernbar ist, dann besitzt er entweder eine unendliche *VC*-Dimension oder es existiert kein konsistenter Lernalgorithmus.

Daß in der Tat Hypothesenräume existieren, die eine unendliche *VC*-Dimension besitzen, zeigt folgendes Beispiel:

Definition 4.9 charakteristische Funktion χ_A
Für eine beliebige Untermenge $A \subseteq \mathbf{R}$ ist die charakteristische Funktion durch

$$X_A(y) = \begin{cases} 1, & if\, y \in A; \\ 0, & sonst. \end{cases}$$

definiert.

Es ist klar, daß jede Beispielmenge B, die aus unterschiedlichen Beispielen besteht, durch eine charakteristische Funktion auf jede beliebige Weise bewertet werden kann. Wenn wir zum Beispiel die Untermenge U von B positiv und die Restmenge $B \backslash U$ negativ bewerten wollen, benützen wir die charakteristische Funktion X_U. Der Hypothesenraum der charakteristischen Funktionen besitzt also einen unendlichen Hypothesenraum.

Notwendige Beispielkomplexität

Eine weitere Anwendung der *VC*-Dimension ist die Abschätzung der zum Erlernen notwendigen Probengröße. Wir betrachten den Fall, daß der Hypothesenraum C durch sich selbst erlernt werden muß. Falls dann ein konsistenter Lernalgorithmus existiert und die *VC*-Dimension $d = VCdim(C)$ endlich ist, läßt sich die zum Erlernen notwendige Probengröße (*Beispielkomplexität*) nach unten und oben durch folgende Formeln abschätzen.

$$m_0(H,\delta,\epsilon) = \Omega\left(\frac{d}{\epsilon} + \frac{1}{\epsilon}\ln(\frac{1}{\delta})\right) \tag{4.1}$$

$$m_0(H,\delta,\epsilon) = O\left(\frac{d}{\epsilon}\ln\left(\frac{1}{\epsilon}\right)\ln(\frac{1}{\epsilon}) + \frac{1}{\epsilon}\ln(\frac{1}{\delta})\right) \tag{4.2}$$

Für weitere Abschätzungen siehe [AB92].

4.2.3 Effizientes *PAC*

Bis hier haben wir uns nur überlegt, wann ein Hypothesenraum überhaupt erlernbar ist. Im allgemeinen ist jedoch die Zeit, die zum Erlernen eines Konzepts zur Verfügung steht, eingeschränkt. Wir führen dafür den Begriff der effizienten Erlernbarkeit ein. Wir werden häufig der Einfachheit halber *effizient* erlernbar statt *in Polynomialzeit berechenbar* schreiben. Wir beginnen mit der effizienten Variante der *PAC* Erlernbarkeitsdefinition.

Definition 4.10 *pPAC*
Ein Genauigkeitswert ϵ und ein Vertrauenswert δ sind gegeben. Es muß eine Probengröße m_0 in Abhängigkeit von ϵ und δ geben, so daß für Proben, die entsprechend einer festen aber unbekannten Verteilung generiert werden und mehr als m_0 Muster enthalten, gilt, daß der Algorithmus mit einer Wahrscheinlichkeit von über $1-\delta$ eine Hypothese h liefert, die bei gleicher Verteilung auf dem Beispielraum nur in ϵ der Fälle einen Fehler macht. Dabei muß die Laufzeit des Lernalgorithmus durch ein Polynom über $1/\epsilon$, $1/\delta$ und n beschränkt sein, falls der Hypothesenraum parametrisiert durch einen Parameter n ist.

Es wird also in polynomieller Zeit entsprechend der *PAC* Definition eine Hypothese erzeugt.

Entsprechend wird die Definition der schwachen Erlernbarkeit abgewandelt:

Definition 4.11 *pw2PAC* **(weak** *PAC***)**
Sei C eine Zielkonzeptklasse und H ein Hypothesenraum über der Beispielmenge X. Dann ist C schwach erlernbar durch H, wenn es einen probabilistischen Algorithmus A gibt, der Zugriff auf zwei Orakel (POS und NEG) besitzt (two-Button Modell) und als Eingabe den Zuverlässigkeitsparameter δ erhält. Die Laufzeit des Lernalgorithmus ist durch ein Polynom über $1/\delta$ und n beschränkt, falls der Hypothesenraum parametrisiert durch einen Parameter n ist. Der Algorithmus muß nun für jedes Zielkonzept, jede Verteilung auf den positiven beziehungsweise negativen Beispielen und für jeden Zuverlässigkeitsparameter δ halten und eine Hypothese $h \in H$ ausgeben, die mit einer Wahrscheinlichkeit von über $1-\delta$ auf den positiven wie negativen Beispielen weniger als $1/2 - 1/p(n)$ Fehler macht. Dabei ist $p(n)$ ein positives Polynom.

Entsprechend wird die Definition *p2PAC* aus der Definition von *pPAC* abgeleitet, indem man zwei statt einem Orakel verwendet.

Man kann sich auch hier überlegen, daß jeder in *p2PAC* erlernbarer Hypothesenraum auch *pw2PAC* erlernbar ist. Darüber hinaus sind alle in *p2PAC* erlernbaren Hypothesenräume auch in *pPAC* erlernbar [PW88] und umgekehrt.

Beispiele

In *pPAC* erlernbare Konzeptklassen sind zum Beispiel *kCNF*, Monome und monotone *DNF*. Nicht erlernbare Konzeptklassen sind unter plausiblen Annahmen zum Beispiel Boolesche Schaltkreise und *DFA's*.

Es gibt verschiedene Techniken Lernalgorithmen zu erstellen. Wir werden eine davon im nächsten Abschnitt vorstellen.

4.2.4 Occams Razor

Eine Grundtechnik zur Konstruktion eines *PAC* Lernalgorithmus ist Occams Razor (siehe zum Beispiel [Sch89]). Nach der Methode von Occam nimmt man eine ausreichend große Anzahl an Mustern und generiert eine möglichst einfache konsistente Hypothese. Unter *möglichst einfach* versteht man hierbei, daß die Repräsentationsgröße der Hypothese klein sein soll.

Definition 4.12 Occam-Lernalgorithmus
Es seien C und H polynomiell auswertbare Boolesche Repräsentationsklassen. Wähle ein festes $\alpha \geq 1$ und $0 \leq \beta < 1$. A heißt Occam-Algorithmus, falls er bei Eingabe einer Probe P für ein Zielkonzept $c \in C_n$, die aus m positiven und m negativen

Beispielen besteht, die entsprechend D^+ und D^- gezogen wurden, eine Hypothese $h \in H_n$ ausgibt, die konsistent mit der Probe P ist und die Bedingung $|h| \leq n^\alpha m^\beta$ erfüllt. Dabei ist $|h|$ die Länge der Repräsentation in Bits.

Satz 4.2 Notwendige Probengröße für einen Occam-Algorithmus
Dann ist A ein Lernalgorithmus für C durch einen Hypothesenraum H. Die Probengröße die dafür benötigt wird, ist:

$$m = O\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{n^\alpha}{\epsilon}\log n^\alpha \epsilon\right)^{1/(1-\beta)}\right).$$

Es muß also nicht die kürzeste Beschreibung gefunden werden, sondern eine die um einen nichtlinearen Faktor kürzer als die Probe ist.

4.2.5 Vorhersage mit Mitgliedschaftsfragen

Bei der Erlernbarkeit mit Mitgliedschaftsfragen erhält der Lernalgorithmus einen aktiven Einfluß auf die Probe. Der Lernalgorithmus darf nicht nur Muster anfordern sondern auch selbstgewählte Beispiele bewerten lassen. Eine um Mitgliedschaftsfragen erweiterte Erlernbarkeitsdefinition scheint also mächtiger zu sein. In der Tat gibt es auf der Grundlage von plausiblen Annahmen Hypothesenräume, die nur mit Mitgliedschaftsfragen erlernbar sind, zum Beispiel *DFA's*.

Es gibt noch eine Reihe weiterer Erlernbarkeitsdefinitionen, die wir im folgenden Abschnitt kurz erläutern werden.

4.2.6 Andere Erlernbarkeitsdefinitionen

Verteilungsabhängiges Lernen

In der *PAC*-Erlernbarkeitsdefinition muß der Lernalgorithmus für alle möglichen Verteilungen auf dem Beispielraum eine korrekte Hypothese liefern. Diese Voraussetzung ist ziemlich stark. Da die erzeugte Hypothese jedoch auch nach der selben Verteilung auf dem Beispielraum bewertet wird, bleibt die Bedingung erfüllbar. In der Praxis verwendet man bei verteilungsabhängigen Erlernbarkeitsresultaten meist die Gleichverteilung.

Absolute mistake bound

Littlestone hat 1988 [Lit88] das *absolute mistake bound* Erlernbarkeitsmodell vorgestellt. Hierbei muß die maximale Fehlerzahl über einer beliebigen Probe polynomiell über der Beispielgröße und der Zielkonzeptgröße beschränkt sein.

Es ist möglich einen Lernalgorithmus des absolute mistake bound (*AMB*) Erlernbarkeitsmodells in einen Lernalgorithmus im *PAC* Modell zu transformieren. Also sind *AMB* erlernbare Hypothesenräume, auch *PAC* erlernbar. Blum hat gezeigt, daß die Umkehrrichtung nicht gilt, wenn Einwegfunktionen existieren [Blu90].

4.3 Wie kann die Nichterlernbarkeit von Repräsentationsklassen gezeigt werden?

Man kann die Nichterlernbarkeit von Repräsentationsklassen entweder durch einen Widerspruch zur Sicherheit von Kryptosystemen oder durch Reduktion zwischen Lernproblemen nachweisen. Ersteres wird in Kapitel 6 intensiv benützt und dort ausführlich erläutert. Die dafür notwendigen Zusammenhänge zwischen Kryptographie und Algorithmischen Lernen werden im nächsten Kapitel beschrieben.

Die Reduktion zwischen Lernproblemen wird in Kapitel 7 verwendet. Es ist dadurch möglich aus der Erlernbarkeit beziehungsweise Nichterlernbarkeit einer Repräsentationsklasse auf eine andere zu schließen.

5

Kryptographie und Algorithmisches Lernen

Die Erfindung des Computers legte den Kryptoanalytikern ein mächtiges Werkzeug in die Hand, das im zweiten Weltkrieg mit Erfolg eingesetzt wurde. Gleichzeitig besaß man aber auch die Möglichkeit, dieses neue Werkzeug für kyptographische Verfahren einzusetzen. Dies führte zu immer aufwendigeren Verschlüsselungsverfahren, insbesondere da die Computer von Jahr zu Jahr immer effizienter wurden.

Die Erlernbarkeitstheorie entwickelte sich aus den Versuchen, Programme Dame spielen zu lassen, und der anfänglichen Euphorie, daß Computer innerhalb kürzester Zeit alle mentalen Fähigkeiten des Menschen übertreffen würden.

Algorithmisches Lernen - wie wir es heute kennen - entwickelte sich erst mit der Einführung der *PAC*-Erlernbarkeit (vergleiche Definition 4.1) durch Valiant [Val84]. Der bemerkenswerteste Durchbruch in der Kryptographie war die Entwicklung von Public-Key Systemen (Definition 3.2) durch Whitfield Diffie und Martin Hellman [DH76].

Erst durch diese einschneidenden Veränderungen in der Kryptographie und der Erlernbarkeitstheorie wurde es möglich, Resultate interagieren zu lassen.

Wir werden uns im folgenden überlegen, welche Zusammenhänge zwischen einem kryptoanalytischem Vorgang und dem Erlernen einer Repräsentationsklasse bestehen. Einen guten Überblick zu diesem Thema liefert auch Rivest [Riv88].

Wenn ein Kryptoanalytiker ein Public-Key System brechen will, wird er versuchen, den verwendeten geheimen Schlüssel zu ermitteln. Dabei geht man im allgemeinen davon aus, daß die verwendete Entschlüsselungsfunktion durch den geheimen Schlüssel eindeutig bestimmt und die Beziehung bekannt ist. Die Entschlüsselungsfunktion stammt also aus einer bekannten Funktionsfamilie, welche durch den Schlüssel indiziert ist. Demnach entspricht der geheime Schlüssel eines Kryptosystems der gesuchten Zielfunktion in einem Lernproblem. Genauer formuliert entspricht der Schlüsselraum eines Kryptosystems der Klasse der möglichen Zielkonzeptrepräsentationen eines Lernproblems.

Um das Public-Key System zu brechen, kann der Kryptoanalytiker je nach Angriffsschema (vergleiche Abschnitt 3.3) eigene Klartexte verschlüsseln oder sogar eigene Kryptotexte entschlüsseln lassen. Die Möglichkeit eigene Klartexte zu verschlüsseln entspricht dabei dem Beispielorakel eines *PAC* Lernalgorithmus, denn man kann sich dadurch beliebige Klartext-Kryptotextpaare generieren lassen. Diesen Zusammenhang werden wir in Kapitel 6 benützen. Die Möglichkeit Kryptotexte entschlüsseln zu lassen entspricht einer Mitgliedschaftsfrage, denn wir können die gesuchte Funktion für verschiedene Werte abfragen. Wir werden dies in Abschnitt 7 noch genauer erläutern. Der Nachrichtenraum des Kryptosystems entspricht dabei der Verteilung auf dem Beispielraum eines Lernproblems. Die Beispiele, die Nachrichten entsprechen, die nicht versendet werden, erhalten dabei die Wahrscheinlichkeit 0.

Wenn das entsprechende Lernproblem also *PAC* erlernbar ist, sind wir in der Lage aus einem beliebigen Kryptotext mit hoher Wahrscheinlichkeit auf den verschlüsselten Klartext zu schließen. Es besteht also ein direkter Zusammenhang zwischen der Sicherheit eines Public-Key Verfahrens und der Erlernbarkeit der damit verbundenen Konzeptklasse.

Kearns und Valiant (Kapitel 6) haben diese Zusammenhänge dazu benutzt, auf der Grundlage kryptographischer Annahmen die nicht effiziente Erlernbarkeit von Konzeptklassen nachzuweisen. Hierzu entwickelten sie eine Technik, die es ermöglicht, die kryptographischen Entschlüsselungsfunktionen durch sehr einfache Schaltkreise auszuführen. Diese Technik besteht darin, daß die Schaltkreise zusätzliche Informationen erhalten, die die Sicherheit des verwendeten Systems nicht gefährden, jedoch die notwendige Schaltkreiskomplexität, die zur Repräsentation der Entschlüsselungsfunktionen notwendig sind, wesentlich verringern. Weiter konnten sie nachweisen, daß sich diese Resultate auf Optimierungsprobleme anwenden lassen (Abschnitt 6.3).

Angluin und Kharitonov (Kapitel 7) verfeinerten dieses Beweisschema von Kearns und Valiant (Kapitel 6) um auch die nicht effiziente Erlernbarkeit mit Mitgliedschaftsfragen nachweisen zu können. Hierfür verwendeten sie ein gegen gewählte Kryptotextangriffe sicheres Public-Key Verfahren von Naor und Yung ([NY90], Abschnitt 3.3.1). Die Beantwortung der Mitgliedschaftsfragen erreichten sie durch gewählte Kryptotextanfragen des potentiellen Angreifers.

Schließlich verwendeten sie noch Unterschriftsschemata (Kapitel 8) um von einigen

Komplexitätsklassen nachweisen zu können, daß sie ohne Mitgliedschaftsfragen effizient erlernbar sind oder überhaupt nicht.

Wir werden nun in den folgenden drei Kapiteln diese Vorgehensweisen und Resultate diskutieren und nachvollziehen.

6

Boolesche Formeln sind nicht effizient erlernbar

In diesem Kapitel werden wir, den Veröffentlichungen von Kearns und Valiant ([KV89], [KV94]) folgend, nachweisen, daß verschiedene Boolesche Repräsentationsklassen (Abschnitt 4.1) nicht in *pw2PAC* (vergleiche Abschnitt 4.2.3) unter plausiblen kryptographischen Annahmen erlernbar sind. Das bedeutet, daß man in Polynomialzeit keine Hypothese finden kann, die die Beispiele wenigstens um einen polynomiellen Anteil besser klassifizieren kann als durch Münzwurf. Darunter sind insbesondere die Klassen Boolesche Formeln *BF*, Zyklenfreie Deterministische Endliche Automaten *ADFA* und Schwellenschaltkreise mit konstanter Tiefe *dTC*. Diese Nichterlernbarkeitsresultate erweisen sich sogar als unabhängig von dem verwendeten Hypothesenraum. Das heißt, sie sind auch durch einen beliebig mächtigen Hypothesenraum nicht effizient erlernbar (Abschnitt 4.2.1).

Wie wir aus Abschnitt 4.2.3 wissen, ist *pw2PAC* die am einfachsten zu erfüllende polynomialzeitbeschränkte Erlernbarkeitsdefinition, abgesehen von der Erlernbarkeit mit Mitgliedschaftsfragen. Die oben genannten Klassen sind daher ohne Mitgliedschaftsfragen nicht effizient erlernbar. Wir werden im nächsten Kapitel noch näher darauf eingehen.

6.1 Erläuterungen und bekannte Erlernbarkeitsresultate

Wir wollen ein möglichst allgemeingültiges Nichterlernbarkeitsresultat herleiten. Deshalb ist es wichtig, die nichteffiziente Erlernbarkeit für irgend einen Hypothesenraum, eine einfache Erlernbarkeitsdefinition und eine einfache Repräsentationsklasse nachzuweisen.

In der *pw2PAC*-Erlernbarkeitsdefinition wird nur eine um einen polynomiellen Anteil gegenüber Raten verbesserte Vorhersage der Beispiele gefordert. Daher folgt aus der Nichterlernbarkeit in *pw2PAC* auch die Nichterlernbarkeit in *pPAC*. Noch stärkere Nichterlernbarkeitsresultate zeigen wir in Kapitel 7. Dort sind auch trotz Hinzunahme von Mitgliedschaftsfragen viele Repräsentationsklassen nicht erlernbar.

Wir geben nun ein Beispiel dafür, daß eine Repräsentationsklasse schwieriger durch sich selbst zu erlernen ist als durch eine mächtigere Klasse. Wir zeigen damit, wie wichtig es ist, für einen möglichst mächtigen Hypothesenraum die nichteffiziente Erlernbarkeit nachzuweisen. Valiant [Val84] hat gezeigt, daß die Repräsentationsklassen *kCNF*, *kDNF* (Abschnitt 4.1) durch sich selbst erlernbar sind. Durch Ausmultiplizieren einer *kTERM-DNF* erhält man eine *kCNF* und durch Ausmultiplizieren einer *kKLAUSEL-CNF* erhält man eine *kDNF* (siehe auch Abschnitt 4.1). Also ist *kTERM-DNF* beziehungsweise *kKLAUSEL-CNF* durch die Repräsentationsklasse *kCNF* beziehungsweise *kDNF* erlernbar. Jedoch haben Pit und Valiant [PV88] nachweisen können, daß die Erlernbarkeit von *kTERM-DNF* und *kKLAUSEL-CNF* durch sich selbst NP-harte Probleme sind.

An dieser Stelle wären repräsentationsabhängige Erlernbarkeits- und Nichterlernbarkeitsresultate interessant, um die Frage zu klären, wie groß ein Hypothesenraum mindestens sein muß, damit *kTERM-DNF* und *kKLAUSEL-CNF* erlernbar werden. Allgemein formuliert sind repräsentationsabhängige Nichterlernbarkeitsresultate immer dann von Interesse, wenn eine Klasse durch einen beliebigen Hypothesenraum erlernbar ist, jedoch nicht durch sich selbst.

Das einzige ältere repräsentationsunabhängige Nichterlernbarkeitsresultat ist von Goldreich, Goldwasser und Micali [GGM86] und besagt, daß die Größte in Polynomialzeit auswertbare Repräsentationsklasse (Boolesche Schaltkreise *BSK*; Abschnitt 4.1) nicht effizient durch eine beliebige polynomiell auswertbare Repräsentationsklasse erlernbar ist, wenn Einwegfunktionen existieren. Hierbei ist noch anzumerken, daß Schapire [Sch89] nachgewiesen hat, daß Repräsentationsklassen, die nicht in Polynomialzeit auswertbar sind, auch nicht effizient erlernt werden können. Daher beschränken wir uns von nun an auf effizient auswertbare Klassen.

Wir werden zuerst die nichteffiziente Erlernbarkeit der Repräsentationsklasse NC^1 (Abschnitt 2.7) nachweisen. Daraus läßt sich dann durch bereits bekannte Resultate und Komplexitätsvergleiche mit anderen Repräsentationsklassen die nicht effiziente Erlernbarkeit der in der Einleitung erwähnten Repräsentationsklassen Boolesche Formeln *BF*, Zyklenfreie Deterministische Endliche Automaten *ADFA* und Schwellenschaltkreise mit konstanter Tiefe dTC ableiten.

Eine etwas weiter eingeschränkte echte Teilklasse von NC^1 ist die Klasse AC^0 (Abschnitt 2.8), welche einen unbeschränkten fan-in und fan-out besitzt und nur eine konstante Schaltkreistiefe benutzen darf. Es wird jedoch unmittelbar klar sein, daß diese Repräsentationsklasse AC^0 nicht für die Nichterlernbarkeitsbeweise herangezogen werden kann. Damit ist NC^1 die einfachste Repräsentationsklasse, für die die Beweise aus den Abschnitten 6.2.2, 6.2.3 und 6.2.4 durchgeführt werden können.

Bevor wir jedoch die nichteffiziente Erlernbarkeit von NC^1 nachweisen werden, beschreiben wir eine verallgemeinerte Vorgehensweise für eine beliebige Boolesche Repräsentationsklasse R_B.

Der Nachweis der nichteffizienten Erlernbarkeit von R_B wird durch einen Widerspruchsbeweis geführt. Man nimmt zuerst an, daß R_B in *pw2PAC* (Abschnitt 4.2.3) repräsentationsunabhängig erlernbar ist. Dann zeigen wir, daß verschiedene Umkehrfunktionen von Falltürfunktionen (Abschnitt 3.1.2) in R_B unter Verwendung der Falltürinformation repräsentierbar sind. Das heißt, daß die Auswertung der Repräsentation die Umkehrfunktion berechnet. Ein potentieller Lernalgorithmus für die Boolesche Repräsentationsklasse R_B führt dann zur Erlernbarkeit der Falltürumkehrfunktion ohne Falltürinformation. Da man jedoch annimmt, daß die verwendeten Falltürfunktionen sicher sind, kann R_B nicht effizient erlernbar sein.

6.2 Nichteffiziente Erlernbarkeit von Repräsentationsklassen aufgrund der Sicherheit von Public-Key Systemen

Wir werden in diesem Abschnitt drei Lernprobleme betrachten, deren effiziente Erlernbarkeit im Widerspruch zur Sicherheit der unten aufgeführten Public-Key Systeme steht. Jedem System liegt eine Falltürfunktion zugrunde, welche die effiziente Entschlüsselung ohne Falltürinformation verhindert. Folgende Systeme und Falltürfunktionen werden verwendet:

i.) *RSA* Public-Key Verfahren (Abschnitt 3.2.1).
Das *RSA* Verfahren benützt die modulare Potenzfunktion modulo einer zusammengesetzten Zahl als Falltürfunktion. Als Falltürinformation wird hier die Faktorisierung der zusammengesetzten Zahl verwendet und nicht wie in der Originaldefinition der aus der Faktorisierung effizient berechenbare Umkehrexponent. Da Verfahren existieren, die in zu erwartender

Polynomialzeit aus dem Ver- und Entschlüsselungsexponenten die Faktorisierung der zusammengesetzten Zahl berechnen, wird die Sicherheit des Verfahrens dadurch nicht beeinträchtigt.

ii.) Probabilistisches Public-Key System von Shafi Goldwasser und Silvio Micali [GM83] (Abschnitt 3.2.2).
Die Sicherheit dieses Systems beruht auf der Quadratischen Reste Annahme (modulo einer Blum-Integer n; Abschnitt 3.1.1). Die Falltürinformation ist wiederum die Faktorisierung von n.

iii.) Rabin Public-Key Verfahren (Abschnitt 3.2.1)
Die Sicherheit des Rabin Verfahrens beruht nachweisbar auf der nicht effizienten Faktorisierbarkeit von Blum-Integern (Abschnitt 3.1.1).

Wir werden zuerst im Abschnitt 6.2.1 die allgemeine Vorgehensweise erläutern. Dann werden wir für jedes der obengenannten Systeme nachweisen, daß die *pw2PAC* Erlernbarkeit (Kapitel 4.2.3) der Entschlüsselungsfunktionen im Widerspruch zur Sicherheit der zugrundeliegenden Falltürfunktionen steht.

6.2.1 Zugrundeliegendes Schema

In diesem Abschnitt wird eine verallgemeinerte Vorgehensweise zum Nachweis der nichteffizienten Erlernbarkeit von Repräsentationsklassen vorgeführt.

Wir beginnen mit einer Falltürfunktionsfamilie. Als Beispiel betrachten wir die möglichen Verschlüsselungsfunktionen des *RSA* Public-Key Verfahrens (Abschnitt 3.2.1). Eine Instanz aus unserer Beispielfamilie ist das 3-Tupel (p, q, e). Dabei sind p und q die verwendeten Primzahlen und e der öffentliche Schlüssel. Dabei ist zu beachten, daß mit (p, q, e) auch der Entschlüsselungsexponent eindeutig definiert ist und somit auch die Entschlüsselungsfunktion.

Für jede Instanz (p, q, e) aus der Falltürfunktionsfamilie wird eine Repräsentation der Umkehrfunktion unter Verwendung der Falltürinformation erstellt. Die Repräsentation wird aus einer Booleschen Repräsentationsklasse (Abschnitt 4.1) gewählt. Boolesche Repräsentationsklassen können jedoch nur 0 oder 1 ausgeben und wir benötigen eine Darstellung der Umkehrfunktion, die alle n-Bit des Klartextes ausgibt. Es ist also notwendig für jedes Bit des Klartextes eine eigene Repräsentation anzugeben

Wenn man davon ausgeht, daß die Repräsentationsklasse erlernbar ist, dann kann man die Repräsentationen der Umkehrfunktion für die Bits des Klartextes anhand von zufälligen Beispielen erlernen. Solche Beispiele lassen sich leicht erzeugen. Man wählt zum Beispiel einen Klartext, der an der i-ten Bitposition eine 1 besitzt und verschlüsselt ihn. Dadurch hat man ein positives Beispiel für die i-te Repräsentation der Umkehrfunktion erzeugt. Wenn ein Lernalgorithmus in der Lage ist, die Repräsentationen zu erlernen, kann er die damit verbundene Falltürfunktion invertieren.

Aber was bedeutet das für unser Beispiel? Wenn wir die Sicherheit des *RSA* Public-Key Verfahrens voraussetzen, ist es nicht in Polynomialzeit möglich ohne die Falltürinformation zu entschlüsseln. Wenn die Repräsentationsklasse, in der die Umkehrfunktion des *RSA* Systems berechenbar ist, erlernbar wäre, könnte ein effizienter Lernalgorithmus die Repräsentation der Entschlüsselungsfunktion erlernen. Dies steht aber im Widerspruch zur Sicherheit des *RSA* Verfahrens. Daher kann die Repräsentationsklasse nicht erlernbar sein.

Wir benötigen für den Hauptsatz dieses Abschnitts noch einen genauen Begriff einer Falltürfunktionsfamilie. Wir verwenden die von Yao [Yao82] eingeführte Definition, wobei wir insbesondere an Falltürfunktionen interessiert sind, die in Public-Key Systemen verwendet werden.

Die Definition einer Falltürfunktionsfamilie wird, um den Zusammenhang zu verdeutlichen, als Public-Key System beschrieben. Eine Falltürfunktionsfamilie f ist durch einen Sicherheitsparameter n parametrisiert. Eine Verteilung P_n erzeugt dabei die n-Bit langen öffentlichen und privaten Schlüssel (k, k'). Wir gehen dann von der vereinfachten Annahme aus, daß die Nachrichten die selbe Länge wie die Schlüssel besitzen. Für jeden öffentlichen Schlüssel k beschreibt Q_k die Verteilung auf dem $|k|$-Bit langen Nachrichtenraum. Diese Verteilung wird auch verwendet, um Nachrichten auszusondern, die mit dem verwendeten öffentlichen Schlüssel nicht verschlüsselt werden können. Dies wird später anhand des *RSA* Verschlüsselungverfahrens noch genauer erläutert.

Die Falltürfunktion $f_{(k,k')}(m)$ bildet dann einen entsprechend Q_k erzeugten Klartext m auf einen Kryptotext c ab. Auffällig ist, daß in dieser Definition noch nichts über die Sicherheit der Falltürfunktion ausgesagt wird. Sie wird erst durch die Definition 6.2 formalisiert.

Es folgt noch die formale Definition von Yao [Yao82]:

Definition 6.1 Falltürfunktionsfamilie f
Sei $P = P_n$ eine Klasse von Wahrscheinlichkeitsverteilungen, wobei für $n \geq 1$ die Verteilung über Paare $(k, k') \in \{0,1\}^n \times \{0,1\}^n$ definiert ist. Dabei steht k für den n-Bit langen öffentlichen und k' für den n-Bit langen privaten Schlüssel. Sei $Q = \{Q_k\}$ eine durch k parametrisierte Klasse von Wahrscheinlichkeitsverteilungen. Wenn n die Länge von k ist ($n = |k|$) gilt, daß die Verteilung Q_k über binäre Strings der Länge n ($\{0,1\}^n$) definiert ist. Q_k steht für die Verteilung des Nachrichtenraums.

Die Falltürfunktion $f_{(k,k')}$ bildet einen öffentlichen Schlüssel $k \in \{0,1\}^n$ und eine Nachricht $m \in \{0,1\}^n$ auf einen Kryptotext $f_{(k,k')}(x) \in \{0,1\}^n$ ab. Die Falltürfunktionsfamilie ist dann $f = \bigcup_{n \geq 1, |k|=n} f_{(k,k')}$.

Unter Verwendung der letzten Definition können wir die Sicherheit einer solchen Falltürfunktionsfamilie formalisieren. Dies benötigen wir als Grundlage für den Widerspruchsbeweis.

Definition 6.2 $\alpha(n)$**-sicheres Falltürschema**
Es gelten folgende 5 Eigenschaften:

1.) *Existenz eines effizienten Schlüsselerzeugers:*
Es existiert ein probabilistischer Polynomialzeitalgorithmus (Abschnitt 2.3) G, der bei Eingabe 1^n ein Schlüsselpaar (k,k') entsprechend der Verteilung P_n ausgibt.

2.) *Existenz eines effizienten Nachrichtenerzeugers:*
Es existiert ein probabilistischer Polynomialzeitalgorithmus M, der bei Eingabe k eine Nachricht x entsprechend der Verteilung Q_k ausgibt.

3.) *Polynomialzeitverschlüsselung:*
Es existiert ein Polynomialzeitalgorithmus E, der bei Eingabe von k und x den Funktionswert $f(k,x)$ ausgibt.

4.) *Polynomielle Entschlüsselung ist unwahrscheinlicher als $\alpha(n)$:*
Sei A ein beliebiger probabilistischer Polynomialzeitalgorithmus.
Wir führen nun für ein beliebiges n folgendes Experiment aus:
Erzeuge ein Schlüsselpaar (k,k') entsprechend der Verteilung P_n und eine Nachricht x entsprechend der Verteilung Q_k. Dann muß die Wahrscheinlichkeit, daß der Algorithmus A bei Eingabe von k und $f(k,x)$ die Klartextnachricht x ausgibt, kleiner als $\alpha(n)$ sein.
Wir wissen also, daß kein probabilistischer Polynomialzeitalgorithmus bei Schlüsselgröße $n = |k|$ mit einer Wahrscheinlichkeit größer als $\alpha(n)$ die zugrundeliegende Nachricht entschlüsseln kann.

5.) *Polynomielle Entschlüsselung mit Falltürinformation:*
Es gibt einen Polynomialzeitalgorithmus D, der bei Eingabe des Schlüsselpaares (k,k') und des Kryptotextes $f(k,x)$ die enthaltene Nachricht x berechnen kann.
Der legale Empfänger kann also die Nachricht effizient berechnen.

Wir verdeutlichen uns diese Definitionen anhand des *RSA* Public-Key Verfahrens. Der Schlüssel wird erzeugt, indem N gleichverteilt aus der Menge der n-Bit langen Primzahlen generiert wird. Dies ist möglich, da probabilistische Primzahltests existieren, die in zu erwartender Polynomialzeit halten. Dann wird e unter Gleichverteilung aus $Z^*_{\varphi(p \cdot q)}$ gezogen und $d = e^{-1} \mod \varphi(n)$ durch den extended Euklid Algorithmus berechnet. e und d entsprechen dem Schlüsselpaar (k, k'). P_n ist dann die Verteilung, die sich für $k = (p \cdot q, e)$ entsprechend dem oben beschriebenen Verfahren ergibt. Die Nachrichtenverteilung Q_k ist die Gleichverteilung auf $Z_{p \cdot q}$. Die Nachrichten können ohne weiteres entsprechend der Verteilung Q_k erzeugt werden.

Die Verschlüsselungsfunktion ist dann die modulare Potenzfunktion $f(k, x) = f((p \cdot q, e), x) = x^e \mod p \cdot q$. Sie ist also auch effizient berechenbar (Abschnitt 3.1.1). Man nimmt an, daß das *RSA* Verfahren sicher ist, damit ist Bedingung 4 der Definition erfüllt. Die Entschlüsselung ist mit d in Polynomialzeit möglich und damit ist auch Bedingung 5 erfüllt.

Da wir ja die polynomialzeitliche Nichterlernbarkeit einer Konzeptklasse zeigen wollen, müssen wir uns noch überlegen, wie man die Umkehrfunktion einer Falltürfunktion in einer Booleschen Repräsentationsklasse darstellen kann. Weil man in einer Booleschen Repräsentation nur die Ausgaben 0 und 1 erhält, ist es notwendig eine Repräsentation für jedes Bit des Klartextes zu finden. Die n Repräsentationen bilden dann zusammen die Umkehrfunktion. Es folgt noch die formale Definition.

Definition 6.3 $(\mathbf{P}, \mathbf{Q}, \mathbf{f})$ invertierbar in C
Sei $C = \bigcup_{n \geq 1} C_n$ eine parametrisierte Boolesche Repräsentationsklasse. Ein Falltürschema (P, Q, f) ist invertierbar in C bei gegebener Falltürinformation, wenn für jeden Sicherheitsparameter $n \geq 1$, für alle Schlüsselpaare $(k, k') \in \{0,1\} \times \{0,1\}$ und für alle $i : 1 \leq i \leq n$ eine Repräsentation $c^i_{(k,k')} \in C_n$ existiert, die für alle $x \in \{0,1\}^n$ bei Eingabe von $f_{(k,k')}(x)$, das i-te Bit von x ausgibt.

Die Repräsentation $c^i_{(k,k')} \in C_n$ invertiert also die Verschlüsselungsfunktion $f_{(k,k')}$ für das i-te Bit des Klartextes.

Mit den vorangegangenen Definitionen sind wir nun in der Lage, den Hauptsatz dieses Abschnitts zu formulieren. Der Satz lautet wie folgt:

Satz 6.1 *Sei $p(n)$ irgendein Polynom, und sei $\alpha(n) \geq \frac{1}{p(n)}$. Sei (P, Q, f) ein α-sicheres Falltürschema und sei C eine parametrisierte Boolesche Repräsentationsklasse. Dann gilt: Wenn (P, Q, f) invertierbar in C bei Verwendung der Falltürinformation ist, dann ist C nicht in Polynomialzeit pPAC erlernbar.*

Beweisskizze:
Wir führen einen Widerspruchsbeweis aufgrund der Annahme, daß die Boolesche Repräsentationsklasse C_n effizient in $pPAC$ erlernbar ist.

Da das Falltürschema (P, Q, f) *invertierbar in C unter Verwendung der Falltürinformation* ist, existiert für jede Bitposition i der Klartextnachricht eine Repräsentation der Umkehrfunktion in C. Wenn C jedoch effizient erlernbar ist, existiert ein effizienter Lernalgorithmus, der diese Repräsentationen anhand von Beispielen erlernt. Man erzeugt solche Beispiele, indem man einen Klartext x generiert, der an der i-ten Bitposition eine 1 (bzw. 0) besitzt und x daraufhin verschlüsselt. Die Verschlüsselungsfunktion ist nach Bedingung 3 des α-sicheren Falltürschemas in Polynomialzeit berechenbar. Der Kryptotext von x ist dann ein positives (bzw. negatives) Beispiel für die Repräsentation $c^i_{(k,k')}$.

Es kann nur das Problem auftreten, daß ein angefordertes positives (bzw. negatives) Beispiel nicht in Polynomialzeit gefunden werden kann, wenn Q_k nur sehr selten Klartexte erzeugt, die an der zu erlernenden Bitposition eine 1 (bzw. 0) besitzen. In diesem Fall kann man jedoch anhand von Chernoff Schranken [Che52] zeigen, daß nach einer ausreichenden Anzahl an Versuchen die konstanten Hypothesen $h = 1$ (bzw. $h = 0$) mit ausreichender Wahrscheinlichkeit die gesuchte Hypothese annähern.

Diese Vorgehensweise wird für jedes Bit wiederholt. Man kann nun die Zuverlässigkeits- und Genauigkeitsparameter der Lernalgorithmen für die einzelnen Bits des Klartextes so wählen, daß die richtige Vorhersage des kompletten Klartextes wahrscheinlicher als $\alpha(n)$ wird. Dies steht im Widerspruch zur vierten Bedingung des α-sicheren Falltürschemas. Also ist die Repräsentationsklasse C_n nicht erlernbar.
□

Beweis:
Angenommen A ist ein Polynomialzeit Lernalgorithmus für die Repräsentationsklasse C. Wir werden den Algorithmus A als Unterprogramm für einen polynomialzeitbeschränkten Invertierungsalgorithmus B benutzen, der bei Eingabe des öffentlichen Schlüssels k und des Kryptotextes $f(k, x)$ die Klartextnachricht x mit hoher Wahrscheinlichkeit ausgibt. Damit erhalten wir dann den Widerspruch zur vierten Bedingung des Falltürschemas.

Sei das Schlüsselpaar (k, k') entsprechend der Verteilung P_n und der Klartext x entsprechend der Verteilung Q_k erzeugt worden. Der Invertierungsalgorithmus B verhält sich dann wie folgt:

B erhält als Eingabe k und $f(k, x)$. Es reicht dabei zu zeigen, daß ein Algorithmus B existiert, der in zu erwartender polynomieller Zeit in n hält. Für jedes Bit $1 \leq i \leq n$ des Klartextes simuliert der Invertierungsalgorithmus B den Lernalgorithmus A. Dabei ist das Ziel des i-ten Durchlaufs, das i-te Bit des Klartextes aus dem Kryptotext zu entschlüsseln. A erhält als Eingabe den Genauigkeitsparameter $\epsilon = \alpha(n)/n$. Der Zuverlässigkeitsparameter δ wird hier nicht betrachtet, da er so klein gewählt werden kann, daß die Zuverlässigkeit bei polynomiell vielen Anfragen gewährleistet ist. Zudem läßt sich die Gesamthypothese, die der Algorithmus B ausgibt überprüfen. Wir werden darauf am Ende des Beweises noch näher eingehen.

Wenn A ein positives Muster anfordert, läßt B den effizienten Nachrichtenerzeuger so lange Nachrichten x' generieren, bis das i-te Bit von x' eins ist. Dann berechnet B den Kryptotext $f(k, x')$ mit dem effizienten Verschlüsselungsalgorithmus und übergibt das Muster $(f(k, x'), 1)$ an A. Für ein negatives Beispiel wird entsprechend verfahren.

Wenn dabei nach $O(\frac{1}{\epsilon} \cdot \ln(\frac{n}{\delta}))$ Versuchen das erwartete 1 beziehungsweise 0 Bit nicht erscheint, nimmt B an, daß die Verteilung Q_k in den meisten Fällen Zahlen ausgibt, bei denen das i-te Bit 0 (bzw. 1) ist. B setzt die Hypothese h_k^i dann auf konstant 1 (beziehungsweise 0). Man kann so vorgehen, da die Genauigkeit des Lernalgorithmus A mit derselben Verteilung auf den Klartexten getestet wird, mit der wir versuchen, das entsprechende Bit zu erzeugen. Wir werden nach diesem Beweis mit Hilfe von Chernoff Schranken [Che52] nachweisen, daß die Wahrscheinlichkeit, daß der Algorithmus dabei in mehr als ϵ der Fälle einen Fehler macht, kleiner als $\frac{\delta}{n}$ bleibt.

Man kann nach Voraussetzung $1/\alpha(n)$ nach oben durch ein Polynom abschätzen, also ist auch $1/\epsilon$ polynomiell in n. Also hält der simulierte Lernalgorithmus nach polynomieller Zeit. Das $\alpha(n)$-sichere Falltürschema (P, Q, f) besitzt nach Voraussetzung für jedes Bit i eine Repräsentation $c^i_{(k,k')} \in C_n$. Darum sind die berechneten Muster konsistent mit einem Konzept aus C_n und der Lernalgorithmus gibt entsprechend seiner Eingabeparameter (ϵ, δ) korrekte Hypothesen $h_k^i \; \forall 1 \leq i \leq n$ aus.

Der Invertierungsalgorithmus B muß dann nur noch die Gesamthypothese

$$h_k^1(f(k,x)), \cdots, h_k^n(f(k,x))$$

ausgeben um $f(k, x)$ zu invertieren. Wir müssen uns an dieser Stelle noch überlegen, warum der Zuverlässigkeitsparameter δ für den Lernalgorithmus A keine Rolle spielt.

Wir können die Gesamthypothese wiederum mit Chernoff Schranken überprüfen, indem wir sie mit einer polynomiellen Anzahl von verschlüsselten Klartexten testen. Wenn der Test ergibt, daß die Hypothese mit großer Wahrscheinlichkeit einen größeren Fehler als ϵ aufweist, wird der Invertierungsalgorithmus B erneut gestartet. Da δ so gewählt werden kann, daß nur in einem polynomiell kleinen Bruchteil ein solcher Fehler auftritt, ist diese Vorgehensweise in zu erwartender polynomieller Zeit erfolgreich.

Die Wahrscheinlichkeit, daß ein Bit dieser Ausgabe sich von x unterscheidet, ist dann maximal $n \cdot \epsilon < \alpha(n)$. Dies widerspricht der Annahme, daß (P, Q, f) ein $\alpha(n)$-sicheres Falltürschema ist. $\square$

Es bleibt noch zu zeigen, daß $O(\frac{1}{\epsilon} \cdot \ln(\frac{n}{\delta}))$ Fehlversuche genügen, um mit ausreichender Genauigkeit ϵ und Zuverlässigkeit δ die Hypothese auf Konstant 1 (bzw. 0) setzen zu können. Als Grundlage für diesen Beweis benötigen wir *Chernoff Schranken* [Che52].

Im obigen Beweis wird der Klartext x $m = O(\frac{1}{\epsilon} \cdot \ln(\frac{n}{\delta}))$ mal entsprechend der Verteilung Q_k erzeugt. Die Wahrscheinlichkeit des Auftretens eines Ereignisses E sei $0 \leq p \leq 1$. Die einzelnen Ereignisse sind voneinander unabhängig. Dies entspricht einer Binomialverteilung [BS91]. Für die Abschätzung benötigen wir noch zwei Definitionen. Sei $LE(p, m, r)$ die Wahrscheinlichkeit für das Auftreten von echt weniger als r Ereignissen, bei m Versuchen und Wahrscheinlichkeit p für das Auftreten eines Ereignisses. Entsprechend sei $GE(p, m, r)$ die Wahrscheinlichkeit, daß echt mehr als r Ereignisse auftreten. Durch folgende Chernoff Schranken läßt sich die Abweichung vom Erwartungswert nach unten und oben abschätzen.

$$\begin{aligned} LE(p, m, (1-\alpha)mp) &\leq exp(-\alpha^2 mp/2) \\ GE(p, m, (1+\alpha)mp) &\leq exp(-\alpha^2 mp/3) \end{aligned}$$

Sei r die Häufigkeit des Auftretens des Ereignisses. Wir zeigen nun, daß der ermittelte Schätzwert $\hat{p} = \frac{r}{m}$ nach maximal $m = \frac{3}{\epsilon} \ln\left(\frac{2n}{\delta}\right)$ Versuchen mit Wahrscheinlichkeit $1 - \frac{\delta}{n}$ in den Schranken $p - \epsilon < \hat{p} < p + \epsilon$ bleibt. Dies ist genau dann der Fall, wenn die beiden Wahrscheinlichkeiten, daß das gemessene $\hat{p}$ oberhalb beziehungsweise unterhalb der Schranken liegt ($\hat{p} \leq p - \epsilon$ und $\hat{p} \geq p + \epsilon$), kleiner als $\frac{\delta}{2n}$ sind. Es folgt nun die Abschätzung für die untere Schranke.

Falls $p = 0$ gelten die Chernoff Schranken für ein beliebiges m. Wir betrachten im folgenden nur den Fall $p \neq 0$. Wir wählen ein $\alpha < \frac{\epsilon}{p}$ denn dann gilt:

$$(1-\alpha)mp \geq (1 - \frac{\epsilon}{p})mp = m(p - \epsilon)$$

also wird mit $m(p - \epsilon)$ ein größerer Bereich abgeschätzt und es gilt:

$$LE(p, m, m(p-\epsilon)) \leq LE(p, m, (1-\alpha)mp)$$

Es reicht also zu zeigen, daß für ein beliebiges $\alpha < \frac{\epsilon}{p}$ und $m = \frac{2}{\epsilon} \ln\left(\frac{2n}{\delta})\right)$ die Gleichung $LE(p, m, (1-\alpha)mp) \leq \frac{\delta}{2n}$ erfüllt ist.

$$
\begin{aligned}
LE(p, m, (1-\alpha)mp) \quad & \underbrace{\leq}_{Chernoff} \quad \exp\left(\frac{-\alpha^2 mp}{2}\right) \\
& \underbrace{=}_{m=\frac{2}{\epsilon}\ln\left(\frac{2n}{\delta}\right)} \quad \exp\left(\frac{-\alpha^2 \frac{2}{\epsilon}\ln\left(\frac{2n}{\delta}\right)p}{2}\right) \\
& \underbrace{\leq}_{\epsilon \geq \alpha p} \quad \exp\left(\alpha \cdot \left(-\ln\left(\frac{2n}{\delta}\right)\right)\right) \\
& \underbrace{=}_{-\ln\left(\frac{\alpha}{\beta}\right)=\ln\left(\frac{\beta}{\alpha}\right)} \quad \exp(\alpha)^{\ln\left(\frac{\delta}{2n}\right)} \\
& \underbrace{\leq}_{\alpha \in (0,1)} \quad e^{\ln\left(\frac{\delta}{2n}\right)} = \frac{\delta}{2n}
\end{aligned}
$$

Also ist die Wahrscheinlichkeit, bei $m = \frac{2}{\epsilon}\ln\left(\frac{2n}{\delta}\right))$ Versuchen die untere Schranke zu unterschreiten, kleiner als $\frac{\delta}{2n}$. Der Beweis für die obere Schranke läuft analog mit $m = \frac{3}{\epsilon}\ln\left(\frac{2n}{\delta}\right))$.

Man kann also m in polynomieller Abhängigkeit von $\frac{1}{\epsilon}$ und $\frac{1}{\delta}$ so wählen, daß die Wahrscheinlichkeit innerhalb der Schranken $p - \epsilon < \hat{p} < p + \epsilon$ zu bleiben größer als $1 - \frac{\delta}{n}$ ist.

Wenn wir also bei allen m Versuchen immer eine 1 erhalten haben, ist die tatsächliche Ereigniswahrscheinlichkeit p mit Sicherheit $1 - \frac{\delta}{n}$ in den Grenzen $1 \geq p > \hat{p} - \epsilon$. Der Parameter δ kann hier wiederum so klein gewählt werden, daß er für einen Polynomialzeitalgorithmus keine Rolle spielt (vergleiche Abschnitt 2.6). Die Fehlerwahrscheinlichkeit ist damit ϵ. Der andere Fall folgt analog. $\square$

6.2.2 Nichteffiziente Erlernbarkeit beruhend auf der Sicherheit des *RSA* Public-Key Systems

Im letzten Abschnitt haben wir gesehen wie man aus der Sicherheit von kryptographischen Falltürfunktionen Rückschlüsse auf die Erlernbarkeit von Konzeptklassen ziehen kann. In diesem Abschnitt werden wir zeigen, daß die Klasse NC^1 (Abschnitt 2.7) nicht effizient erlernt werden kann, unter der Voraussetzung, daß das *RSA* Public-Key Verfahren sicher ist. Zur Erinnerung geben wir noch einmal die Verschlüsselungsfunktion des *RSA* Verfahrens an.

$$RSA(x, N, e) = x^e \mod N$$

N ist dabei die aus zwei Primzahlen zusammengesetzte Zahl und $e \in Z^*_{\varphi(n)}$ der Verschlüsselungsexponent (siehe auch Abschnitt 3.2.1).

Um nicht wie im vorherigen Abschnitt für jedes Bit des Klartextes eine Repräsentation angeben zu müssen, bedienen wir uns des Satzes von Alexi, Chor, Goldreich und Schorr [ACG88].

Satz 6.2 *Sei x ein Klartext, N das Produkt zweier Primzahlen und e der Verschlüsselungsexponent. Dann sind die folgenden zwei Probleme probabilistisch polynomialzeitredzuierbar (Abschnitt 2.6) zu einander.*

(i) Gib bei Eingabe des Kryptotextes $RSA(x, N, e)$, der zusammengesetzten Zahl N und des Verschlüsselungsexponenten e den Klartext x aus.

(ii) Gib bei Eingabe des Kryptotextes $RSA(x, N, e)$, der zusammengesetzten Zahl N und des Verschlüsselungsexponenten e das niederwertigste Bit $LSB(x)$ von x mit einer Wahrscheinlichkeit über $1/2 + 1/p(l)$ aus. Dabei bezeichnet $p()$ ein bestimmtes Polynom, $l = \log(N)$ und $LSB(x)$ steht für das niederwertigste Bit von x.

Die Wahrscheinlichkeit wird entsprechend der Variable x, die gleichverteilt aus Z_N gezogen wird, und dem Zufallsverhalten der probabilistischen Algorithmen berechnet.

Es reicht also aus die Repräsentation der Umkehrfunktion für das niederwertigste Bit des Klartextes zu betrachten. Denn aus der Vorhersagbarkeit des niederwertigsten Bits folgt mit Satz 6.2 auch die Invertierung der RSA-Verschlüsselungsfunktion.

Wir wollen jedoch auch von einfacheren Repräsentationsklassen nachweisen, daß sie nicht effizient erlernbar sind. Um den Widerspruch für eine möglichst einfache Repräsentationsklasse herleiten zu können, bedienen wir uns eines Tricks. Wir versehen die Beispiele mit erweiterten Informationen, die die Berechnung der Umkehrfunktion erleichtern. Hierfür benötigen wir eine Hilfsdefinition:

Definition 6.4 powers(z,N)

$$powers(z, N) = z \mod N, z^{(2^1)} \mod N, z^{(2^2)} \mod N, \ldots, z^{(2^{\lceil \log(N) \rceil})} \mod N$$

Jedoch darf diese Zusatzinformation die Sicherheit der Falltürfunktion nicht gefährden. Dies ist auf jeden Fall gewährleistet, wenn die zusätzlichen Informationen in Polynomialzeit aus den Beispielen berechenbar sind. Der Lernalgorithmus kann daraus keinen Vorteil ziehen, denn wenn die Zusatzinformation nicht bereits im Beispiel vorhanden wäre, könnte sie nachträglich durch den Lernalgorithmus in Polynomialzeit berechnet werden. Dies ist auch bei unserer Definition $powers(z, N)$ der Fall, denn es werden nur in Abhängigkeit der Eingabegröße viele Multiplikationen berechnet.

Unter Verwendung dieses Tricks können wir die Repräsentierbarkeit der Umkehrfunktion des RSA Verschlüsselungssystems in der Repräsentationsklasse NC^1 nachweisen.

Wir wissen jetzt also, daß man die Nichterlernbarkeit einer Repräsentationsklasse C nachweisen kann, indem man eine Falltürfunktion findet (eventuell mit in die Beispiele kodierten Zusatzinformationen), die in C repräsentierbar ist.

Wir konstruieren nun die Umkehrfunktion einer Falltürfunktion unter Verwendung der Falltürinformation. Dafür definieren wir eine Repräsentationsklasse C_n die Umkehrfunktionen der RSA Systeme bei gegebenem p, q und e repräsentieren soll.

Die Repräsentationsklasse C_n ist auf der Grundmenge $\{0, 1\}^n$ definiert. Wir benötigen jedoch nur einen Teil dieser Grundmenge.

Der Beispielraum ist mehrfach eingeschränkt:

(i) Zulässige Nachricht:
m muß in den Grenzen 0 und $n - 1$ liegen.

(ii) Zusatzinformationen:
Es werden zur Erleichterung der Repräsentierung der Umkehrfunktion weitere Informationen (powers) in die Muster eingebettet.

Definition 6.5 C_n
Sei l die größte natürliche Zahl, die die Gleichung $4l^2+4l \leq n$ erfüllt. Jede Repräsentation aus C_n ist durch ein 3-Tupel p, q, e definiert. Hierbei sind p und q Primzahlen mit genau l Bits und $e \in Z^*_{\varphi(N)}$. Dabei gilt $N = p \cdot q$. (Aus der Zahlentheorie wissen wir, daß dann $ggT(e, \varphi(N))$ gleich 1 ist.)

Hierbei beschreibt das 3-Tupel p, q, e das RSA Kryptosystem vollständig. Denn bei gegebenem Primzahlen p, q und Verschlüsselungsexponent e sind auch $N = p \cdot q$ und der Entschlüsselungsexponent d eindeutig definiert.

Die Bedingung $4l^2 + 4l \leq n$ sorgt in Verbindung mit der folgenden Definition dafür, daß C_n gleichzeitig durch die Repräsentations- und die Beispielgröße parametrisiert ist (Abschnitt 4.1).

Definition 6.6 Betrachtete Muster
Für $r_{(p,q,e)} \in C_n$ ist ein betrachtetes Muster ein Binärstring der Form:

$$< binary(powers(RSA(x, N, e), N), N, e), LSB(x)) >, x \in Z_N$$

Wobei $binary(a_1, \ldots, a_m)$ *die binäre Repräsentation der Zahlenfolge* a_1 *bis* a_m *ist.*

Da l die Länge der verwendeten Primzahlen ist, kann N maximal $2l$ Bits besitzen. Die maximale Länge des Beispiels ist damit: $2l \cdot 2l + 2l + 2l = 4l^2 + 4l \leq n$.

Nach der Definition der Erlernbarkeit muß der Lernalgorithmus für alle Verteilungen auf dem Beispielraum ein gleichermaßen korrektes Ergebnis liefern. Da wir die nichteffiziente Erlernbarkeit zeigen wollen, reicht es aus, den Widerspruch für eine bestimmte Verteilung herzuleiten. Wir nehmen als Verteilung der positiven Muster D^+ für unser Lernproblem die Gleichverteilung auf den *betrachteten positiven Mustern* und entsprechend für D^- die *betrachteten negativen Muster*, für ein festes $r_{(p,q,e)}$.

Es reicht hier aus nur eine spezielle Verteilung D^+, D^- für jede Zielkonzeptrepräsentation anzugeben, da die *pw2PAC* Erlernbarkeitsdefinition die Erlernbarkeit für jedes mögliche Verteilungspaar gewährleistet. Der Widerspruch läßt sich also schon aufgrund eines Verteilungspaares herleiten.

Um den Widerspruchsbeweis führen zu können, werden wir als nächstes zeigen, daß die Entschlüsselungsfunktion des *RSA* Public-Key Systems sich in NC^1 repräsentieren läßt.

Nachweis der Existenz einer Repräsentation der Entschlüsselungsfunktion in NC^1

Wir zeigen, daß für jede Repräsentation $c_{(p,q,e)}$ ein äquivalenter NC^1-Schaltkreis existiert. Es existiert also ein Schaltkreis mit logarithmischer Tiefe und polynomieller Größe, der bei Eingabe von $binary(powers(RSA(x, N, e), N), N, e)$ für ein beliebiges $x \in Z_N$ das niederwertigste Bit $LSB(x)$ ausgibt. Wobei N das Produkt der zwei Primzahlen p und q ist und $r = RSA(x, N, e)$ die Verschlüsselung von x mit der *RSA*-Verschlüsselungsfunktion.

Da e aus $Z^*_{\varphi(N)}$ ist, existiert ein inverses Element d (Abschnitt 3.1.1) und damit ein Entschlüsselungsexponent mit der Eigenschaft:

$$d \cdot e \equiv_{\varphi(N)} 1$$

Und es gilt:

$$r^d \equiv_N x^{e \cdot d} \equiv_N x^{k \cdot \varphi(N)+1} \equiv_N x$$

Der Schaltkreis muß also r^d berechnen und das 0-te Bit des Ergebnisses ausgeben. Dies ist möglich durch Multiplikation der entsprechenden Potenzen r^{2^i}, die im Beispiel bereitgestellt sind (siehe auch Abschnitt 3.1.1). P. Beame, S. Cook und H. Hoover [BCH86] haben gezeigt, daß diese Funktion durch einen NC^1-Schaltkreis berechenbar ist.

Mit den obigen Ergebnissen können wir nun den Beweis der nicht effizienten Erlernbarkeit auf der Grundlage der Sicherheit des *RSA* Systems führen. Der Beweis wird im Prinzip wie im vorhergehenden Abschnitt geführt, jedoch können wir aus dem Satz über die Sicherheit des *RSA* Verschlüsselungssystems zwei Vorteile ziehen. Zum einen reicht es die nichteffiziente Erlernbarkeit der Umkehrfunktion für das niederwertigste Klartextbit zu zeigen. Wir können uns also auf die Repräsentation der niederwertigsten Bits beschränken. Zum anderen folgt schon aus der schwachen Erlernbarkeit mit zwei Orakeln (*pw2PAC*) der Widerspruch.

Beweis der nichteffizienten Erlernbarkeit der NC^1-Schaltkreise

Wir nehmen an, daß ein schwacher polynomialzeitbeschränkter Lernalgorithmus A (4.2.3) für die Repräsentationsklasse NC^1 existiert. Wir beschreiben nun, wie man einen Invertierungsalgorithmus B konstruieren kann, der durch Simulation von A die *RSA* Verschlüsselungsfunktion invertiert. Dabei ist zu beachten, daß der Parameter δ so klein gewählt werden kann, daß trotzdem ein polynomiell verwertbarer Vorteil gegenüber Raten verbleibt. Wir wählen zum Beispiel $\delta = \frac{1}{p+3}$, wenn die Genauigkeit des Algorithmus $\frac{1}{2} + \frac{1}{p(n)}$ beträgt. In dieser Argumentation liegt jedoch das Problem, daß man die Genauigkeit des Algorithmus nicht im voraus kennt. Die Hypothese läßt sich hier auch noch nicht testen, da nur das niederwertigste Bit des Klartextes erlernt wird und nicht der gesamte Klartext wie in Abschnitt 6.2.1.

Deshalb verwenden wir eine andere Vorgehensweise. Wir wählen zum Beispiel die maximale Fehlerwahrscheinlichkeit $\delta = \frac{1}{4}$. Wir erhalten dann mit einer Wahrscheinlichkeit von $\frac{3}{4}$ eine Hypothese, die einen überpolynomiellen Vorteil gegenüber Raten besitzt. Wir wenden nun die Reduktion von Satz 6.2 an. Dadurch erhalten wir einen probabilistischen Algorithmus, der die *RSA* Verschlüsselungsfunktion invertiert. Hier können wir nun durch Chernoff Schranken die Entschlüsselungsfunktion testen. Wenn der so erzeugte Invertierungsalgorithmus erfolgreich ist, dann ist auch unsere Hypothese ausreichend korrekt und wir sind fertig. Wenn die Überprüfung eine fehlerhafte Vorhersage ergibt, beginnen wir erneut von vorne. Diese Vorgehensweise ist in zu erwartender Polynomialzeit erfolgreich.

Sei N das Produkt zweier unbekannter l-Bit Primzahlen p und q. Der Verschlüsselungsexponent sei e. Dem Invertierungsalgorithmus B ist nur N und e bekannt. Der Algorithmus B beantwortet die Orakelanfragen des Lernalgorithmus A auf folgende Weise: Wenn A ein positives Muster anfordert, wählt B unter Gleichverteilung ein $x \in Z_N$ mit niederwertigstem Bit gleich eins und gibt das Muster

$$binary(powers(RSA(x, N, e), N), N, e), 1)$$

an A zurück. Sowohl die *RSA* Verschlüsselung als auch die Erzeugung der Potenzen sind in Polynomialzeit durchführbar, also bleibt die Ausführungszeit von B effizient. Zudem erzeugt B dabei die gewünschte Verteilung D^+. Die Anfrage eines negativen Musters wird analog durch Erzeugen eines $x \in Z_N$ mit niederwertigstem Bit gleich null und der Ausgabe

$$binary(powers(RSA(x, N, e), N), N, e), 0)$$

beantwortet. Dabei wird die Verteilung D^- erzeugt.

Nach Ablauf der Simulation gibt A die Hypothese h aus. Alle Orakelanfragen von A wurden entsprechend der *pw2PAC* Definition korrekt beantwortet, daher gilt für ein gleichverteilt aus Z_N gewähltes x, daß $h(binary(powers(r, N), N, e))$ gleich $LSB(x)$ mit einer Wahrscheinlichkeit von mehr als $1/2 + 1/p(l)$ für ein beliebiges aber festes Polynom p ist. Wir können hier mit $p(l)$ statt mit $p(n)$ abschätzen, da l und n polynomiell voneinander abhängen.

Wir können also das niederwertigste Bit mit einem polynomiellen gegenüber Raten verbesserten Anteil vorhersagen. Damit folgt direkt aus Satz 6.2, daß die *RSA* Verschlüsselungsfunktion ohne Kenntnis der Falltürinformation invertierbar ist. Also kann die Annahme, daß NC^1 in *pw2PAC* erlernbar ist nicht wahr sein. □

6.2.3 Nichteffiziente Erlernbarkeit beruhend auf der Quadratischen Reste Annahme

Wir werden in diesem Abschnitt erneut die nichteffiziente Erlernbarkeit der Repräsentationsklasse NC^1 nachweisen. Der Beweis beruht diesmal auf der Quadratischen Reste Annahme (Abschnitt 3.1.2). Diese wird besonders in probabilistischen Public-Key Verfahren gerne verwendet (Abschnitt 3.2.2).

Um mit der Quadratischen Reste Annahme verschlüsseln zu können [GM83], benötigen wir 2 Primzahlen p und q als geheimen Schlüssel. Der öffentliche Schlüssel ist

(N, y), wobei $N = p \cdot q$ und y ein Quadratischer Nichtrest ist ($y \in QNR$). Um ein Bit $b \in \{0, 1\}$ zu verschlüsseln, wird zunächst ein x zufällig aus $Z_n^*(+1)$ [1] (Abschnitt 3.1.1) gezogen. Dann wird mit folgender Formel verschlüsselt:

$$QR(b, N, y) := y^{(1-b)} \cdot x^2 \bmod N, x \in_R Z_n^*(+1)$$

Dabei ist x^2 immer ein Quadratischer Rest und das Produkt ist genau dann aus QR wenn b gleich 1 ist. Es wird also 1 durch einen Quadratischen Rest verschlüsselt und 0 durch einen Quadratischen Nichtrest. Die Sicherheit dieses Systems beruht dann auf der Quadratischen Reste Annahme.

Annahme 6.1 Quadratische Reste Annahme
Sei $N = p \cdot q$, wobei p und q Primzahlen der Länge l sind. Für jedes $x \in Z_N^*(+1)$, definieren wir $QR(x, N) = 1$, wenn x ein quadratischer Rest ist und $QR(x, N) = 0$ sonst. Sei A ein beliebiger probabilistischer Polynomialzeitalgorithmus, der als Eingabe N und x erhält. Dann besagt die Quadratische Reste Annahme, daß für unendlich viele N gilt:

$$Pr[A(x, N) = QR(x, N)] < \frac{1}{2} + \frac{1}{p(l)}$$

Dabei ist $p()$ ein beliebiges aber festes Polynom und die Wahrscheinlichkeit wird entsprechend den x, die gleichverteilt aus $Z_N^*(+1)$ gezogen werden, und dem Zufallsverhalten von A ermittelt.

Bei der Definition fällt auf, daß unendlich viele N existieren können für die die Wahrscheinlichkeitsabschätzung nicht gilt. Wir werden aber bald sehen, daß das für unseren Beweis keine Rolle spielt.

Wir wissen aus dem Kapitel 3.1.1, daß genau die Hälfte der x aus $Z_N^*(+1)$ Quadratische Reste sind. Der obige Satz besagt also, daß kein Polynomialzeitalgorithmus mit *polynomiell verwertbarer Wahrscheinlichkeit* (Abschnitt 4.2.3) Quadratische Reste von Quadratischen Nichtresten unterscheiden kann.

Wir wollen diesmal einen Widerspruch zur Quadratischen Reste Annahme herleiten. Dafür benötigen wir - ähnlich wie im vorherigen Abschnitt - die Repräsentation der Entschlüsselungsfunktion. Damit kann dann der Widerspruchsbeweis für die *pw2PAC* Erlernbarkeit von NC^1 geführt werden.

Wir definieren nun die Repräsentationsklasse C_n auf der Grundmenge $\{0, 1\}^n$.

[1] $Z_n^*(+1)$ beschreibt die Menge aller $x \in Z_N^*$ mit Jakobisymbol J(x,N) gleich 1

Definition 6.7 Repräsentationsklasse $\mathbf{C_n}$
Sei l die größte natürliche Zahl, die die Ungleichung $4l^2 + 2l \leq n$ erfüllt. Jede Repräsentation aus C_n wird durch ein Paar von l-Bit Primzahlen (p, q) beschrieben. Wir nennen diese Repräsentation $r_{(p,q)} \in C_n$.

Aufgrund dieser Definition beschreiben aufeinanderfolgende C_n häufig die gleiche Klasse, denn der Parameter l ändert sich erst, wenn n genügend Spielraum läßt. Nichtsdestoweniger sind jedoch alle möglichen N in $\bigcup_{n\geq 1} C_n$. Daher stört es nicht, daß der Satz 6.1 nur für unendlich viele N gilt.

Definition 6.8 Betrachtete Muster *Wir betrachten eine Repräsentation $r_{(p,q)}$ aus C_n und sei $N = p \cdot q$. Für jedes $x \in Z_N^*(+1)$ ist*

$$< binary(powers(x, N), N), QR(x, N) >$$

ein betrachtetes Muster. Die Länge eines solchen Musters ist maximal $(2l)\cdot(2l)+2l \leq n$, damit ist die Repräsentationsklasse C_n sowohl auf den Mustern als auch auf der Repräsentationsgröße durch n parametrisiert. Wir betrachten für den Widerspruchsbeweis die Gleichverteilung auf den positiven und den negativen Mustern (D^+, D^-).

Wir zeigen jetzt, daß eine Repräsentation der Entschlüsselungsfunktion für jedes $r_{(p,q)} \in C_n$ in NC^1 existiert.

Nachweis der Existenz einer Repräsentation der Entschlüsselungsfunktion in NC^1
Wir zeigen, daß für jede Repräsentation $r_{(p,q)}$ aus C_n ein NC^1-Schaltkreis existiert, der das beschriebene Konzept repräsentiert.

Der NC^1-Schaltkreis erhält als Eingabe die binäre Zeichenkette

$$binary(powers(x, N), N),$$

wobei $N = p \cdot q$ und $x \in Z_N^*(+1)$ ist. Der Schaltkreis ist in 4 Bereiche aufgeteilt:

Phase i: Berechnung der Potenzen modulo p und q.
Die Potenzen modulo n sind nach Definition in das Muster hineinkodiert und haben jeweils die Länge $2l$. Die Potenzen modulo p und q kann man unter Verwendung der vorliegenden Potenzen anhand der Formel

$$\begin{aligned} a \bmod p &= (a \bmod n) \bmod p \\ &= (a \bmod n) - \left\lfloor \frac{(a \bmod n)}{p} \right\rfloor \cdot p \end{aligned}$$

berechnen. Die drei Rechenoperationen Integerdivision, Multiplikation und Subtraktion können durch einen NC^1-Schaltkreis ausgeführt werden [BCH86]. Dabei ist zu beachten, daß diese $4l$ Berechnungen parallel durchgeführt werden, da sonst die Tiefe des Schaltkreises nicht mehr logarithmisch in l wäre.

Phase ii: Berechnung der Legendresymbole modulo p und q.
Wie wir aus der Zahlentheorie wissen, gilt für eine Primzahl p und ein $x \in Z_p^*$ die Formel

$$L(x,p) = x^{\frac{p-1}{2}} \bmod p.$$

Es müssen also nur noch die entsprechenden Potenzen aus Phase II modular multipliziert werden (siehe auch Abschnitt 3.1.1). Dies ist auch durch einen NC^1-Schaltkreis möglich.

Phase iii: Vergleich der Ergebnisse mit -1 und 1.
Aus Phase II haben wir zwei $2l$-Bit lange Ergebnisse. Wir wissen aus der Zahlentheorie, daß die Ergebnisse nur -1 oder 1 sein können. Diese Phase gibt also 2 Bits aus, die genau dann gleich 1 sind, wenn die entsprechende $2l$-Bit Zahl eine Eins repräsentiert. Dies ist ohne weiteres durch einen NC^1-Schaltkreis möglich.

Phase iv: Auswertung der Ergebnisse.
Die Zahl x ist genau dann ein Quadratischer Rest, wenn sie sowohl modulo p als auch modulo q ein Quadratischer Rest ist. Wir geben daher genau dann 1 aus, wenn die Ergebnisse der Phase III beide 1 sind. Dies ist auch durch einen NC^1-Schaltkreis möglich.

Wir können also jede Repräsentation $r_{(p,q)} \in C_n$ durch einen NC^1-Schaltkreis darstellen. Auf dieser Grundlage können wir nun den Beweis der Nichterlernbarkeit in *pw2PAC* für die Repräsentationsklasse NC^1 auf der Quadratischen Reste Annahme führen.

Beweis: NC^1 ist nicht in *pw2PAC* erlernbar
Nehmen wir an, daß A ein polynomialzeitbeschränkter schwacher Lernalgorithmus

(*pw2PAC*) für die Repräsentationsklasse NC^1 ist. Wir haben bereits gezeigt, daß alle Konzepte $r_{(p,q)}$ aus C_n sind auch in NC^1 repräsentierbar sind. Also ist der Algorithmus A in der Lage die Repräsentationen $r_{(p,q)}$ schwach zu erlernen.

Wir werden zeigen, wie man den Lernalgorithmus A einsetzen kann, um Quadratische Reste zu erkennen. Sei N das Produkt zweier geheimer l-Bit Primzahlen p und q. Wir simulieren Algorithmus A, wobei A nur N aber nicht die Faktorisierung bekannt ist. A erhält auch den Zuverlässigkeitswert δ. Die Hypothese wird nach der Simulation überprüft und δ beeinflußt das Ergebnis deshalb nicht.

Wenn A ein positives Beispiel anfordert, erzeugen wir gleichverteilt ein $y \in Z_N^*$ und geben das positive Muster

$$< binary(powers(y^2 \mod N, N), N), 1 >$$

an A zurück. Da y^2 ein Quadratischer Rest ist, ist das Jakobisymbol $J(x, N)$ gleich 1. Weiter besitzt jeder quadratische Rest exakt 4 Wurzeln in Z_N^* und daher erzeugen wir bei der beschriebenen Vorgehensweise die gewünschte Gleichverteilung D^+ auf den Quadratischen Resten modulo N.

Um auch negative Muster erzeugen zu können, benötigen wir einen Quadratischen Nichtrest u modulo n. Wir erzeugen zu Beginn der Simulation wiederholt ein u aus Z_N^*, bis das Jakobisymbol $J(u, N)$ gleich 1 ist. Da genau die Hälfte der Zahlen aus Z_N^* Jakobisymbol 1 besitzen, ist dies in erwarteter Polynomialzeit[2] möglich. Wenn nun ein negatives Beispiel angefordert wird, wählen wir gleichverteilt ein y aus Z_N^* und geben das Muster

$$< binary(powers(u \cdot y^2 \mod N, N), N), 1 >$$

an A zurück. Wenn u ein Quadratischer Nichtrest ist, ist $u \cdot y^2$ für jedes $y \in Z_N^*$ auch ein Quadratischer Nichtrest. Es werden also die Anfragen von A an das negative Orakel (*NEG*) korrekt beantwortet und A gibt eine Hypothese h mit Genauigkeit $1/2 + 1/p(l)$ entsprechend den Verteilungen D^+ und D^- aus. In diesem Fall wäre unser Algorithmus zur Erkennung von Quadratischen Resten erfolgreich.

Wenn nun aber u ein Quadratischer Rest ist, erhält der Lernalgorithmus falsche Muster und die ausgegebene Hypothese h muß nicht korrekt sein. Die Gesamtwahrscheinlichkeit, daß A korrekt arbeitet ist damit mindestens $\frac{1}{2}(1-\delta)$.

Wir sind aber in der Lage die ausgegebene Hypothese h auf ihre Zuverlässigkeit zu überprüfen. Hierzu müssen wir uns die Möglichkeiten der Chernoff Schranken

[2] Der Erwartungswert der Laufzeit ist polynomiell

[Che52] aus Abschnitt 6.2.1 in Erinnerung rufen. Mit ihrer Hilfe sind wir in der Lage eine Hypothese durch eine polynomielle Anzahl an Tests so abzuschätzen, daß sie mit Wahrscheinlichkeit $1 - \delta$ einen Fehler kleiner als $\frac{1}{p(n)}$ aufweist. Dies geschieht folgendermaßen.

Mit einer Wahrscheinlichkeit von $\frac{1}{2}(1 - \delta)$ gibt der Algorithmus eine Hypothese aus, die auf den positiven D^+ und den negativen Beispielen D^- mit einer Wahrscheinlichkeit von über $\frac{1}{2} + \frac{1}{p(l)}$ eine korrekte Antwort liefert. Wir können wie oben beschrieben positive Beispiele erzeugen und die Hypothese h dadurch testen, ob sie einen Anteil von über $\frac{1}{2} + \frac{1}{2 \cdot p(l)}$ richtig bewertet mit einer Irrtumswahrscheinlichkeit von $\frac{\delta}{2}$. Wenn dies der Fall ist wissen wir, daß die positiven Beispiele entsprechend der *pw2PAC* Erlernbarkeit korrekt beantwortet werden.

Wir müssen nur noch die negativen Beispiele testen. Da wir aber negative Beispiele nicht sicher erzeugen können, testen wir statt dessen die Hypothese h für beliebige gleichverteilt gezogene Beispiele. Hierzu wählt man ein x aus $Z_N^*(+1)$ und berechnet $h(binary(powers(x, N), N))$. Wiederum schätzen wir die erhaltene Verteilung mit Chernoff Schranken so ab, daß der Erwartungswert für $h(binary(powers(x, N), N))$ mit Wahrscheinlichkeit $1 - \delta/2$ in den Grenzen $\frac{1}{2} - \frac{1}{4p(l)}$ und $\frac{1}{2} + \frac{1}{4p(l)}$ liegt. Denn genau dann wenn dies der Fall ist muß die Hypothese h auf den negativen Beispielen mit Wahrscheinlichkeit von $1 - \delta/2 - \delta/2 = 1 - \delta$ mindestens einen Anteil von

$$\frac{1}{2} + \frac{1}{2p(l)} - \frac{1}{4p(l)} = \frac{1}{2} + \frac{1}{4p(l)}$$

korrekt beantworten. Die Hypothese h liefert also mit großer Wahrscheinlichkeit auf den negativen und positiven Beispielen einen polynomiell verwertbaren Vorteil.

Wir wiederholen diese Vorgehensweise so lange, bis wir eine solche Hypothese erhalten. Dies ist in Polynomialzeit zu erwarten. Also gilt für ein x, das gleichverteilt aus der Menge Z_N^* gezogen wird:

$$h(binary(powers(x, N), N)) = QR(x, N)$$

mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2} + \frac{1}{4p(l)}$ und dies widerspricht der Quadratischen Reste Annahme. □

Daher kann die Repräsentationsklasse NC^1 nicht erlernbar sein.

6.2.4 Nichteffiziente Erlernbarkeit beruhend auf der Annahme der Nichtfaktorisierbarkeit von Blum-Integern

Wir wissen bereits, daß die Annahme der Sicherheit des *RSA* Public-Key Verfahrens oder die Quadratische Reste Annahme ausreichen, um die polynomialzeitliche schwache Nichterlernbarkeit (*pw2PAC*) von NC^1 nachzuweisen. In diesem Abschnitt werden wir das Ergebnis noch weiter untermauern, indem wir zeigen, daß auch die Annahme, daß Blum-Integer (Abschnitt 3.1.1) nicht effizient faktorisierbar sind, ausreicht um die Nichterlernbarkeit von NC^1 zu zeigen.

Wir werden im Folgenden nicht direkt die Annahme, daß Blum-Integer nicht effizient faktorisierbar sind, benützen. Denn der folgende Satz besagt, daß Blum-Integer faktorisieren probabilistisch polynomialzeitreduzierbar zu Wurzelziehen modulo n ist.

Satz 6.3 [Rab79]
Sei N das Produkt zweier Primzahlen p und q der Länge l und x aus Z_N^*. Dann sind folgende Probleme probabilistisch polynomialzeitreduzierbar zueinander.

(i) Ausgabe eines nichttrivialen Faktors bei Eingabe von N.

(ii) Bei Eingabe von N und $x^2 \bmod N$ gib ein y aus mit der Eigenschaft $x^2 \equiv_N y^2$.

Die Wahrscheinlichkeit wird entsprechend dem Zufallsverhalten der probabilistischen Algorithmen und den gleichverteilt aus Z_N gezogenen x ermittelt.

Die zweite Eigenschaft im obigen Satz entspricht wiederum der Sicherheit des Rabin Public-Key Verfahrens (Abschnitt 3.2.1). Das originale Rabin Verfahren benutzte zwei Primzahlen p und q der Länge l. Für $N = p \cdot q$ und $x \in Z_N^*$ war die Verschlüsselungsfunktion als $R(x, N) = x^2 \bmod N$ definiert. Die Falltürinformation bestand aus den Primzahlen p und q, die es dem rechtmäßigen Empfänger Alice ermöglichten die Quadratwurzel aus dem Kryptotext und damit den Klartext zu berechnen. Das Hauptproblem dieses Verfahrens besteht darin, daß die Quadratwurzeln nicht eindeutig definiert sind. Es muß also eine zusätzliche Information übertragen werden, die es dem rechtmäßigen Empfänger ermöglicht, die zuvor verschlüsselte Nachricht aus den zur Auswahl stehenden Wurzeln zu identifizieren. Aus diesem Grund verwenden wir eine veränderte Rabin Verschlüsselungsfunktion [ACG88].

Definition 6.9 Veränderte Rabin Funktion
Seien die zwei Primzahlen p und q der Länge l kongruent zu 3 modulo 4, das heißt $N = p \cdot q$ ist eine Blum-Integer. Wir definieren eine Untermenge M_n von Z_N^* durch:

$$M_N = \left\{ x : 0 \leq x \leq \frac{N}{2} \, und \, x \in Z_N^*(+1) \right\}$$

Für $x \in M_N$ ist die veränderte Rabin Verschlüsselungsfunktion wie folgt definiert:

$$MR(x, N) = \begin{cases} x^2 \mod N, & wenn \, x^2 \mod N \in M_N, \\ (N - x^2) \mod N, & sonst \end{cases}$$

Jede Quadratzahl modulo N besitzt Jakobisymbol 1 und da $p, q \equiv_4 3$ sind, gilt J(x,N)=J(-x,N). Es wird also eine Permutation (Abschnitt 3.1.1) auf M_N definiert.

Wir überlegen uns nun, daß aus der Invertierbarkeit der veränderten Rabin Verschlüsselungsfunktion auch die Invertierbarkeit der Originalfunktion zumindest für Blum-Integer N folgt. Hierbei kann man voraussetzen, daß man nur aus Quadratischen Resten versucht die Wurzel zu ziehen und diese besitzen immer ein positives Jakobisymbol. Wir sind also im Definitionsbereich M_N der Umkehrfunktion von $MR()$. Wenn $MR(x, N)$ invertierbar ist, kann man auch $R(x, N)$ invertieren, indem man versucht MR für die beiden Werte $R(x, N)$ und $N - R(x, N)$ zu invertieren. Die Ergebnisse lassen sich leicht durch Quadrieren überprüfen und man kann das korrekte Ergebnis ausgeben.

Dieser Beweisschluß bewirkt, daß wir nur einen Widerspruch zur Faktorisierbarkeit bei Blum-Integern und nicht für beliebige Produkte zweier Primzahlen erhalten.

Weiter gilt folgender Satz [ACG88].

Satz 6.4 *Seien x und N wie gehabt. Dann sind folgende Probleme probabilistisch polynomialzeitreduzierbar zueinander:*

(i) Ausgabe von x bei Eingabe von $MR(x, N)$ und N.

(ii) Gib das niederwertigste Bit $LSB(x)$ von x mit einer Wahrscheinlichkeit von über $1/2 + 1/p(l)$ bei Eingabe von $MR(x, N)$ und N aus. Wobei $l = \log(N)$ die Länge von N und $p()$ ein festgelegtes Polynom ist.

Die Wahrscheinlichkeit wird dabei über alle x berechnet, die gleichverteilt aus M_N gezogen werden und über die Münzwürfe des probabilistischen Algorithmus.

Wenn wir die obigen Sätze und Reduktionen kombinieren erhalten wir, daß Blum-Integer faktorisieren probabilistisch polynomialzeitreduzierbar zu Problem (ii) im letzten Satz ist.

Daher benötigen wir - wie auch in den beiden vorangegangenen Abschnitten - nur die Repräsentation für das niederwertigste Bit des Klartextes entsprechend der veränderten Rabinverschlüsselung. Und wir können die schwache polynomielle Nichterlernbarkeit der Repräsentationsklasse NC^1 nachweisen.

Wir definieren nun die Repräsentationsklasse C_n, die die Entschlüsselungsfunktion des veränderten Rabin Verfahrens repräsentieren soll.

Definition 6.10 C_n
Sei l die größte natürliche Zahl, die die Gleichung $4l^2+2l \leq n$ erfüllt. Jede Repräsentation in C_n ist durch ein Paar von l-Bit Primzahlen p und q definiert, wobei beide kongruent zu 3 modulo 4 sind. Damit ist $N = p \cdot q$ eine Blum-Integer und wir nennen diese Repräsentation $r_{(p,q)}$.

Wie in den vorherigen Abschnitten definieren wir nun die betrachteten Muster, die dann die Beispielverteilungen D^+ und D^- erzeugen.

Definition 6.11 Betrachtete Muster für $r_{(p,q)} \in Z_N^*$
Für jedes $x \in M_N$ ist

$$< binary(powers(MR(x,N),N),N), LSB(x) >$$

ein betrachtetes Muster für $r_{(p,q)} \in C_n$. Die Verteilung D^+ für $r_{(p,q)}$ ist gleichverteilt auf den positiven Beispielen, also denjenigen Beispielen mit $LSB(x) = 1$. Die Verteilung D^- ist entsprechend auf den negativen Beispielen definiert.

Wir müssen noch zeigen, daß jede Repräsentation aus C_n auch in NC^1 repräsentierbar ist. Hierfür werden wir verschiedene Sätze aus der Zahlentheorie benützen.

Nachweis der Existenz einer Repräsentation der Entschlüsselungsfunktion in NC^1
Wir zeigen, daß für jede Repräsentation $r_{(p,q)}$ aus C_n ein äquivalenter NC^1-Schaltkreis existiert. Der Schaltkreis erhält als Eingabe

$$binary(powers(r,N),N),$$

wobei $N = p \cdot q$ und $r = MR(x,N)$ für ein beliebiges $x \in M_N$. Wir sind in der Lage unter Zuhilfenahme verschiedener Sätze aus der Zahlentheorie die Wurzel aus $MR(x,N)$ und damit $LSB(r)$ zu berechnen. Dabei ist zu beachten, daß das Ergebnis von $MR(x,N)$ unter Umständen negiert wurde. Wir werden also zuerst in Phase i überprüfen, ob $MR(x,N)$ oder $N - MR(x,N)$ ein Quadratischer Rest ist und dann die Wurzel berechnen (siehe auch Abschnitt 3.1.1).

Phase i: Bestimmung ob $MR(x, N)$ ein quadratischer Rest ist.
Im letzten Abschnitt haben wir gezeigt, daß man mit einem NC^1-Schaltkreis berechnen kann, ob eine Zahl r ein Quadratischer Rest modulo N ist. Weiter gilt nach Theorem 3.1, daß genau eine der beiden Zahlen $MR(x, N)$, $(N - MR(x, N))$ ein Quadratischer Rest ist. Aus dieser Zahl müssen wir nun die Wurzel ziehen.

Wir benützen also den NC^1-Schaltkreis aus dem letzten Abschnitt um zu bestimmen, ob die veränderte Rabin Verschlüsselung das Ergebnis negiert hat oder nicht und rechnen dann mit dem Quadratischen Rest a weiter.

Für jedes z gilt, daß $z^{2i} = (N - z)^{2i}$ für alle $i \geq 1$. Wir können also das Zwischenergebnis des verwendeten NC^1-Schaltkreises durch eine leichte Veränderung (nur die erste Potenz kann falsch sein) in folgende Ergebnisse abwandeln:

$$a, a^2 \mod p = (\pm r)^2 \mod p, \cdots, a^{2^{2l}} \mod p = (\pm r)^{2^{2l}} \mod p$$

und

$$a, a^2 \mod q, \cdots, a^{2^{2^2 l}} \mod q = (\pm r)^{2^{2l}} \mod q$$

Diese Zwischenergebnisse benötigen wir in der nächsten Phase.

Phase ii: Berechnung der Wurzeln modulo p und q
Nach dem Satz 3.1.1 aus der Zahlentheorie gilt für alle Quadratischen Reste a modulo p und $p \equiv_4 3$:

$$\sqrt{a} \mod p = \pm a^{\frac{p+1}{4}} \mod p$$

Diese Modulare Exponentiation (Abschnitt 3.1.1) läßt sich wie schon in den vorausgehenden Abschnitten durch einen NC^1 Schaltkreis verwirklichen. Wir erhalten also jeweils zwei Lösungen modulo p und modulo q.

$$\sqrt{a_{p1,2}} \mod p = \pm a^{\frac{p+1}{4}} \mod p$$

und

$$\sqrt{a_{q1,2}} \mod q = \pm a^{\frac{p+1}{4}} \mod q$$

Phase iii: Verbinden der Resultate durch den Chinesischen Restsatz
Wir verbinden nun unter Verwendung des Chinesischen Restsatzes (Abschnitt 3.1.1 die Lösungen modulo p und q zu vier Lösungen modulo $N = p \cdot q$.

Da p und q gegeben sind und $ggT(p, q) = 1$ gibt es nach Abschnitt 3.1.1 eine Linearkombination $\lambda p + \mu q = 1$. Hierbei ist zu beachten, daß wir die Werte λ und μ nicht berechnen müssen, die Existenz der Werte reicht aus um schließen zu können, daß ein NC^1-Schaltkreis existiert. Wir berechnen dann die 4 Lösungen mit den Formeln:

$$\begin{array}{ll} a_{n1} \equiv_n \lambda p a_{q1} + \mu q a_{p1} & a_{n2} \equiv_n \lambda p a_{q2} + \mu q a_{p1} \\ a_{n3} \equiv_n \lambda p a_{q1} + \mu q a_{p2} & a_{n4} \equiv_n \lambda p a_{q2} + \mu q a_{p2} \end{array}$$

Wir benötigen hierzu nur eine konstante Anzahl von modularen Multiplikations und Additionsschritten. Dies ist durch einen NC^1-Schaltkreis möglich [BCH86].

Phase iv: Überprüfung, welches der Resultate in M_N liegt
Das gesuchte Resultat muß kleiner als $N/2$ sein und Jakobisymbol 1 besitzen.

Das Jakobisymbol von a_n ist genau dann 1, wenn entweder a_p und a_q Quadratische Reste oder beide Quadratische Nichtreste sind. Denn es gilt mit verschiedenen Sätzen aus der Zahlentheorie:

$$\begin{array}{lcl} J(a_n, N) & = & J(a_n, p) \cdot J(a_n, q) \\ & \underbrace{=}_{a_n = \lambda p a_q + \mu q a_p} & J(\lambda p a_q + \mu q a_p, p) \cdot J(\lambda p a_q + \mu q a_p, q) \\ & \underbrace{=}_{J(x,p) = J(x \mod p, p)} & J(\mu q a_p, p) \cdot J(\lambda p a_q, q) \\ & \underbrace{=}_{\mu q \equiv_p 1, \lambda p \equiv_q 1} & J(a_p, p) \cdot J(a_q, q) \end{array}$$

Die Bestimmung ob a_p und a_q Quadratische Reste sind, kann wie im letzten Abschnitt beschrieben durch einen NC^1-Schaltkreis realisiert werden. Es kommen dann also nur noch zwei Ergebnisse a_n in Frage, von denen genau eines kleiner als $N/2$ ist (vergleiche Abschnitt 3.2.1).

Wir ermitteln also das entsprechende Resultat a_{ni}, das die Bedingungen erfüllt und geben das niederwertigste Bit $LSB(a_{ni})$ aus. Die letzten Schritte sind auch durch einen NC^1-Schaltkreis ausführbar.

Wir können nun den Beweis der nichteffizienten Erlernbarkeit der Repräsentationsklasse NC^1 auf der Grundlage der Faktorisierbarkeit von Blum-Integern führen.

Beweis der nichteffizienten Erlernbarkeit der Repräsentationsklasse NC^1
Wir haben bereits nachgewiesen, daß für alle Repräsentationen $r_{(p,q)} \in \bigcup_{n\geq 1} C_n$ eine Repräsentation aus NC^1 existiert. Wenn also die Repräsentationsklasse NC^1 in *pw2PAC* erlernbar ist, gibt es einen Lernalgorithmus A, der die Repräsentationen $r_{(p,q)}$ erlernt.

Dieser Lernalgorithmus existiert insbesondere unabhängig davon, welche Verteilung auf den Beispielen vorliegt. Also auch für die Gleichverteilungen D^+ und D^-, die durch die betrachteten Muster generiert werden. Wir werden nun beschreiben, wie der Lernalgorithmus A durch einen Algorithmus B simuliert werden kann, um Blum-Integer zu faktorisieren. Genauer gesagt reicht es aus mit den vorangegangenen Sätzen nachzuweisen, daß wir das niederwertigste Bit des Klartextes berechnen können.

Unser Invertierungsalgorithmus B startet als erstes den Algorithmus A mit der Eingabe N. Wenn A ein *positives Muster* anfordert, wählen wir ein x aus M_N gleichverteilt, so daß das niederwertigste Bit $LSB(x) = 1$ ist. Dann geben wir das Beispiel

$$< binary(powers(MR(x, N), N), N), 1 >$$

an A zurück. Ein solches Beispiel kann in zu erwartender Polynomialzeit bei Eingabe von N erzeugt werden.

Wenn der Lernalgorithmus A ein *negatives Muster* anfordert, wählen wir entsprechend ein x gleichverteilt aus M_N mit $LSB(x) = 0$ und übergeben das Muster

$$< binary(powers(MR(x, N), N), N), 0 >$$

an A. Dies ist auch in zu erwartender Polynomialzeit möglich. Diese zwei Orakelsimulationen erzeugen genau die gewünschten Musterverteilungen D^+ und D^-.

Wenn A die Hypothese h ausgibt, gibt B seinerseits h aus. Da die Muster entsprechend den Verteilungen D^+ und D^- erzeugt wurden, erfüllt die Ausgabe die Bedingung $h(binary(powers(r, N), N)) = LSB(x)$ mit Wahrscheinlichkeit $1/2 + 1/p(l)$ für ein festes Polynom $p()$. Dabei wird die Wahrscheinlichkeit über den $r = MR(x, N)$ gemessen, wobei die x gleichverteilt aus M_N gezogen werden.

Dies steht im direkten Widerspruch zu Satz 6.4, da der Satz aussagt, daß die Invertierung des niederwertigsten Bits für irgendein Polynom $p()$ die probabilistische Invertierbarkeit der veränderten Rabin-Verschlüsselungsfunktion nach sich ziehen würde.

Da aus der Invertierbarkeit der veränderten Rabin-Verschlüsselungsfunktion auch die Invertierbarkeit der originalen Rabinfunktion folgen würde, ergibt sich mit Satz 3.2 auch die Faktorisierbarkeit von Blum-Integern und damit der Widerspruch. □

6.3 Resultate und Anwendungsmöglichkeiten

NC^1-Schaltkreise besitzen polynomielle Größe und logarithmische Tiefe. Die Lernprobleme der letzten drei Abschnitte waren immer in NC^1 repräsentierbar. Also gibt es ein Polynom $p(n)$, so daß das Erlernen eines NC^1-Schaltkreises mit einer Größe von mindestens $p(n)$ (und logarithmischer Tiefe) mindestens so schwer ist, wie die Probleme:

- *RSA* invertieren
- Quadratische Reste erkennen
- Blum-Integer faktorisieren

Dieses Ergebnis läßt sich auch auf weitere Repräsentationsklassen übertragen. Wir zeigen als nächstes, daß auch Boolesche Formeln nicht effizient schwach erlernbar sein können.

Satz 6.5 *Es existiert ein Polynom $p(n)$, so daß jeder NC^1-Schaltkreis (Abschnitt 2.7) mit fan-in 2, $p(n)$ Ein- und $q(n)$ Ausgängen auch durch eine Boolesche Formel der Größe $r(n)$ repräsentiert werden kann. Dabei sind $p(), q(), r()$ nichtverkleinernde Polynome.*

Beweis: Sei C ein üblicher NC^1-Schaltkreis mit fan-in 2, $p(n)$ Ein- und $q(n)$ Ausgängen. $p(n)$ und $q(n)$ sind nichtverkleinernde Polynome. Die Größe ist dabei zwangsläufig polynomiell.

Da sowohl NC^1-Schaltkreise als auch Boolesche Repräsentationen auf *AND*, *OR* und *NOT* Gattern definiert sind, können wir diese bei der Transformation direkt übernehmen. Es verbleibt zu zeigen, daß die Größe der entstehenden Booleschen Repräsentation polynomiell beschränkt ist.

Wir bauen die Boolesche Repräsentation des NC^1-Schaltkreises rekursiv für jeden der $q(n)$ Ausgänge vom Ausgang aus auf. Es ist klar, daß diese Rekursion nach $O(log(n))$ Schritten abbricht. Die Größe $G(i)$ der Repräsentation auf Rekursionsstufe i läßt sich - da der fan-in 2 ist - durch $2G(i+1)+O(1)$ nach oben abschätzen. Damit erhalten wir eine Abschätzung für $G(1)$. Wir berechnen $G(1)$ indem wir die Summe über den $O(\log(n))$ Tiefenstufen des sich aufspannenden Binärbaumes berechnen.

$$\begin{aligned} G(1) &= \sum_{i=1}^{O(\log(n))} O(1) \cdot 2^i \\ &= O(2^{k \cdot \log(n)}) \\ &= O\left(\left(2^{\log(n)}\right)^k\right) = O\left(n^k\right) \end{aligned}$$

Die Boolesche Repräsentation für alle Ausgänge besitzt also die Größe $O(n^k) \cdot q(n)$ und ist somit polynomiell beschränkt. □

Mit Satz 6.5 sind wir in der Lage auf die Erlernbarkeit der Repräsentationsklasse $BF_n^{p(n)}$ Rückschlüsse zu ziehen. Dabei ist zu beachten, daß es leichter ist eine Repräsentationsklasse in *pw2PAC* zu erlernen als in *pPAC* oder *p2PAC*.

Korollar 6.1 *Sei $BF_n^{p(n)}$ die Repräsentationsklasse der Booleschen Formeln über n Variablen und maximaler Größe $p(n)$. Sei $BF^{p(n)} = \bigcup_{n\geq 1} BF_n^{p(n)}$. Dann gilt für irgendein ein Polynom $p(n)$, daß die Probleme: RSA Verschlüsselung invertieren, Quadratische Reste erkennen und Blum-Integer faktorisieren probabilistisch polynomialzeitreduzierbar zur pwPAC-Erlernbarkeit von $BF^{p(n)}$ sind.*

Weiterhin haben Pit und Warmuth [PW88] nachgewiesen, daß wenn die Klasse *ADFA* *pwPAC* erlernbar ist, dann ist die Klasse *BF* auch *pwPAC* erlernbar. Wenn *ADFA* also *pwPAC* erlernbar wäre, würde sofort der Widerspruch zu Theorem 6.1 folgen, also haben wir folgenden Satz bewiesen:

Korollar 6.2 *Sei $ADFA_n^{p(n)}$ die Klasse der deterministisch endlichen Automaten mit maximaler Größe $p(n)$, die nur Zeichenketten der Länge n akzeptieren. Weiter gilt: $ADFA^{p(n)} = \bigcup_{n\geq 1} ADFA_n^{p(n)}$. Dann gilt für irgendein Polynom $p(n)$, daß die Probleme: RSA Verschlüsselung invertieren, Quadratische Reste erkennen und Blum-Integer faktorisieren probabilistisch polynomialzeitreduzierbar zur pwPAC-Erlernbarkeit von $ADFA^{p(n)}$ sind.*

Dieses Resultat ist insbesondere bemerkenswert, da *DFA's* mit Mitgliedschaftsfragen erlernbar sind. Hier kann also aus der Benutzung von Mitgliedschaftsfragen ein Vorteil gezogen werden. Wir werden diese Zusammenhänge noch näher in Kapitel 7 betrachten.

Die Resultate von [CSV84] [BCH86] und [Rei87] zeigen, daß die verwendeten Repräsentationen nicht nur durch einen NC^1-Schaltkreis sondern auch durch einen dTC[3] (Abschnitt 4.1) berechenbar sind. Der letzte Satz gilt also auch für die Repräsentationsklasse dTC.

Zusammenfassend haben wir also gezeigt, daß BF, $ADFA$ und dTC unter den genannten Voraussetzungen nicht in Polynomialzeit schwach ($pwPAC$) erlernbar sind.

Es ist wichtig darauf hinzuweisen, daß diese Resultate unabhängig von dem verwendeten Hypothesenraum sind (Abschnitt 4.2.1). Es existiert also keine Repräsentationsklasse, mit der eine der Klassen BF, $ADFA$ und dTC schwach effizient erlernbar ($pwPAC$) wäre. Wir wissen aus Abschnitt 4.2.1, daß aus der $pPAC$-Erlernbarkeit einer Klasse direkt die $pwPAC$ Erlernbarkeit folgt. Außerdem folgt aus der Erlernbarkeit einer Klasse C durch sich selber die Erlernbarkeit von C durch eine beliebige größere Klasse H (Abschnitt 4.2.1).

Daher sind die Klassen BF, $ADFA$ und dTC auch nicht durch sich selber in $pPAC$ erlernbar. Aber selbst für diese schon recht eingeschränkte Aussage gibt es keine Nichterlernbarkeitsresultate, die sich nur auf die Annahme $P \neq NP$ (Abschnitt 2.1) stützen. Also gewinnt man den Eindruck, daß solche Resultate schwer auf der schwachen Annahme $P \neq NP$ zu erreichen sind.

Man kann aus diesen Ergebnissen auch durch *Klasseninklusion* die nichteffiziente Erlernbarkeit von anderen Klassen zeigen. Hiervon wird im Kapitel 7 reger Gebrauch gemacht. Zum Beispiel ist die Repräsentationsklasse NC_1 in $L = DSPACE(log(n))$ (Abschnitt 2.2) enthalten, also auch sind alle Klassen, die L enthalten nicht effizient erlernbar. Die meisten dieser Aussagen sind jedoch in den Nichterlernbarkeitsaussagen des Kapitels 7 enthalten. Die Ausnahme bildet wie schon erwähnt die Repräsentationsklasse DFA, die mit Mitgliedschaftsfragen erlernbar wird.

Die in diesem Kapitel gezeigten Resultate können aber auch auf einen ganz anderen Problemtyp angewendet werden - sogenannte *Optimierungsprobleme*. Hierbei ist die Frage nicht mehr, ob eine Konzeptklasse erlernbar ist, sondern wie nah man einer optimalen Lösung in polynomieller Zeit kommen kann. Ein typisches Beispiel hierfür ist Graph-Coloring.

Beim Graph-Coloring Problem ist die Eingabe ein Graph (P, K) mit Punkten P und Kanten K, wobei jede Kante genau zwei Punkte verbindet. Die Frage ist nun, wieviele Farben man mindestens benötigt, um die Punkte so einzufärben, daß keine Kante zwei gleichfarbige Punkte verbindet.

[3]Ein dTC ist ein polynomiell großer Schwellenschaltkreis mit konstanter Tiefe.

Wir betrachten zum Beispiel den Graph, der aus n Punkten, die alle untereinander mit Kanten verbunden sind. Die optimale Lösung für einen solchen Graphen ist n, da keine zwei Punkte die selbe Farbe besitzen dürfen.

Um die Nichterlernbarkeitsresultate dieses Kapitels auf die Nichtapproximierbarkeit von Optimierungsproblemen anwenden zu können, bedienen wir uns Occams Razor (siehe auch [Sch95] [AB92] oder Abschnitt 4.2.4).

Zur Erinnerung: Ein Occam Algorithmus liefert eine konsistente Hypothese und komprimiert die erhaltene Lerninformation. Wenn ein effizienter Occam Algorithmus für eine Repräsentationsklasse existiert, ist sie effizient erlernbar. Dementsprechend gilt auch der Umkehrschluß: Wenn eine Repräsentationsklasse nicht effizient erlernbar ist, dann kann es auch keinen effizienten Occam Algorithmus geben.

Sei n der Abstufungsparameter des Zielkonzepts $t \in C_n$, das erlernt werden soll, und m die Größe der Probe, die der Lernalgorithmus betrachtet. Dann folgt aus der nicht effizienten Erlernbarkeit der Zielkonzeptklasse unter Zuhilfenahme von Occams-Razor, daß es keinen konsistenten Lernalgorithmus gibt, der eine Hypothese mit der Eigenschaft $|h| \leq n^\alpha m^\beta$ ausgibt, für feste $\alpha \geq 1$ und $0 \leq \beta < 1$.

Die minimale Größe einer mit der Probe S konsistenten Hypothese $h = opt_{Con}(S)$ ist nach oben durch ein Polynom über n beschränkt, da auch die Größe der Zielhypothese polynomiell auf n beschränkt ist. Daraus folgt, daß sich in zu erwartender polynomieller Zeit für eine nichterlernbare Klasse keine Hypothese finden läßt, deren Repräsentationsgröße kleiner als $opt_{Con}(S)^\alpha |S|^\beta$ ist. Auf diese Weise erhält man die Aussage, daß die bisher betrachteten Lernprobleme probabilistisch polynomialzeitreduzierbar zur Annäherung des Konsistenzproblems $Con(ADFA)$[4] sind.

Das Problem $Con(DFA, DFA)$ ist eine Verschärfung des $Con(ADFA)$ Problems und daher unter den selben kryptographischen Annahmen nicht erlernbar. Es muß dann nur noch gezeigt werden, daß die Annäherung von $Con(DFA, DFA)$ mindestens so schwer ist, wie eine Verallgemeinerung des Graphcoloring Problems.

[4] Con(C) ist das Problem, für eine gegebene Probe S, die mit einem Konzept aus C konsistent ist, eine minimale Hypothese zu finden, die mit S konsistent ist. Der verwendete Hypothesenraum ist hierbei beliebig. Bei $Con(C, H)$ muß die Hypothese aus H gewählt werden

7

Nichterlernbarkeit trotz Mitgliedschaftsfragen

In der ursprünglichen verteilungsunabhängigen Erlernbarkeitsdefinition werden die über das Orakel angeforderten Muster entsprechend einer unbekannten, aber festen Verteilung ausgewählt. Der Lernalgorithmus arbeitet also passiv die vorgelegten Muster ab, ohne selber Fragen zur Zugehörigkeit von selbstgewählten Beispielen stellen zu können.

Durch die Erweiterung um Mitgliedschaftsfragen kann nun der Lernalgorithmus gezielt die Zugehörigkeit von Beispielen erfragen, die den verbleibenden konsistenten Hypothesenraum am stärksten einschränken. Dies ist aber nur dann möglich, wenn die Struktur der zu erlernenden Konzeptklasse eine solche Vorgehensweise zuläßt, wie zum Beispiel für das Reelle Perceptron (Kapitel 4.2.2). Ein Gegenbeispiel bilden die charakteristischen Funktionen auf den reellen Zahlen (Kapitel 4.2.2), da hier von bereits bekannten Mustern keine Rückschlüsse auf weitere gezogen werden können.

Es ist unmittelbar klar, daß dies in vielen Fällen zu einer Verkürzung der Laufzeit führt, jedoch muß sie dadurch noch nicht polynomiell werden. Es existieren aber in der Tat Konzeptklassen, die mit Mitgliedschaftsfragen effizient erlernbar sind, in der ursprünglichen *PAC*-Erlernbarkeitsdefinition aber noch nicht in Polynomialzeit erlernbar waren. Zum Beispiel hat D. Angluin ([Ang87]) gezeigt, daß *DFA's* in Polynomialzeit mit Mitgliedschaftsfragen erlernbar sind. Wir wissen jedoch aus dem vorausgehenden Kapitel, daß die Erlernbarkeit ohne Mitgliedschaftsfragen von *DFA's* die Widerlegung kryptographischer Grundannahmen nach sich zieht.

Boolesche Schaltkreise sind ein Beispiel für eine Konzeptklasse, bei der sich durch die Verwendung von Mitgliedschaftsfragen kein ausreichender Vorteil ergibt, denn Valiant ([Val84]) konnte die Nichterlernbarkeit von Booleschen Schaltkreisen sogar mit

Mitgliedschaftsfragen auf der Grundlage von Pseudozufalls- und Einwegfunktionen nachweisen.

In diesem Kapitel werden wir von verschiedenen Konzeptklassen zeigen, daß sie auch mit Mitgliedschaftsfragen nicht erlernbar sind. Darunter sind auch die Konzeptklassen Boolesche Formeln *BF* und Schwellenschaltkreise mit konstanter Tiefe *CDTC*, von denen wir im vorigen Kapitel bereits die Nichterlernbarkeit in *pw2PAC* gezeigt haben.

7.1 Einführung

Aus dem vorausgehenden Kapitel wissen wir, daß Boolesche Formeln, *DFA's* und Schwellenschaltkreise mit konstanter Tiefe nicht in *pw2PAC* erlernbar sind. Dies bedeutet, daß sie selbst mit zwei Orakeln und geringen Anforderungen an die Vorhersagequalität nicht erlernbar sind. Damit folgt direkt (Kapitel 4.2.3) die Nichterlernbarkeit in den Erlernbarkeitsklassen *p2PAC* und *pPAC*.

Wir werden in diesem Kapitel auf der Grundlage von kryptographischen Annahmen und der Existenz verschiedener gegen gewählte Kryptotextangriffe sicherer Public-Key Verschlüsselungsverfahren (Abschnitt 3.3.1) zeigen, daß jede Repräsentationsklasse, die mächtig genug ist, die Entschlüsselungsfunktion des Verschlüsselungsverfahrens polynomiell zu repräsentieren, nicht *pPAC* erlernbar mit Mitgliedschaftsfragen sein kann.

Im letzten Kapitel konnten wir die Nichterlernbarkeit von verschiedenen Repräsentationsklassen nachweisen, indem wir den potentiellen Lernalgorithmus durch einen gewählten Klartextangreifer simulieren ließen. Da der gewählte Klartextangreifer Zugriff auf die Verschlüsselungsfunktion besaß, konnte er die vom Lernalgorithmus geforderten Muster erzeugen. Die Erlernbarkeit der Repräsentationsklasse stand damit im Widerspruch zur Sicherheit des Kryptosystems gegen gewählte Klartextangriffe.

Wir werden in diesem Kapitel von Verschlüsselungsverfahren ausgehen, die sogar gegen gewählte Kryptotextangriffe sicher sind (siehe auch Abschnitt 3.3.1). Damit ist es uns möglich auch Mitgliedschaftsfragen durch den gewählten Kryptotextangreifer zu beantworten.

Der Beweis läuft ähnlich wie in Kapitel 6 ab. Wir zeigen zuerst, daß eine Repräsentationsklasse die Entschlüsselungsfunktion des gegen gewählte Kryptotextangriffe sicheren Public-Key Verschlüsselungsverfahrens polynomiell repräsentieren kann. Man nimmt nun an, daß ein Algorithmus A existiert, der diese Repräsentationsklasse mit Mitgliedschaftsfragen polynomiell erlernt. Dann ist A aber in der Lage die Entschlüsselungsfunktion anhand von Beispielen und Mitgliedschaftsfragen zu erlernen und man kann ihn dazu verwenden, einen erfolgreichen gewählten Kryptotextangreifer zu konstruieren. Dabei werden die Orakelanfragen durch den Angreifer beantwortet. Die Beantwortung der Mitgliedschaftsfragen ist möglich, da der Angreifer Zugriff auf die Entschlüsselungsfunktion besitzt. Dies steht jedoch im Widerspruch zur Annahme, daß das Verfahren gegen gewählte Kryptotextangriffe sicher ist.

Das in Abschnitt 3.3.1 vorgestellte und im Beweis verwendete Public-Key Verfahren benutzt die Quadratische Reste Annahme als Verschlüsselungsfunktion. Im Kapitel 6 haben wie gezeigt, daß diese Entschlüsselungsfunktion als Boolesche Formel oder in NC^1 repräsentiert werden kann. Wir benutzen dann eine Reduktion für die Vorhersage mit Mitgliedschaftsfragen, um auch von komplexeren Klassen die Nichterlernbarkeit nachzuweisen.

Damit folgt dann, daß die Repräsentationsklassen Boolesche Formeln, 3μ Boolesche Formeln, Schaltkreise mit konstanter Tiefe, endliche Vereinigungen und Schnitte von deterministischen endlichen Automaten, 2-Weg *DFA's*, *NFA's* und kontextfreie Grammatiken nicht mit Mitgliedschaftsfragen erlernbar sind (siehe auch Abschnitt 4.1).

7.2 Mitgliedschaftsfragen und gewählte Kryptotextangriffe

Wir werden uns zuerst den Zusammenhang zwischen Mitgliedschaftsfragen und gewählten Kryptotextangriffen verdeutlichen.

Wir betrachten ein probabilistisches Public-Key System (Abschnitt 3.2.2), in dem die versendeten Nachrichten entweder gleich 0 oder 1 sind. Bei einem gewählten Kryptotextangriff (Abschnitt 4), auch Mitternachts- oder Kaffeepausenangriff genannt, hat ein Kryptoanalytiker für einen gewissen Zeitraum Zugriff auf die Entschlüsselungsfunktion. Er hat also die Möglichkeit, sich einen selbst gewählten Kryptotext c entschlüsseln zu lassen. Dies entspricht einer Mitgliedschaftsfrage, wenn man als Zielkonzeptklasse des Lernproblems diejenigen Kryptotexte c ansieht, die sich zu 1 entschlüsseln lassen. Zudem hat er die Möglichkeit sich gelöste Probleme generieren zu lassen, da die Verschlüsselungsfunktion öffentlich ist. Dies entspricht einer Anfrage für ein bewertetes Beispiel. Nach dem der Kodebrecher keinen Zugriff mehr auf die Entschlüsselungsfunktion D mehr besitzt, wird ein Bit b gleichverteilt aus der Menge $\{0, 1\}$ gezogen und verschlüsselt. Der Kryptoanalytiker muß nun b bestimmen. Er ist erfolgreich, wenn die Wahrscheinlichkeit für eine korrekte Klassifizierung um einen polynomiell verwertbaren Anteil $\frac{1}{2}$ überschreitet (vergleiche auch *pw2PAC*, Abschnitt 4.2.3).

Daher impliziert die Sicherheit eines Public-Key Systems gegen einen gewählten Kryptotextangriff die Nichterlernbarkeit einer damit verbundenen Konzeptklasse in polynomieller Zeit mit Mitgliedschaftsfragen. Denn wenn die Klasse der zu 1 entschlüsselbaren Kryptotexte erlernbar wäre, dann könnte man Aussagen über die Mitgliedschaft eines Kryptotextes c zu einem Klartext 0 bzw. 1 machen und das Kryptosystem wäre nicht sicher.

Wenn man also eine Repräsentationsklasse auf ihre Erlernbarkeit überprüfen will, muß man durch sie die Entschlüsselungsfunktion repräsentieren können. Wie auch schon im vorangegangenen Kapitel stellt sich die Frage, wie stark man die Komplexität einer Entschlüsselungsfunktion vermindern kann, ohne ihre kryptographische Sicherheit zu gefährden.

Wir benötigen noch die Definition für Vorhersage mit Mitgliedschaftsfragen. Sie entspricht der um ein Mitgliedschaftsorakel ergänzten Erlernbarkeitsdefinition *pPAC*. Die Definition des Lernalgorithmus als probabilistische Turingmaschine enthält implizit den Zuverlässigkeitsparameter δ (vergleiche Abschnitt 2.3).

Definition 7.1 Vorhersage mit Mitgliedschaftsfragen

Der probabilistische Lernalgorithmus erhält als Eingabe eine Schranke s für die Zielkonzeptbeschreibung, eine Schranke n für die Beispielgröße und eine Genauigkeitsschranke ϵ.

Der Lernalgorithmus hat Zugriff auf drei Orakel, die abhängig von der Verteilung auf dem Beispielraum und vom Zielkonzept sind.

- *Mitgliedschaftsorakel:*
  Die Mitgliedschaftsfrage erhält als Eingabe das Beispiel x und gibt als Ausgabe die zugehörige Bewertung b zurück.

- *Beispielorakel:*
  Beim Aufruf des Beispielorakels wird ein entsprechend der Verteilung auf dem Beispielraum gezogenes Muster $< x, b >$ zurückgegeben.

- *Orakel für ein zu bewertendes Element x:*
  Beim Aufruf dieses Orakels wird entsprechend der Verteilung auf dem Beispielraum ein Beispiel x erzeugt und an den Lernalgorithmus zurückgegeben.

Das letzte Orakel wird am Ende der Berechnung aufgerufen worauf der Algorithmus die Bewertung berechnet und ausgibt. Der Lernalgorithmus sagt das Zielkonzept korrekt voraus, wenn er für alle $s, n, \epsilon, u \in X^{[s]}$ und D auf $X^{[n]}$ eine korrekte Aussage für *das Beispiel aus dem letzten Orakelaufruf bezüglich u mit einer Wahrscheinlichkeit größer als $1 - \epsilon$ macht.*

Eine Repräsentationsklasse C ist polynomiell vorhersagbar mit Mitgliedschaftsfragen (vmm) genau dann, wenn es einen vmm-Algorithmus gibt, der polynomielle Laufzeit hat.

Schapire [Sch89] hat gezeigt, daß diese Definition der *pw2PAC* Erlernbarkeit mit Mitgliedschaftsfragen entspricht. Wir werden daher den Widerspruchsbeweis für ein festes $\epsilon = 1/4$ führen. Dieses Ergebnis gilt dann nach Schapire auch für die schwache Erlernbarkeit mit Mitgliedschaftsfragen.

Um die Repräsentation der Entschlüsselungsfunktion zu vereinfachen (vergleiche Abschnitt 6.2.1) verwenden wir ein Probabilistisches Public-Key System, in dem nur einzelne Bits verschlüsselt werden.

Definition 7.2 Bitverschlüsselndes Probabilistisches Public-Key Verfahren
Ein Public-Key System besteht aus drei probabilistischen Turingmaschinen: einem Schlüsselerzeuger G, einem Verschlüsselungsmechanismus E und einem Entschlüsselungsmechanismus D. Jede der Turingmaschinen hält innerhalb zu erwartender polynomieller Zeit entsprechend ihrer Eingabelängen. Die Länge der Ausgaben ist auch polynomiell entsprechend der Eingabelängen beschränkt.

- *Der Schlüsselerzeuger G liest den Sicherheitsparameter n als eine Folge von n Einsern gefolgt von einer Null ein und gibt das Schlüsselpaar (PK, SK) aus. Dabei ist PK der öffentliche und SK der geheime Schlüssel.*
- *Der Verschlüsselungsmechanismus E liest (b, PK) ein, wobei b ein Bit und PK der öffentliche Schlüssel ist und gibt den Kryptotext x aus.*
- *Der Entschlüsselungsmechanismus D liest das 3-Tupel (PK, SK, x) ein und gibt das entsprechende Klartextbit b aus.*

Die verschlüsselten Bits müssen für alle durch G erzeugten Schlüsselpaare korrekt entschlüsselt werden. Es gilt also für alle durch G erzeugten Schlüsselpaare (PK, SK) und ein beliebiges Bit b:

$$D(PK, SK, E(b, PK)) = b$$

Wir formalisieren nun, was wir in diesem Zusammenhang unter einem gewählten Kryptotextangriff verstehen.

Definition 7.3 gewählter Kryptotextangriff
Ein gewählter Kryptotextangreifer ist ein probabilistisches Programm A, das n und PK einliest und in erwarteter Polynomialzeit entsprechend seiner Eingabelänge stoppt. Der Algorithmus hat Zugriff auf zwei Orakel, (1) die Entschlüsselung und (2) Ausgabe eines Musters. Jeder Orakelaufruf benötigt eine Zeiteinheit. Nach dem Aufruf des Orakels (2) dürfen keine weiteren Orakelaufrufe erfolgen.

Das Ergebnis der Orakelaufrufe hängt von n, PK und der Menge c ab, die aus den Zeichenketten besteht, die durch PK zu 1 entschlüsselt werden.

$$c = \{x | D(PK, SK, x) = 1\}$$

Bei Eingabe von n und PK arbeiten die Orakel wie folgt: Das Orakel (1) liest eine Zeichenkette x und gibt 1 aus wenn $x \in c$ ist, und sonst 0. Es entspricht damit der Entschlüsselungsfunktion. Das Orakel (2) liefert ein zu bewertendes Beispiel. Hierfür wird ein Bit b gleichverteilt gezogen und zu $x = E(b, PK)$ verschlüsselt zurückgegeben.

Der Angriffsalgorithmus gibt ein Bit b' aus, von dem er annimmt, daß es die korrekte Entschlüsselung des durch das Orakel (2) erhaltenen Kryptotextes ist. Der gewählte Kryptotextangreifer war erfolgreich, wenn das Bit b' eine korrekte Entschlüsselung des durch Orakel (2) zurückgegebenen x ist.

Die Zufallsvariable $T_A(n)$ ist gleich 1 wenn der Algorithmus für einen durch G erzeugten Schlüssel (PK, SK) und ein Orakel, das die Konzeptbeschreibungsmenge $c = \{x | D(PK, SK, x) = 1\}$ verwendet, eine korrekte Aussage macht. Einen solchen Lernalgorithmus nennen wir einen vmm-Algorithmus.

Definition 7.4 Sicherheit eines Verschlüsselungssystems
Das Verschlüsselungssystem ist sicher, wenn für jedes positive Polynom $p(n)$ gilt, daß für alle Kryptotextangreifer A

$Pr\{T_A(n) = 1\} < \frac{1}{2} + \frac{1}{p(n)}$

ist.

Dabei wird die Wahrscheinlichkeitsverteilung entsprechend der zufälligen Auswahl von b und des Zufallsverhaltens von G, E und A gemessen.

Dementsprechend existiert bei einem unsicheren Kryptosystem ein Angriffsalgorithmus A, dessen Vorhersagewahrscheinlichkeit um mindestens einen polynomiellen Anteil besser ist als direktes Raten.

7.3 Repräsentation von kryptographischen Entschlüsselungsfunktionen

Ähnlich wie in Kapitel 6 benötigen wir eine Repräsentation der Entschlüsselungsfunktion.

Definition 7.5 Polynomielle Repräsentation der Entschlüsselungsfunktion eines Kryptosystems (G, E, D) in einer Repräsentationsklasse C
$c(PK, SK)$ ist die Menge, welche aus denjenigen Zeichenketten x besteht, die zu 1 entschlüsselt werden. Eine Repräsentationsklasse C repräsentiert die Entschlüsselungsfunktion von (G, E, D) polynomiell genau dann, wenn es für alle möglichen Verschlüsselungsfunktionen $G(n)$ eine Konzeptbeschreibung u aus C gibt, deren Länge polynomiell durch den Sicherheitsparameter n beschränkt ist und die $c(PK, SK)$ repräsentiert.

Auf dieser Grundlage sind wir in der Lage das folgende Theorem zu formulieren.

Satz 7.1 *Wenn (G, E, D) ein Public-Key System ist und C polynomiell die Entschlüsselung von (G, E, D) repräsentiert, dann gilt:*
Wenn (G, E, D) sicher gegen einen gewählten Kryptotextangriff ist, dann ist C nicht polynomiell vorhersagbar mit Mitgliedschaftsfragen.

Der Beweis dieses Theorems wird geführt, indem man annimmt, daß C erlernbar ist und dann den polynomiellen Lernalgorithmus für C in einen gewählten Kryptotextangreifer umwandelt.

Die Einschränkung der Länge dieser Konzeptbeschreibungen durch ein Polynom auf dem Sicherheitsparameter n bewirkt, daß auch ein potentieller Lernalgorithmus für die Konzeptklasse in polynomieller Zeit entsprechend n hält und somit auch der daraus generierte gewählte Kryptotextangreifer.

Beweis:
Sei (G, E, D) ein Public-Key Verschlüsselungssystem, das sicher ist gegen gewählte Kryptotextangriffe. Sei $p(n)$ ein Polynom, so daß für jede Ausgabe von $x = E(b, PK)$ für $b \in \{0, 1\}$, $|x| \leq p(n)$ gilt. Dies ist möglich, da die Länge der Ausgaben in unserer bitverschlüsselnden Probabilistischen Public-Key Definition polynomiell durch n beschränkt ist.

Sei C eine Repräsentationsklasse, welche die Entschlüsselungsfunktion nach Definition 7.5 polynomiell repräsentiert. Sei $q(n)$ ein Polynom, das für jedes n und für jede Ausgabe (PK, SK) des Schlüsselerzeugers $G(n)$ die Größe der Konzeptrepräsentation von $c(PK, SK)$ beschränkt. Dies ist wegen der Beschränkung der Länge der Konzeptbeschreibung in der Definition 7.5 möglich.

Wir beginnen einen Widerspruchsbeweis, indem wir annehmen, daß C polynomiell mit Mitgliedschaftsfragen vorhersagbar ist. Wir erzeugen nun aus dem Vorhersagealgorithmus A' einen erfolgreichen gewählten Kryptotextangreifer A wie folgt.

Der gewählte Kryptotextangreifer A erhält die Eingaben n und PK und simuliert den vmm-Algorithmus A$'$, mit den Eingaben $q(n)$ (Beschränkung der Größe der Konzeptbeschreibung), $p(n)$ (Beschränkung der Beispiele) und dem Genauigkeitswert $\frac{1}{4}$. Die Polynome $p(n)$ und $q(n)$ müssen nicht bekannt sein, da bereits die Existenz für unseren Widerspruchsbeweis ausreichend ist. Die Orakelanfragen werden wie folgt beantwortet:

Punkt 1.) Bei einer Mitgliedschaftsfrage wird x entschlüsselt. Dies ist möglich, da der Angreifer Zugriff auf die Entschlüsselungsfunktion besitzt.

Punkt 2.) Wenn ein zufälliges Beispiel gefordert wird, wirft A eine Münze und verschlüsselt das Ergebnis.

Punkt 3.) Wenn A$'$ ein zu bewertendes Beispiel einlesen will, fragt A nach einer Herausforderung und gibt diese an A$'$ zurück.

Die Vorhersage, die A$'$ am Ende der Berechnung ausgibt wird direkt übernommen.

Die Verteilung, die durch das Vorgehen bei Punkt 2 erzeugt wird, weist Zeichenketten, die länger als $p(n)$ sind, die Wahrscheinlichkeit 0 zu. Da der Lernalgorithmus jedoch nach Definition für beliebige Verteilungen eine $\epsilon-$ optimale Vorhersage liefert, ist auch hier das Ergebnis entsprechend $c(PK, SK)$ $\epsilon-$ optimal und damit ist die Wahrscheinlichkeit, daß A das Kryptosystem (G, E, D) bricht, größer als $\frac{3}{4}$. Also ist der gewählte Kryptotextangreifer erfolgreich. $\square$

7.4 Erste Ergebnisse

Aus Abschnitt 3.3.1 wissen wir, daß gegen gewählte Kryptotextangriffe sichere Public-Key Verfahren existieren. Diese Verfahren beruhten jeweils auf einer der drei folgenden Annahmen: Quadratischen Reste Annahme, *RSA* Verschlüsselung invertieren und Blum-Integer faktorisieren. Es ist möglich, durch die im letzten Kapitel vorgestellte Technik (siehe auch [BCH86]) die Repräsentierbarkeit der in diesen Verfahren verwendeten Verschlüsselungsfunktionen in NC^1 nachzuweisen. Da wir aufgrund der Sicherheit der oben genannten Verfahren gegen gewählte Kryptotextangriffe auch Mitgliedschaftsfragen beantworten können, führt die Erlernbarkeit der in NC^1 repräsentierbaren Umkehrfunktionen zum Widerspruch. Unter Zuhilfenahme des Satzes 6.5 und der Resultate von [CSV84], [BCH86] und [Rei87] aus dem letzten Kapitel können wir das folgende Korollar aufstellen:

Korollar 7.1 Folgesatz
Wenn die folgenden drei Annahmen: Quadratische Reste Annahme (Abschnitt 6.1), Nicht effiziente Invertierbarkeit der RSA Verschlüsselungsfunktion (Abschnitt 3.2) und Nicht effiziente Faktorisierbarkeit von Blum Integern (Abschnitt 3.1) korrekt sind, dann sind die folgenden Konzeptrepräsentationsklassen nicht polynomiell mit Mitgliedschaftsfragen vorhersagbar:

C_{BF}, C_{dTC}.

7.5 Weitere Ergebnisse durch Reduktion

Wir können wie in Abschnitt 4.3 erwähnt durch Reduktion weitere Nichterlernbarkeitsresultate herleiten. Dafür führen wir folgende Reduktion für die Vorhersage mit Mitgliedschaftsfragen ein:

Definition 7.6 Reduktion für die Vorhersage mit Mitgliedschaftsfragen $C \leq_{vmm} C'$

Wir definieren eine Reduktion von einer Repräsentationsklasse C auf C'. Hierzu benötigen wir drei Funktionen f, g, h.

- *Konzeptfunktion: $g(s, n, u)$*
  Die Funktion $g(s, n, u)$ ist für alle $s, n, u \in X^{[s]}$ definiert. Sie liefert für alle Konzeptbeschreibungen $u \in C$ in Abhängigkeit von der Größe s der Konzeptbeschreibung und der Größe der Beispiele n eine Konzeptbeschreibung für C', deren Länge durch ein nichtverkleinerndes Polynom q auf s, n beschränkt wird.

- *Beispielfunktion: $f(s, n, x)$*
  Die Funktion f ist für alle $s, n, x \in X^{[n]}$ definiert. Sie ist eine in Polynomialzeit auf s, n berechenbare Funktion über s, n und x, wobei x ein Beispiel der Länge n ist. Das Ergebnis ist ein Beispiel x' für das Konzept das durch $g(s, n, u)$ repräsentiert wird, wobei gilt, daß $x \in \kappa_C(u) \Leftrightarrow x' \in \kappa_{C'}(g(s, n, u)) \,\forall\, u \in X^{[s]}$.

Das Beispiel x wird also von u genauso bewertet wie $f(x)$ durch $g(s,n,u)$.

- *Mitgliedschaftsfunktion:* $h(s,n,x')$
  Die Mitgliedschaftsfunktion $h(s,n,x')$ ist für alle $x,n,u \in X'[s], x' \in X$ definiert. Sie erhält als Eingabe die Variablen s,n und x' und liefert $\top, \bot$ oder eine Zeichenkette x. Wobei gilt:
  $h(s,n,x') = \top \Rightarrow x' \in \kappa_{C'}(g(s,n,u))$
  $h(s,n,x') = \bot \Rightarrow x' \notin \kappa_{C'}(g(s,n,u))$
  $h(s,n,x') = x \backslash \{\top, \bot\} : x' \in \kappa_{C'}(g(s,n,u)) \Leftrightarrow x \in \kappa_C(u)$

  Das Beispiel x' wird also entsprechend dem Konzept $g(s,n,u)$ auf die gleiche Weise bewertet wie $h(s,n,x')$ durch u.

Man konstruiert einen Algorithmus zum Erlernen der Konzeptklasse C, indem man den Lernalgorithmus A' für C' auf folgende Weise verändert. Wenn nun A ein Konzept aus C erlernen will, das durch u repräsentiert wird, dann existiert ein $g(s,n,u)$, das durch A' erlernbar wäre. Hierbei ist zu beachten, daß $|g(s,n,u)|$ nur polynomiell größer als u sein darf, da sonst A' nicht mehr effizient auf $|u|$ laufen würde. Damit nun A das vorherzusagende Beispiel richtig bewertet, wird der Lernalgorithmus A' so abgewandelt, daß er die Beispiele aus dem Orakel für C entgegennimmt und so umwandelt, daß A' korrekt auf C' arbeiten kann.

Dies geschieht im Einzelnen folgendermaßen:

Immer wenn A' ein Muster anfordert, liest A seinerseits ein Beispiel $\langle x, b\rangle$ und übergibt dann an A' das Beispiel $\langle f(s,n,x), b\rangle$. Aufgrund der Definition von f ist die Bewertung dieses Beispiels auch für das Zielkonzept aus C' korrekt.

Auf die Anfrage nach einem *vorherzusagendem Beispiel* wird ebenso verfahren.

Die Bewertungsanforderung bei einer *Mitgliedschaftsfrage* muß korrekt erfolgen, um das korrekte Ablaufen des Lernalgorithmus A' zu gewährleisten. Wenn das zu bewertende Beispiel x' im Bild von $x' = f(s,n,x)$ enthalten ist, kann man x durch das Orakel von C bewerten lassen. Wenn dies jedoch nicht der Fall ist, muß man zumindest ein Element an das Orakel von C übergeben, daß Bewertungsäquivalent auf u zu x' auf C' ist. Diese Bedingung wird durch die Definition von h erfüllt.

Eine Konzeptrepräsentation ist $\leq_{vmm}$-vollständig in einer Komplexitätsklasse S, wenn und nur wenn $C \in S$ und wenn für jede Konzeptrepräsentation $C' \in S$ gilt, daß $C' \leq_{vmm} C$.

7.6 Beweis der polynomiellen Nichterlernbarkeit von $C_{3\mu}$ durch Reduktion

Wir werden nun stellvertretend für einige weitere Resultate durch Reduktion die Nichterlernbarkeit von $C_{3\mu}$ aus der Nichterlernbarkeit von C_{BF} ableiten. Die Repräsentationsklasse $C_{3\mu}$ besteht aus den Booleschen Formeln, in denen jede Variable höchstens dreimal vorkommen darf.

Beweis für $C_{BF} \leq_{vmm} C_{3\mu}$

Die für die Reduktion nötigen Funktionen werden am Beispiel $n = 4, s = 6$ veranschaulicht. Die *Beispielfunktion* f erhält als Eingabe eine Zeichenkette der Länge n und tauscht jedes Bit der Eingabe durch s Kopien des Bits. Die Idee hierbei ist, daß jede Variable maximal s-mal vorkommen kann. In der 3μ Formel werden dann für jede Variable der Booleschen Formel s einzelne Variablen eingeführt. Eine Erweiterung der Formel sorgt dafür, daß diese Variablengruppen immer die gleichen Werte besitzen müssen.

$$f(s, n, x) = f(6, 4, 1101) = 111111111111000000111111$$

Die *Konzeptfunktion* g erhält als Eingabe eine Boolesche Formel ϕ die auf n Variablen definiert ist und die maximale Größe s besitzt. Als Ausgabe liefert g eine 3μ Formel auf $n \cdot s$ Variablen, indem zuerst die mehrfachen Variablenvorkommen einer Variable X_i in der Eingabe durch disjunkte aufeinanderfolgende Variablen aus dem Bereich $X_{(i-1)s+1}, \ldots, X_{is}$ ersetzt werden. Man erhält dadurch eine read once Formel ϕ'. Dann wird die Formel ϕ' noch mit einer Formel verbunden, die sicherstellt, daß die zusammengehörenden Variablengruppen die selben Werte besitzen. Zum Beispiel wenn

$$\phi = (\neg X_1 \vee X_3 \vee X_4) \wedge (X_1 \vee X_2 \vee \neg X_3)$$

ist, erhalten wir

$$\phi' = (\neg X_1 \vee X_{13} \vee X_{19}) \wedge (X_2 \vee X_7 \vee \neg X_{14})$$

Damit die Variablen der ersten Gruppe die selben Werte erhalten, benötigen wir:

$$\Delta_1 = (X_1 \to X_2) \wedge (X_2 \to X_3) \wedge (X_3 \to X_4) \wedge (X_4 \to X_5) \wedge (X_5 \to X_6) \wedge (X_6 \to X_1)$$

Ähnliche Formeln benötigt man auch für die 4 anderen Variablengruppen. Die $2\mu-$ Formel

$$\psi = \Delta_1 \wedge \Delta_2 \wedge \Delta_3 \wedge \Delta_4$$

sorgt also dafür, daß alle Variablen aus einer Gruppe gleiche Werte besitzen. Die Konzeptfunktion g gibt schließlich die 3μ Formel $\phi' \wedge \psi$ (über $4 \cdot 6$ Variablen) aus.

Die *Mitgliedschaftsfunktion* h gibt für eine Zeichenkette y der Länge $n \cdot s$ und $y = f(s, n, x)$, x aus, wenn x existiert, andernfalls $\perp$. Es gilt also beispielsweise

$$h(6, 4, 111111000000111111000000) = 1010$$

jedoch

$$h(6, 4, 111101000000111111000000) = \perp$$

Wie man sich leicht überlegen kann, erfüllen die Formeln die folgenden *Reduktionsbedingungen*:

- $|g(s, n, u)|$ ist durch ein Polynom auf s und n beschränkt
- $x \in \kappa_C(u)$ genau dann wenn $f(s, n, x) \in \kappa_{C'(g(s,n,u))}$
- $f(s, n, x)$ ist in Polynomialzeit in $s, n, |x|$ berechenbar.
- h ist in Polynomialzeit berechenbar.
- Wenn $h(s, n, x') = \perp$ dann gilt $x' \notin \kappa_{C'}(g(s, n, u)) \,\forall\, u \in X^{[s]}$
- Der Fall $h(s, n, x') = \top$ kommt hier nicht vor.
- $x' \in \kappa_{C'}(g(s, n, u)) \Leftrightarrow h(s, n, x') \in \kappa_C(u)$, falls $n(s, n, x') \neq \perp$

□

Durch diese Reduktion und noch einige weitere (siehe [AK95]) kann man folgenden Satz beweisen.

Korollar 7.2 Folgerung:
Wenn die folgenden drei Annahmen Quadratische Reste Annahme (Abschnitt 6.1, Nicht effiziente Invertierbarkeit der RSA Verschlüsselungsfunktion (Abschnitt 3.2 und Nicht effiziente Faktorisierbarkeit von Blum Integern (Abschnitt 3.1) gelten, dann sind die folgenden Konzeptrepräsentationsklassen nicht polynomiell mit Mitgliedschaftsfragen vorhersagbar:

C_{BF}, C_{dTC} *für hinreichend große* d, $C_{3\mu}, C_{\cup DFA}, C_{\cap DFA}, C_{2DFA}, C_{NFA}, C_{CFG}$.

8

Mitgliedschaftsfragen helfen nicht bei CNF und DNF

In diesem Kapitel werden wir dem Beweis von D. Angluin und M. Kharitonov folgend [AK95] nachweisen, daß man für eine Polynomialzeitvorhersage bei *CNF* und *DNF* Formeln entweder keine Mitgliedschaftsfragen benötigt oder daß sie auch mit Mitgliedschaftsfragen nicht in Polynomialzeit vorhersagbar sind. Also nützen Mitgliedschaftsfragen nichts beim Erlernen von *CNF* und *DNF* Formeln. Voraussetzung für dieses Resultat ist die Existenz von Einwegfunktionen, die nicht durch polynomiell beschränkte Schaltkreise invertiert werden können.

8.1 Einführung

Wir werden im nächsten Abschnitt zuerst ein Unterschriftsschema definieren. Ein solches Unterschriftsschema muß gegen einen polynomialzeitbeschränkten Fälscher bestehen, der vorübergehend selbst Unterschriften erzeugen lassen kann und schließlich eine neue Nachricht x wählt, die er mit einer Unterschrift y versieht. Wir werden uns auf der Grundlage von verschiedenen Veröffentlichungen überlegen, unter welchen Voraussetzungen gegen solche Fälscher sichere Unterschriftsschemata existieren.

Wir verwenden in diesem Kapitel die Erlernbarkeitsdefinition mit Mitgliedschaftsfragen aus dem letztem Kapitel (Definition 7.1), wobei zusätzlich die Zielkonzeptgröße polynomiell beschränkt sein muß. Als Erlernbarkeitsdefinition ohne Mitgliedschaftsfragen werden wir ebenfalls Definition 7.1 verwenden, jedoch ohne das Mitgliedschaftsorakel. Auch hier muß die Zielkonzeptgröße polynomiell beschränkt sein. Der Unterschied zur üblichen Definition 4.10 besteht darin, daß dem Lernalgorithmus obere Schranken für die Zielkonzeptbeschreibung und für die Beispiellänge gegeben werden, daß die Zielkonzeptgröße polynomiell beschränkt ist und daß ein vorgelegtes Beispiel bewertet wird, statt eine Hypothese auszugeben.

Um den Hauptsatz dieses Kapitels beweisen zu können, benötigen wir eine Hilfskonstruktion. Hierfür betrachten wir ein Unterschriftsschema (G, SP, V) (Abschnitt 3.5) und eine Repräsentationsklasse C. Jede Repräsentation u aus C beschreibt eine Menge von Nachrichten $\kappa_C(u)$. Wir benötigen nun eine andere Repräsentationsklasse C', die für jedes u aus C eine Repräsentation u' enthält, so daß ein Nachrichten-Unterschriftspaar x, y (in kodierter Form) in $\kappa_{C'}(u')$ enthalten ist, falls x aus $\kappa_C(u)$ und y eine gültige Unterschrift für x ist. Die hierbei verwendete Kodierung des Nachrichten-Unterschriftspaars x, y muß in Polynomialzeit invertierbar sein. Wir sagen dann, daß C' polynomiell Unterschriften für C entsprechend einem Unterschriftsschema (G, SP, V) repräsentiert.

Weiter sagen wir, daß eine Repräsentationsklasse C polynomiell Unterschriften darstellen kann, wenn für jedes Unterschriftsschema (G, SP, V) die Repräsentationsklasse C polynomiell Unterschriften für C repräsentiert.

Angluin und Kharitonov [AK95] haben nachgewiesen, daß die Repräsentationsklassen C_{3CNF}, C_{CNF}, $C_{dTC} \forall d \geq 2$, C_{BF}, $C_{\cap DFA}$, C_{2DFA} polynomiell Unterschriften repräsentieren können. Es läßt sich auch nachweisen, daß Unterschriftsschemata existieren, die selbst dem zu Anfang dieses Abschnitts beschriebenen Fälscher widerstehen. Auf dieser Grundlage läßt sich nun der Hauptsatz skizzieren.

Wir nehmen nun an, daß eine Repräsentationsklasse C', die Unterschriften für eine Repräsentationsklasse C darstellt, mit Mitgliedschaftsfragen erlernbar wäre. Es existiert dann ein Lernalgorithmus A$'$, der Mitgliedschaftsfragen verwendet und beliebige Zielkonzepte aus C' erlernt. Wir konstruieren einen Lernalgorithmus für die Repräsentationsklasse C, der keine Mitgliedschaftsfragen stellt. Eine Beispielorakelanfrage der Lernalgorithmus A$'$ kann ohne weiteres beantwortet werden, indem zuerst durch das Beispielorakel von A ein Muster (x, b) angefordert wird. Wir speichern dieses Muster (x, b) zur weiteren Verwendung bei einem Mitgliedschaftsorakelaufruf. Das Muster wird durch den Unterschriftenerzeuger SP mit y unterschrieben. Das Nachrichten-Unterschriftspaar x, y wird kodiert und das so erzeugte Muster wird an A$'$ zurückgegeben.

Eine Mitgliedschaftsfrage von A$'$ muß jedoch durch A ohne Verwendung eines Mitgliedschaftsfsorakels beantwortet werden. Wir beantworten einen solchen Orakelaufruf von A$'$, indem zuerst überprüft wird, ob sich das Beispiel in ein Nachrichten-Unterschriftspaar x, y zerlegen läßt und dann ob ein Paar (x, b) bereits in einem Beispielorakelaufruf als Muster zurückgegeben wurde. In diesem Fall wird die gespeicherte Bewertung zurückgegeben und andernfalls wird das Beispiel negativ bewertet.

Wenn sich das Nachrichten-Unterschriftspaar nicht zerlegen läßt, ist es auf jeden Fall ein negatives Beispiel und somit ist die Antwort korrekt. Wenn das Paar zerlegbar und bereits aufgetreten ist, wird die Orakelanfrage ebenfalls korrekt beantwortet. Somit existiert bei dieser Form der Beantwortung der Mitgliedschaftsorakelanfragen des simulierten Lernalgorithmus A′ genau ein Fall, der zur falschen Antwort führen kann. Dieser Fall tritt auf, wenn das Beispiel eine korrekt unterschriebene Nachricht x, y darstellt und x in $\kappa_C(u)$ liegt, jedoch das Paar x, b noch nicht durch das Beispielorakel zurückgegeben wurde. Dies würde aber bedeuten, daß wir eine Fälschung gefunden haben und wir könnten den Algorithmus zu einem erfolgreichen Fälscher umkonstruieren. Dies steht dann im Widerspruch zur Sicherheit des zugrundeliegenden Unterschriftsschemas. Man kann also aus einem Lernalgorithmus mit Mitgliedschaftsfragen für eine Repräsentationsklasse C' einen Lernalgorithmus ohne Mitgliedschaftsfragen für eine Repräsentationsklasse C erzeugen, wenn C' polynomiell Unterschriften für C bezüglich einem Unterschriftsschema (G, SP, V) repräsentiert. Daraus folgt, daß man zum Erlernen einer Konzeptklasse C, die polynomiell Unterschriften darstellen kann, keine Mitgliedschaftsfragen benötigt.

8.2 Beschränkte *pPAC* Erlernbarkeit

Wie schon erwähnt verwenden wir in diesem Kapitel eine eingeschränkte Erlernbarkeitsdefinition. Der Lernalgorithmus erhält dabei (siehe auch Kapitel 7) obere Schranken für die Beispielgröße n und die Größe s der Zielkonzeptbeschreibung. Zusätzlich muß die Größe der Zielkonzeptbeschreibung durch ein Polynom auf der oberen Schranke der Beispielgröße beschränkt sein. Es folgt die Definition:

Definition 8.1 beschränkt polynomiell vorhersagbar mit Mitgliedschaftsfragen
Eine Repräsentationsklasse C ist beschränkt polynomiell vorhersagbar mit Mitgliedschaftsfragen genau dann, wenn ein Polynomialzeit-vmm-Algorithmus A (Definition 7.1) existiert, so daß für alle Polynome $p(n)$, für alle bis auf endlich viele positive Integer n, für alle Zeichenketten u mit maximaler Länge $p(n)$ und für alle Verteilungen D gilt, daß der Lernalgorithmus A gestartet auf den Eingaben $p(n)$ und n nur mit einer Wahrscheinlichkeit von maximal $1/4$ einen Vorhersagefehler macht. Dabei sind die Orakel durch das Konzept $\kappa_C(u)$ und der Verteilung D bestimmt.

Es reicht aus, daß der Algorithmus mit einer festgelegten Fehlerwahrscheinlichkeit von $\epsilon = 1/4$ arbeitet, da Schapire [Sch89] nachgewiesen hat, daß aus der schwachen Erlernbarkeit einer Repräsentationsklasse in *pwPAC* auch die Erlernbarkeit in *pPAC* folgt. Dies gilt sogar dann, wenn Mitgliedschaftsfragen zugelassen sind.

Für die Definition „*beschränkt polynomiell vorhersagbar*“ verwenden wir die obige Definition, jedoch darf der Lernalgorithmus keine Mitgliedschaftsfragen stellen.

8.3 Definitionen für Unterschriftsschemata

Wir beschreiben nun, was wir im folgenden unter einem Unterschriftsschema verstehen werden.

Definition 8.2 Unterschriftsschema
Ein Unterschriftsschema besteht aus drei Turingmaschinen. Einem Schlüsselerzeuger G, einem Unterschriftsprogramm SP und einem Unterschriftsbestätiger V.

1.) *Der Schlüsselerzeuger G liest n in Einheitsdarstellung ein und gibt das Public-Key Schlüsselpaar (PK, SK) aus. Dabei ist PK der öffentliche und SK der geheime Schlüssel.*

2.) *Das Unterschriftsprogramm SP liest (PK, SK) und eine Nachricht x ein und gibt die Unterschrift y aus.*

3.) *Der Unterschriftsbestätiger V liest den öffentlichen Schlüssel PK, die Nachricht x und die Unterschrift y ein und gibt ein Bit aus, das angibt, ob er die Unterschrift akzeptiert.*

Die Turingmaschinen G und SP sind probabilistische Turingmaschinen. Die Turingmaschine V ist deterministisch. Die probabilistischen Turingmaschinen G und SP halten in zu erwartender polynomieller Zeit entsprechend ihrer Eingabelängen. Die Laufzeit von V ist polynomiell auf n beschränkt. Der Sicherheitsparameter n beschränkt die Länge der verwendeten Nachrichten. Der Überprüfer V muß eine durch SP - entsprechend einem durch G erzeugten Schlüsselpaar (PK, SK) - unterschriebene Nachricht x, y akzeptieren.

Ein Fälscher hat die Möglichkeit sich Unterschriften zu beliebigen Nachrichten generieren zu lassen und muß schließlich eine gültige Unterschrift für eine weitere Nachricht erzeugen, die er selbst wählen darf.

Definition 8.3 Nichtuniformer Fälscher
Ein Nichtuniformer Fälscher ist eine probabilistische Turing Maschine M_F mit vordefinierten Informationsbändern $t_n, n \geq 1$. Sie liest n und PK ein und hat Zugriff

auf das Informationsband t_n. M_F *läuft in Polynomialzeit entsprechend* n *und* $|PK|$. M_F *hat Zugriff auf ein Orakel, das gültige Unterschriften generieren kann und* M_F *gibt schließlich einen Fälschungsversuch* x', y' *aus. Das Paar* x', y' *darf in keinem vorausgegangenen Orakelaufruf ermittelt worden sein.*

Ein Unterschriftsschema ist sicher, wenn kein Fälscher existiert, der mit einer nicht vernachlässigbaren Wahrscheinlichkeit ($> 1/p(n)$) eine erfolgreiche Fälschung generiert.

Definition 8.4 sicheres Unterschriftsschema

Wir definieren zuerst eine Zufallsvariable $T_F(n)$ *über* n. *Wir starten* G *mit Eingabe* n *und erhalten* (PK, SK). *Dann führen wir* M_F *bei Eingabe* n, PK *mit dem Informationsband* t_n *aus. Wenn* M_F *einen Orakelaufruf für eine Nachricht* x *tätigt, berechnen wir* $y = SP(PK, SK), x)$ *und geben* y *an* M_F *zurück. Der Wert der Zufallsvariablen ist gleich* 1 *genau dann, wenn der Überprüfer* V *den Fälschungsversuch, der von* M_F *ausgegeben wird, akzeptiert* ($V(PK, x', y') = 1$). *Das Unterschriftsschema* (G, SP, V) *ist sicher, wenn für jeden Nichtuniformen Fälscher und für jedes Polynom* $p(n)$ *für ausreichend große* n *gilt:*

$$Pr\{T_F(n) = 1\} < \frac{1}{p(n)}$$

Hierbei wird die Wahrscheinlichkeit entsprechend dem Zufallsverhalten von G,SP und M_F *berechnet.*

Von Goldwasser, Micali und Rivest [GMR88] stammt das erste sichere Unterschriftsschema, unter der Annahme, daß das Faktorisierungsproblem nicht effizient lösbar ist. Durch Bellare und Micali [BM92] wurde diese Annahme auf die Existenz von Falltürpermutationen reduziert. Naor und Yung [NY89] schwächten die Annahme weiter auf Einwegpermutationen ab und schließlich wurde sie durch Rompel [Rom90] bis auf die Existenz von Einwegfunktionen vermindert. Daher erhalten wir den folgenden Satz:

Korollar 8.1 Folgesatz

Wenn es Einwegfunktionen gibt, die nicht durch polynomiell große Schaltkreise invertiert werden können, dann gibt es sichere Unterschriftsschemata.

Wir müssen für den Hauptsatz dieses Kapitels definieren, was wir darunter verstehen, daß eine Repräsentationsklasse C' unterschriebene Versionen von Konzepten aus einer

Klasse C entsprechend einem Unterschriftsschema (G, SP, V) repräsentiert. Intuitiv bedeutet nachfolgende Definition, daß für jede Repräsentationsbeschreibung u aus C und jedes Schlüsselpaar (PK, SK) eine Repräsentationsbeschreibung u' aus C' existiert, die aus genau den unterschriebenen Versionen der Zeichenketten x aus u besteht.

Definition 8.5 Polynomielle Repräsentation von Unterschriften
C' repräsentiert unterschriebene Konzepte aus C bezüglich einem Unterschriftsschema (G, SP, V) wenn es zwei Funktionen g und f und zwei Polynome $q_1(s,n)$ und $q_2(s,n)$ mit den folgenden Eigenschaften gibt:

1.) *$g(PK, s, n, u)$:*
Die Funktion $g(PK, s, n, u)$ ist für alle PK, s, n und $u \in X^{[s]}$ definiert. Sie erhält als Eingabe den öffentlichen Schlüssel PK, die obere Schranke s der Konzeptbeschreibungslänge, die obere Schranke der Nachrichtenlänge n und die Konzeptbeschreibung u. Als Ausgabe liefert die Funktion g eine in der Länge durch das Polynom $q_1(s,n)$ beschränkte Konzeptbeschreibung u' aus C'.

2.) *$f(PK, s, n, x, y)$:*
Die Funktion f ist für alle $PK, s, n, x \in X^{[n]}, y \in X$ definiert. Sie verschmilzt die Nachricht x mit der Unterschrift y zu einem Wort w. Als Eingabe erhält sie hierfür den öffentlichen Schlüssel PK, die obere Schranke s des zugrundeliegenden Konzepts u aus C, die Beschränkung n der Beispiellänge, die Nachricht x und die Unterschrift y. Die Ausgabelänge von w ist durch das Polynom $q_2(s,n)$ beschränkt.

3.) *Invertierbarkeit von f:*
Wenn PK, s und n festgelegt sind ist die Funktion f in Polynomialzeit invertierbar. Das bedeutet, es existiert ein Polynomialzeitalgorithmus, der bei Eingabe von PK, s, n, w die zugehörige Nachricht x und die Unterschrift y errechnet, so daß gilt:

$$f(PK, s, n, x, y) = w \textit{ falls die Lösung existiert}$$

4.) *Repräsentation von unterschriebenen Nachrichten in C':*
Die Zeichenkette w ist genau dann in $\kappa_C(g(PK, s, n, u))$, für ein $u \in X^{[s]}$, wenn w zu einer Nachricht $x \in X^{[n]}$ und einer Unterschrift y aufgespalten werden kann, $x \in \kappa_C(u)$ ist und y eine gültige Unterschrift für x ist ($V(PK, x, y) = 1$).

Da wir aus der Erlernbarkeit einer Repräsentationsklasse C mit Mitgliedschaftsfragen Rückschlüsse auf die Erlernbarkeit ohne Mitgliedschaftsfragen der selben Klasse ziehen wollen, fassen wir die letzte Definition weiter zusammen:

Definition 8.6 C **repräsentiert polynomiell Unterschriften**
Wir sagen, daß C polynomiell Unterschriften repräsentiert genau dann, wenn für jedes Unterschriftsschema (G, SP, V) C polynomiell Unterschriften für C unter Berücksichtigung des Unterschriftsschemas (G, SP, V) repräsentiert.

Angluin und Kharitonov [AK95] haben von einigen Repräsentationsklassen nachgewiesen, daß sie polynomiell Unterschriften repräsentieren können. Wir erhalten daher folgendes Lemma:

Lemma 8.1 Existenz von Unterschriftsschemata
Die folgenden Konzeptrepräsentationen repräsentieren polynomiell Unterschriften:
$C_{3CNF}, C_{CNF}, C_{dTC} \forall d \geq 2, C_{BF}, C_{\cap DFA}, C_{2DFA}$

8.4 Ergebnis

Satz 8.1 Hauptsatz
Seien C und C' in Polynomialzeit auswertbare Konzeptrepräsentationen. Wenn (G, SP, V) ein sicheres Public-Key Unterschriftsschema ist, C' polynomiell Unterschriften für C entsprechend (G, SP, V) repräsentiert und C' beschränkt polynomiell vorhersagbar mit Mitgliedschaftsfragen ist, dann ist C beschränkt polynomiell vorhersagbar (ohne Mitgliedschaftsfragen).

Beweis:

Die Repräsentationsklasse C' repräsentiert polynomiell Unterschriften für die Repräsentationsklasse C entsprechend (G, SP, V). Daher existieren die Funktionen f und g und ein Polynom q_1, das die obere Schranke für die Konzeptbeschreibung in C' liefert und ein Polynom q_2, das eine obere Schranke für die Beispiellänge der Beispiele aus $\kappa_C(u')$ für ein u' aus C' liefert (siehe auch Definition 7.1). Sei A$'$ der vmm-Algorithmus für C'.

Wir konstruieren nun aus dem Lernalgorithmus mit Mitgliedschaftsfragen A$'$ einen Lernalgorithmus A, der C ohne Mitgliedschaftsfragen erlernt.

A erhält als Eingaben eine obere Schranke für die Zielkonzeptbeschreibung s und eine obere Schranke für die Beispiellänge n und simuliert A$'$, dessen Eingaben $s' = q_1(s, n)$, $n' = q_2(s, n)$ und der Genauigkeitswert $\frac{1}{8}$ sind. A läßt (PK, SK) durch G erzeugen.

Die Orakelaufrufe von A′ werden wie folgt behandelt:

- Beispielorakel:
  Wenn A′ ein zufälliges Beispiel anfordert, fordert seinerseits A durch das Beispielorakel ein Muster (x, b) an. Dieses Muster wird gespeichert, um später erkennen zu können, ob es bereits aufgetreten ist. Die Unterschrift $y = SP((PK, SK), x)$ wird erzeugt und $(f(PK, s, n, x, y), b)$ an A′ übergeben. Zur Erinnerung, die Funktion f erzeugt aus dem Paar (x, y) einen Binärstring w, der von der Konzeptbeschreibung $g(PK, s, n, u)$ auf die gleiche Weise bewertet wird wie x durch u.

- Orakel für vorherzusagendes Beispiel:
  Bei der Anfrage des vorherzusagenden Elements wird mit Ausnahme der Speicherung ebenso verfahren.

- Mitgliedschaftsfrage:
  Bei einer Mitgliedschaftsfrage für eine Zeichenkette w werden durch die Formel $f(PK, s, n, x, y) = w$ die Zeichenketten x und y bestimmt, falls sie existieren. Dies ist aufgrund der Bedingung 3 der Definition 8.6 möglich. Es wird überprüft, ob y eine korrekte Unterschrift für x ist. Wenn das der Fall ist, und das Paar (x, b) in einem früheren Schritt gespeichert wurde, wird b zurückgegeben sonst 0.

Wenn A′ eine Vorhersage macht, gibt A diese aus.

Der Lernalgorithmus A benötigt also keine Mitgliedschaftsfragen. Die Orakelanfragen von A′ werden bis auf einen einzigen Fall korrekt beantwortet: Wenn A eine Mitgliedschaftsfrage für ein w stellt, das sich in x und y aufteilen läßt, y eine gültige Unterschrift für x ist und das Beispiel x noch nicht aufgetreten ist und x in $\kappa_C(u)$ liegt. In diesem Fall wäre jedoch das Paar (x, y) eine erfolgreiche Fälschung.

Wenn alle Mitgliedschaftsfragen korrekt beantwortet wurden, liefert der erzeugte Algorithmus mit Wahrscheinlichkeit von über $\frac{7}{8}$ eine korrekte Antwort. In diesem Fall wird die Genauigkeitsschranke von $\frac{1}{4}$ eingehalten und wir erhalten nach Schapire [Sch89] auch die Erlernbarkeit in *pw2PAC*.

Wenn jedoch unendlich viele Konzepte und Verteilungen existieren, bei denen A einen Fehler von über $\frac{1}{4}$ macht, dann kann man A so umwandeln, daß A ein erfolgreicher Fälscher wird mit einer Wahrscheinlichkeit von über $\frac{1}{8}$. Dies würde aber im Widerspruch zur Sicherheit des Unterschriftsschemas stehen.

Wenn A also keine polynomiell beschränkte Vorhersage für C erreicht, dann existiert ein Polynom $p(n)$, das die Größe der Zielkonzeptbeschreibung beschränkt, und unendlich viele $n_1, n_2, \ldots$, so daß für jedes n_i ein Konzeptname u_i mit maximaler Länge $p(n_i)$ und eine Verteilung D_i auf $X^{[n_i]}$. Wenn nun A mit den Eingaben $s = p(n_i)$ und $n = n_i$, Konzept $\kappa_C(u_i)$ und Verteilung D_i gestartet wird, übersteigt die Wahrscheinlichkeit, daß A einen Fehler macht $\frac{1}{4}$.

Wir konstruieren nun einen Fälscher wie folgt:

Für jedes i, enthält das Informationsband t_{n_i} eine Zeichenkette u_i , entsprechend des unvorhersagbaren Konzepts für n_i. Der Angreifer hat Zugriff auf ein Orakel D, welches dazu benützt werden kann, um Beispiele entsprechend der Verteilung D_i auf $X^{[n_i]}$ zu erzeugen. Die anderen Informationsbänder enthalten nur die leere Zeichenkette.

Der Fälschungsalgorithmus läuft im Prinzip wie der vorhergegangene Lernalgorithmus ab, jedoch wird zur Erzeugung der Beispiele das Orakel des Angreifers benützt und man läßt den simulierten Lernalgorithmus das ungünstige Konzept, das auf dem Informationsband gespeichert ist, erlernen. Dadurch wird die Wahrscheinlichkeit einer fehlerhaften Ausgabe und damit auch die Wahrscheinlichkeit des Auftretens einer erfolgreichen Fälschung erhöht.

Es läßt sich nun beweisen, daß die Wahrscheinlichkeit für das Erzeugen einer erfolgreichen Fälschung über $\frac{1}{8}$ liegt. Also wäre das verwendete Unterschriftsschema (G, SP, V) nicht sicher und wir erhalten den erwarteten Widerspruch. □

8.5 Folgerungen

Durch Kombination dieses Theorems mit Lemma 8.5, der Repräsentation von Unterschriftsschemata und Lemma 8.1, der Existenz von sicheren Unterschriftsschemata, erhalten wir das folgende Resultat:

Satz 8.2 Folgerung
Wenn Einwegfunktionen existieren, die nicht durch polynomiell große Schaltkreise invertiert werden können dann sind die folgenden Repräsentationsklassen entweder beschränkt polynomiell vorhersagbar oder nicht beschränkt polynomiell vorhersagbar mit Mitgliedschaftsfragen:

$$C_{CNF}, C_{DNF}, C_{dTC} \forall d \geq 2, C_{BF}, C_{\cap DFA}, C_{\cup DFA}, C_{2DFA}, C_{NFA}, C_{CFG}$$

Also nützen unter den gegebenen Annahmen Mitgliedschaftsfragen bei der Vorhersage dieser Repräsentationsklassen nichts.

Kearns und Valiant [KV94] haben unter den folgenden drei Annahmen Quadratische Reste Annahme (Abschnitt 6.1), nicht effiziente Invertierbarkeit der RSA Verschlüsselungsfunktion (Abschnitt 3.2) und nicht effiziente Faktorisierbarkeit von Blum Integern (Abschnitt 3.1) nachgewiesen, daß die Repräsentationsklassen C_{BF}, $C_{\cap DFA}$, $C_{\cup DFA}$, C_{2DFA}, C_{NFA}, C_{CFG} des vorangegangenen Satzes nicht effizient erlernbar sind. Wir können also aufgrund der obengenannten Annahmen und der Annahme, daß Einwegfunktionen existieren, die nicht durch polynomiell große Schaltkreise invertiert werden können, folgern, daß die zuletzt genannten Klassen auch mit Mitgliedschaftsfragen nicht erlernbar sind.

Man könnte einem Lernalgorithmus auch gestatten, nach dem Einlesen eines vorherzusagenden Beispiels noch Mitgliedschaftsfragen zu stellen, ausgenommen für das vorherzusagende Beispiel selbst. Aber es wurde von Dolev, Dwork und Naor [DDN91] nachgewiesen, daß die hier gezeigten, nicht erlernbaren Konzeptrepräsentationen auch in dem erweiterten Erlernbarkeitsmodell unvorhersagbar bleiben.

Das wichtigere Resultat ist jedoch, daß CNF, DNF Formeln und Schaltkreise mit konstanter Tiefe entweder ohne Mitgliedschaftsfragen effizient erlernbar oder überhaupt nicht effizient erlernbar sind.

9

Diskussion und Ausblick

Wir haben in den vorausgehenden Kapiteln auf der Grundlage kryptographischer Annahmen eine Reihe von Resultaten in der Erlernbarkeitstheorie nachgewiesen. Wir überlegen uns nun, inwieweit es möglich ist, Erlernbarkeitsresultate auf kryptographische Systeme anzuwenden.

Da viele negative Resultate in der Erlernbarkeitstheorie auf kryptographischen Annahmen beruhen ist es nicht sinnvoll diese wieder auf Public-Key Systeme anzuwenden. Möglicherweise kann man jedoch, aus negativen Erlernbarkeitsresultaten, die nicht auf kryptographischen Annahmen beruhen, sichere kryptographische Systeme erzeugen.

Aus positiven Erlernbarkeitsresultaten kann man ableiten, welche Funktionsklassen ungünstig für klassische kryptographische Systeme sind. Als Beispiel für eine solche Anwendung gibt Rivest [Riv88] Kryptotext-Rückkopplungssysteme an. Hierbei wird eine Nachricht Bit für Bit verschlüsselt, indem eine Funktion über den letzten n erzeugten Kryptotextbits ausgeführt wird und das Klartextbit mit dem Ergebnis durch „exklusives oder“ verknüpft wird. Dem geheimen Schlüssel entspricht hierbei die verwendete Funktion, die beiden Teilnehmern bekannt sein muß.

Bei einem bekannten Klartextangriff sind dem Angreifer einige Klartext-Kryptotextpaare bekannt. Ein Lernalgorithmus, der anhand von Beispielen die Entschlüsselungsfunktion erlernt, wäre für den Angreifer sehr nützlich. Bei einem gewählten Kryptotextangriff könnte man dem Lernalgorithmus sogar gestatten, Mitgliedschaftsfragen zu stellen.

Beim Entwurf eines Kryptotext-Rückkopplungssystem wird gefordert, daß die verwendete Verschlüsselungsfunktion mit gleicher Wahrscheinlichkeit die Nachrichten 0 und 1 ausgibt. Daraus sollte aber auch folgen, daß die erzeugten Kryptotexte und damit auch die Eingabe für die Verschlüsselungsfunktion gleichverteilt erzeugt werden. Die Korrektheit dieser Behauptung muß allerdings im Einzelfall nachgeprüft werden.

Daraus würde also folgen, daß Funktionsklassen, die effizient durch gleichverteilt erzeugte Beispiele erlernbar sind, nicht in einem Kryptotext-Rückkopplungssystem verwendet werden sollten.

Es gibt in der Tat eine Reihe von Erlernbarkeitsresultaten, die auf dieser Grundlage aufbauen. Zum Beispiel haben Linial, Mansour und Nisan [LMN89] nachgewiesen, daß AC^0-Schaltkreise bei Gleichverteilung auf dem Beispielraum erlernbar sind.

Wenn sogar gewählte Kryptotextangriffe auf unser Kryptotext-Rückkopplungssystem möglich sind, sollten Repräsentationsklassen, die mit Mitgliedschaftsfragen auf einem gleichverteilten Beispielraum effizient erlernbar sind, vermieden werden.

Wir sehen also, daß auch Erlernbarkeitsresultate Auswirkungen auf den Entwurf kryptographischer Systeme haben können.

Literaturverzeichnis

[AB92] M. Anthony, N. Biggs: *Computational Learning Theory*. Cambridge University Press, 1992.

[ACG88] W. Alexi, B. Chor, O. Goldreich, C. Schorr: RSA and Rabin functions: Certain parts are as hard as the whole. In: *SIAM Journal of Computing 17,2*. pages: 194-209, 1988.

[AK91] Dana Angluin, Michael Kharitonov: When Won't Membership Queries Help. In: *23rd Annual ACM Symposium on Theory of Computing*. pages: 444-454, 1991.

[AK95] Dana Angluin, Michael Kharitonov: When Won't Membership Queries Help. In: *Journal of Computer and System Sciences 50*. pages: 336-355, 1995.

[Ang87] Dana Angluin: Learning reqular sets from queries and counterexamples. In: *Information and Computation*. pages: 75:87-106, 1987.

[Ang92] Dana Angluin: Computational Learning Theory: Survey and Selected Bibliography. In: *24th Annual ACM Symposium on Theory of Computing*. pages 351-369, 1992.

[BCH86] P. W. Beame, S. A. Cook, H. J. Hoover: Log depth circuits for division and related problems. In: *SIAM Journal on Computing, 15 (4)*. pages: 994-1003, 1986.

[BFM88] M. Blum, P. Feldmann, S. Micali: Non-Interactive Zero-Knowledge Proof Systems. In: *Proc. 20th Annual Symposium on the Theory of Computing*. pages: 103-112, 1988.

[Blu90] A. Blum: Separating distribution-free and mistake-bound learning models over the boolean domain. In: *Proc. 31st Annual Symposium on Foundations of Computer Science*. pages: 211-218, 1990.

[BM92] M. Bellare, S. Micali: How to sign given any trapdoor function. In: *Journal of the Association for Computing Machinery, Vol 40.* pages: 214-233, 1992.

[BS91] Bronstein, Semendjajew: *Taschenbuch der Mathematik.* Verlag Nauka Moskau, 1991.

[BSW91] Albrecht Beutelspacher, Jorg Schwenk, Klaus-Dieter Wolfenstetter: *Moderne Verfahren der Kryptographie.* vieweg Mathematik, 1991.

[Che52] H. Chernoff: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. In: *The Annuals of Mathematical Statistics 23.* pages: 493-509, 1952.

[CSV84] A. Chandra, L. Stockmeyer, U. Vishkin: Constant depth reducibility. In: *SIAM Journal of Computing 13,2.* pages: 423-432, 1984.

[DDN91] D. Dolev, C. Dwork, M. Naor: Non-malleable cryptographie. In: *Proceedings, 23d Annual ACM Symposium on Theory of Computing.* pages: 542-552, 1991.

[DH76] W. Diffie und M. E. Hellmann: New Directions in Cryptography. In: *IEEE Transactions on Information Theory.* pages: 644-654,1976.

[FLS90] U. Feige, D. Lapidot, A. Shamir: Multiple non-interactive zero knowledge proofs based on a single random string. In: *Proceedings of the 31st Annual Symposium on Foundations of Computer Science.* pages: 308-317, 1990.

[GGM86] O. Goldreich, S. Goldwasser, S. Micali: How to construct random functions. In: *Journal of the ACM 33 4(Oct).* pages: 792-807, 1986.

[GM83] Shafi Goldwasser and Silvio Micali: Probabilistic Encryption. In: *Journal of Computer and System Sciences 28.* pages: 270-299,1983.

[GMR88] S. Goldwasser, S. Micali, C. Rivest: A digital signature scheme secure against adaptive chosen message attacks. In: *SIAM Journal on Computing 17.* pages: 281-308, 1988.

[HKL88] D. Haussler, M. Kearns, N. Littlestone, M. Warmut: Equivalence of models for polynomial learnability. In: *Proceedings of the 1988 Workshop on Computational Learning Theory.* pages: 42-55, 1988.

[HU90] John E. Hopcroft, Jeffrey D. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitatstheorie.* Addison-Wesley, 1990.

[Kob94] Johannes Köbler: *Vorlesungsskript Kryptographie*. Universitat Ulm, 1994.

[KuV94] M. J. Kearns, Umesh V. Vazirani: *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[KV89] Michael Kearns, Leslie G. Valiant: Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In: *21st ACM Symposium on Theory of Computing*. pages 433-444, 1989.

[KV94] Michael Kearns, Leslie Valiant: Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In: *Journal of the Association for Computing Machinery, Vol 41*. pages 67-95, 1994.

[Lee90] Jan van Leeuwen: *Algorithms and Complexity*. Elsevier, 1990.

[Lit88] N. Littlestone: Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. In: *Machine Learning*. pages: 285-318, 1988.

[LMN89] Nathan Linial, Yishay Mansour, Noam Nisan: Constant depth circuits, fourier transform, and learnability. In: *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*. pages: 574-579, 1989.

[NY89] M. Naor, M-Yung: Universal one-way hash funktions and their cryptographic applications. In: *Proceedings, 21st Annual ACM Symposium on Theory of Computing*. pages: 33-43, 1989.

[NY90] Moni Naor, Moti Yung: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*. pages: 427-437, 1990.

[Pap94] Papadimitriou: *computational complexity*. Addison Wesley, 1994.

[PV88] L. Pit, L. Valiant: Computational limitations on learning from exampoles. In: *Journal of the ACM, 4 (Okt)*. pages: 965-984, 1988.

[PW88] L. Pitt, M. Warmuth: Reductions among prediction problems: on the difficulty of predicting automata. In: *Proceedings of the 3rd IEEE Conference on Structure in Complexity Theory*. pages: 60-69, 1988.

[Rab79] M. Rabin: Digital signatures and public-key functions as intractable as factoring. In: *Technical Report TM-212 Lab. Comput. Sci.* MIT Cambridge, Mass, 1979.

[Rei87] J. Reif: On threshold circuits and polynomial computations. In: *Proceedings of the 2nd Structure in Complexity Theory Conference.* pages: 118-125, 1987.

[Riv88] Ronald L. Rivest: Cryptographie and Machine Learning. In: *email: rivest@theory.lcs.mit.edu.* 1988.

[Rom90] J. Rompel: One-way functions are necessary and sufficient for secure signatures. In: *Proceedings, 22d Annual Symposium on THeory of Computing.* pages: 387-394, 1990.

[Sca91] Robert E. Schapire: Learning probabilistic read-once formulas on product distributions. In: *Proceedings of the Fourth Annual Workshop on Computational Learning Theorie.* pages: 184-198, 1991.

[Sch89] R. Schapire: On the strength of weak learnability. In: *Proceedings of the 30th IEEE Symposium on the Foundations of Computer Science.* pages: 28-33, 1989.

[Sch94] Uwe Schöning: *Theoretische Informatik - kurzgefasst.* Spektrum, akademischer Verlag, 1994.

[Sch95] Uwe Schöning: *Perlen der Theoretischen Informatik.* B · I · Wissenschaftsverlag, 1995.

[Sch96] Bruce Schneier: *Applied Cryptography (second edition).* WILEY, 1996.

[SMP87] A. De Santis, S. Micali, G. Persiano: *Non-Interactive Zero-Knowledge Proof Systems.* Proceedings of Cryptograpy 1987.

[VC71] V. N. Vapnik, A. Ya. Chervonenkis: On the uniform convergence of relative frequencies of events to their probabilities. In: *Theory of Probability and its Applications.* pages: 264-280, 1971.

[Val84] L. G. Valiant: A theory of the learnable. In: *C. ACM 27.* pages: 1134-1142, 1984.

[Yao82] A. Yao: Theory and application of trapdoor functions. In: *Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science.* pages: 80-91, 1982.

Symbolverzeichnis und Worterklärungen

Bezeichnung	Beschreibung
n	Maximale Beispielgröße
s	Maximale Größe der Zielkonzeptbeschreibung
u	Konzeptrepräsentation
m	Größe einer Probe
effizient	In Polynomialzeit berechenbar
log()	Logarithmus zur Basis zwei
$p(n)$, $q(n)$	positive Polynome
X	Menge der binären Zeichenketten
$X^{[n]}$	Die Menge der binären Zeichenketten mit maximaler Länge n ($\{w \in X \mid \lvert w \rvert < n\}$)
Einheitsdarstellung	Um eine Zahl n in Einheitsdarstellung zu repräsentieren verwendet man n Einsbits gefolgt von einem Nullbit.
$L(s)$	Ausgabe eines Lernalgorithmus bei Eingabe der Probe s
$C_{X\ldots X}$	Repräsentation der Konzeptklasse $X \ldots X$.
C_{BF}	Repräsentationsklasse der Booleschen Formeln
C_{DFA}	Repräsentationsklasse der Deterministischen Endlichen Automaten
C_{NFA}	Repräsentationsklasse der Nichtdeterministischen Endlichen Automaten
C_{dTC}	Repräsentationsklasse der Schwellenschaltkreise mit konstanter Tiefe d
$C_{3\mu-\mathrm{BF}}$	Repräsentationsklasse der Booleschen Formeln in denen jede Variable nur maximal dreimal auftreten darf
$C_{\cup\mathrm{DFA}}$	Repräsentationsklasse der endlichen Vereinigungen von *DFA's*. Wird durch Hintereinanderschreiben der Repräsentationen der verwendeten *DFA's* dargestellt
$C_{\cap\mathrm{DFA}}$	Repräsentationsklasse der endlichen Schnitte von *DFA's*. Repräsentation wie $\cup$*DFA*.
$C_{2\mathrm{DFA}}$	Repräsentationsklasse der Zweiweg *DFA's*
C_{CFG}	Repräsentationsklasse der Kontextfreien Grammatiken
$\kappa_C(u)$	Die Menge der Eingaben die zu akzeptierenden Auswertungen einer Konzeptbeschreibung u aus einer Repräsentationsklasse C führen.
konsistent	Eine Hypothese h ist mit einer Probe s konsistent, wenn h die Beispiele aus der Probe auf die selbe Weise bewertet.
$Con(C)$	Konsistenzproblem: Es muß eine Hypothese h gefunden werden, die mit einer gegebenen Probe s konsistent ist. Wobei die Probe mit einem Zielkonzept aus C konsistent ist.
$Con(C,H)$	Wie $Con(C,H)$, jedoch muß die gesuchte Hypothese h aus dem Hypothesenraum H gewählt werden.
vernachlässigbare Wahrscheinlichkeit	Eine Folge von Wahrscheinlichkeiten die für jedes Polynom $p(n)$ für ausreichend große n kleiner als $1/p(n)$ bleibt.

Name: **Guido Kehrle** Matrikelnummer: **0234823**

Erklärung

Ich erkläre, daß ich die Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 31. Mai 1996 ....

(Unterschrift)