

Reduzierung von Interaktion in kryptographischen Protokollen



Diplomarbeit

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

eingereicht von: Martin Stigge

Gutachter: Prof. Dr. Johannes Köbler
Prof. Dr. Ernst-Günter Giessmann

Berlin, den 08.08.2008

Inhaltsverzeichnis

1	Einleitung	1
1.1	Grundlagen	3
2	Pseudozufallsgeneratoren und Hitting Set Generatoren	11
2.1	Zufall versus Determinismus	11
2.2	Pseudozufallsgeneratoren	13
2.2.1	„BMY-Typ“ Generatoren	14
2.2.2	„NW-Typ“ Generatoren	16
2.3	Hitting Set Generatoren	19
2.3.1	Derandomisierung mittels Hitting Set Generatoren	20
2.3.2	Alternative Definitionen	21
2.3.3	Existenz von HSG	22
2.3.4	HSG gegen nichtdeterministische Beobachter	24
3	Bit Commitment	29
3.1	Bit Commitment Protokolle	30
3.2	Nicht-interaktives Bit Commitment via injektiver Einwegfunktion	33
3.3	Interaktives Bit Commitment via Einwegfunktionen	36
3.4	Derandomisierung von Naor’s Protokoll	40
3.5	Partiell-injektive Einwegfunktionen	44
4	Zero Knowledge Proofs und Witness Indistinguishability	47
4.1	Zero Knowledge Beweissysteme	48
4.1.1	Beispiele	51
4.1.2	Zero Knowledge mit Privater Eingabe	53
4.1.3	Existenz	54
4.1.4	Abgeschlossenheit	56
4.2	Witness Indistinguishability	58
4.3	Nicht-interaktives Zero Knowledge	62
4.4	Zaps	65
4.5	Derandomisierung von Zaps	70
5	Oblivious Transfer	73
5.1	1-aus-2 Oblivious Transfer	74
5.2	Dwork und Naor’s Protokoll	75
5.3	Derandomisierung von Dwork und Naor’s Protokoll	78
6	Zusammenfassung und Ausblick	81

Kapitel 1

Einleitung

Kryptographie beschäftigt sich mit dem Schutz von Daten vor potentiellen „Gegnern“ mittels mathematischer Transformationen. Dazu zählen Schutzziele wie Geheimhaltung oder Integrität, die von den Gegnern bedroht werden können. Das klassische Szenario ist hierbei, dass zwei Parteien sich gegenseitig Nachrichten zusenden möchten, ohne Dritten die Möglichkeit zu geben, diese zu lesen oder unbemerkt zu verändern. Darüber hinaus haben sich im Laufe der Zeit, insbesondere im letzten Jahrhundert durch Verfügbarkeit moderner Rechen- und Kommunikationstechnik, viele weitere situationsabhängige Schutzziele wie Anonymität, Identifizierung oder (Un-)Abstreitbarkeit ergeben, die man mittels kryptographischer Methoden zu lösen versucht.

Allem gemein ist, dass die zumeist mindestens zwei Teilnehmer auf eine wie auch immer gegebene Weise kommunizieren und sich dabei an ein zuvor vereinbartes Protokoll halten sollen. Die Eigenschaften dieser *kryptographischen Protokolle* werden untersucht und beinhalten neben den gewünschten kryptographischen Eigenschaften (wie eben zum Beispiel einem gewissen Grad an Geheimhaltung) auch Merkmale wie den Aufwand der nötigen *Interaktion* zwischen den teilnehmenden Parteien. In vielen Situationen ist es dabei wünschenswert, dass die Interaktion nach Möglichkeit minimiert wird, sei es aus Gründen der Geheimhaltung, der Kosten, oder auch schlicht der Unmöglichkeit, ein hohes Maß an Interaktion praktisch zu realisieren. So ist zum Beispiel vorstellbar, dass zwei Teilnehmer räumlich eine große Distanz zu überbrücken haben und jeder Transport von Daten mit immensen Kosten oder auch Risiken verbunden ist. Noch nötiger ist eine Beschränkung der Interaktion, wenn die Teilnehmer *zeitlich* voneinander getrennt sind, das Protokoll also einen langen Zeitrahmen überspannt. In dem Fall kann eine Interaktion vielleicht sogar nur in eine Richtung stattfinden, weil für einen Rückkanal einer der Teilnehmer gar nicht mehr existiert. (Als Beispiel sei die Hinterlassenschaft eines Geheimnisses über Jahrhunderte genannt.)

Mit einer solchen *Reduzierung* oder gar *Minimierung* von Interaktion in kryptographischen Protokollen befasst sich diese Arbeit. Dabei sind verschiedene Ansätze denkbar, die davon abhängen, welche Aspekte der betrachteten Protokolle man untersucht. Wir lenken unseren Fokus dabei auf den Aspekt von *Probabilismus*: Um die Gegenseite vom „Schummeln“ abzuhalten, also vom bewussten Brechen der vorgesehenen Schutzziele, funktioniert ein großer Teil kryptographischer Protokolle *randomisiert*. Die Teilnehmer wählen im Laufe eines solchen Protokolls Zufallszahlen, um den Ablauf des Protokolls in gewissem Maße unvorhersehbar zu gestalten, insbesondere für wie auch immer agierende Gegner. Ein solcher kann somit nicht vollständig vorhersehen, mit welchen Daten er es zu tun hat, und seine Reaktionen mit potentiell „böswilliger Absicht“ daran anpassen. Der Nachteil ist jedoch, dass ein großer Teil der kostbaren Ressource „Interaktion“ allein dadurch entsteht, dass sich die Parteien schlicht gegenseitig Zufallszahlen zuschicken.

Es liegt daher nahe, sich Methoden der *Derandomisierung* zu bedienen. Diese Art von Techniken ist in der Komplexitätstheorie bereits recht umfassend untersucht. Dort wird Derandomisierung verwendet, um zum Beispiel Beziehungen zwischen probabilistischen und deterministischen Komplexitätsklassen zu betrachten – also zu untersuchen, ob Rechenmodelle, die Zufall in Berechnungen einfließen lassen, einen echten Zugewinn an Berechnungsstärke bedeuten. Zum Einsatz kommen hier sogenannte Pseudozufallsgeneratoren, die in ähnlicher Form in der Kryptographie bereits Verwendung finden (und in der Tat zuerst in der Kryptographie eingeführt wurden). Dies sind Algorithmen, deren Ausgaben wie zufällige Daten „aussehen“. Die entscheidende Beobachtung bei der Derandomisierung (in Anlehnung an den Artikel [BOV07] von Barak, Ong und Vadhan) wird sein, dass viele in kryptographischen Protokollen verwendete Zufallszahlen gar nicht wirklich zufällig sein müssen, um ihre Aufgabe zu erfüllen. Sie müssen nur mit „hinreichender Güte“ gewählt werden, wobei ein Pseudozufallsgenerator helfen kann. Auf diese Weise ist es möglich, die Interaktion in den betreffenden Protokollen zu verringern, indem Schritte, in denen Zufallszahlen ausgetauscht werden, durch bestimmte lokale Generator-Aufrufe der Teilnehmer ersetzt werden.

Wir betrachten in der Arbeit drei Arten von Protokollen: Bit Commitment, Zeugen-ununterscheidbare Beweissysteme, und Oblivious Transfer. In allen drei Fällen werden bereits vorhandene Protokolle zunächst untersucht, und anschließend wird mittels Derandomisierung die Anzahl der nötigen Kommunikations-Runden reduziert. In den ersten beiden Fällen wird dabei Nicht-Interaktivität erreicht, d.h. die Reduktion auf eine einzige gesendete Nachricht, also das Optimum. Im Falle von Oblivious Transfer erreichen wir eine Reduktion auf 2 Runden, was bei dieser Protokoll-Art auch dem Optimum entspricht.

Die Arbeit ist nun wie folgt aufgebaut: Den Rest des ersten Kapitels bildet eine Übersicht über die verwendeten formalen Grundlagen, also Definitionen und grundlegende Resultate aus den Bereichen Kryptographie und Komplexitätstheorie. In Kapitel 2 werden die später verwendeten Derandomisierungs-Techniken erarbeitet, also Pseudozufallsgeneratoren¹ und verwandte Objekte. Kapitel 3 wird dann Bit Commitment Protokolle definieren, konkrete Instanzen des Protokolles vorstellen, und eines von diesen schließlich derandomisieren. Im Anschluss daran beleuchtet Kapitel 4 das Konzept von Zero Knowledge und die verwandte Zeugen-Ununterscheidbarkeit samt grundlegender Resultate. Ein 2-Runden Protokoll wird am Ende dieses Kapitels wiederum derandomisiert. Eine Anwendung dieses Resultats stellt schließlich Kapitel 5 dar, in dem Oblivious Transfer Protokolle untersucht werden. Einige Schlussbemerkungen finden sich schließlich in Kapitel 6.

1.1 Grundlagen

In diesem Abschnitt führen wir die verwendete Notation für die folgenden Kapitel ein. Es handelt sich dabei um Grundbegriffe und -resultate aus Komplexitätstheorie und Kryptographie, wobei wir uns an etablierte Standard-Schreibweisen halten.

Sprachen: Die Basis sämtlicher Betrachtungen bilden *Sprachen*. Eine Sprache L ist eine Menge von Wörtern $x \in L$ über einem endlichen Alphabet Σ , d.h. $L \subseteq \Sigma^* = \bigcup_{i \geq 0} \Sigma^i$. Das Standard-Alphabet ist meist $\Sigma = \{0, 1\}$. Die *charakteristische Funktion* $\chi_L : \Sigma^* \rightarrow \{0, 1\}$ einer Sprache L ist definiert als $\chi_L(x) = 1 \iff x \in L$.

Uniformes Berechnungsmodell: Zur Modellierung der Berechnungsstärke von in unseren Szenarien agierenden Individuen verwenden wir zwei verschiedene Berechnungsmodelle. Das *uniforme* Berechnungsmodell ist die klassische Turingmaschine (TM) mit einer endlichen Zustandsmenge und konstant vielen, unendlich langen Arbeitsbändern, einem Eingabeband, und je nach Kontext auch einem Ausgabeband. Diese ist zunächst *deterministisch*, zu einer Konfiguration gibt es also in Abhängigkeit des aktuellen Zustandes und der Zeichen unter den Lese-/Schreibköpfen stets maximal eine Folge-

¹In dieser Arbeit werden wir sogenannte *Hitting Set Generatoren* verwenden, die eine spezielle Form der Pseudozufallsgeneratoren sind. Ein früher Arbeitstitel der Arbeit lautete entsprechend auch „Derandomisierung in der Kryptographie mittels Hitting Set Generatoren“.

konfiguration. Schreiben wir $M(x)$ für die Ausgabe einer TM M bei Eingabe x , so ist die akzeptierte Sprache:

$$L(M) := \{x \in \Sigma^* \mid M(x) = 1\}$$

Bezeichnet $\text{time}_M(x)$ die Anzahl der Schritte, die M bei Eingabe x ausführt, so ist M $t(n)$ -zeitbeschränkt, falls für alle x gilt $\text{time}_M(x) \leq t(|x|)$. Damit nennen wir die Klasse der Sprachen, die durch eine $t(n)$ -zeitbeschränkte Maschine M entschieden werden können, $\text{DTIME}(t(n))$. Für den Fall von polynomiellen Zeitschranken schreiben wir auch kurz $\mathbf{P} := \text{DTIME}(n^{O(1)})$. Eine polynomiell in ihrer Laufzeit beschränkte, deterministische TM, kurz „P-TM“, wird auch *effizient* genannt.

Wir betrachten drei Erweiterungen des deterministischen Modells der Turingmaschine:

Nichtdeterminismus ist eine erste Erweiterung des Modells. Hier darf die Maschine zwischen verschiedenen Folgekonfigurationen wählen. Diese nichtdeterministischen Entscheidungen können explizit durch einen zweiten Eingang y beschrieben werden, sodass wir für die akzeptierte Sprache haben:

$$L(M) := \{x \in \Sigma^* \mid \exists y \in \Sigma^* : M(x, y) = 1\}$$

Je nach Kontext wird dieser zweite Eingang auch weggelassen und die nichtdeterministischen Entscheidungen werden somit implizit getroffen. Die Laufzeit $\text{time}_M(x)$ einer solchen nichtdeterministischen TM, kurz „NTM“, ist dann das Maximum über alle nichtdeterministischen Entscheidungen, und alle derart entscheidbaren Sprachen fassen wir in $\text{NTIME}(t(n))$ zusammen. Für den polynomiellen Fall schreiben wir $\mathbf{NP} := \text{NTIME}(n^{O(1)})$. Analog betrachtet man auch *co-nichtdeterministische* Maschinen M , deren Akzeptanzkriterium dann lautet:

$$L(M) := \{x \in \Sigma^* \mid \forall y \in \Sigma^* : M(x, y) = 1\}$$

Eine weitere Charakterisierung der Sprachen $L \in \mathbf{NP}$ besteht in der Existenz einer in \mathbf{P} entscheidbaren, sogenannten *Zeugen-Relation* R_L , wobei es ein Polynom $p(n)$ geben muss sodass $\forall (x, y) \in R_L : |y| \leq p(|x|)$ und es gilt:

$$x \in L \iff \exists y \in \Sigma^{p(|x|)} : (x, y) \in R_L$$

Ein y mit $(x, y) \in R_L$ wird auch *Zeuge für* $x \in L$ genannt.

Probabilismus ist eine zweite Erweiterung. Dabei wählt die Maschine die Folgekonfigurationen mit einer gewissen Wahrscheinlichkeit. Wie oben kann auch das über einen zweiten Eingang r beschrieben werden, der für die Akzeptanzbedingung dann jedoch nicht Existenz-quantifiziert, sondern zufällig gewählt wird. (Zur Vereinfachung betrachten wir hier nur noch polynomielle Zeitschranken.) Man unterscheidet zwischen ein- und zweiseitigem Fehler. Für probabilistische Maschinen, die eine Sprache L mit einseitigem Fehler akzeptieren und $p(n)$ -zeitbeschränkt sind, soll für die Fehlerwahrscheinlichkeiten gelten:

$$\begin{aligned} x \in L &\Rightarrow \Pr_{r \in \Sigma^{p(|x|)}}[M(x, r) \neq 1] \leq 1/2 \\ x \notin L &\Rightarrow \Pr_{r \in \Sigma^{p(|x|)}}[M(x, r) \neq 0] = 0 \end{aligned}$$

Es sind also nur bei positiven Eingaben Fehler mit einer Wahrscheinlichkeit von $1/2$ zugelassen. Diese Sprachen L sind in der Klasse **RP** zusammengefasst. Für zweiseitigen Fehler lässt sich verkürzt schreiben:

$$\forall x \in \Sigma^* : \Pr_{r \in \Sigma^{p(|x|)}}[M(x, r) \neq \chi_L(x)] \leq 1/3$$

Der Fehler liegt hier also in beiden Fällen bei höchstens $1/3$. Diese Sprachen bilden die Klasse **BPP**, in der offenbar **RP** enthalten ist. Für eine probabilistische, polynomiell in der Laufzeit beschränkte Maschine schreiben wir auch kurz „pp-TM“. Wie auch für den Nichtdeterminismus wird oft der zweite Eingang weggelassen und die probabilistischen Entscheidungen werden implizit, also intern getroffen (engl. *internal coin tosses*). Dabei kann die Fehlerschranke von $1/3$ reduziert werden auf eine exponentiell kleine Schranke $2^{-p(|x|)}$ für ein beliebiges Polynom, indem M hinreichend oft erneut aufgerufen wird.

Interaktivität betrachten wir als dritte Erweiterung des Modells der Turingmaschine. Damit soll modelliert werden, dass Maschinen miteinander Nachrichten austauschen können. Dies geschieht formal über zusätzliche *gemeinsame* Kommunikations-Bänder, auf welche eine solche *interaktive Turingmaschine* (ITM) die zu übertragende Nachricht schreibt. Anschließend geht sie in einen speziellen Zustand, um die Kontrolle an die Maschine zu übergeben, an welche die Nachricht geschickt wurde. Die beteiligten Maschinen wechseln sich so in ihrer Arbeit ab. Wir betrachten nur Interaktionen von *zwei* Maschinen P und V , die beide auch probabilistisch sein können, und bezeichnen die Menge der bei Eingabe x ausgetauschten Nachrichten mit $\langle P, V \rangle(x)$ und die Ausgabe der gemeinsamen Berechnung mit $[P, V](x)$ (dies ist die Ausgabe von V sobald V in den Haltezustand geht). Von einem *interaktiven Beweissystem* (P, V) für eine Sprache L spricht man, falls (P, V) ein Paar von ITMs ist, V außerdem polynomiell in der Laufzeit beschränkt (d.h. „pp-ITM“), und es gilt:

Completeness: Für alle $x \in L$ ist $\Pr_{P,V} [[P, V](x) \neq 1] \leq 1/3$.

Soundness: Für alle $x \notin L$ und jede ITM P^* gilt:

$$\Pr_{P^*,V} [[P^*, V](x) \neq 0] \leq 1/3.$$

Im positiven Fall soll also P als „Prover“ den „Verifier“ V mit hoher Wahrscheinlichkeit von $x \in L$ überzeugen können. Im negativen Fall soll es kein P^* geben, der V fälschlicherweise mit mehr als $1/3$ Wahrscheinlichkeit von $x \in L$ überzeugen kann. Die Sprachklasse aller Sprachen L , die ein solches interaktives Beweissystem besitzen, wird mit IP bezeichnet. Wie auch bei BPP kann hier die Fehlerschranke von $1/3$ auf $2^{-p(|x|)}$ für beliebige Polynome reduziert werden. Es ist sogar ausreichend, einen Unterschied der Akzeptanzwahrscheinlichkeiten (für positive bzw. negative Instanzen) von $1/p(|x|)$ für ein Polynom $p(\cdot)$ zu fordern, falls diese als FP-Funktionen gegeben sind. Im Gegensatz dazu wird von *perfekter* Completeness bzw. Soundness gesprochen, wenn die Wahrscheinlichkeit sogar jeweils 1 ist, der Fehler also 0.

Neben der gemeinsamen Eingabe x können beide Teilnehmer P und V auch noch *private Eingaben* y bzw. z bekommen. Ein solches *interaktives Beweissystem mit privaten Eingaben* hat dann erweiterte Akzeptanzbedingungen:

Completeness: Für alle $x \in L$ gibt es ein $y \in \{0, 1\}^*$, sodass für alle $z \in \{0, 1\}^*$:

$$\Pr_{P,V} [[P(y), V(z)](x) \neq 1] \leq 1/3$$

Soundness: Für alle $x \notin L$, jede ITM P^* und alle $y, z \in \{0, 1\}^*$ gilt:

$$\Pr_{P^*,V} [[P^*(y), V(z)](x) \neq 0] \leq 1/3$$

Nicht-uniformes Berechnungsmodell: Als zweites Berechnungsmodell betrachten wir *Schaltkreisfamilien*. Der Unterschied zum uniformen Modell ist hier, dass mit der Größe der Eingabe nicht nur die Laufzeit, sondern auch die „Größe“ der Maschine (d.h. ihre Beschreibungskomplexität) wachsen können soll.

Ein (*boolescher*) *Schaltkreis* c der Größe $\text{size}(c) = m$ mit n Eingängen ist ein gerichteter Knoten-gelabelter Graph $c = (V, E, \text{label})$ ohne (gerichtete) Kreise. Die Knoten $V = \{1, \dots, m\}$ werden *Gatter* genannt und haben Label² $\text{label} : V \rightarrow \{\perp, \top, \wedge, \neg, X_1, X_2, \dots, X_n\}$. Die Gatter v mit $\text{label}(v) \in \{\perp, \top, X_1, X_2, \dots, X_n\}$ haben Eingangsgrad 0 und heißen die *Eingänge von c*. Die $v \in V$ mit $\text{label}(v) = \neg$ haben Eingangsgrad 1, und

²Zur Vereinfachung fehlt \vee als Label, welches durch \wedge und \neg simuliert werden kann.

mit $\text{label}(v) = \wedge$ Eingangsgrad 2. Alle Knoten $v \in \{1, \dots, m-1\}$ haben beliebigen Ausgangsgrad, und der Knoten m hat Ausgangsgrad 0 und wird *der Ausgang von c* genannt. Der Wert $\text{val}(v)$ eines Knotens v bei Eingabe $x = x_1 \cdots x_n \in \{0, 1\}^n$ ist definiert als:

$$\text{val}(v) := \begin{cases} 0 \text{ bzw. } 1 \text{ bzw. } x_i, & \text{falls } \text{label}(v) = \perp \text{ bzw. } \top \text{ bzw. } X_i \\ 1 - \text{val}(u), & \text{falls } \text{label}(v) = \neg \text{ und } (u, v) \in E \\ \text{val}(u_1) \cdot \text{val}(u_2), & \text{falls } \text{label}(v) = \wedge \text{ und } (u_1, v), (u_2, v) \in E \end{cases}$$

Die Ausgabe $c(x)$ des Schaltkreises c bei Eingabe $x \in \{0, 1\}^n$ ist dann $\text{val}(m)$ wenn den Eingängen die Werte der Bits von x zugewiesen werden. Die von c akzeptierte Sprache ist $L(c) := \{x \in \{0, 1\}^n \mid c(x) = 1\}$. Sei L eine Sprache, dann kann für $L_n := L \cap \{0, 1\}^n$ ein solcher Schaltkreis zum Entscheiden der Zugehörigkeit eines Wortes x zu L_n verwendet werden, und für ganz L eine Schaltkreisfamilie $C = \{C_n\}_{n \in \mathbb{N}}$ wobei C_n genau L_n entscheidet. (C selbst wird auch oft einfach nur „Schaltkreis“ genannt.) Das Berechnungsmodell ist damit *nicht-uniform*, da zu jeder Eingabelänge n ein vollkommen anderer Schaltkreis verwendet werden kann, wodurch C keine endliche Beschreibung besitzen muss (im Gegensatz zur Turingmaschine).

Analog der Laufzeitschranken für das uniforme Berechnungsmodell bezeichnen wir mit $\text{DSIZE}(t(n))$ die Menge der Sprachen, die mit Schaltkreisen der Größe höchstens $t(n)$ entschieden werden können. Der Sonderfall für polynomielle Größenschranken wird mit $\text{P/poly} := \text{DSIZE}(n^{O(1)})$ bezeichnet, wobei diese polynomiell beschränkten Schaltkreise auch *effizient* genannt werden, analog dem uniformen Fall. Ein wichtiges Resultat hierbei ist das folgende:

Theorem 1.1 ([Sav72]). *Es gilt: $\text{DTIME}(t(n)) \subseteq \text{DSIZE}(O(t(n)^2))$.*

Dies liefert nicht nur $\text{P} \subseteq \text{P/poly}$, also dass jedes $L \in \text{P}$ auch durch polynomielle Schaltkreise entschieden werden kann, sondern sogar dass zu jeder TM A mit Laufzeitschranke $p(n)$ eine Schaltkreisfamilie C der Größe $O(p(n)^2)$ existiert mit $L(C) = L(A)$.

Umgekehrt lässt sich P/poly auch interpretieren als die Menge der Sprachen L , die durch eine P -Turingmaschine entschieden werden, welche zu jeder Eingabelänge noch einen polynomiell langen *Advice* bekommt. Zum Beispiel kann sie einen Schaltkreisauswerter implementieren, und der Advice für Eingabelänge n ist eine Beschreibung des Schaltkreises C_n mit $L(C_n) = L_n$. Aus diesem Grund spricht man bei P/poly auch von *nicht-uniformer Polynomialzeit*.

Wir bezeichnen mit $S(L)$ die minimale Größe einer L entscheidenden Schaltkreisfamilie:

$$S(L) := (n \mapsto \min\{m \mid \exists c : \text{size}(c) = m \wedge L(c) = L_n\})$$

Analog zum uniformen Fall können ebenso nichtdeterministische bzw. probabilistische Schaltkreise definiert werden. Die minimale Schaltkreis-Größe bezeichnen wir im nichtdeterministischen Fall mit $S_N(L)$.

Weitere Grundbegriffe: Wir nennen eine Funktion $\nu(\cdot)$ *vernachlässigbar*, falls für jedes Polynom $p(\cdot)$ und hinreichend große n gilt: $\nu(n) < 1/p(n)$. Wir schreiben außerdem $\text{poly}(n)$ als Platzhalter für ein beliebiges Polynom, welches sich aus dem Kontext ergibt (zumeist durch polynomielle Beschränkung der betrachteten Maschine).

Schließlich wenden wir uns noch Grundbegriffen aus der Kryptographie zu. Zunächst definieren wir *Einwegfunktionen*. Dies sind Funktionen, die zwar effizient berechnet, jedoch nicht effizient umgekehrt werden können:

Definition 1.2 (Einwegfunktion). *Eine Funktion $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ heißt Einwegfunktion, falls folgendes gilt:*

- f ist in Polynomialzeit berechenbar (d.h. $f \in \text{FP}$)
- Für alle pp-TMs A , Polynome $p(\cdot)$ und hinreichend große n gilt:

$$\Pr_{U_n, A}[A(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

Dabei bezeichnet U_n eine Zufallsvariable, deren Wert gleichverteilt aus $\{0, 1\}^n$ gewählt wird. Wir nennen obige Wahrscheinlichkeit auch die Erfolgswahrscheinlichkeit von A bei Argumentlänge n .

Die Funktion f heißt *längenerhaltend*, falls $\forall x \in \{0, 1\}^* : |f(x)| = |x|$. Eine *längenerhaltende Einwegfunktion*, die injektiv ist, wird auch *Einwegpermutation* genannt.

Als nächstes betrachten wir den Begriff der *Ununterscheidbarkeit* von Zufallsvariablen. Intuitiv besagt dies für ein Paar von Zufallsvariablen, dass ein Beobachter, der Werte einer der beiden Variablen als Eingabe bekommt, nicht entscheiden können soll, welche der beiden er als Eingabe bekam. Seine jeweiligen Akzeptanzwahrscheinlichkeiten dürfen sich also nicht nennenswert unterscheiden. Man unterscheidet zwischen drei verschiedenen starken Arten der Ununterscheidbarkeit. Die schwächste, für unsere Zwecke jedoch ausreichende, ist die Ununterscheidbarkeit durch effiziente Algorithmen (*uniform ununterscheidbar*) bzw. Schaltkreise (*nicht-uniform ununterscheidbar*).

Definition 1.3. 1. Eine Folge $\{X_i\}_{i \in \mathbb{N}}$ bzw. $\{X_w\}_{w \in \Sigma^*}$ heißt *Zufalls-Ensemble*, falls die X_i bzw. X_w Zufallsvariablen sind.

2. *Indexmenge \mathbb{N}* : Zwei Zufalls-Ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ und $Y = \{Y_i\}_{i \in \mathbb{N}}$ mit Wertebereich Σ^* heißen *uniform ununterscheidbar* (bzw. *nicht-uniform ununterscheidbar*), falls für jede pp-TM D (bzw. jeden polynomiell größenbeschränkten Schaltkreis D), jedes Polynom $p(\cdot)$ und alle hinreichend großen $n \in \mathbb{N}$ gilt:

$$|\Pr[D(X_n, 1^n) = 1] - \Pr[D(Y_n, 1^n) = 1]| < \frac{1}{p(n)}$$

3. *Indexmenge Σ^** : Zwei Zufalls-Ensembles $X = \{X_w\}_{w \in \Sigma^*}$ und $Y = \{Y_w\}_{w \in \Sigma^*}$ mit Wertebereich Σ^* heißen *uniform ununterscheidbar* (bzw. *nicht-uniform ununterscheidbar*), falls für jede pp-TM D (bzw. jeden polynomiell größenbeschränkten Schaltkreis D), jedes Polynom $p(\cdot)$ und alle hinreichend langen $w \in \Sigma^*$ gilt:

$$|\Pr[D(X_w, w) = 1] - \Pr[D(Y_w, w) = 1]| < \frac{1}{p(|w|)}$$

Dabei soll $|X_i|$ bzw. $|X_w|$ nur von i bzw. w abhängen. Die Differenz der betrachteten Akzeptanzwahrscheinlichkeiten nennen wir den Erfolg von D .

Ein Unterscheider bekommt also den jeweiligen Index n bzw. w als zusätzliche Information – hat aber selbst dann nur vernachlässigbaren Erfolg. Dabei ist der Fall der Indexmenge \mathbb{N} nur ein Spezialfall von Σ^* , wenn man eine natürliche Zahl n mit ihrer unären Kodierung 1^n identifiziert.

Diese Ununterscheidbarkeit durch effiziente uniforme oder nicht-uniforme Beobachter (engl. auch *computational indistinguishability*) ist schwächer als die *statistische Ununterscheidbarkeit*, bei der allein schon der statistische Unterschied nur vernachlässigbar groß ist:

Definition 1.4. *Zwei Zufalls-Ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ und $Y = \{Y_i\}_{i \in \mathbb{N}}$ heißen statistisch ununterscheidbar, falls die statistische Differenz $\Delta(n)$ vernachlässigbar ist:*

$$\Delta(n) := \frac{1}{2} \sum_{\alpha} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|$$

Analog für Indexmenge Σ^* .

Die stärkste Ununterscheidbarkeits-Forderung jedoch, ist die der *identischen Verteilung*. Für zwei identisch verteilte Zufalls-Ensembles ist in der Tat $\Delta(n) = 0$.

Kapitel 2

Pseudozufallsgeneratoren und Hitting Set Generatoren

In diesem Kapitel werden wir uns mit den grundlegenden Konzepten der Pseudozufallsgeneratoren befassen. Nach einer allgemeinen Betrachtung zur Verwendung von und dem Umgang mit Zufallsgrößen in Berechnungsmodellen liegt der Fokus dabei zunächst auf den sogenannten „kryptographischen“ Pseudozufallsgeneratoren und später auf den „nicht-kryptographischen“, die bei der Derandomisierung Verwendung finden. Es werden Implikationen hinsichtlich der Existenz und Leistungsfähigkeit der Generatoren aufgezeigt und bewiesen. Insbesondere auf Hitting Set Generatoren, die in späteren Kapiteln angewendet werden, wird gegen Ende dieses Kapitels eingegangen.

2.1 Zufall versus Determinismus

Wie reale Rechner auch, sind in den *klassischen* Berechnungsmodellen (Turingmaschinen, Schaltkreise, . . .) die betrachteten (theoretischen) Maschinen zunächst von *deterministischer* Natur: Eine Konfiguration des Gesamtsystems – sein *Zustand* – hat genau einen Nachfolgezustand, den das System durch den nächsten Zustandswechsel einnimmt. Dies entspricht auch dem zunächst intuitiven Algorithmen-Begriff. Eine theoretische Erweiterung dieses Modells aus den 60er und 70er Jahren ist der *Nichtdeterminismus*, der einer berechnenden Maschine bei manchen Schritten die Wahl lässt, welcher Nachfolgezustand eingenommen werden soll. Um zur Lösung zu gelangen, wählt sie „automatisch den richtigen Weg“ durch den nun entstandenen Berechnungsbaum. Dieses Modell ist zwar theoretisch interessant, lässt es sich doch auch interpretieren als das *Überprüfen* einer geratenen Lösung (anstelle des *Findens* der Lösung), bzw. gar als gewisse eingeschränkte Form von

Parallelismus. Für praktische Zwecke ist es allerdings wenig brauchbar – der Aufwand, deterministisch nach einem akzeptierenden Berechnungspfad zu suchen, explodiert.

In den 80er Jahren wurden jedoch Betrachtungen intensiviert, die ein ergänzendes Konzept in die Berechnungsmodelle einfügten: Die Benutzung von zufälligem Verhalten. Statt „automatisch den richtigen Weg“ durch den Baum ihrer möglichen Entscheidungen zu wählen, entscheidet die betrachtete Maschine zufällig, welches der Nachfolgezustand wird. Diese *probabilistischen Algorithmen* eröffnen sehr simple Lösungen für eine Reihe von bekannten Problemen (wie Primzahltests, Erfüllbarkeitsproblem, ...), auch wenn ihre Ausgabe nun eine Zufallsgröße ist und somit potentiell fehlerbehaftet. Dieses Modell ist im Gegensatz zum Nichtdeterminismus allerdings ohne großen Effizienzverlust auch real in Computern umsetzbar und daher besonders interessant. Der Begriff des Probabilismus hat inzwischen sogar Einzug in den intuitiven Begriff der „effizienten Berechnung“ gefunden.

Dabei entsteht nun aber ein ganz praktisches Problem, welches in Bereichen, die ohnehin bereits zufälliges Verhalten benötigen (z.B. die Schlüsselerzeugung in der Kryptographie, oder die Auswahl von Stichproben bei statistischen Simulationen) auch existiert: Die praktische Simulation von zufälligem Verhalten, also die Erzeugung von Zufallszahlen. Die Eigenschaften, die theoretisch von Zufallsgrößen gefordert werden – wie z.B. Gleichverteilung und Unabhängigkeit – sind praktisch so leicht nicht umzusetzen, sind doch reale Geräte nach wie vor deterministische Maschinen.

Basierend auf physikalischen Phänomenen in elektronischen Bauteilen, können „echte“ Zufallsgrößen generiert werden. Dies ist ein aufwändiger Prozess, und in der nötigen Dimension oft schlicht zu aufwändig. Aus diesem Grund gibt es Ansätze, „wenige echte“ Zufallsbits durch Transformationen in mehr Bits zu verwandeln, die immer noch „hinreichend zufällig aussehen“ und so für die betrachtete Anwendung noch immer „gut genug“, eben *pseudozufällig* sind. Abbildungen bzw. Algorithmen die dies leisten, heißen daher *Pseudozufallsgeneratoren*. Das erlaubt auch das Konzept der Derandomisierung: Zwar ist ein *Brute Force* Ansatz auch bei probabilistischen Algorithmen zu aufwändig (also das Durchprobieren aller möglichen Werte der verwendeten Zufallsgrößen), wenn jedoch der einfließende „echte“ Zufall nur gering ist, kann es praktikabel sein, ihn komplett durchzuprobieren – und damit den Algorithmus wieder deterministisch zu machen und insbesondere das Fehlerrisiko zu beseitigen.

Damit entsteht jedoch ein neues schwieriges Problem: Die Frage, „wie zufällig aussehend“ eine Bitfolge denn tatsächlich ist. Die Mitte der 60er Jahre unabhängig von Solomonoff, Kolmogorow und Chaitin entwickelte *Kolmogorow-Komplexität*, die einer Zeichenkette als Maß ihrer Zufälligkeit die

Länge des kürzesten sie produzierenden Programmes zuordnet, ist dafür ungeeignet – „ontologisch“ in ihrer Natur und ohnehin nicht berechenbar in den gängigen Berechnungsmodellen. Stattdessen wurden in der Statistik eine Vielzahl von Tests entwickelt, die pseudozufällige Größen von „echtem Zufall“ unterscheiden können sollen (z.B. χ^2 Test). Das Problem an diesem Ansatz ist, dass alle diese Tests eine Bitfolge zwar als zufällig klassifizieren könnten, sie jedoch durch einen neuen – vielleicht noch nicht entdeckten – Test sofort als pseudozufällig, und damit als künstlich, entlarvt würde. Ein solcher Ansatz ist in der grundlegenden Betrachtung der Kryptographie und auch der Komplexitätstheorie also ungeeignet, könnte doch gerade ein neuer Verschlüsselungsalgorithmus die Pseudozufallsbits in einer Weise gebrauchen, die diesem noch nicht entdeckten Test entspricht.

In ihren Anwendungen fällt allerdings auf, dass nur einige bestimmte Eigenschaften von Zufallsgrößen benötigt werden. Entscheidend ist nicht, wie sie entstanden sind, sondern nur, ob sie durch einen ihrerseits in einem Berechnungsmodell formulierten *Beobachter* von „zufällig“ erzeugten Eingaben unterschieden werden können. Ist dies nicht der Fall, so ist die Güte der pseudozufälligen Größen hinreichend für ihre Anwendung, und aus diesem Grunde wird dies die Idee hinter den späteren Definitionen sein.

Die Methode, „schwierige“ Probleme zur Erzeugung von Pseudozufall heranzuziehen, hat einen Effekt, der auch *Hardness versus Randomness Tradeoff* genannt wird: Für viele Probleme kann Zufälligkeit ausgetauscht werden durch die Benutzung gewisser komplexitätstheoretischer Objekte, die *hart* sind in dem Sinne, dass sie nur mit einem gewissen (beweisbaren) Mindestaufwand gelöst werden können. Je größer die vorausgesetzte Härte, desto mehr Pseudozufall kann generiert, also Zufall beseitigt werden, und umgekehrt.

Die gebräuchlichen Konzepte der Pseudozufallsgeneratoren und Hitting Set Generatoren wollen wir im folgenden nun etwas näher beleuchten.

2.2 Pseudozufallsgeneratoren

Ein *Pseudozufallsgenerator* (auch „PRNG“ nach engl. „pseudo random number generator“) ist ein deterministischer Algorithmus, welcher eine kurze Eingabe in eine längere Ausgabe verwandelt. Diese Ausgabe hat die Eigenschaft, von einem „effizienten“ Beobachter nur mit „geringem Erfolg“ von einer zufälligen Ausgabe unterschieden werden zu können, falls die Eingabe zufällig gewählt war. Aus diesem Grund nennt man die Ausgabe dann auch „pseudozufällig“. Was unter „geringem Erfolg“ und „Effizienz“ des Beobachters zu verstehen ist, hängt dabei vom jeweiligen Anwendungsfall ab und wird meist durch polynomielle Schranken bzw. die Leistungsfähigkeit von proba-

bilistischen polynomiellen Algorithmen oder von Schaltkreisen beschrieben. Wir beschränken uns hier auf die Definition mit Schaltkreisen:

Definition 2.1 (Pseudozufallsgenerator, PRNG). *Eine Funktion $G : \{0, 1\}^l \rightarrow \{0, 1\}^m$ mit $l < m$ ist ein (s, ϵ) -Pseudozufallsgenerator gegen Schaltkreise¹, falls für alle Schaltkreise $C : \{0, 1\}^m \rightarrow \{0, 1\}$ von höchstens Größe s gilt:*

$$|\Pr_{U_l}[C(G(U_l)) = 1] - \Pr_{U_m}[C(U_m) = 1]| < \epsilon$$

Dabei sind U_l und U_m Zufallsvariablen mit Werten gleichverteilt über $\{0, 1\}^l$ bzw. $\{0, 1\}^m$. Wir nennen die Differenz der Wahrscheinlichkeiten auch den Erfolg von C .

Die Wahrscheinlichkeit, dass ein „Unterscheider-Schaltkreis“ C auf den Ausgaben des Generators mit 1 antwortet, unterscheidet sich also nicht um mehr als ϵ von seiner Akzeptanzwahrscheinlichkeit bei zufälliger Eingabe.

Es sei angemerkt, dass Pseudozufallsgeneratoren gegen (uniforme) Algorithmen analog definiert werden, wobei dann als Beschränkung die Laufzeit an Stelle der Größe herangezogen wird.

Je nach Anwendungsfall (ob Kryptographie oder Komplexitätstheorie) haben sich zwei verschiedene Klassen von PRNGs etabliert: Die nach Blum, Micali und Yao benannten „BMY-Typ“ Generatoren in der Kryptographie, und die nach Nisan und Wigderson benannten „NW-Typ“ Generatoren in der Komplexitätstheorie. Mit beiden Typen beschäftigen wir uns nun näher.

2.2.1 „BMY-Typ“ Generatoren

In der Kryptographie sind besonders Pseudozufallsgeneratoren interessant, die durch Algorithmen berechnet werden können, deren Laufzeit polynomiell in ihrer Eingabelänge beschränkt ist. Diese Anforderung kommt durch die praktische Notwendigkeit in der Anwendung, diese Generatoren effizient ausführen zu können. So ist ein PRNG zentraler Teil von vielen Protokollen, die effiziente Implementierungen erfordern.

Eine zweite nötige Eigenschaft für kryptographische Anwendungen entsteht durch die Rolle des Beobachters: In diesem Szenario ist der Beobachter zumeist ein „Angreifer“, der die kryptographischen Eigenschaften zu brechen versucht. Im allgemeinen ist dieser unbekannt, es müssen ihm also bedeutende Rechenressourcen zugestanden werden, wenngleich diese trotzdem noch effizient sein müssen – seine Laufzeit ist also zwar polynomiell beschränkt, jedoch

¹In der Literatur findet man eine Vielzahl von größtenteils äquivalenten Definitionen. Aufgrund der später betrachteten Anwendungen halten wir uns hier nah an die Definitionen aus [BOV07].

kann das Polynom beliebig sein. Zusätzlich könnte er sich damit zufrieden geben, nur minimale (wenn auch nicht-vernachlässigbare) Erfolgsaussichten zu haben. Trotz dieser Eigenschaften des Angreifers soll die Ausgabe des PRNG für ihn von zufälliger Ausgabe nicht unterscheidbar sein.

Die folgende Definition macht beide Anforderungen explizit. Der Generator ist dort hinsichtlich der Laufzeit polynomiell in der Eingabelänge beschränkt. Der Angreifer, gegen den der Generator „resistent“ sein soll, ist jedoch ein beliebig polynomieller Schaltkreis mit beliebig polynomielltem Erfolg.

Definition 2.2 (BMY-Typ Generator). *Eine Funktion $G = \bigcup_m G_m : \{0, 1\}^l \rightarrow \{0, 1\}^m$ ist ein BMY-Typ Generator mit Seed-Länge $l = l(m)$, falls gilt:*

1. G ist berechenbar in Zeit $l^{O(1)}$, und
2. Für alle c und hinreichend große m ist G_m ein (m^c, m^{-c}) -Pseudozufallsgenerator.

Entscheidend für den BMY-Typ ist also die Eigenschaft, dass die Laufzeit des Generators zwar abhängig von seiner Eingabelänge durch ein festes Polynom bestimmt ist, seine Ausgabe jedoch resistent gegen Schaltkreise beliebig polynomieller Größe (bzw. gegen uniforme Algorithmen beliebig polynomieller Laufzeit) sein muss².

Es stellt sich nun die Frage, ob solche Generatoren existieren, bzw. unter welchen Annahmen man sie konstruieren kann. In der selben Arbeit [BM84], in der Blum und Micali die obige Definition von PRNGs durch Ununterscheidbarkeit von „echtem“ Zufall vorschlugen, entwickelten sie einen Generator auf Basis der diskreten Logarithmusfunktion (DL), unter der Bedingung, dass diese schwer zu berechnen sei. Sie nutzen dafür die Eigenschaft, dass selbst das höchstwertige Bit von DL so schwer zu berechnen ist wie DL selbst.

Andrew C. Yao gab in [Yao82] eine andere Definition von Pseudozufälligkeit durch Unvorhersagbarkeit des Folgebits bei beliebig bekanntem Präfix der Generatorausgabe an. Mit dieser äquivalenten Formulierung konnte er zeigen, dass sich BMY-Generatoren aus beliebigen Einwegpermutationen konstruieren lassen. Zu einer Einwegpermutation $f(x)$ kann man hierzu eine Einwegpermutation $g(x')$ mit einem zugehörigen sogenannten *Hard-Core Prädikat* $h_g(x')$ konstruieren. Dies ist ein Prädikat, welches zwar aus x' , jedoch nicht aus seinem Bild $g(x')$ effizient berechnet werden kann. (Später in

²Wir bemerken folgenden Zusammenhang zum Begriff der Ununterscheidbarkeit aus Definition 1.3: Für die Ausgaben eines BMY-Typ Generators darf es keinen effizienten Unterscheider von zufälligen Werten geben. Damit ist also gerade $\{G(U_{l(n)})\}_{n \in \mathbb{N}}$ nicht-uniform ununterscheidbar von $\{U_n\}_{n \in \mathbb{N}}$.

Abschnitt 3.2 werden wir im Zusammenhang mit nicht-interaktiven Bit Commitment Protokollen ein solches konkretes Hard-Core Prädikat konstruieren.) Die Verkettung von $g(x')$ und $h_g(x')$ liefert dann einen Pseudozufallsgenerator, unter der Voraussetzung dass f Einwegpermutation ist.

Schließlich konnten Håstad et al. in [HILL99] dieses Resultat verbessern, indem sie zeigten, dass eine beliebige Einwegfunktion ausreicht. Eine solche ist jedoch auch notwendig, also fassen wir dieses grundlegende Resultat im folgenden Theorem zusammen:

Theorem 2.3 ([HILL99]). *Pseudozufallsgeneratoren vom BMY-Typ gegen Schaltkreise (bzw. uniforme Algorithmen) gibt es genau dann, wenn es Einwegfunktionen gegen Schaltkreise (bzw. uniforme Algorithmen) gibt.*

Die grundsätzliche Annahme für die Existenz von „kryptographischen“ Pseudozufallsgeneratoren (d.h. vom BMY-Typ) ist also die Existenz von Einwegfunktionen. Diese Annahme ist in der Kryptographie durchaus plausibel, denn Einwegfunktionen sind ohnehin eine vielfach eingesetzte kryptographische Primitive. Jedoch ist die Annahme recht stark (sie impliziert u.a. $P \neq NP$) und wird auch als eine „kryptographische Annahme“ bezeichnet. In der Komplexitätstheorie werden Pseudozufallsgeneratoren auch für schwächere Annahmen benötigt, müssen aber oft auch keine derart starken Eigenschaften haben: Die „NW-Typ“ Generatoren.

2.2.2 „NW-Typ“ Generatoren

In ihrer einflussreichen Arbeit [NW94] machen Nisan und Wigderson die Beobachtung, dass die Eigenschaften der BMY-Generatoren viel stärker sind, als für komplexitätstheoretische Betrachtungen (wie Derandomisierung) nötig. So ist eine polynomielle Laufzeit (abhängig von der *Eingabe*) zwar für kryptographische Anwendungen wichtig, jedoch nicht für die Simulation von randomisierten Algorithmen. Der entscheidende Parameter ist hier die Ausgabelänge, und von besonderem Interesse gerade die Generatoren, deren Eingabelänge nur logarithmisch in der Ausgabelänge ist (deren Laufzeit daher dann allerdings auch exponentiell in der Eingabelänge sein muss). Daher wird die Bedingung an die Laufzeit (nun polynomiell in der Ausgabelänge) von der an die „Zufälligkeit“ der Ausgabe abgekoppelt, gemäß folgender Definition:

Definition 2.4 (NW-Typ Generator). *Eine Funktion $G = \bigcup_m G_m : \{0, 1\}^l \rightarrow \{0, 1\}^m$ ist ein NW-Typ Generator mit Seed-Länge $l = l(m)$, falls gilt:*

1. G ist berechenbar in Zeit $m^{O(1)}$, und
2. Für hinreichend große m ist G_m ein (m^2, m^{-2}) -Pseudozufallsgenerator.

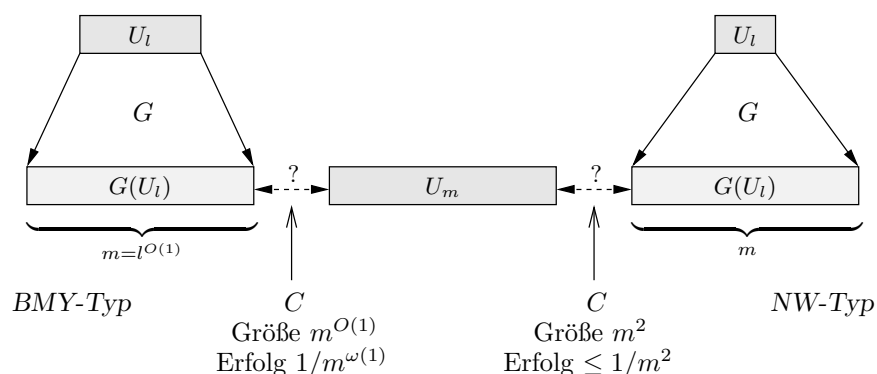


Abbildung 2.1: Gegenüberstellung von BMY-Typ und NW-Typ Pseudozufallsgeneratoren.

Im direkten Vergleich zum kryptographischen BMY-Typ Generator aus Definition 2.2 (vgl. auch Abbildung 2.1) fällt also auf, dass ein NW-Typ Generator eine in der Eingabe-Größe exponentielle Laufzeit haben darf und der „getäuschte“ Beobachter durch ein festes Polynom beschränkt ist. (Die Beschränkung auf $p(m) = m^2$ ist via Padding äquivalent zu jedem anderen festen mindestens linearen Polynom.) Die NW-Typ Generatoren sind somit in gewissem Sinne schwächer, also weniger effizient und weniger sicher, als die BMY-Typ Generatoren. Wie wir gleich sehen werden, reichen zu ihrer Existenz jedoch auch schwächere Annahmen aus.

Zunächst betrachten wir jedoch ein einfaches Resultat, nämlich wie mit Hilfe eines NW-Typ Generators randomisierte Algorithmen mit zweiseitigem Fehler direkt derandomisiert werden können:

Theorem 2.5. Falls es einen NW-Typ Generator G mit Seed-Länge l gibt, so gilt:

$$\text{BPP} \subseteq \text{DTIME} \left(2^{l(n^{O(1)})} \cdot \text{time}_G(l(n^{O(1)})) \right)$$

Beweis. Sei $L \in \text{BPP}$ und A eine pp-TM mit Fehlerschranken von $1/3$ und $L = L(A)$. Sei p ein Polynom, so dass A auf Eingaben der Länge n höchstens $p(n)$ Zufallsbits benötigt. Ein deterministischer Algorithmus B wird nun wie folgt konstruiert:

- Auf Eingabe x setze $k := l(p(|x|))$.
- Für alle $s \in \{0, 1\}^k$ berechne $\tilde{r} := G(s)$ und simuliere $A(x, \tilde{r})$
- Die Ausgabe ist der Mehrheitsentscheid über alle Ausgaben von A .

Falls es (unendlich viele) x gibt mit $B(x) \neq \chi_L(x)$, der Algorithmus B also eine falsche Ausgabe erzeugt, dann unterscheidet sich für jedes dieser x

die Mehrheit der $A(x, \tilde{r})$ mit pseudozufälligem \tilde{r} deutlich von der Mehrheit der $A(x, r)$ mit zufälligem r . Mindestens die Hälfte der $s \in \{0, 1\}^k$ erzeugen nämlich eine falsche Ausgabe von A (denn sie dominieren den Mehrheitsentscheid), aber höchstens ein Drittel der $r \in \{0, 1\}^{p(|x|)}$ (nach Voraussetzung von A).

$$|\Pr_{U_k}[A(x, G(U_k)) = 1] - \Pr_{U_{p(|x|)}}[A(x, U_{p(|x|)}) = 1]| \geq \frac{1}{6}$$

Damit kann aber $A(x, \cdot)$ benutzt werden, um $G(U_k)$ von $U_{p(|x|)}$ zu unterscheiden: Aus $A(x, \cdot)$ mit Laufzeit $p(|x|)$ kann ein Schaltkreis C der Größe $O(p(|x|)^2)$ konstruiert werden, der dies mit Wahrscheinlichkeit von mindestens $1/6$ leistet. Dann ist jedoch G kein PRNG (vom NW-Typ).

Offenbar hat B auch die angegebene Laufzeit, denn für jeden der 2^k Seeds s wird G einmal mit Eingabe s aufgerufen. \square

Nun ist klar, dass PRNGs mit (mindestens) exponentieller Expansion besonders interessant sind: Sie derandomisieren BPP sofort zu P. In diesem Fall ist nämlich die Anzahl der möglichen Seeds polynomiell in der Anzahl der zu erzeugenden Zufallsbits beschränkt. Da die Laufzeit des Generators ohnehin polynomiell in der Ausgabelänge beschränkt ist, folgt:

Korollar 2.6. *Falls es einen NW-Typ Generator G mit Seed-Länge $l = O(\log m)$ gibt, so gilt $\text{BPP} = \text{P}$.*

Nachdem wir nun also gesehen haben, welche Folgen die Existenz von „schwächeren“ Generatoren des NW-Typs hat, stellt sich die Frage, unter welchen Bedingungen sie denn konstruiert werden können, insbesondere welche mit maximaler Expansion. Bereits in [NW94] besprechen Nisan und Wigderson eine Konstruktion basierend auf der Existenz einer Sprache $L \in \text{EXP}$, die hart *im Durchschnitt* bezüglich Schaltkreisen ist. Im Laufe der Zeit wurde diese Voraussetzung durch verschiedene Arbeiten weiter abgeschwächt und schließlich konnten Impagliazzo und Wigderson in [IW97] zeigen, dass eine Sprache $L \in \text{E}$ mit Schaltkreiskomplexität $S(L) = 2^{\Omega(n)}$, also *im Worst Case*, hinreichend ist:

Theorem 2.7 ([IW97]). *Falls es eine Sprache $L \in \text{E}$ mit $S(L) = 2^{\Omega(n)}$ gibt, so existiert ein Pseudozufallsgenerator vom NW-Typ gegen Schaltkreise mit Seed-Länge $l(m) = O(\log m)$. Insbesondere gilt dann $\text{P} = \text{BPP}$.*

Der Beweis hierfür ist sehr umfangreich und technisch. Das Resultat zeigt jedoch, dass nicht beides gelten kann: Dass es Probleme in E gibt, die exponentielle Schaltkreisgröße benötigen *und* dass Randomisierung die Ausdrucksstärke von Polynomialzeitalgorithmen erhöht. Ein Beweis der einen

Aussage würde die andere falsifizieren. Dies ist bemerkenswert im Zusammenhang mit dem *Hardness versus Randomness Tradeoff* aus der Einleitung.

Wir wenden uns nun einer etwas spezielleren Form der NW-Typ Generatoren zu, die für den Rest der Arbeit von besonderer Bedeutung ist: den Hitting Set Generatoren.

2.3 Hitting Set Generatoren

Die grundlegende Beobachtung ist die unterschiedliche Anwendung von Pseudozufallsgeneratoren: Sie werden in der Kryptographie benutzt, um aus *einem* zufällig gewählten Seed genau *eine* Ausgabe zu erhalten, die, betrachtet als einzelnes Element des Bildbereiches $\{0, 1\}^m$, pseudozufällig ist. Wie wir bereits gesehen haben, ist die Verwendung zur Derandomisierung eine andere: Hier interessiert das gesamte Bild des Generators *als Menge*, die als Teilmenge des Bildbereiches $\{0, 1\}^m$ auch „pseudozufällig“ ist. Das heißt, dass mit ihr die Akzeptanzwahrscheinlichkeiten beliebiger Schaltkreise approximiert werden können.

Zur Derandomisierung von Algorithmen mit beidseitigem Fehler ist dies notwendig, dort soll bestimmt werden, ob die Akzeptanzwahrscheinlichkeit sich im Intervall $[0, 1/3]$ oder im Intervall $[2/3, 1]$ bewegt. Bei der Derandomisierung von Algorithmen mit einseitigem Fehler ist dies jedoch gar nicht notwendig: Interessant ist hier ja nur, ob es überhaupt einen Zufallsstring gibt, der zu einer akzeptierenden Rechnung führt.

Diese Abschwächung der Anforderungen wird als *Hitting Set* formal formuliert: Ein Hitting Set ist eine Menge von Strings, die für *jeden* Schaltkreis, der mindestens einen gewissen Anteil seiner Eingaben akzeptiert, *mindestens eine* akzeptierende Eingabe enthält. Damit wird also jeder Schaltkreis mit einer gewissen Mindest-Akzeptanzwahrscheinlichkeit „getroffen“. Ein *Hitting Set Generator* ist ein Algorithmus, der ein solches Hitting Set erzeugt:

Definition 2.8 (Hitting Set Generator, HSG). *Ein deterministischer Algorithmus $H(1^m, 1^s)$, der eine Menge $M \subseteq \{0, 1\}^m$ ausgibt, ist ein ϵ -Hitting Set Generator gegen Schaltkreise, falls für alle Schaltkreise $C : \{0, 1\}^m \rightarrow \{0, 1\}$ von höchstens Größe s gilt:*

$$\Pr_{U_m}[C(U_m) = 1] \geq \epsilon \implies \exists y \in H(1^m, 1^s) : C(y) = 1$$

H ist ein ϵ -Hitting Set Generator gegen uniforme Algorithmen, falls für alle uniformen Algorithmen $A : \{0, 1\}^ \rightarrow \{0, 1\}$, alle A in der Laufzeit beschränkenden Funktionen $s(\cdot)$, und für alle hinreichend großen m gilt:*

$$\Pr_{U_m}[A(U_m) = 1] \geq \epsilon \implies \exists y \in H(1^m, 1^{s(m)}) : A(y) = 1$$

H ist effizient, falls seine Laufzeit polynomiell in m und s beschränkt ist.

Analog definiert man Hitting Set Generatoren gegen nichtdeterministische und co-nichtdeterministische Schaltkreise bzw. uniforme Algorithmen.

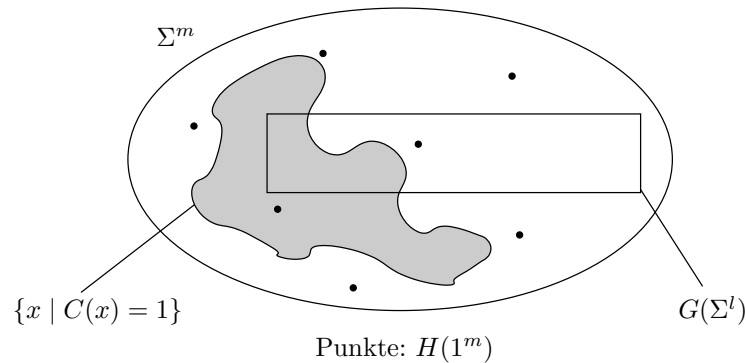


Abbildung 2.2: Gegenüberstellung von Ausgaben eines Pseudozufallsgenerators G und eines Hitting Set Generators H : Die Ausgaben von G approximieren den Anteil der vom Schaltkreis C akzeptierten Elemente in Σ^m , also dessen Akzeptanzwahrscheinlichkeit. Die Ausgaben von H müssen seine Akzeptanzmenge „treffen“.

2.3.1 Derandomisierung mittels Hitting Set Generatoren

Ein effizienter Hitting Set Generator H kann nun also analog Theorem 2.5 direkt zur Derandomisierung von RP verwendet werden: Bei einer RP-Sprache L enthält die Zeugen-Menge $R(x)$ eines jeden $x \in L$ laut Definition mindestens die Hälfte aller möglichen Strings $\{0, 1\}^{p(|x|)}$. Mindestens ein Element davon muss in der Ausgabe von H liegen, denn sonst wäre ein L auf x entscheidender Schaltkreis nicht von H erfasst. Daher müssen nur alle (polynomiell vielen) Elemente der Ausgabe von H als mögliche Zeugen für x untersucht werden, um jedes x zu entscheiden: $x \in L$ gilt genau dann, wenn für ein $r \in H$ der ursprüngliche Algorithmus akzeptiert.

Theorem 2.9. *Falls es effiziente 1/2-Hitting Set Generatoren gegen Schaltkreise gibt, so gilt $\text{RP} = \text{P}$.*

Hitting Set Generatoren sind offenbar genau so definiert, dass sie sich gut zur Derandomisierung von Algorithmen mit einseitigem Fehler eignen (also z.B. für $\text{RP} = \text{P}$), wobei Pseudozufallsgeneratoren vom NW-Typ für den zweiseitigen Fall notwendig scheinen (also z.B. für $\text{BPP} = \text{P}$). In einer Reihe von Arbeiten ([ACR98], [GVW00], ...) wurde allerdings gezeigt, dass

auch Hitting Set Generatoren zum Derandomisieren von zum Beispiel BPP Verwendung finden können. Wir werden gleich sehen, dass die Existenz von effizienten Hitting Set Generatoren aber sogar äquivalent zur Existenz von NW-Typ Pseudozufallsgeneratoren mit maximaler Expansion ist.

2.3.2 Alternative Definitionen

In obiger Definition 2.8 der Hitting Set Generatoren wurde gleich dem Umstand Rechnung getragen, dass bei der Derandomisierung das gesamte Bild eines Generators interessiert, also die Menge aller generierten Strings. Im allgemeinen sind Generatoren aber Algorithmen, die zu einer Eingabe (zum Beispiel einem Index) ein Element (also einen String) „generieren“, also ausgeben. Einige Arbeiten verwenden aus Konsistenzgründen daher eine äquivalente Definition von Hitting Set Generatoren, bei der $H'(1^m, 1^s)$ eine Funktion ist, die zum Beispiel von $\{0, 1\}^l$ mit $l = \log |H(1^m, 1^s)|$ in $\{0, 1\}^m$ abbildet, wobei das Argument einen Index der Ausgabe im Hitting Set bezeichnet. Diese Definition lehnt sich also eher an die Funktionsweise der Pseudozufallsgeneratoren an.

Eine weitere äquivalente Definition übergibt H nicht zwei Parameter m und s für die Größe der auszugebenden Strings bzw. der zu „täuschenden“ Schaltkreise, sondern nur einen Parameter n für beides:

Definition 2.10 (HSG, alternative Def.). *Ein deterministischer Algorithmus $H(1^n)$, der eine Menge $M \subseteq \{0, 1\}^n$ ausgibt, ist ein ϵ -Hitting Set Generator gegen Schaltkreise, falls für alle Schaltkreise $C : \{0, 1\}^n \rightarrow \{0, 1\}$ von höchstens Größe n gilt:*

$$\Pr_{U_n}[C(U_n) = 1] \geq \epsilon \implies \exists y \in H(1^n) : C(y) = 1$$

Diese Definition eignet sich für manche Betrachtungen besser und ist äquivalent zu Definition 2.8:

Lemma 2.11. *Für jedes ϵ gilt: Genau dann, wenn es effiziente ϵ -Hitting Set Generatoren gemäß Definition 2.8 gibt, gibt es auch welche gemäß Definition 2.10.*

Beweis. Die eine Richtung ist klar: Falls es einen ϵ -HSG $H(1^m, 1^s)$ gemäß Definition 2.8 gibt, so muss ein Algorithmus $H'(1^n)$ diesen nur mit $m = s = n$ aufrufen und ist damit ein ϵ -HSG gemäß Definition 2.10.

Sei umgekehrt $H(1^n)$ ein ϵ -HSG gemäß Definition 2.10. Ein Algorithmus $H'(1^m, 1^s)$ kann dann zunächst $H(1^n)$ mit $n = \max(m, s)$ aufrufen und dann sämtliche m -Präfixe der erhaltenen n -Strings ausgeben. Sei nun C ein Schaltkreis der Größe s mit m Eingabe-Bits, der mindestens einen ϵ -Anteil seiner

möglichen Eingaben akzeptiert. Falls $m = n$, dann ist also $s \leq n$ und es muss in dieser Menge damit ein Element geben, welches C akzeptiert. Falls $m < n$, so kann C als ein Schaltkreis der Größe $s = n$ angesehen werden, der n Eingabe-Bits hat, der jedoch alle Eingabe-Bits außer den ersten m ignoriert. Insbesondere akzeptiert C auch unter dieser Interpretation mindestens einen ϵ -Anteil seiner möglichen Eingaben. Somit muss auch hier C mindestens einen String der Ausgabe von H' akzeptieren. \square

Nach den Vorbetrachtungen zur Verwendung und verschiedenen Definitionen von Hitting Set Generatoren beschäftigen wir uns nun damit, unter welchen Voraussetzungen sie existieren.

2.3.3 Existenz von HSG

Es stellt sich heraus, dass die Existenz von Hitting Set Generatoren äquivalent zur Existenz von Pseudozufallsgeneratoren vom NW-Typ ist:

Theorem 2.12. *Die folgenden Aussagen sind äquivalent:*

1. *Es gibt Pseudozufallsgeneratoren vom NW-Typ mit Seed-Länge $l(m) = O(\log m)$.*
2. *Es gibt effiziente ϵ -Hitting Set Generatoren für jedes $\epsilon > 0$.*
3. *Es gibt eine Sprache $L \in \mathbf{E}$ mit $S(L) = 2^{\Omega(n)}$.*

Beweis. Wir zeigen die Ring-Implikation $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$:

$1 \Rightarrow 2$: Das Bild eines PRNG G vom NW-Typ ist bereits ein Hitting Set. Wir können daher aus einem gegebenem PRNG G mit Seed-Länge $l(m) = O(\log m)$ folgenden Hitting Set Generator $H(1^n)$ gemäß Definition 2.10 konstruieren: Bei Eingabe 1^n rufe $G(s)$ für alle Seeds $s \in \{0, 1\}^{l(n)}$ auf. Die erhaltene Menge wird ausgegeben.

Die Laufzeit von H ist polynomiell in n , da das Generieren des Bildes von G nur polynomiell in der Länge der Bilder von G ist, analog den Betrachtungen zum Derandomisieren von BPP im Beweis von Theorem 2.5. Die Ausgabe ist das gewünschte Hitting Set: Angenommen, es gibt einen Schaltkreis C der Größe n , der mindestens einen ϵ -Anteil seiner Eingaben akzeptiert, aber kein Element der Ausgabe von H . Damit gilt:

$$\Pr_{U_n}[C(U_n) = 1] \geq \epsilon \quad \wedge \quad \forall x \in G(\{0, 1\}^{l(n)}) : C(x) = 0$$

Also hätten wir:

$$\left| \underbrace{\Pr_{U_{l(n)}}[C(G(U_{l(n)})) = 1]}_{=0} - \underbrace{\Pr_{U_n}[C(U_n) = 1]}_{\geq \epsilon} \right| \geq \epsilon$$

Mit hinreichend großem n ist aber $\epsilon > 1/n^2$, also ein Widerspruch zur PRNG-Eigenschaft von G .

2 \Rightarrow 3: (Nach [ISW99].) Sei $H(1^n)$ ein 1/2-Hitting Set Generator (gegen Schaltkreise der Größe n). Die Laufzeit von $H(1^n)$ ist in $n^{O(1)}$, es gibt also $c \in \mathbb{N}$ sodass insbesondere die Größe der von $H(1^n)$ ausgegebenen Menge durch n^c beschränkt ist. Wir konstruieren eine Sprache $L \in \mathbf{E}$, indem wir einen Algorithmus angeben, der L akzeptiert:

- Eingabe: x
- Sei $k = |x|$.
- Setze $n := 2^{(k-1)/c}$.
- Rufe $H(1^n)$, sei X_n dessen Ausgabe.
- Akzeptiere genau dann, wenn kein k -Präfix von X_n mit x übereinstimmt.

Es ist also $L \cap \{0, 1\}^k$ die Menge aller y , die *nicht* k -Präfix eines Wortes in $H(1^n)$ sind für $n = 2^{(k-1)/c}$.

Zunächst stellen wir fest, dass damit $L \in \mathbf{E}$ gilt: Die Eingabe von H hat eine Länge von $2^{O(k)}$. Durch die polynomielle Laufzeitschranke von H und die obige Definition von c ist die Größe der Menge X_n durch 2^{k-1} beschränkt, und von jedem dieser Elemente wird das k -Präfix getestet. Damit ist die Laufzeit insgesamt in $2^{O(k)}$.

Nun nehmen wir an, eine Schaltkreisfamilie $C = \{C_k\}$ könnte L erkennen mit $\text{size}(C_k) \leq 2^{(k-1)/c}$. Sei k fix und wie im obigen Algorithmus $n = 2^{(k-1)/c}$ die Größe des Schaltkreises C_k mit k Eingabebits, der x mit $|x| = k$ entscheiden soll. C_k akzeptiert genau die $y \in \{0, 1\}^k$, die *nicht* k -Präfix im Hitting Set X_n sind, denn das ist gerade $L \cap \{0, 1\}^k$. Da $|X_n| \leq 2^{k-1}$ muss C_k also die Hälfte der Eingaben akzeptieren. Nun kann man C_k auch so (vielleicht als C'_k) interpretieren, dass er n Eingabebits hat, davon jedoch nur die ersten k betrachtet und die restlichen ignoriert. Damit wäre C'_k ein Schaltkreis mit n Eingabebits von höchstens Größe n , der mindestens die Hälfte aller möglichen Eingaben akzeptiert. Nun ist X_n ein Hitting Set für genau diese Art Schaltkreis,

also müsste C'_k ein Element aus X_n akzeptieren, was aber nach Konstruktion gerade nicht der Fall ist.

Eine Schaltkreisfamilie dieser Beschränkung kann es folglich nicht geben, also gilt wie behauptet $S(L) = 2^{\Omega(n)}$.

3 \Rightarrow 1: Dies ist gerade das Resultat von [IW97] aus Theorem 2.7.

□

2.3.4 HSG gegen nichtdeterministische Beobachter

Unsere bisherigen Betrachtungen von Hitting Set Generatoren gemäß Definition 2.8 (bzw. Pseudozufallsgeneratoren allgemein) war auf deterministische „Beobachter“ bezogen, die der entsprechende Generator in der Lage war zu „täuschen“. Wie bei der Definition bereits erwähnt, können die Definitionen analog gegen *nichtdeterministische* Beobachter (also uniforme Algorithmen bzw. nicht-uniforme Schaltkreise) aufgestellt werden. Diese Generatoren sind dann mächtigere Objekte, denn selbst die zusätzliche Hilfe des Nichtdeterminismus darf es dem Beobachter nicht möglich machen, den Generator zu „überführen“.

Da die in den späteren Kapiteln verwendeten Hitting Set Generatoren solche gegen nichtdeterministische Beobachter sein müssen, seien hier zwei zu Theorem 2.7 ähnliche Resultate aufgeführt, die Aussagen darüber treffen, wann diese Generatoren existieren. Zunächst der nicht-uniforme Fall:

Theorem 2.13 ([MV99]). *Falls es eine Sprache $L \in \mathbf{E}$ mit $S_N(f) = 2^{\Omega(n)}$ gibt, so gibt es effiziente $1/2$ -Hitting Set Generatoren gegen co-nichtdeterministische Schaltkreise.*

Die Voraussetzung ist also analog der von 2.7, bezieht sich jedoch auf die Größe von *nichtdeterministischen* Schaltkreisen.

Ein grundlegendes Interesse an Hitting Set Generatoren gegen nichtdeterministische Schaltkreise ergibt sich in der Komplexitätstheorie durch Untersuchungen der Klasse \mathbf{AM} , den sogenannten *Arthur-Merlin-Games*, die zuerst von László Babai in [Bab85] eingeführt wurden. Informell kann \mathbf{AM} beschrieben werden als die Klasse derjenigen Sprachen L , die interaktive Beweissysteme der folgenden Form besitzen: Bei einer gemeinsamen Eingabe x tauschen die beiden Teilnehmer, *Arthur* und *Merlin*, abwechselnd Nachrichten aus, beginnend bei Arthur. Diese Nachrichten sind polynomiell in $|x|$ beschränkt und dabei gilt, dass die Nachrichten von Arthur gleichverteilt zufällig gewählte Zeichenketten sind. Arthur ist dabei also eine probabilistische Turingmaschine, Merlin hingegen werden beliebige Rechenressourcen

zugestanden. Nach einer gewissen Anzahl von Runden wird die Menge der gegenseitig gesendeten Nachrichten von einer (deterministischen) P-Maschine V ausgewertet. Es muss dann gelten, dass im Fall $x \in L$ die Akzeptanzwahrscheinlichkeit von V mindestens $2/3$ beträgt, im Fall $x \notin L$ von höchstens $1/3$ (unabhängig davon, welche Nachrichten Merlin dabei sendet).

Die Menge der Sprachen mit AM-Beweissystemen mit $k = k(|x|)$ Runden (d.h. k ausgetauschten Nachrichten) werden mit $\text{AM}[k]$ bezeichnet. Unterschieden werden die Arthur-Merlin-Games mit polynomieller Anzahl von Runden, $\text{AM}[\text{poly}]$, von denen mit konstanter Anzahl, $\text{AM}[\text{const}] = \bigcup_{k \in \mathbb{N}} \text{AM}[k]$. Dabei gilt einerseits $\text{AM}[\text{poly}] = \text{IP}[\text{poly}]$, d.h. bei polynomieller Runden-Anzahl ist obige Beschreibung keine grundsätzliche Einschränkung allgemeiner interaktiver Beweissysteme. Andererseits gilt, dass die $\text{AM}[\text{const}]$ Hierarchie auf der 2. Stufe kollabiert, d.h. $\forall k : \text{AM}[k] \subseteq \text{AM}[2]$ ([Bab85]). Aus diesem Grund bezeichnet man mit AM (ohne Zusatz) zwar die Menge der Sprachen mit $\text{AM}[\text{const}]$ -Beweisen, jedoch ist dies gerade $\text{AM}[2]$. (Analog der Schreibweise aus der Erstveröffentlichung [Bab85] für die Klasse AM kann man dies auch lesen als „Arthur sendet die erste Nachricht, Merlin die zweite.“) Die zusätzliche Eigenschaft, dass der Fehler bei AM einseitig gemacht werden kann ([FGM⁺89]), führt schließlich zur folgenden vergleichsweise simplen Definition:

Definition 2.14. *Die Klasse der Arthur-Merlin-Games AM besteht aus allen Sprachen L für die es eine probabilistische, nichtdeterministische und in der Länge ihrer Eingabe x polynomiell zeitbeschränkte Turingmaschine $M(x, r, y)$ gibt, sodass für alle $x \in \Sigma^*$ gilt:*

$$\Pr_r[\exists y : M(x, r, y) = 1] \begin{cases} = 1, & \text{falls } x \in L \\ \leq 1/2, & \text{falls } x \notin L \end{cases}$$

In gewisser Weise ist AM also eine „randomisierte“ Variante von NP , so wie BPP (bzw. RP) als das randomisierte P gesehen werden kann. Analog der Derandomisierung von RP mit Hitting Set Generatoren gegen (deterministische) Schaltkreise (vgl. Theorem 2.9) kann AM nun mit einem Hitting Set Generator gegen co-nichtdeterministische Schaltkreise derandomisiert werden:

Theorem 2.15. *Falls es einen 1/2-Hitting Set Generator gegen co-nichtdeterministische Schaltkreise gibt, so ist $\text{AM} = \text{NP}$.*

Beweis. Klar ist zunächst, dass $\text{NP} \subseteq \text{AM}$, denn eine $L \in \text{NP}$ akzeptierende NTM M benötigt einfach keine Zufallsbits, die Akzeptanzwahrscheinlichkeit ist also 0 oder 1.

Für die Inklusion $\text{AM} \subseteq \text{NP}$ sei $L \in \text{AM}$ und M die TM aus der Definition. Betrachtet man die jeweiligen Gegenwahrscheinlichkeiten (also die Wahrscheinlichkeiten für das komplementäre Ereignis), muss also gelten:

$$\Pr_r[\forall y : M(x, r, y) = 0] \begin{cases} = 0, & \text{falls } x \in L \\ \geq 1/2, & \text{falls } x \notin L \end{cases}$$

Um die Frage ohne Zufallsbits beantworten zu können, muss also untersucht werden, ob es r gibt mit $\forall y : M(x, r, y) = 0$. Dazu kann ein Hitting Set Generator verwendet werden, denn zu einer Eingabe x kann ein co-nichtdeterministischer Schaltkreis C_x polynomieller Größe konstruiert werden mit:

$$C_x(r) = 1 \iff \forall y : M(x, r, y) = 0$$

(Die co-nichtdeterministischen Entscheidungen werden implizit getroffen, erscheinen also nicht als extra Eingang.)

Damit akzeptiert C_x für $x \notin L$ mehr als die Hälfte der r . Seien $p(\cdot)$ und $q(\cdot)$ polynomielle Größen-Schranken für r und C_x , dann muss die Ausgabe H eines Hitting Set Generators gegen co-nichtdeterministische Schaltkreise $H(1^{p(|x|)}, 1^{q(|x|)})$ also in diesem Fall ein r enthalten wofür C_x akzeptiert. Mithin ist zur derandomisierten (jedoch nichtdeterministischen) Simulation von M nur der Test auf allen r aus H erforderlich, um zu entscheiden, ob $x \in L$ gilt, denn genau dann wird für keines dieser r die Bedingung $\forall y : M(x, r, y) = 0$ gelten, d.h.:

$$x \in L \iff \forall r \in H : \exists y : M(x, r, y) = 1$$

Wir geben noch eine NP-Maschine $M'(x)$ an, die dieses leistet:

- Eingabe: x
- Simuliere $H := H(1^{p(|x|)}, 1^{q(|x|)})$
- Für alle $r \in H$:
 - Wähle y nichtdeterministisch
 - Simuliere $M(x, r, y)$
- Akzeptiere x genau dann, wenn es für jedes $r \in H$ ein y gab mit $M(x, r, y) = 1$.

□

Wir haben nun also gesehen, wie auch HSG gegen co-nichtdeterministische Schaltkreise zur Derandomisierung verwendet werden können. Auch das *uniforme* Pendant werden wir später benutzen, also geben wir schließlich noch das zu Theorem 2.13 analoge Resultat für den uniformen Fall an. Hierbei bezeichnet $\text{AMTIME}(t(n))$ die Klasse der Sprachen, bei denen die Maschine aus Definition 2.14 auf Eingaben der Länge n in der Laufzeit durch $t(n)$ beschränkt ist, und $[\text{i.o.-AMTIME}](t(n))$ die Sprachen, bei denen die Bedingung aus Definition 2.14 für unendlich viele n gilt.

Theorem 2.16 ([GSTS04]). *Falls $E \not\subseteq [\text{i.o.-AMTIME}](2^{\delta n})$ für ein $\delta > 0$, dann gibt es effiziente 1/2-Hitting Set Generatoren gegen co-nichtdeterministische Algorithmen.*

Dabei ist anzumerken dass diese Voraussetzung hier von der Voraussetzung „ $\exists L \in E$ mit $S_N(f) = 2^{\Omega(n)}$ “ aus Theorem 2.13 impliziert wird, also schwächer ist.

Kapitel 3

Bit Commitment

Ein praktisch häufig auftretendes Problem ist, dass in einer Interaktion zwischen zwei Parteien die eine Partei S zu einem späteren Zeitpunkt t_2 die andere Partei R davon überzeugen möchte, sich bereits zu einem früheren Zeitpunkt $t_1 < t_2$ auf einen gewissen Wert b *festgelegt* zu haben. Dies ist einfach, wenn S den Wert b zum Zeitpunkt t_1 bereits veröffentlicht. Der schwierige und in der Kryptographie interessante Fall ist, wenn R den Wert b erst zum Zeitpunkt t_2 erfahren soll.

In seinem Standardwerk [Sch95] beschreibt Bruce Schneier die Situation mit dem folgenden, klassischen Beispiel:

Alice ist Börsen-Spezialistin und möchte den Investor Bob davon überzeugen, sie für Investment-Beratungen zu engagieren.

Bob: Wähle zur Demonstration fünf Aktien für mich aus. Wenn sie sich im nächsten Monat alle als Gewinner herausstellen, hast du den Job.

Alice: Wenn ich für dich fünf Aktien auswähle, könntest du in diese investieren, ohne mich für diesen Tipp zu bezahlen. Schauen wir uns doch einfach meine Empfehlung vom vergangenen Monat an, dann wirst du sehen, dass sie gut war.

Bob: Woher soll ich wissen, dass du dir diese Empfehlung nicht im Nachhinein überlegt hast, nachdem du die Performance der Aktien kanntest? Wenn du mir stattdessen jetzt eine Empfehlung für den nächsten Monat gibst, dann kann ich mir sicher sein, dass du nicht schummelst. Ich verspreche, in deine neue Empfehlung nicht zu investieren, bevor ich dich engagiere. Vertraue mir!

Alice: Ich würde dir lieber die Empfehlung vom letzten Monat zeigen. Ich verspreche auch, nicht zu schummeln. Vertraue mir!

Dieses Problem lässt sich kryptographisch mit einem *Commitment Verfahren* lösen. Solche Verfahren sind grundlegende Primitive in vielen kryptographischen Protokollen. In obigem Beispiel hat Alice die Rolle von S (für *Sender*) und möchte ihre Festlegung (engl. *Commitment*) nicht frühzeitig offenbaren. Diese Eigenschaft wird als *Hiding* bezeichnet. Bob hingegen ist in der Rolle von R (für *Receiver*) und möchte sicherstellen, dass Alice die Festlegung nicht im Nachhinein verändern kann. Diese Eigenschaft wird als *Binding* bezeichnet. Die Betrachtung lässt sich dabei auf den Fall reduzieren, dass der festzulegende Wert b nur ein einziges Bit ist, denn bei einem größeren Wertebereich muss das Verfahren dafür dann nur hinreichend oft parallel ausgeführt werden. Aus diesem Grund sind die untersuchten Commitment Verfahren in diesem Kapitel *Bit Commitment Verfahren*.

Zunächst werden wir eine formale Definition von *Bit Commitment Protokollen* einführen. In den folgenden Abschnitten werden dann einige konkrete Verfahren betrachtet. Von besonderem Interesse sind *nicht-interaktive* Verfahren, in denen nur Nachrichten von S an R übermittelt werden. Wir beginnen daher in Abschnitt 3.2 mit einem einfachen nicht-interaktiven Verfahren, welches auf injektiven Einwegfunktionen beruht. Können nur allgemein Einwegfunktionen vorausgesetzt werden, erhält man zunächst nur ein interaktives Verfahren, mit dem wir uns in Abschnitt 3.3 beschäftigen. Schließlich werden wir jedoch in Abschnitt 3.4 sehen, wie dieses mit der Hilfe von Hitting Set Generatoren aus Kapitel 2 derandomisiert und damit in ein nicht-interaktives Verfahren umgewandelt werden kann.

3.1 Bit Commitment Protokolle

Ein Bit Commitment Protokoll soll also aus zwei Phasen bestehen:

Commit Phase: S legt sich auf ein Bit b gegenüber R fest, indem S mit R interagiert und sich damit ein Commitment $C = \langle S(b), R \rangle$ ergibt.¹ Bei einem *nicht-interaktiven* Protokoll ist C damit eine Funktion in b und den Zufallsbits r_S von S , denn in diesem (entarteten) Fall wird R gar nicht ausgeführt.

Reveal Phase: S „öffnet“ das Commitment gegenüber R und zeigt damit den Wert von b . Im speziellen teilt S dazu R die Werte von b und r_S mit. R kann mit ihnen dann das Commitment C (durch Simulation von $\langle S(b), R \rangle$) überprüfen und akzeptiert genau dann, wenn b und r_S mit C gemäß des Protokolls konsistent sind.

¹Dabei bezeichnet $\langle S(b), R \rangle$ die gesamte Kommunikation zwischen den beiden interaktiven Maschinen S und R .

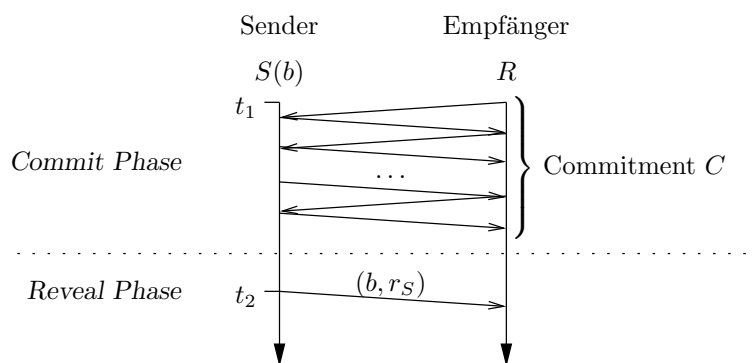


Abbildung 3.1: Allgemeiner Ablauf eines Bit Commitment Protokolls. Dabei ist das Commitment C die Menge der Nachrichten, die in der *Commit Phase* ausgetauscht wurden. In der *Reveal Phase* sendet S sein Bit b und seine Zufallsbits r_S .

Zur praktischen Durchführbarkeit müssen die beiden Teilnehmer effizient, also durch pp-ITMs, implementiert werden können. Dabei sollen die schon genannten Eigenschaften gelten:

Geheimhaltung (*Hiding*): Am Ende der Commit Phase hat der Empfänger R keinerlei Information über den Wert von b . Selbst wenn R sich nicht an das Protokoll hält, muss diese Eigenschaft erfüllt sein.

Eindeutigkeit (*Binding*): Zum Commitment aus der Commit Phase gibt es höchstens einen Wert von b , für den S in der Reveal Phase das Commitment öffnen kann. Selbst wenn S sich nicht an das Protokoll hält, muss diese Eigenschaft erfüllt sein.

Durchführbarkeit (*Viability*): Falls sich R und S an das Protokoll halten, so soll am Ende des Protokolls R tatsächlich den richtigen Wert von b erhalten.

Bit Commitment stellt also zwei „kryptographische“ Anforderungen an das durchgeführte Protokoll: Die *Hiding* Eigenschaft entspricht intuitiv dem Verhalten einer Einwegfunktion, die *Binding* Eigenschaft scheint eine zusätzliche zu sein. Wir werden jedoch später sehen, dass Bit Commitment Protokolle in der Tat äquivalent zur Existenz von Einwegfunktionen sind.

Zunächst geben wir eine formale Definition von Bit Commitment Protokollen. Grundlegend hierfür ist das Konzept der *Ununterscheidbarkeit* gemäß Definition 1.3.

Definition 3.1 (Bit Commitment Protokoll, nach [Gol01]). Ein Paar von *pp-ITMs* (S, R) ist ein Bit Commitment Protokoll, falls gilt:

1. Die gemeinsame Eingabe ist der Sicherheits-Parameter 1^n . Die private Eingabe von S ist ein Bit b .
2. Hiding-Eigenschaft: Für jede *pp-TM* R^* , die mit S interagiert, gilt: $\{\langle S(0), R^* \rangle(1^n)\}_{n \in \mathbb{N}}$ und $\{\langle S(1), R^* \rangle(1^n)\}_{n \in \mathbb{N}}$ sind uniform ununterscheidbar.
3. Binding-Eigenschaft: Sei (C, r_R) die „Empfänger-Sicht“ der Interaktion von R mit S bestehend aus den ausgetauschten Nachrichten, d.h. dem Commitment C und den dabei von R verwendeten Zufallsbits r_R . Wir nennen (C, r_R) mehrdeutig, falls es r_S^0 und r_S^1 gibt, sodass C die Nachrichten beschreibt, wenn
 - S lokale Zufallsbits r_S^0 bei Eingabe $b = 0$ verwendet, und auch wenn
 - S lokale Zufallsbits r_S^1 bei Eingabe $b = 1$ verwendet.

Es muss gelten, dass es höchstens für einen vernachlässigbaren Anteil der $r_R \in \{0, 1\}^{\text{poly}(n)}$ ein C gibt, sodass (C, r_R) mehrdeutig ist.

Es ist zu erkennen, dass die *Hiding*-Bedingung eine *Berechnungs-Resistenz* ist, die *Binding*-Bedingung jedoch eine *informationstheoretische Resistenz*. Diese bezieht sich nicht auf Berechnungs-Stärken des Senders, ein solches C zu generieren, sondern auf die pure Existenz. Aus diesem Grund wird sie „perfekt“ genannt. In englischer Literatur trägt die hier verwendete Art von Bit Commitment daher auch die Bezeichnung *computationally hiding, perfectly binding*. Es sei angemerkt, dass manchmal auch die dazu duale Variante (*perfectly hiding, computationally binding*) betrachtet wird, was wir hier jedoch nicht tun.

Die Definition 3.1 bezieht sich explizit nur auf die Commit Phase, in der eine Interaktion zwischen S und R stattfindet, wodurch das Commitment C als „Aufzeichnung“ der Kommunikation von S und R entsteht. Die Reveal Phase ergibt sich aufgrund dieser Definition kanonisch aus der Binding Eigenschaft:

1. S sendet an R den Wert von b und die Zufallsbits r_S , die S während der Interaktion in der Commit Phase verwendet hatte.
2. R überprüft, ob b und r_S , zusammen mit seinen eigenen Zufallsbits r_R , konsistent mit C sind. Diese Überprüfung kann in polynomieller Zeit geschehen, z.B. durch Simulation von $\langle S(b), R^* \rangle(1^n)$.

Klar ist dadurch auch, dass die Eigenschaft der Durchführbarkeit automatisch gegeben ist.

Bevor wir uns damit beschäftigen, unter welchen Voraussetzungen bestimmte Arten von Bit Commitment Protokollen konstruiert werden können, zunächst ein einfaches Resultat, welches die Existenz von Einwegfunktionen als notwendig herausstellt:

Theorem 3.2. *Falls Bit Commitment Protokolle existieren, so gibt es auch Einwegfunktionen.*

Beweis. Sei (S, R) ein Paar von interaktiven Turingmaschinen, die ein Bit Commitment Protokoll implementieren. Wir definieren eine Funktion f wie folgt:

$$f : (b, r_S, r_R) \mapsto (C, r_R)$$

Diese Funktion ordnet also gegebenen Werten vom Bit b , auf welches das Commitment erzeugt werden soll, und den beiden Zufallsstrings r_S und r_R von S bzw. R , den Ausgang der Commit Phase des Protokolls von (S, R) aus Sicht von R zu, analog der Definition. Klar ist zunächst, dass diese Funktion deterministisch in polynomieller Zeit berechnet werden kann (durch Simulation von $\langle S(b), R \rangle$).

Gäbe es einen Angreifer D für dieses f , der allein ein Paar (C, r_R) als Eingabe bekommt und mit nicht-vernachlässigbarer Wahrscheinlichkeit passende b und r_S bestimmen könnte, könnte man die Hiding Eigenschaft von (S, R) brechen: Zuerst könnte man R zu einem R^* modifizieren, der am Ende seine Zufallsbits r_R mit in das Commitment einfügt. Aus diesem Commitment $C' = \langle S(b), R^* \rangle$ kann somit (C, r_R) abgeleitet werden. Um nun $\langle S(0), R^* \rangle$ von $\langle S(1), R^* \rangle$ zu unterscheiden, kann D aufgerufen werden, der zu (C, r_R) insbesondere ein passendes b bestimmt. Dieses b ist aufgrund der Binding Eigenschaft in allen außer einem zu vernachlässigenden Teil der Fälle eindeutig für (C, r_R) , wodurch die Unterscheidung gelingt.

□

3.2 Nicht-interaktives Bit Commitment via injektiver Einwegfunktion

Wir haben bereits gesehen, dass Einwegfunktionen *im allgemeinen* zumindest notwendig für die Existenz von Bit Commitment Protokollen sind. Eine sogar für nicht-interaktives Bit Commitment hinreichende Bedingung ist bereits seit längerem bekannt: Die Existenz von *injektiven Einwegfunktionen* (engl: *1-1 one-way functions*).

Ein entsprechendes Protokoll gab Manuel Blum in [Blu82] an. Es basiert auf dem *Hard-Core Prädikat* zu einer Einwegfunktion, welches er auch zur Konstruktion von Pseudozufallsgeneratoren benutzt hat. Die eher intuitive Beschreibung aus Abschnitt 2.2.1 ergänzen wir zunächst durch eine formale Definition:

Definition 3.3 (Hard-Core Prädikat). *Sei $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. Eine FP-Funktion $h_f : \{0, 1\}^* \rightarrow \{0, 1\}$ heißt Hard-Core Prädikat von f , falls für alle pp-TMs A , Polynome $p(\cdot)$ und hinreichend große n gilt:*

$$\Pr_{U_n, A}[A(f(U_n)) = h_f(U_n)] < \frac{1}{2} + \frac{1}{p(n)}$$

Ein Hard-Core Prädikat zu einer Funktion f ist also ein Bit, welches aus Elementen des Bildes von f nur schwer, aus deren Urbildern jedoch einfach berechnet werden kann. Es entspricht somit dem intuitiven Verständnis, dass jede Einwegfunktion vielleicht nicht *sämtliche* Informationen über die Urbilder verschleiert, es aber zumindest *irgend eine* Eigenschaft des Urbildes geben muss, die aus dem Bild schwer vorhersagbar ist. Gerade eine solche Eigenschaft wird durch das Hard-Core Prädikat ausgedrückt.

Zu beliebig gegebener Einwegfunktion f kann auch leicht eine Einwegfunktion g konstruiert werden, für die ein Hard-Core Prädikat h_g unmittelbar gegeben ist. Konkret liefert dies das folgende Theorem:

Theorem 3.4. *Sei f eine beliebige Einwegfunktion und $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ definiert durch:*

$$g(x, r) := (f(x), r) \quad \text{für } n := |x| = |r|$$

(Bei Eingaben ungerader Länge sei die Eingabe vorn implizit um ein Bit 0 erweitert und somit der Fall einer Eingabe gerader Länge hergestellt.)

Sei weiterhin $h_g(x, r) := \langle x, r \rangle$ das innere Produkt modulo 2 von x und r (d.h. das Skalarprodukt auf $GF(2)^n$). Dann ist h_g Hard-Core von g .

Beweis (Skizze). Falls h_g nicht Hard-Core Prädikat von g ist, so kann das innere Produkt von x mit beliebigem r aus $f(x)$ mit einer nicht-vernachlässigbaren Wahrscheinlichkeit ermittelt werden. Mit dieser Annahme kann man f ebenfalls invertieren: r wird so gewählt, dass die Bits von x einzeln getestet werden, um x zu rekonstruieren. Dabei werden trickreich viele verschiedene r_i aus wenigen zufällig und unabhängig gewählten s_j so kombiniert, dass die r_i zwar hinreichend unkorreliert sind, jedoch die s_j in ihrer Anzahl trotzdem derart wenige bleiben, dass die Erfolgswahrscheinlichkeit nicht exponentiell verschwindet. \square

Eine wichtige Eigenschaft der Konstruktion in Theorem 3.4 ist, dass die konstruierte Funktion g injektiv bleibt, falls auch f dies war. Damit erhalten wir also insbesondere, dass unter der Voraussetzung der Existenz von injektiven Einwegfunktionen auch solche mit Hard-Core Prädikat existieren.

Wir konstruieren nun das nicht-interaktive Bit Commitment Protokoll von Blum aus einer injektiven Einwegfunktion g mit Hard-Core Prädikat h_g :

Protokoll 3.5 (Nicht-interaktives Bit Commitment Protokoll nach [Blu82]).

Commit Phase: Bei gegebenem Bit b und Sicherheits-Parameter n wählt S zufällig und gleichverteilt $r_S \in_{\mathcal{R}}\{0,1\}^n$ und sendet an R :

$$(\alpha, \sigma) := (g(r_S), h_g(r_S) \oplus b)$$

Reveal Phase (kanonisch): S sendet b und r_S an R . Danach überprüft R , ob gilt:

$$g(r_S) = \alpha \wedge h_g(r_S) \oplus b = \sigma$$

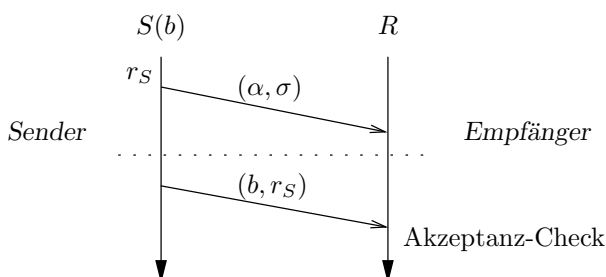


Abbildung 3.2: Nicht-interaktives Bit Commitment von Blum.

Dieses Protokoll stellt in der Tat ein nicht-interaktives Bit Commitment Protokoll dar:

Proposition 3.6. Falls g eine injektive Einwegfunktion ist und h_g ein Hard-Core Prädikat von g , so ist Protokoll 3.5 ein nicht-interaktives Bit Commitment Protokoll.

Beweis. Zunächst zeigen wir die Hiding Eigenschaft: Sie folgt unmittelbar daraus, dass h_g Hard-Core Prädikat von g ist. Jegliche nicht-vernachlässigbare Tendenz für b bei gegebenen Werten von $g(r_S)$ und $h_g(r_S) \oplus b$ liefert direkt eine gleich starke Tendenz für $h_g(r_S)$, die wegen der Hard-Core Eigenschaft von h_g aber vernachlässigbar sein muss.

Außerdem gilt die Binding Eigenschaft: Aufgrund der Injektivität von g gibt es bei gegebenem $\alpha = g(r_S)$ nur ein r_S , mit dem S dieses α erzeugen konnte. Damit ist auch $h_g(r_S)$ eindeutig bestimmt und damit bei gegebenem $\sigma = h_g(r_S) \oplus b$ auch b . Es gibt also *in keinem Fall* eine mehrdeutige Sicht des Empfängers. \square

Anhand des Beweises ist zu sehen, dass die Injektivität von g in diesem Protokoll in der Tat eine kritische Voraussetzung ist, um die Binding Eigenschaft sicher zu stellen. Ohne diese Voraussetzung funktioniert die Konstruktion nicht: Ist $g(x_0) = g(x_1)$ für ein Paar $x_0 \neq x_1$, und gilt außerdem $h_g(x_0) \neq h_g(x_1)$ (diese Eigenschaft kann erfüllt werden, wenn das Paar (g, h_g) aus einer nicht-injektiven Einwegfunktion f gemäß Theorem 3.4 konstruiert wurde), so kann S zum Brechen der Binding-Eigenschaft $r_S = x_0$ wählen und später zu beliebigem b das Commitment mit (b, x_b) öffnen.

Abschließend bemerken wir noch, dass dieses Protokoll zwar nicht-interaktiv ist, zur Konstruktion jedoch auch *eine konkrete* injektive Einwegfunktion notwendig ist. Zur Instanziierung des Protokolls reicht also die *bloße Existenz* einer solchen Funktion *nicht* aus. Stattdessen muss eine konkrete Funktion zur Verfügung stehen. In den folgenden Abschnitten wird die Situation eine andere sein.

3.3 Interaktives Bit Commitment via Einwegfunktionen

Wir setzen nun also keine Injektivität der Einwegfunktion mehr voraus. Wie oben gesehen, funktioniert Blum's einfache Konstruktion mit Hilfe eines Hard-Core Prädikates damit im allgemeinen nicht mehr. Führt man jedoch *Interaktion* in das Verfahren ein, so kann dennoch die Binding Eigenschaft hergestellt werden, indem der Empfänger R Zufall in die Wahl des Commitments einfließen lässt und so den Sender S dazu zwingen kann, sich nicht auf günstig passende r_S zu beschränken, die das Commitment mehrdeutig machen könnten.

Ein solches *interaktives Bit Commitment Protokoll* schlug Moni Naor in [Nao91] vor. Dabei wird benutzt, dass die Existenz von Einwegfunktionen gemäß Theorem 2.3 äquivalent zur Existenz von (kryptographischen) Pseudozufallsgeneratoren ist. Da die Expansion eines beliebig gegebenen PRNG durch Iteration beliebig polynomiell gemacht werden kann, impliziert die allgemeine Annahme (Existenz von Einwegfunktionen) also PRNGs mit beliebig polynomieller Expansion. Sei daher G in der folgenden, zunächst intuitiven Beschreibung der Idee ein PRNG mit $|G(s)| = 3|s|$.

Nehmen wir vorerst an, die Mengen $M_0 := \{G(s) \mid s \in \{0, 1\}^n\}$ und $M_1 := \{G(s) \oplus 1^{3n} \mid s \in \{0, 1\}^n\}$ seien disjunkt². Ein Commitment von S auf ein Bit b könnte dann so aussehen, dass S bei zufällig gewähltem $r_S \in \{0, 1\}^n$ setzt $C := G(r_S) \oplus b^{3n}$. Da G PRNG ist, kann R anhand von C nicht feststellen, ob C im Bild von G liegt (also in M_0), oder nicht (also in M_1), woraus die Hiding Eigenschaft folgen würde. Die Binding Eigenschaft würde aus der Disjunktheit von M_0 und M_1 folgen, denn selbst wenn S den String r_S künstlich wählen würde, mit dem Ziel, das Binding zu brechen, gäbe es keine r_S^0 und r_S^1 die bei unterschiedlichem b auf das selbe Commitment C führen würden. Daher wäre S , egal wie die Wahl von r_S auch aussieht, auf einen Wert von b festgelegt.

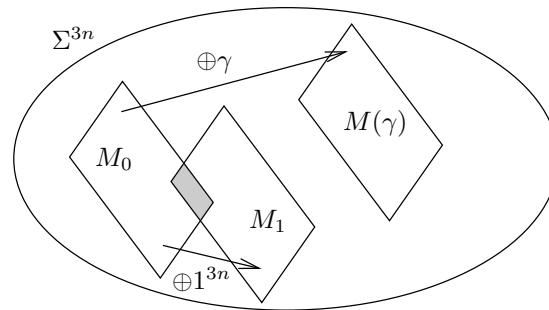


Abbildung 3.3: Für die Generatorausgaben $G(s)$ ist M_0 vielleicht nicht immer disjunkt zu $M_1 := M_0 \oplus 1^{3n}$, dafür aber doch zu den meisten $M(\gamma) := M_0 \oplus \gamma$.

Leider kann im allgemeinen nicht davon ausgegangen werden, dass $M_0 \cap M_1 = \emptyset$ gilt. Sobald es auch nur ein Element gibt, welches in beiden Mengen vorkommt, also s_0 und s_1 mit $G(s_0) = G(s_1) \oplus 1^{3n} \iff G(s_0) \oplus G(s_1) = 1^{3n}$, könnte eine geschickte Wahl von r_S (nämlich $r_S \in \{s_0, s_1\}$) die Binding Eigenschaft brechen. Für beliebiges Bit b könnte das Commitment später mittels (b, s_b) geöffnet werden. Um dies zu beheben, können wir jedoch wie bereits bemerkt Interaktion benutzen. Die Muster, die via bitweisem XOR mit der Ausgabe des Generators verknüpft werden, sind nun nicht mehr fest, sondern variabel – definiere dazu:

$$M(\gamma) := \{G(s) \oplus \gamma \mid s \in \{0, 1\}^n\}$$

Die entscheidende Beobachtung ist nun, dass M_0 und $M(\gamma)$ für die meisten $\gamma \in \{0, 1\}^{3n}$ disjunkt sind (siehe Abbildung 3.3), aufgrund eines einfachen Abzählarguments (siehe Beweis von Lemma 3.9 weiter unten). Solche γ nennen wir *gut*. Die Disjunktheit benötigen wir für die Binding Eigenschaft, also

²Dabei bezeichnet $x \oplus y$ das bitweise XOR von x und y .

darf die Wahl von γ nicht bei S liegen, sondern bei R , dem am Binding gelegen ist. Das dadurch interaktiv gewordene Protokoll wird also so ablaufen: R wählt $r_R \in_{\mathbb{R}}\{0, 1\}^{3n}$, hoffend, dass r_R ein *guter* String sei, und sendet es an S , der wiederum $r_S \in_{\mathbb{R}}\{0, 1\}^n$ wählt. Ein Commitment auf $b = 0$ ist dann $C = G(r_S)$, ein Commitment auf $b = 1$ hingegen $C = G(r_S) \oplus r_R$. Die Hiding Eigenschaft folgt dann aus der PRNG Eigenschaft von G (selbst wenn r_R nicht zufällig war), die Binding Eigenschaft wiederum aus der wahrscheinlichen Disjunktheit von M_0 und $M(r_R)$ (selbst wenn r_S nicht zufällig war).

Wir fassen das Protokoll zusammen. Dabei sei wie oben G ein kryptographischer PRNG mit $|G(s)| = 3|s|$.

Protokoll 3.7 (Bit Commitment Protokoll nach [Nao91]).

Commit Phase: *Das Commitment läuft als interaktiver Prozess in 2 Runden:*

R: Wähle gleichverteilt zufällig $r_R \in_{\mathbb{R}}\{0, 1\}^{3n}$ und sende r_R an S .

S: Wähle gleichverteilt zufällig $r_S \in_{\mathbb{R}}\{0, 1\}^n$ und sende α an R mit:

$$\alpha := \begin{cases} G(r_S), & \text{falls } b = 0 \\ G(r_S) \oplus r_R, & \text{falls } b = 1. \end{cases}$$

Reveal Phase (kanonisch): S sendet (b, r_S) an R . R macht die Akzeptanz abhängig von der Gültigkeit folgender Bedingung:

$$(b = 0 \wedge \alpha = G(r_S)) \vee (b = 1 \wedge \alpha = G(r_S) \oplus r_R)$$

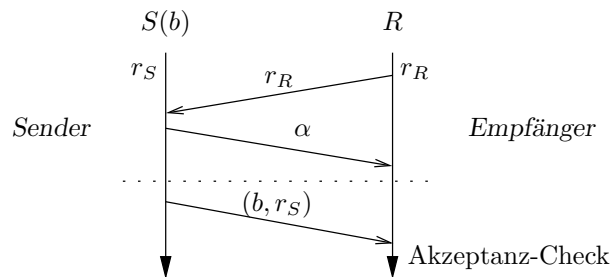


Abbildung 3.4: Interaktives Bit Commitment von Naor.

Wir beweisen nun wieder die beiden Eigenschaften *Hiding* und *Binding* dieses Protokolls:

Lemma 3.8. *Protokoll 3.7 hat die Hiding Eigenschaft. Das heißt insbesondere: Für jedes $r \in \{0, 1\}^{3n}$ sind $\{G(U_n)\}_{n \in \mathbb{N}}$ und $\{G(U_n) \oplus r\}_{n \in \mathbb{N}}$ uniform ununterscheidbar.*

Beweis. Für die Hiding Eigenschaft muss die (uniforme) Ununterscheidbarkeit von $\langle S(0), R^* \rangle (1^n)$ und $\langle S(1), R^* \rangle (1^n)$ gezeigt werden. Da sich die Protokoll-Abläufe nur aufgrund des Wertes von b unterscheiden und davon insbesondere die Wahl der Zufallsstrings r_S und r_R von S bzw. R nicht abhängt, genügt eine Betrachtung von $\{G(U_n)\}_{n \in \mathbb{N}}$ und $\{G(U_n) \oplus r_S\}_{n \in \mathbb{N}}$ für alle r_S .

Falls G ein PRNG gegen Schaltkreise ist, ist für jeden effizienten Angreifer C und jedes Polynom $p(\cdot)$ die folgende Ungleichung für hinreichend große n erfüllt:

$$|\Pr_{U_n}[C(G(U_n)) = 1] - \Pr_{U_{3n}}[C(U_{3n}) = 1]| < \frac{1}{p(n)}$$

C kann aufgrund seiner Nicht-Uniformität eine XOR Verknüpfung seiner Eingabe mit bekanntem Bitmuster im ersten Schritt entfernen, also haben wir für beliebige $r \in \{0, 1\}^{3n}$ jedoch auch:

$$|\Pr_{U_n}[C(G(U_n) \oplus r) = 1] - \Pr_{U_{3n}}[C(U_{3n} \oplus r) = 1]| < \frac{1}{p(n)}$$

Da U_{3n} und $U_{3n} \oplus r$ identisch verteilt sind, folgt zusammen mit der Dreiecksungleichung:

$$|\Pr_{U_n}[C(G(U_n)) = 1] - \Pr_{U_n}[C(G(U_n) \oplus r) = 1]| < \frac{1}{p(n)} + \frac{1}{p(n)} < \frac{1}{q(n)}$$

Zu beliebigem $q(\cdot)$ kann $p(\cdot)$ so gewählt werden dass diese Ungleichung stimmt, folglich sind $\{G(U_n)\}_{n \in \mathbb{N}}$ und $\{G(U_n) \oplus r\}_{n \in \mathbb{N}}$ nicht-uniform ununterscheidbar, also insbesondere uniform ununterscheidbar. \square

Lemma 3.9. *Protokoll 3.7 hat die Binding Eigenschaft. Das heißt insbesondere: $\Pr_{r \in \{0, 1\}^{3n}}[r \text{ ist gut}] \geq 1 - 2^{-n}$*

Beweis. Die Binding-Eigenschaft ist etabliert, wenn durch das Commitment $C = \langle S(b), R \rangle (1^n)$ der Wert von b eindeutig bestimmt ist. Dies ist sicherlich dann der Fall, wenn die Mengen M_0 und $M(r_R)$ disjunkt sind, denn die von S während des Commitments an R gesendete Nachricht α stammt aus einer der beiden Mengen. Sind sie disjunkt, folgt daraus die Festlegung auf genau einen Wert von b . Solche r_R , für die $M_0 \cap M(r_R) = \emptyset$ ist, nannten wir *gut*.

Es genügt also zu zeigen, dass für den überwiegenden Teil der r_R (d.h. alle bis auf vernachlässigbar viele) diese Eigenschaft erfüllt ist.

Hierzu sehen wir, dass r_R genau dann *gut* ist, wenn für alle Paare (s, s') gilt, dass $G(s) \neq G(s') \oplus r_R$. Umgekehrt ist daher r_R genau dann *nicht gut*, wenn es (s, s') gibt sodass $G(s) \oplus G(s') = r_R$. Die Menge der *nicht guten* r_R ist also folgende Menge N :

$$N = \{G(s) \oplus G(s') \mid s, s' \in \{0, 1\}^n\}$$

Damit ist unmittelbar klar, dass es höchstens $|N| \leq 2^{2n}$ *nicht gute* r_R gibt. Da allerdings $r_R \in \{0, 1\}^{3n}$ gewählt wird, folgt:

$$\Pr_{r \in \{0, 1\}^{3n}}[r \text{ ist nicht gut}] \leq \frac{2^{2n}}{2^{3n}} = 2^{-n} \implies \Pr_{r \in \{0, 1\}^{3n}}[r \text{ ist gut}] \geq 1 - 2^{-n}$$

□

Aus Lemma 3.8 und 3.9 folgt unmittelbar:

Theorem 3.10. *Falls G ein (kryptographischer) Pseudozufallsgenerator gegen Schaltkreise mit $|G(s)| = 3|s|$ ist, so ist Protokoll 3.7 ein interaktives Bit Commitment Protokoll.*

Abschließend bemerken wir hier, dass im Unterschied zum nicht-interaktiven Protokoll von Blum keine *konkrete* Funktion gegeben sein muss, um das Protokoll zu instanzieren. Hierzu kann nämlich eine (konkrete) *universelle Einwegfunktion* Verwendung finden, deren Eigenschaft bereits aus der *Existenz* von Einwegfunktionen folgt. Ebenso ist das Resultat von [HILL99] konstruktiv, sodass auch hier ein konkreter Pseudozufallsgenerator entsteht.

3.4 Derandomisierung von Naor's Protokoll

Um das obige Protokoll von Naor zu derandomisieren und es so in ein nicht-interaktives umzuwandeln, ist die folgende Beobachtung entscheidend: Die Interaktion wird dadurch verursacht, dass R einen Zufallsstring $\gamma := r_R$ an S sendet, den S verwenden soll, um das Commitment aus M_0 und $M(\gamma)$ auszuwählen. Die Wahl liegt bei R , um S daran zu hindern, den Wert von γ künstlich so zu wählen, dass M_0 und $M(\gamma)$ nicht disjunkt sind und das Commitment somit mehrdeutig gemacht werden kann. Solange also sichergestellt ist, dass S ein *gutes* γ verwendet, ist diese Interaktion nicht notwendig.

Der Test darauf, ob ein $\gamma \in \{0, 1\}^{3n}$ *gut* ist, ist deterministisch aufwändig, denn es muss gelten:

$$M_0 \cap M(\gamma) = \emptyset \iff \forall s, s' : G(s) \oplus G(s') \neq \gamma \quad (*)$$

Ein Brute-Force Ansatz hätte hier also exponentielle Komplexität. Offenbar kann diese Bedingung jedoch *co-nichtdeterministisch* in polynomieller Zeit überprüft werden. Wir hatten in Lemma 3.9 gesehen, dass der Test in der überwiegenden Mehrheit der Fälle einen positiven Ausgang hat. Daher hat ein testender (co-nichtdeterministischer) Algorithmus eine hohe Erfolgsquote, und es bietet sich der Einsatz eines *Hitting Set Generators* gegen co-nichtdeterministische Algorithmen an. Ein solcher HSG gibt nämlich ein Hitting Set aus, welches mindestens ein Element enthalten muss, das der testende Algorithmus akzeptiert – und damit also mindestens ein *gutes* γ .

Die Idee ist nun also, die Interaktion dadurch zu beseitigen, dass nicht R an S einen Zufallsstring $r_R \in_{\mathcal{R}} \{0, 1\}^{3n}$ sendet, sondern S stattdessen die Ausgabe eines Hitting Set Generators verwendet. Statt wie im ursprünglichen Protokoll $\gamma = r_R$ abhängig von b mit der PRNG-Generator Ausgabe $G(r_S)$ zu einem α bitweise XOR zu verknüpfen, wird dies nun mit *allen* $\gamma^{(i)}$ *parallel* geschehen (etwa zu $\alpha^{(i)}$), die der HSG ausgibt. Mindestens ein solches $\gamma^{(i)}$ wird ein *gutes* sein und damit die Binding Eigenschaft sicherstellen.

Dies wird natürlich das Commitment vergrößern, allerdings um einen maximal polynomiellen Faktor. Außerdem muss S für jedes dieser polynomiell vielen Elemente $\alpha^{(i)}$ neue, unabhängige Zufallsbits $r_S^{(i)}$ verwenden. Das Protokoll wird damit also zwar nicht-interaktiv, jedoch wachsen Commitment-Größe und Bedarf an Zufalls-Bits seitens des Senders polynomiell an.

Wir fassen wiederum das entstehende Protokoll zusammen. Dabei sei wieder G ein kryptographischer PRNG mit $|G(s)| = 3|s|$ und sei H ein 1/2-Hitting Set Generator gegen co-nichtdeterministische Algorithmen gemäß Definition 2.8. Sei außerdem $p_A(\cdot)$ ein Polynom, welches die Laufzeit eines co-nichtdeterministischen Algorithmus A beschränkt, der Eigenschaft $(*)$ testet.

Protokoll 3.11 (Nicht-interaktives Bit Commitment nach [BOV07]).

Commit Phase: *Das Commitment läuft nicht-interaktiv in einer Runde.*

S: Sei $\gamma = (\gamma^{(1)}, \dots, \gamma^{(p_H(n))})$ die Ausgabe von $H(1^{3n}, 1^{p_A(3n)})$, wobei $p_H(\cdot)$ ein Polynom ist, welches die Größe des ausgegebenen Hitting Sets bei dieser Eingabe beschränkt. Wähle für jedes $i \in \{1, \dots, p_H(n)\}$ gleichverteilt zufällig und unabhängig $r_S^{(i)} \in_{\mathcal{R}} \{0, 1\}^n$ und setze:

$$\alpha^{(i)} := \begin{cases} G(r_S^{(i)}), & \text{falls } b = 0 \\ G(r_S^{(i)}) \oplus \gamma^{(i)}, & \text{falls } b = 1. \end{cases}$$

Sende schließlich $\alpha = (\alpha^{(1)}, \dots, \alpha^{(p_H(n))})$ an R .

Reveal Phase (kanonisch): S sendet b und alle $r_S^{(i)}$ an R . R ruft ebenfalls $H(1^{3n}, 1^{p_A(3n)})$ auf und macht die Akzeptanz abhängig von der Gültigkeit folgender Bedingung:

$$\left(b = 0 \wedge \forall i : \alpha^{(i)} = G(r_S^{(i)}) \right) \vee \left(b = 1 \wedge \forall i : \alpha^{(i)} = G(r_S^{(i)}) \oplus \gamma^{(i)} \right)$$

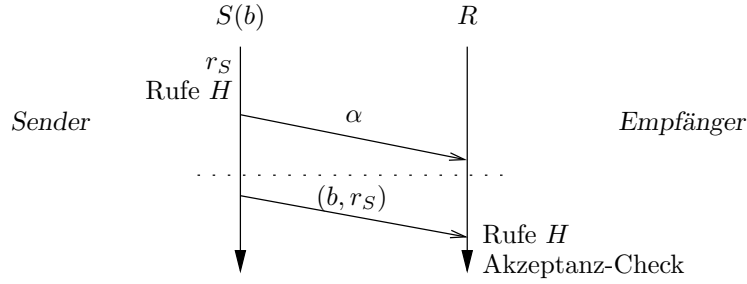


Abbildung 3.5: Derandomisiertes, nicht-interaktives Bit Commitment nach [BOV07]. Im Vergleich zu Protokoll 3.7 (vgl. Abbildung 3.4) wird die dortige erste Runde (R sendet an S Zufallszahlen r_R) durch lokale Aufrufe des Hitting Set Generators H ersetzt.

Zu zeigen sind wieder die *Hiding* und *Binding* Eigenschaften:

Lemma 3.12. *Protokoll 3.11 hat die Hiding Eigenschaft.*

Beweis. Wieder müssen $\{\langle S(0), R^* \rangle(1^n)\}_{n \in \mathbb{N}}$ und $\{\langle S(1), R^* \rangle(1^n)\}_{n \in \mathbb{N}}$ uniform ununterscheidbar sein. Analog Lemma 3.8 reicht es hier, unsere Betrachtungen auf die beiden möglichen Strings α zu beschränken, die S senden kann:

$$\begin{aligned} \alpha_0 &= (G(U_n^1), \dots, G(U_n^{p_H(n)})) \\ \alpha_1 &= (G(U_n^1) \oplus \gamma^{(1)}, \dots, G(U_n^{p_H(n)}) \oplus \gamma^{(p_H(n))}) \end{aligned}$$

Nach Lemma 3.8 wissen wir bereits für jedes i , dass $G(U_n^i)$ und $G(U_n^i) \oplus \gamma^{(i)}$ uniform ununterscheidbar sind. Mit einem Hybrid-Argument erweitert sich das nun auf obige Ensembles: Setze

$$\mathcal{H}_i := (G(U_n^1), \dots, G(U_n^i), G(U_n^{i+1}) \oplus \gamma^{(i+1)}, \dots, G(U_n^{p_H(n)}) \oplus \gamma^{(p_H(n))})$$

Damit haben wir gerade $\mathcal{H}_0 = \alpha_1$ und $\mathcal{H}_{p_H(n)} = \alpha_0$. Außerdem wissen wir, dass für jedes i die Hybride \mathcal{H}_i und \mathcal{H}_{i+1} uniform ununterscheidbar sind. Es folgt für jedes Polynom $q(\cdot)$ und hinreichend große n :

$$\begin{aligned} |\Pr[A(\alpha_0)] - \Pr[A(\alpha_1)]| &= |\Pr[A(\mathcal{H}_0)] - \Pr[A(\mathcal{H}_{p_H(n)})]| \\ &\leq \sum_{k=0}^{p_H(n)-1} |\Pr[A(\mathcal{H}_k)] - \Pr[A(\mathcal{H}_{k+1})]| \\ &\leq p_H(n) \cdot \frac{1}{q(n)} \end{aligned}$$

Für hinreichend große n und alle Polynome $q'(\cdot)$ folgt damit:

$$|\Pr[A(\alpha_0)] - \Pr[A(\alpha_1)]| < \frac{1}{q'(n)}$$

Also sind α_0 und α_1 uniform ununterscheidbar. \square

Lemma 3.13. *Protokoll 3.11 hat die Binding Eigenschaft.*

Beweis. Wir definieren den co-nichtdeterministischen Algorithmus A zum Test der Eigenschaft (*) wie folgt:

$$A(\gamma) = 1 : \iff \forall s, s' : G(s) \oplus G(s') \neq \gamma$$

Dazu erzeugt A nichtdeterministisch s und s' aus $\{0, 1\}^n$ und testet, ob deren XOR-Summe ungleich γ ist. Damit ist $A(\gamma) = 1$ genau dann, wenn γ gut ist, und es gilt nach Lemma 3.9:

$$\Pr_{U_{3n}}[A(U_{3n}) = 1] \geq 1 - 2^{-n} \geq 1/2$$

Außerdem kann die Laufzeit von A durch ein Polynom $p_A(\cdot)$ beschränkt werden. Damit ist A ein co-nichtdeterministischer Algorithmus, der mindestens $1/2$ aller Eingaben akzeptiert und durch $p_A(\cdot)$ Laufzeit-beschränkt ist. Für einen $1/2$ -Hitting Set Generator H gegen co-nichtdeterministische Algorithmen gilt demnach:

$$\exists \gamma \in H(1^{3n}, 1^{p_A(3n)}) : \forall s, s' \in \{0, 1\}^n : G(s) \oplus G(s') \neq \gamma$$

Für die Ausgabe $(\gamma^{(1)}, \dots, \gamma^{(p_H(n))})$ von $H(1^{3n}, 1^{p_A(3n)})$ gibt es damit keine $r_S^{(1)}, \dots, r_S^{(p_H(n))}, t_S^{(1)}, \dots, t_S^{(p_H(n))}$ mit:

$$\left(G(r_S^{(1)}), \dots, G(r_S^{(p_H(n))}) \right) = \left(G(t_S^{(1)}) \oplus \gamma^{(1)}, \dots, G(t_S^{(p_H(n))}) \oplus \gamma^{(p_H(n))} \right)$$

An mindestens einer Stelle müssen sich diese Tupel unterscheiden. Damit kann es kein mehrdeutiges Commitment geben, egal welche Seeds S auch wählt. \square

Aus Lemma 3.12 und 3.13 folgt wiederum:

Theorem 3.14. *Falls G ein (kryptographischer) Pseudozufallsgenerator mit $|G(s)| = 3|s|$ ist, und H ein 1/2-Hitting Set Generator gegen co-nichtdeterministische Algorithmen, so ist Protokoll 3.11 ein nicht-interaktives Bit Commitment Protokoll.*

Analog den Schlussbetrachtungen zu Naor's Protokoll in Abschnitt 3.3 gilt auch hier, dass keine *konkrete* Funktion mit den nötigen Eigenschaften zur Instanziierung des Protokolls vorausgesetzt werden muss. Dies gilt nicht nur für eine Einwegfunktion (für den PRNG), sondern auch für den Hitting Set Generator, da für ihn nur die *Existenz* einer hinreichend „schweren“ Sprache in E (gemäß Theorem 2.7 und 2.12) ausreicht: Eine konkrete Sprache kann dann zum Beispiel mittels des beschränkten Halteproblems konstruiert werden, welches vollständig für E ist. Damit erhalten wir:

Korollar 3.15. *Falls es Einwegfunktionen und 1/2-Hitting Set Generatoren gegen co-nichtdeterministische Algorithmen gibt, so existieren nicht-interaktive Bit Commitment Protokolle.*

3.5 Partiiell-injektive Einwegfunktionen

Am Anfang des Kapitels hatten wir gesehen, dass Bit Commitment Protokolle im allgemeinen Einwegfunktionen implizieren (Theorem 3.2) und umgekehrt (Theoreme 2.3 und 3.10 für Naor's Protokoll). Für *injektive* Einwegfunktionen hatten wir die Implikation zur Existenz von *nicht-interaktiven* Bit Commitment Protokollen gesehen. Eine interessante Frage ist nun, ob hier auch die Umkehrung gilt, also ob nicht-interaktive Bit Commitment Protokolle ihrerseits auch Einwegfunktionen implizieren, die injektiv sind.

In ihrem Artikel [BOV07] gelingt den Autoren diese Äquivalenz für eine abgeschwächte Form von Injektivität für Einwegfunktionen:

Definition 3.16 (Partiiell-injektive Einwegfunktion). *Eine Funktion $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ ist eine partiell-injektive Einwegfunktion, falls die folgenden drei Bedingungen gelten:*

1. f kann in polynomieller Zeit berechnet werden.
2. Aus $x \neq x'$ folgt $f(x, y) \neq f(x', y)$.
3. Für jede pp-TM A und jedes Polynom $p(\cdot)$ gilt für hinreichend große n :

$$\Pr_{A, U_n, U'_n} [A(f(U_n, U'_n), 1^n) = U_n] < \frac{1}{p(n)}$$

Eine solche *partiiell-injektive Einwegfunktion* ist also eine Einwegfunktion, welche auf dem ersten Argument injektiv ist. Die Bedingung der schweren Umkehrbarkeit ist „eng“ für das erste Argument definiert (und gilt damit auch für die gesamte Eingabe), weil sich sonst jede Einwegfunktion durch „Vorschalten“ identisch ausgegebener Bits trivial in eine Funktion dieser Art umbauen ließe.

Nun gilt folgender Zusammenhang, der das Analogon zu allgemeinen Einwegfunktionen und Bit Commitment Protokollen bildet:

Theorem 3.17. *Partiiell-injektive Einwegfunktionen existieren genau dann, wenn nicht-interaktive Bit Commitment Verfahren existieren.*

Beweis (Skizze). Sei f eine partiiell-injektive Einwegfunktion. Für ein Commitment auf ein Bit b bei Sicherheits-Parameter 1^n wählt S drei Zufallszahlen $r_S^1, r_S^2, r_S^3 \in_{\mathcal{R}} \{0, 1\}^n$ und sendet:

$$C := (f(r_S^1, r_S^2), r_S^3, \langle r_S^1, r_S^3 \rangle \oplus b)$$

Dabei ist $\langle r_S^1, r_S^3 \rangle$ wie in Theorem 3.4 das innere Produkt von r_S^1 und r_S^3 .

Die *Hiding* Eigenschaft dieses Protokolls folgt dann wieder aus der *Hard-Core* Eigenschaft von $\langle \cdot, \cdot \rangle$, analog Protokoll 3.5. Die *Binding* Eigenschaft folgt daraus, dass die Angabe von $f(r_S^1, r_S^2)$ im Commitment den Wert von r_S^1 eindeutig durch die Injektivität im ersten Argument von f festlegt. Da r_S^3 ohnehin direkt im Commitment vorkommt, ist damit auch $\langle r_S^1, r_S^3 \rangle$ festgelegt, also auch b .

Sei nun umgekehrt (S, R) ein nicht-interaktives Bit Commitment Protokoll. Dies kann durch Parallelisierung auf ein nicht-interaktives Bitstring Commitment Protokoll erweitert werden, indem für jedes Bit unabhängig eine Ausgabe von S erzeugt wird. Sei $p(\cdot)$ ein Polynom, welches die Anzahl der Zufallsbits r_S von S beschränkt. Eine partiiell-injektive Einwegfunktion $f(x, y)$ kann dann so definiert werden, dass das Bild ein Commitment auf den String x darstellt, bei dem S die Zufallsbits $r_S = y0^{p(|x|)-|y|}$ verwendet.

Klar ist, dass f in polynomieller Zeit berechnet werden kann (durch Simulation von S). Aus der *Binding* Eigenschaft folgt die Injektivität auf dem ersten Argument. Die schwere Umkehrbarkeit folgt aus der *Hiding* Eigenschaft, denn wäre ein Angreifer A für f erfolgreich, so könnte damit auch x vorhergesagt werden, dessen Geheimhaltung das Commitment Protokoll aber gerade sicherstellt. \square

Partiiell-injektive Einwegfunktionen werden nun also insbesondere von nicht-interaktiven Bit Commitment Protokollen impliziert. Im Zusammenhang mit dem Derandomisierungs-Resultat aus Abschnitt 3.4 und dem dor-

tigen Korollar 3.15 ergibt sich nun eine allgemeine Aussage über den Zusammenhang von Einwegfunktionen und deren Injektivität unter einer „nicht-kryptographischen“ Annahme:

Korollar 3.18. *Falls es $1/2$ -Hitting Set Generatoren gegen co-nichtdeterministische Algorithmen gibt, so gilt: Die Existenz von Einwegfunktionen impliziert die Existenz von partiell-injektiven Einwegfunktionen.*

Ob diese Implikation auf vollständig injektive Einwegfunktionen verschärft werden kann, ist weiterhin Gegenstand aktueller Forschung. Die „Lücke“ zwischen Einwegfunktionen im allgemeinen und injektiven Einwegfunktionen ist durch dieses Resultat aber zumindest ein Stück weit kleiner geworden.

Kapitel 4

Zero Knowledge Proofs und Witness Indistinguishability

Die nächste Art von Protokollen, mit denen wir uns befassen wollen, sind *Zero Knowledge Proofs* und die damit verwandte Eigenschaft der *Zeugen-Ununterscheidbarkeit* (engl. „*Witness Indistinguishability*“). Das intuitiv zugrunde liegende Szenario ist hierbei, dass es wieder eine Interaktion zwischen zwei Parteien P und V gibt. Dabei möchte P sein Gegenüber V von der Gültigkeit einer Aussage überzeugen, jedoch darüber hinaus *nichts* weiter offenbaren – also kein zusätzliches Wissen an V preisgeben oder V neue Fähigkeiten ermöglichen. Insbesondere soll V nicht in die Lage versetzt werden, einer dritten Partei die betreffende Aussage zu beweisen (falls V dies vor der Interaktion mit P noch nicht konnte).

Diese in gewisser Weise „maximale“ Stufe der Geheimhaltung während einer Interaktion hat vielfältige praktische Anwendungen, wie zum Beispiel Identifikations-Protokolle. In den Abschnitten 4.1 und 4.2 werden wir beide Konzepte zunächst eingehender einführen. Da wir insbesondere an Protokollen mit minimaler Interaktion interessiert sind, wird im Abschnitt 4.3 der Sonderfall für *nicht-interaktive* Protokolle mit Zero Knowledge Eigenschaft untersucht. Schließlich betrachten wir in Abschnitt 4.4 sogenannte „Zaps“ (Protokolle in 2 Runden mit interessanten Geheimhaltungs-Eigenschaften), die wir in Abschnitt 4.5 wiederum zu nicht-interaktiven Protokollen derandomisieren werden. Sie bilden auch die Basis für die Konstruktionen im späteren Kapitel 5.

4.1 Zero Knowledge Beweissysteme

Als einleitendes Beispiel betrachten wir die in [QGB89] sehr anschaulich ausgeführte Story, welche sich stark vereinfacht wie folgt schildern lässt:

Tief in einem Berg gibt es eine Höhle mit einem Ring-Gang, der nur einen Eingang und einen Ausgang hat, welche beide in eine T-Kreuzung münden (vgl. Skizze in Abbildung 4.1). Mitten in diesem Gang befindet sich eine „magische Wand“, die den Gang wie eine Sackgasse erscheinen lässt, sich jedoch durch ein „magisches Wort“ öffnen lässt und den Durchgang freigibt. Ein Eingeborener lebt in dieser Höhle und ist der einzige, der dieses magische Wort kennt.

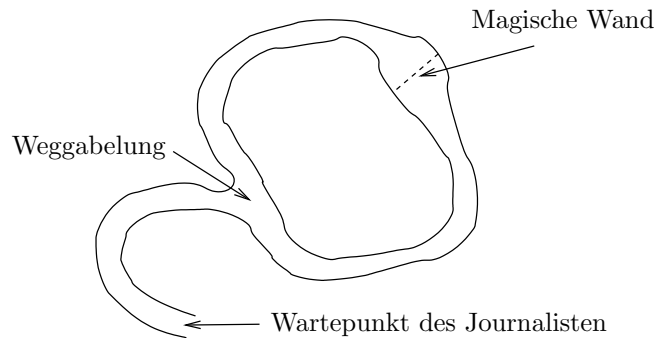


Abbildung 4.1: Skizze der Höhle.

Eines Tages besucht ihn ein Journalist, der von diesem kuriosen Phänomen gehört hatte, und möchte eine Dokumentation darüber drehen. Der Eingeborene ist froh über den Besuch und möchte dem Journalisten die Eigenschaft der Wand gerne demonstrieren. Er ist jedoch nicht gewillt, das magische Wort dem Journalisten preis zu geben. Mehr noch: Er möchte zwar den Journalisten davon überzeugen, dass die Wand wirklich magisch ist, jedoch möchte er weltweiter Publicity und daraus folgenden Besucherstürmen in seine gemütliche Höhle auf jeden Fall vorbeugen.

Er ersinnt folgende List: Er geht mit dem Journalisten bis zu einer Stelle kurz bevor die Weggabelung zum Ring-Gang zu sehen ist. Der Journalist muss dort stehenbleiben und warten, während der Eingeborene zur Gabelung und weiter in einen beliebigen der beiden Eingänge des Ring-Ganges geht. Anschließend darf der Reporter bis zur Gabelung kommen. Er wirft dort eine Münze, und

je nach Ausgang des Münzwurfs ruft er dem Eingeborenen zu, aus welchem Ausgang des Ring-Ganges dieser wieder erscheinen soll. Für den Eingeborenen ist dies einfach zu erfüllen, schließlich kann er bei Bedarf die magische Wand öffnen und somit stets erscheinen, wo er möchte. Nachdem der Journalist zum Ausgangspunkt außer Sichtweite der Gabelung zurückkehrt, wiederholen die beiden dies 20 mal. Schließlich ist der Journalist davon überzeugt, dass der Eingeborene die Wand tatsächlich öffnen kann, denn er hält es für ausgeschlossen, dass der Eingeborene 21 mal hintereinander den Ausgang seines Münzwurfes erraten könnte.

Der Journalist ist froh, fährt nach Hause und möchte seine Dokumentation veröffentlichen. Jedoch wird ihm nun bewusst, warum der Eingeborene dieses seltsame Verfahren gewählt hatte, um ihn zu überzeugen: Zwar ist er selbst sicher, dass der Eingeborene in der Lage ist, die Wand zu öffnen, jedoch wird der Journalist sein Publikum nicht davon überzeugen können. Sein Filmmaterial hätte auch entstehen können, indem er die Szenen, in denen der Eingeborene nicht aus dem richtigen Ausgang hervorkommen konnte, einfach wegschneidet. Obwohl er selbst also überzeugt wurde, ist er nicht in der Lage, anderen seine Aufnahmen als authentisch zu präsentieren. Und so bleibt der Eingeborene schließlich von übermäßigem Trubel in seiner Höhle verschont.

Diese kleine Geschichte demonstriert das Wesentliche eines *Zero Knowledge Proofs*: In einer Interaktion zwischen einem *Prover* P und einem *Verifier* V soll P in der Lage sein, V mit überwältigender Wahrscheinlichkeit von der Wahrheit einer Aussage (modelliert durch „ $x \in L$ “) überzeugen können, jedoch soll V unter keinen Umständen (selbst bei beliebigem Abweichen vom Protokoll) irgend ein zusätzliches Wissen aus der Interaktion ziehen können.

Formal ist es natürlich schwierig, den intuitiven Begriff von „zusätzlichem Wissen“ klar zu definieren. Da bei interaktiven Beweissystemen die beiden Teilnehmer V und P durch Turingmaschinen mit zum Beispiel polynomiellen Laufzeitschranken repräsentiert werden, wird dieses Problem im allgemeinen so gelöst, dass die Berechnungsstärke von V durch die Interaktion *nicht wachsen* darf. Kann V nämlich nach der Interaktion nur Berechnungen durchführen, zu denen er schon vorher in der Lage war, ist das „Wissen“ von V sicherlich nicht angewachsen.

Um also sicherzustellen, dass V keinen Zuwachs an Berechnungsstärke erfährt, schlugen Goldwasser, Micali und Rackoff, als sie das Zero Knowled-

ge Konzept erstmalig in [GMR85] einführen¹, folgenden Ansatz vor: Wir fordern dass alles, was von V nach der Interaktion mit P bei Eingabe x berechnet werden kann, auch direkt aus x berechnet werden können soll. (Wir fordern dies sogar für beliebige V^* , die mit P interagieren. Die *Zero Knowledge Eigenschaft* ist also eine Eigenschaft des Provers P .) Diese Forderung lässt sich durch die Existenz eines *Simulators* ausdrücken: Für jedes V^* welches mit P interagiert, soll es einen Simulator M^* geben, der bei Eingabe x eine Ausgabe produziert, die von $\langle P, V^* \rangle(x)$, also der Kommunikation zwischen V^* und P bei Eingabe x , nicht unterscheidbar ist. Dies führt zu folgender Definition:

Definition 4.1 (Zero Knowledge). *Sei (P, V) ein interaktives Beweissystem für eine Sprache L . Dann hat P die Zero Knowledge Eigenschaft, falls es für jede pp-ITM V^* eine pp-TM M^* gibt, für welche die beiden folgenden Ensembles uniform ununterscheidbar sind:*

- $\{\langle P, V^* \rangle(x)\}_{x \in L}$, also Interaktion von V^* mit P bei gemeinsamer Eingabe x , und
- $\{M^*(x)\}_{x \in L}$, also die Ausgabe von M^* bei Eingabe x , ohne jegliche Interaktion.

Falls $M^(x)$ auf höchstens der Hälfte der Eingaben ein spezielles Symbol \perp ausgibt (d.h. $\Pr[M^*(x) = \perp] \leq 1/2$) und $m^*(x)$ eine ZV ist mit $\Pr[m^*(x) = \alpha] := \Pr[M^*(x) = \alpha \mid M^*(x) \neq \perp]$, dann hat P die Perfect Zero Knowledge Eigenschaft falls die folgenden Ensembles identisch verteilt sind:*

- $\{\langle P, V^* \rangle(x)\}_{x \in L}$, also die Interaktion von V^* mit P bei gemeinsamer Eingabe x , und
- $\{m^*(x)\}_{x \in L}$, also Verteilung der von \perp verschiedenen Ausgaben von M^* bei Eingabe x , ohne jegliche Interaktion.

Der Unterschied bei der Perfect Zero Knowledge Eigenschaft liegt also darin, dass die Verteilungen nicht nur durch beschränkte Beobachter ununterscheidbar sind, sondern in der Tat identisch. Dabei kann durch wiederholte Ausführung des Simulators M^* die Wahrscheinlichkeit, dass er \perp ausgibt, exponentiell klein gemacht werden. Daher ist das spezielle Symbol „ \perp “ im ersten Teil der Definition (engl. auch *Computational Zero Knowledge*) aufgrund der Ununterscheidbarkeits-Definition nicht notwendig.

¹Wie Oded Goldreich in [Gol02] beschreibt, gab es frühe Versionen des Artikels von Goldwasser, Micali und Rackoff bereits 1982, jedoch wurde er dreimal auf großen Konferenzen (FOCS83, STOC84 und FOCS84) abgelehnt, bevor er schließlich veröffentlicht werden konnte.

Wir merken außerdem an, dass die Zero Knowledge Eigenschaft (in beiden Varianten) nur auf Elementen *in* der Sprache definiert ist. Über das Verhalten bei negativen Eingaben wird nichts verlangt.

Die Verwendung eines solchen Simulators wird auch als das *Simulations-Paradigma* bezeichnet. Dieses Prinzip kommt häufig zum Einsatz, wenn Eigenschaften beschrieben werden sollen, bei denen keinerlei Wissen durch eine Interaktion bekannt werden soll: Was auch immer durch einen Teilnehmer mittels einer Interaktion berechnet werden kann, soll von diesem auch selbst ohne Interaktion erhalten werden können. Sicherlich ist dies ein recht restriktiver Ansatz, denn auch „uninteressanter“ Wissenszuwachs wird hier verboten.

4.1.1 Beispiele

Klar ist zunächst, dass jede Sprache $L \in \text{BPP}$ ein (triviales) Zero Knowledge Beweissystem besitzt: Der Prover P tut dabei nichts und V entscheidet die Zugehörigkeit zu L selbst. Damit ist $L(\langle P, V \rangle) = L$ und P hat automatisch die (Perfect) Zero Knowledge Eigenschaft, denn V^* kann als sein eigener Simulator verwendet werden.

Ein etwas komplexeres, und für Zero Knowledge klassisches Beispiel ist das *Graphen Isomorphie Problem*: Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph* ($G_1 \sim G_2$), falls es eine Permutation $\pi : V_1 \rightarrow V_2$ gibt mit $\forall u, v : (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$. Das Graphen Isomorphie Problem GI beschreibt dann die Äquivalenzklassen dieser Relation:

$$GI := \{(G_1, G_2) \mid G_1 \sim G_2\}$$

Für GI geben wir nun ein Zero Knowledge Beweissystem an. (Interessant ist dieses NP-Problem auch aus komplexitätstheoretischer Sicht, weil weder bekannt ist, ob es in P bzw. BPP liegt, noch ob es NP-vollständig ist.)

Die Idee hinter dem Protokoll ist, dass bei zwei gegebenen, isomorphen Graphen G_1 und G_2 eine zufällige Permutation G' eines (zufälligen) der beiden Graphen keinerlei Rückschlüsse zulässt, aus welchem der beiden G_1 oder G_2 er entstand. Isomorph zu *beiden* ist er jedoch nur genau dann, wenn G_1 und G_2 isomorph sind. Aus dieser intuitiven Formulierung ergibt sich folgendes Protokoll:

Protokoll 4.2 (ZK Beweissystem für GI).

Eingabe: Zwei Graphen $G_1 = (X, E_1)$ und $G_2 = (X, E_2)$ auf der Knotenmenge X .

P (Schritt P_1): Wähle eine zufällige Permutation auf X : $\pi \in_R \text{Perm}[X]$.
Permutiere G_1 entsprechend² π : $G' := \pi(G_2)$. Sende G' an V .

V (Schritt V_1): Wähle $\sigma \in_R \{1, 2\}$ und sende σ an P . (Gemeint als Bitte, nun einen Isomorphismus zwischen G_σ und G' zu senden.)

P (Schritt P_2): Sei ϕ ein Isomorphismus³ zwischen G_1 und G_2 .
Sende ψ an V mit:

$$\psi := \begin{cases} \pi \circ \phi, & \text{falls } \sigma = 1 \\ \pi, & \text{falls } \sigma = 2. \end{cases}$$

V (Schritt V_2): Akzeptiere genau dann, wenn $\psi(G_\sigma) = G'$.

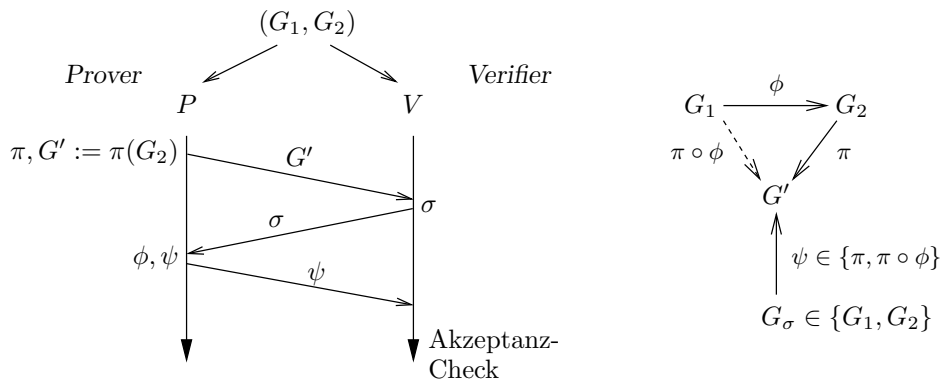


Abbildung 4.2: Links: Zero Knowledge Beweissystem für GI . Rechts ist die Beziehung zwischen den beteiligten Graphen dargestellt.

Theorem 4.3. *Protokoll 4.2 ist ein interaktives (Perfect) Zero Knowledge Beweissystem für GI .*

Beweis. Es sind drei Eigenschaften nachzuweisen:

Completeness: Falls G_1 und G_2 isomorph sind, d.h. $\phi(G_1) = G_2$, dann ist G' zu beiden isomorph. Egal welches $\sigma \in \{1, 2\}$ durch V im Schritt V_1 gewählt wird, ist P durch Kenntnis von π in Schritt P_2 stets in der Lage, das korrekte ψ zu berechnen. Somit wird V in Schritt V_2 stets akzeptieren.

²Für einen Graphen $G = (V, E)$ und eine Permutation $\pi \in \text{Perm}[V]$ sei $\pi(G) := (V, \pi(E))$ mit $\pi(E) := \{(\pi(u), \pi(v)) \mid (u, v) \in E\}$.

³D.h.: $\forall u, v : (u, v) \in E_1 \iff (\phi(u), \phi(v)) \in E_2$.

Soundness: Falls G_1 und G_2 nicht isomorph sind, so gibt es keinen Isomorphismus ϕ zwischen G_1 und G_2 . Egal wie auch G' von einem „böswilligen“ P^* konstruiert wird, gibt es also mit einer Wahrscheinlichkeit von $1/2$ auch kein ψ mit $\psi(G_\sigma) = G'$, da σ von V zufällig gewählt wird.

(Perfect) Zero Knowledge: Zu einer beliebigen pp-ITM V^* welche mit P interagiert, können wir einen Simulator M^* für $\langle P, V^* \rangle$ angeben:

Eingabe: Zwei isomorphe Graphen $G_1 = (X, E_1)$ und $G_2 = (X, E_2)$ auf der Knotenmenge X .

Schritt M_1^* : Wähle $\tau \in_{\mathbb{R}} \{1, 2\}$ und $\pi \in_{\mathbb{R}} \text{Perm}[X]$. Setze $G'' := \pi(G_\tau)$.

Schritt M_2^* : Simuliere V^* in Schritt V_1^* mit Nachricht G'' . Die Ausgabe von V_1^* sei $\sigma \in \{1, 2\}$.

Schritt M_3^* : Falls $\sigma = \tau$, so simuliere V_2^* mit $\psi := \pi$. Andernfalls halte mit Ausgabe \perp .

Diese pp-TM M^* simuliert tatsächlich $\langle P, V^* \rangle$, denn die Verteilung der τ und G'' aus Schritt M_1^* ist stochastisch unabhängig, und insbesondere τ unabhängig von σ , denn V_1^* kennt bei der Festlegung auf σ nur G'' (und kann daraus τ nicht ermitteln). Damit gilt:

- $\Pr[\sigma \neq \tau] = 1/2$, d.h. $\Pr[M^*(G_1, G_2) = \perp] = 1/2$
- Falls $\sigma = \tau$, entspricht die Verteilung der Ausgaben von M^* denen von $\langle P, V^* \rangle$, denn G' im ursprünglichen Protokoll und G'' im Simulator sind identisch auf der Menge der zu G_2 isomorphen Graphen verteilt, und ψ ist stets der angeforderte Isomorphismus. \square

4.1.2 Zero Knowledge mit Privater Eingabe

Dieses Beispiel für GI hat aus praktischer Sicht den Nachteil, dass der Prover P keiner Einschränkung unterliegt. So muss er in der Lage sein, einen Isomorphismus zwischen zwei isomorphen Graphen auch tatsächlich zu bestimmen, und es ist nicht bekannt, ob dies (deterministisch oder probabilistisch) in polynomieller Laufzeit möglich ist. Aus diesem Grunde gesteht man dem Prover eine *private Eingabe* (engl. *Private Input* bzw. *Auxiliary Input*) zu, womit die Prozedur P dann durch eine pp-ITM implementiert werden kann (da der „Zeuge“ ϕ als private Eingabe gegeben werden kann).

Ebenso kann man dem Verifier V eine private Eingabe zugestehen. Dies hat ebenfalls praktische Gründe, schließlich kann so modelliert werden, dass ein Angreifer auf die Zero Knowledge Eigenschaft zusätzliches Vorwissen in den Ablauf des Protokolls einbringt, um mit dessen Hilfe die Eigenschaft leichter zu brechen. Führt man dies also mit in die Definition ein, so wird das resultierende Protokoll robuster:

Definition 4.4 (Zero Knowledge mit privaten Eingaben). Sei (P, V) ein interaktives Beweissystem für eine Sprache L . Sei $P_L(x)$ die Menge der y mit:

$$\Pr[\langle P(y), V(z) \rangle(x) = 1] \geq \frac{2}{3} \text{ für alle } z \in \{0, 1\}^*$$

Dann hat P die Zero Knowledge Eigenschaft mit privaten Eingaben, falls es für jede pp-ITM V^* eine pp-TM M^* gibt, sodass für alle Folgen $y_x \in P_L(x)$ die beiden folgenden Ensembles uniform ununterscheidbar sind:

- $\{\langle P(y_x), V^*(z) \rangle(x)\}_{x \in L, z \in \{0, 1\}^*}$ und
- $\{M^*(x, z)\}_{x \in L, z \in \{0, 1\}^*}$.

Dabei sind die Laufzeiten des Unterscheiders und von M^* polynomiell in $|x|$, und der Unterscheidungs-Erfolg (gemäß Definition 1.3) bezieht sich nur auf hinreichend lange x . Analog für Perfect Zero Knowledge.

Zu dieser Definition ist anzumerken, dass sie nicht-uniforme Ununterscheidbarkeit impliziert: Der Simulator M^* wie auch ein Unterscheider D besitzen jeweils eine in $|x|$ polynomielle Zeitschranke, jedoch wird D erst in Abhängigkeit von M^* angegeben. Die private Eingabe z kann also so lang sein, dass dessen Suffix nur von D , nicht jedoch von M^* gelesen werden kann. Gäbe es also einen nicht-uniformen Unterscheider (eine Schaltkreisfamilie) $C = \{C_n\}$, so ließe sich eine Kodierung von C_n an z via Padding derart anhängen, dass M^* die Kodierung nicht erreicht, jedoch ein uniformer Unterscheider D diese Kodierung lesen könnte. Mit einem Polynomialzeit-Auswerter für Schaltkreise gelänge dann auch diesem uniformen Unterscheider seine Aufgabe. Damit ist also mit dem Ausschluss uniformer Unterscheider in dieser Definition auch der Ausschluss nicht-uniformer Unterscheider impliziert.

4.1.3 Existenz

Nach dem Beispiel für GI interessiert natürlich die Frage, welche Sprachen überhaupt ein Zero Knowledge Beweissystem besitzen. Entsprechend der folgenden Betrachtungen ist es mittels Bit Commitment Verfahren möglich, für jede NP-Sprache ein Zero Knowledge Beweissystem zu konstruieren. Dies ist besonders für praktische Anwendungen interessant, wie wir später sehen werden.

In [GMW91] konstruieren Goldreich, Micali und Wigderson ein Zero Knowledge Beweissystem für das Problem der Graphen-3-Färbbarkeit⁴ $G3C$:

$$G3C = \{G \mid G \text{ ist 3-färbbar}\}$$

⁴Ein Graph $G = (V, E)$ ist k -färbbar, falls es eine Abbildung $\psi : V \rightarrow \{1, \dots, k\}$ mit $\forall (u, v) \in E : \psi(u) \neq \psi(v)$ gibt. Dann heißt ψ auch k -Färbung von G .

Da $G3C$ ein NP-vollständiges Problem ist, kann somit jedes NP-Problem L mittels einer Reduktionsfunktion f_L darauf reduziert werden. Die zusätzliche Eigenschaft dieser Reduktion ist, dass es eine FP-Funktion g_L gibt, für die gilt, dass jedes Paar (x, w) aus der Zeugen-Relation R_L von L durch g_L auf eine 3-Färbung von $f_L(x)$ abgebildet wird. Dies erlaubt schließlich die Konstruktion eines Zero Knowledge Beweissystems für L aus einem für $G3C$: Zu einer Eingabe x und einem Zeugen w , den P als private Eingabe erhält, kann damit die Reduktion auf eine Instanz G von $G3C$ durchgeführt werden und mittels g_L kann P den Zeugen w in eine 3-Färbung des Graphen G umformen. Die Existenz dieser 3-Färbung kann dann in Zero Knowledge (interaktiv) bewiesen werden.

Nun skizzieren wir noch das Zero Knowledge Beweissystem (mit privater Eingabe) für $G3C$ (nach [Gol02]). Vorausgesetzt wird hierzu ein nicht-interaktives Bit Commitment Protokoll C_s mit Seeds s . (Diese Voraussetzung ist nur zur Vereinfachung gewählt – allgemein genügt auch ein interaktives Bit Commitment Protokoll.)

Protokoll 4.5 (ZK Beweissystem für $G3C$).

Eingabe: Ein Graph $G = (X, E)$ mit $X = \{1, \dots, n\}$.

Private Eingabe von P : Eine 3-Färbung $\psi : X \rightarrow \{1, 2, 3\}$ von G .

P (Schritt P_1): Wähle eine Permutation $\pi \in_R \text{Perm}[\{1, 2, 3\}]$ und setze:

$$\forall i : \phi(i) := \pi(\psi(i))$$

Sende dann für jedes dieser i ein Commitment $c_i := C_{s_i}(\phi(i))$ mit zufälligen $s_i \in_R \{0, 1\}^n$ auf die Farbe des Knotens i .

V (Schritt V_1): Wähle zufällig eine Kante $e \in_R E$ und sende sie an P . (Gemeint als Bitte, nun die beiden Farben der beteiligten Knoten zu senden.)

P (Schritt P_2): Für die empfangene Kante $e = (u, v)$ setze $\sigma_u := \phi(u)$ und $\sigma_v := \phi(v)$. Sende (s_u, σ_u) und (s_v, σ_v) als Öffnungen der Commitments c_u und c_v an V .

V (Schritt V_2): Akzeptiere genau dann, wenn die Commitments korrekt geöffnet wurden und $\sigma_u \neq \sigma_v$.

Die Idee des Beweissystems ist folgende: Bei einem 3-färbbaren Graphen ist ϕ eine 3-Färbung und somit erfüllt jede Kante (u, v) , die P in P_2 erfragen

kann, die Bedingung $\phi(u) \neq \phi(v)$. Bei einem nicht 3-färbbaren Graphen besteht jedoch das „Risiko“, dass V nach einer Kante fragt, bei der die beiden adjazenten Knoten gleich gefärbt sind. Die Wahrscheinlichkeit dafür beträgt $1/|E|$ und ist damit nicht vernachlässigbar. Die Idee für die Zero Knowledge Eigenschaft ist, dass ein Simulator die Knoten zufällig färbt, und für jede Kante somit die Wahrscheinlichkeit $2/3$ beträgt, dass die adjazenten Knoten unterschiedliche Färbung haben. Dies genügt für eine hinreichend gute Simulation von $\langle P, V \rangle$.

Zusammen ergibt sich:

Theorem 4.6 ([GMW91]). *Angenommen, es gibt Bit Commitment Protokolle (gegen Schaltkreise). Dann hat jede Sprache $L \in \text{NP}$ ein Zero Knowledge Beweissystem (P, V) mit privater Eingabe. Der Prover P kann als eine pp-ITM implementiert werden, falls P zu positiven Eingaben $x \in L$ einen Zeugen $w \in R_L(x)$ als private Eingabe erhält.*

Dieses wichtige Resultat findet zum Beispiel Anwendung in Situationen, in denen die eine Partei Eigenschaften eines Geheimnisses offenbaren möchte, ohne das Geheimnis selbst preis zu geben. Analog [Gol01, Abschnitt 4.4.3] ist zum Beispiel die Situation vorstellbar, in der S ein Commitment auf einen Wert x gegenüber einem Empfänger R erstellt. Dieses Commitment C kann dargestellt werden als $C = C(x, r_S)$ in Abhängigkeit der von S verwendeten Zufallsbits r_S (siehe Kapitel 3). Zu einem späteren Zeitpunkt, jedoch vor dem endgültigen Öffnen des Commitments, möchte S beweisen, dass $x > z$ für irgend eine Schranke z ist. Die Bedingung „es gibt x und r_S mit $C = C(x, r_S)$ und $x > z$ “ ist offenbar eine NP-Bedingung, zu der S den entsprechenden Zeugen kennt – diesen jedoch nicht verraten will. Betrachte nämlich die folgende Sprache:

$$L := \{(C, z) \mid \exists x, r_S : C = C(x, r_S) \wedge x > z\}$$

Damit ist L die gewünschte NP-Sprache, und S kann mit R einen Zero Knowledge Beweis führen, dass $(C, z) \in L$ tatsächlich gilt. Der Zeuge (x, r_S) ist mittels der Zero Knowledge Eigenschaft geschützt.

4.1.4 Abgeschlossenheit

Eine wichtige Anforderung von kryptographischen Protokollen ist, dass sie ihre jeweiligen Eigenschaften auch dann behalten, wenn sie mehrfach ausgeführt werden – also bei sequentieller oder auch paralleler Wiederholung. Wie die folgenden beiden Theoreme zeigen, bleibt die Zero Knowledge Eigenschaft zwar bei sequentieller Komposition erhalten, jedoch im allgemeinen

nicht bei paralleler. Aufgrund dieses Mankos betrachten wir im nächsten Abschnitt dann die etwas schwächere Eigenschaft der Zeugen-Unterscheidbarkeit, die auch bei paralleler Komposition erhalten bleibt.

Zunächst das *positive Resultat für sequentielle Komposition* für Zero Knowledge mit privaten Eingaben.

Theorem 4.7 ([GO94] und [Gol01]). *Zero Knowledge Beweissysteme mit privater Eingabe sind unter sequentieller Komposition abgeschlossen:*

- Sei P eine pp-ITM mit der Zero Knowledge Eigenschaft mit privater Eingabe.
- Sei $q(\cdot)$ ein Polynom und P_q eine pp-ITM, die bei Eingabe x in $q(|x|)$ Phasen arbeitet und in jeder Phase P mit Eingabe x aufruft.

Dann hat P_q ebenfalls die Zero Knowledge Eigenschaft mit privater Eingabe.

Beweis (Skizze). Ein Verifier V_q^* , der mit P_q interagiert, kann interpretiert werden als wiederholte Ausführung eines Verifiers V^* , der mit einer Phase von P_q interagiert, also mit P . Dieser V^* benötigt jedoch die zusätzliche Eigenschaft, aus seiner privaten Eingabe eine Anfangs-Konfiguration herzustellen und am Ende seiner Berechnungen seine End-Konfiguration auszugeben. Damit entspricht V_q^* also dem in jeder der $q(|x|)$ Phasen erneuten Ausführen von V^* , wobei V^* am Anfang einer Phase $i + 1$ als private Eingabe die Konfiguration bekommt, die V^* in Phase i am Ende hatte. Auf diese Weise lässt sich die Kommunikation von P_q mit V_q^* in die $q(|x|)$ Phasen unterteilen, durch die P_q aus P konstruiert wurde. Für jede Phase existiert jedoch ein Simulator M^* für V^* , da P die Zero Knowledge Eigenschaft mit privater Eingabe hat. Damit lässt sich ein Simulator M_q^* für $\langle P_q, V_q^* \rangle$ konstruieren, wiederum durch wiederholten Aufruf von M^* mit der privaten Eingabe, die auch V^* bekäme. \square

Wir schließen den Abschnitt mit dem negativen Resultat der *fehlenden Abgeschlossenheit unter paralleler Komposition*. Bemerkenswert ist, dass sich noch mehrere Jahre nach Einführung der Zero Knowledge Beweissysteme zunächst die Vermutung hielt, diese seien unter paralleler Komposition abgeschlossen. Dies hat sich jedoch als falsch heraus gestellt:

Theorem 4.8 ([GK96]). *Zero Knowledge Beweissysteme sind unter paralleler Komposition nicht abgeschlossen: Es gibt eine pp-ITM P mit der Zero Knowledge Eigenschaft, die diese Eigenschaft bei paralleler Ausführung von mindestens 2 Instanzen jedoch verliert.*

Beweis (Skizze). Wir konstruieren P wie folgt: Auf eine Eingabe $b \in \{1, 2\}$ gibt es zwei mögliche Verhalten von P :

1. **Fall ($b = 1$):** In diesem Fall stellt P an V eine zufällig gewählte „schwere Frage“, die eine pp-ITM mit höchstens vernachlässigbarer Wahrscheinlichkeit beantworten kann (jedoch P zu beantworten in der Lage ist), und deren Antworten pseudozufällig aussehen (eine sogenannte *pseudozufällige Funktion*). Nur bei korrekter Antwort von V sendet P ein „geheimen Wissen“ an V (z.B. einen Isomorphismus zwischen zwei Graphen o.ä.).
2. **Fall ($b = 2$):** Tritt dieser Fall ein, so bittet P den Verifier V um eine „schwere Frage“ – analog zum ersten Fall, nur dass sich diesmal V diese Frage aussuchen darf. Nach Erhalt dieser Frage sendet P an V die Antwort darauf.

Dieses P hat die Zero Knowledge Eigenschaft, denn in beiden Fällen für sich gesehen kann das Verhalten von P simuliert werden: Im ersten Fall wird V die Frage nicht beantworten können und somit können die Antworten von P (außer mit vernachlässigbarer Wahrscheinlichkeit) stets simuliert werden. Im zweiten Fall sehen die Antworten von P pseudozufällig aus, sind also auch simulierbar.

Wird nun P zweimal instanziiert und parallel ausgeführt, kann ein Angreifer V^* an die eine Instanz $P^{(1)}$ eine 1 schicken, erhält eine Frage, schickt darauf an die zweite Instanz $P^{(2)}$ diese Frage, und verfährt umgekehrt mit der entsprechenden Antwort. Damit erhält der Angreifer V^* stets das „geheimen Wissen“ – dieses Verhalten der Komposition $P^{(1)} \parallel P^{(2)}$ gegenüber V^* kann jedoch nicht simuliert werden. \square

4.2 Witness Indistinguishability

Wir hatten gesehen, dass viele praktisch besonders interessanten Sprachen, also jene aus NP, stets ein Zero Knowledge Beweissystem besitzen (unter kryptographischen Annahmen, Theorem 4.6). Das Konzept der Zero Knowledge Beweissysteme ist ein recht leistungsfähiges, jedoch für den praktischen Einsatz in kryptographischen Protokollen möglicherweise etwas zu restriktiv, denn zuweilen ungeeignet:

- Zero Knowledge Beweissysteme sind nicht abgeschlossen unter allgemeiner Komposition, insbesondere nicht unter paralleler Komposition (Theorem 4.8).

- Die Möglichkeiten, unter denen Zero Knowledge Beweissysteme *nicht-interaktiv* gemacht werden können, sind stark eingeschränkt – insbesondere ist dazu im allgemeinen eine zusätzliche *Setup Voraussetzung* notwendig. (Siehe später Abschnitt 4.3.)

Aus diesen Gründen entwickelten Feige und Shamir in [FS90] das Konzept von *Zeugen-Ununterscheidbarkeit* (engl. *Witness Indistinguishability, WI*) und *Zeugen-Geheimhaltung* (engl. *Witness Hiding, WH*). Die Idee hierbei ist, dass bei interaktiven Beweisen für Zugehörigkeit eines Wortes x zu einer NP-Sprache L insbesondere der vom Prover P verwendete Zeuge $w \in R_L(x)$ geschützt werden soll. Dabei haben beide Konzepte verschiedene Ziele:

Zeugen-Ununterscheidbarkeit: Der Verifier V soll nicht in der Lage sein, anhand des Beweises für $x \in L$ zu erkennen, welcher Zeuge $w \in R_L(x)$ durch den Prover P verwendet wurde – selbst wenn V alle Zeugen kennt.

Zeugen-Geheimhaltung: Der Verifier V soll durch den interaktiven Beweis für $x \in L$ nicht befähigt werden, einen Zeugen $w \in R_L(x)$ effizienter zu bestimmen, als vor der Durchführung des interaktiven Beweises.

Beide Eigenschaften sind also auch wieder Eigenschaften des Provers P und offenbar schwächer als das allgemeinere Konzept von Zero Knowledge, denn dort erhält der Verifier V *keinerlei* zusätzliches Wissen. Wir beschränken uns im Folgenden auf die Betrachtung von *Zeugen-Ununterscheidbarkeit* und geben zunächst eine formale Definition an:

Definition 4.9 (Zeugen-Ununterscheidbarkeit, nach [Gol01]). *Sei (P, V) ein interaktives Beweissystem für eine Sprache $L \in \text{NP}$. Sei außerdem R_L eine Zeugen-Relation für L . Dann hat P die Eigenschaft der Zeugen-Ununterscheidbarkeit für R_L , falls für jede pp-ITM V^* und alle Folgen $W^1 = \{w_x^1\}_{x \in L}$ und $W^2 = \{w_x^2\}_{x \in L}$ von Zeugen $w_x^1, w_x^2 \in R_L(x)$ die folgenden Zufalls-Ensembles nicht-uniform ununterscheidbar sind:*

- $\{\langle P(w_x^1), V^*(z) \rangle(x)\}_{x \in L, z \in \{0,1\}^*}$ und
- $\{\langle P(w_x^2), V^*(z) \rangle(x)\}_{x \in L, z \in \{0,1\}^*}$

Wir nennen (P, V) dann auch ein WI-Beweissystem.

Zunächst sieht man direkt, dass Zeugen-Ununterscheidbarkeit eines interaktiven Beweissystems (P, V) für eine Sprache $L \in \text{NP}$ von der Zero Knowledge Eigenschaft (mit privaten Eingaben) impliziert wird: Die Verteilung für

jeden Zeugen wird durch den (selben) Simulator M^* für jeden Verifier V^* approximiert. Durch die Transitivität der Ununterscheidbarkeits-Relation folgt die Zeugen-Ununterscheidbarkeit.

Die Umkehrung gilt jedoch nicht: Für jedes R_L mit eindeutigen Zeugen für jedes $x \in L$ ist ein entsprechendes Beweissystem offenbar trivial Zeugen-ununterscheidbar, da es stets nur einen einzigen Zeugen gibt. (Besonders interessant sind damit also WI-Beweissysteme für rechts-mehrdeutige Zeugen-Relationen R_L .) Es muss jedoch nicht Zero Knowledge sein: Sei zum Beispiel f eine Einweg-Permutation (also eine längenerhaltende, injektive Einwegfunktion), und $\{0, 1\}^*$ sei charakterisiert als NP-Sprache durch die folgende Zeugen-Relation:

$$R_L = \{(f(w), w) \mid w \in \{0, 1\}^*\}$$

Damit ist zu jedem $x \in \{0, 1\}^*$ der Zeuge w mit $f(w) = x$ eindeutig bestimmt. Ein Beweissystem (P, V) , in dem P für jedes x genau dieses w an V sendet, ist damit (trivial) Zeugen-ununterscheidbar, jedoch nicht Zero Knowledge, denn V ist damit in der Lage, f zu invertieren.

Wir hatten gesehen, dass Zero Knowledge Beweissysteme diese Eigenschaft unter *sequentieller Komposition* behalten. Analog gilt das auch für WI-Beweissysteme. Eine wichtige Eigenschaft von diesen ist jedoch, dass sie auch unter *paralleler Komposition* abgeschlossen sind:

Theorem 4.10 ([Gol01]). *Zeugen-ununterscheidbare Beweissysteme sind unter paralleler Komposition abgeschlossen:*

- Sei $L \in \text{NP}$, R_L eine Zeugen-Relation von L und (P, V) ein interaktives Beweissystem, welches Zeugen-ununterscheidbar für R_L ist und wobei P und V pp-ITMs sind.
- Sei $q(\cdot)$ ein Polynom und

$$\begin{aligned}\bar{x} &= (x_1, \dots, x_{q(n)}) \in \{0, 1\}^n \times \dots \times \{0, 1\}^n, \\ \bar{w} &= (w_1, \dots, w_{q(n)}) \in \{0, 1\}^* \times \dots \times \{0, 1\}^*\end{aligned}$$

- Sei P_q eine pp-ITM, die bei gemeinsamer Eingabe \bar{x} und privater Eingabe \bar{w} die pp-ITM P in $q(n)$ parallelen Instanzen jeweils auf x_i mit privater Eingabe w_i aufruft.

Dann ist P_q Zeugen-ununterscheidbar für:

$$R_L^q := \{(\bar{x}, \bar{w}) \mid \forall i : (x_i, w_i) \in R_L\}$$

Beweis (Skizze). Gibt es einen Verifier V_q^* , eine Eingabe \bar{x} und zwei Zeugen \bar{w}^1, \bar{w}^2 , sodass V_q^* Interaktionen mit P_q bei Verwendung jeweils der Zeugen \bar{w}^1 und \bar{w}^2 unterscheiden kann, so gilt dies nach einem Hybrid-Argument (analog dem Beweis zu Lemma 3.12) auch für zwei Hybride $\bar{h}^{(i)}$ und $\bar{h}^{(i+1)}$ mit:

$$\bar{h}^{(j)} := (w_1^1, \dots, w_j^1, w_{j+1}^2, \dots, w_{q(n)}^2)$$

Damit kann ein V^* konstruiert werden, der mit P interagiert, wobei V^* dann Interaktionen bei denen P als Zeugen w_{i+1}^1 bekommt von denen mit dem Zeugen w_{i+1}^2 unterscheiden kann. Dazu simuliert V^* sowohl $q(n) - 1$ Instanzen von P parallel (dazu ist essentiell, dass P eine pp-ITM ist), als auch deren Interaktion mit V_q^* . Allein die $(i + 1)$ -te Instanz von P wird nicht simuliert, sondern stattdessen mit P kommuniziert. Durch die Unterscheidbarkeit der Hybride folgt dann die Unterscheidbarkeit der beiden Zeugen.

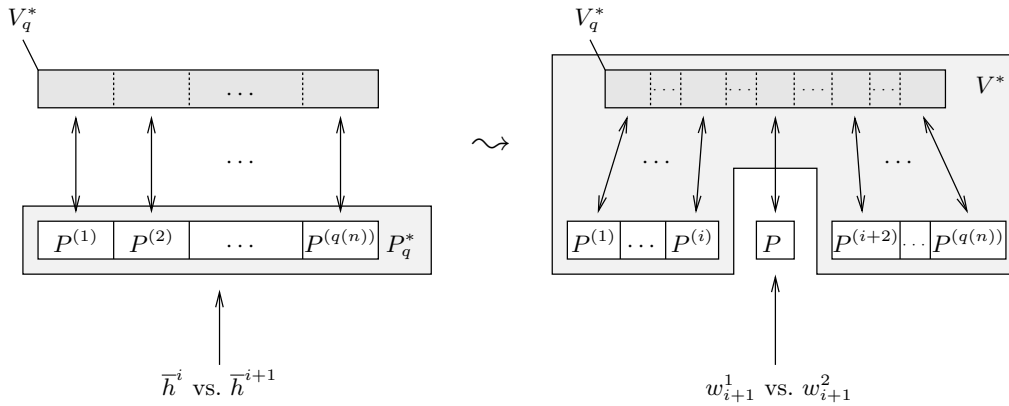


Abbildung 4.3: Verwendung von V_q^* , der die Zeugen \bar{h}^i und \bar{h}^{i+1} mittels P_q^* unterscheiden kann zur Konstruktion von V^* der die Zeugen w_{i+1}^1 und w_{i+1}^2 mittels P unterscheiden kann.

□

Dieses Resultat ist wichtig für die Reduzierung von Interaktion, denn um die Fehlerwahrscheinlichkeit zu verringern, werden oft wenige Runden eines interaktiven Beweises wiederholt ausgeführt (zum Beispiel in Protokoll 4.5 für $G3C$). Können diese Wiederholungen nun parallel statt sequenziell stattfinden, so erhöht sich nicht die nötige Anzahl der Runden, und damit bleibt die Interaktion beschränkt. Bevor wir uns in den folgenden Abschnitten nun der Frage zuwenden, wie denn Interaktion minimiert werden kann, folgt damit aus Theorem 4.10 analog Theorem 4.6 zunächst das folgende Resultat für Zeugen-Ununterscheidbarkeit:

Korollar 4.11. *Angenommen, es gibt Bit Commitment Protokolle (gegen Schaltkreise). Dann hat jede Sprache $L \in \text{NP}$ ein WI-Beweissystem (P, V) mit konstanter Anzahl Runden.*

4.3 Nicht-interaktives Zero Knowledge

Nach den grundsätzlichen Betrachtungen der verschiedenen Modelle wenden wir uns nun der Untersuchung zu, auf welches Mindestmaß an Interaktion Beweissysteme mit Geheimhaltungs-Eigenschaften begrenzt werden können. In diesem Abschnitt betrachten wir zunächst erneut die Zero Knowledge Beweissysteme, also diejenigen mit der stärkeren Anforderung an die Geheimhaltung.

Minimal ist der Aufwand an Interaktion sicherlich dann, wenn Kommunikation nur in eine Richtung stattfindet – in dem Fall spricht man auch von einem *nicht-interaktiven* Protokoll. Im Hinblick auf die Zero Knowledge Eigenschaft gibt es in dieser Hinsicht zunächst das negative Resultat, dass Interaktion notwendig ist, um ein Zero Knowledge Beweissystem für nicht-triviale Sprachen (d.h. Sprachen $L \notin \text{BPP}$) anzugeben. (Den Fall $L \in \text{BPP}$ bezeichnen wir hier als trivial, da hierzu kein Prover P nötig ist und V die Sprachzugehörigkeit selbst entscheiden kann – wodurch das „Protokoll“ offensichtlich die Zero Knowledge Eigenschaft hat.)

Theorem 4.12 ([GO94]). *Sei L eine Sprache mit einem Zero Knowledge Beweissystem (P, V) , bei dem nur eine einzige Nachricht ausgetauscht wird. Dann ist $L \in \text{BPP}$.*

Beweis. Sei α der String, den P an V schickt, und seien r die Zufallsbits, die die pp-ITM V verwendet. (Insbesondere wird α unabhängig von r gewählt.) Damit kann die Akzeptanz des Protokolls als ein deterministisches P-Prädikat $\rho(x, \alpha, r)$ ausgedrückt werden, wobei x die gemeinsame Eingabe von P und V ist (und damit das zu entscheidende Wort). Wir bezeichnen (x, α, r) als die *Konversation*, und es gilt:

$$\begin{aligned} x \in L &\Rightarrow \exists \alpha : \Pr_r[\rho(x, \alpha, r) = 1] \geq 2/3 \\ x \notin L &\Rightarrow \forall \alpha : \Pr_r[\rho(x, \alpha, r) = 1] \leq 1/3 \end{aligned}$$

Sei o.B.d.A. V so, dass er seine Zufallsbits r ausgibt. Die Zero Knowledge Eigenschaft garantiert nun einen Simulator M für $\langle P, V \rangle$. Unser Ziel ist, diesen Simulator zu benutzen, um L mit einer pp-TM zu entscheiden. Der Simulator erzeugt ebenfalls eine Konversation (x, α, r) , wobei diese im Fall $x \in L$ von derjenigen ununterscheidbar ist, die $\langle P, V \rangle$ erzeugt. In diesem

Fall wird also bis auf endlich viele x die Akzeptanzwahrscheinlichkeit nur vernachlässigbar von der von $\langle P, V \rangle$ abweichen.

Ist jedoch $x \notin L$, so ist von der Ausgabe von M nichts garantiert. Zwar darf es in der Konversation (x, α, r) von P und V zu jedem α nur $1/3$ der r geben, die zu $\rho(x, \alpha, r) = 1$ führen, aber es könnte sein, dass M nur solche r erzeugt, für die ρ zu 1 evaluiert. Wir müssen daher selbst Zufallsbits r' wählen und die Auswertung von ρ selbst vornehmen – dies funktioniert dann auch für den Fall $x \in L$.

Unser Entscheider M' für L wird also wie folgt funktionieren. Sei dabei $p(\cdot)$ ein Polynom, welches die Anzahl der Zufallsbits von V beschränkt.

- Eingabe: x
- Simuliere M mit Eingabe x . Falls M verwirft, so verwerfe.
- Falls M akzeptiert, so sei (x, α, r) die von M produzierte Konversation.
- Wähle $r' \in_{\mathbb{R}} \{0, 1\}^{p(|x|)}$.
- Akzeptiere genau dann, wenn $\rho(x, \alpha, r') = 1$.

Da im Fall $x \in L$ das α der Konversation auf mindestens (bzw. nur vernachlässigbar weniger als) $2/3$ aller $r' \in_{\mathbb{R}} \{0, 1\}^{p(|x|)}$ zu Akzeptanz führt, ist der Algorithmus in diesem Fall korrekt. Im Fall $x \notin L$ gibt es kein α , bei dem mehr als $1/3$ der $r' \in_{\mathbb{R}} \{0, 1\}^{p(|x|)}$ zu Akzeptanz führen, also entscheidet der Algorithmus auch in diesem Fall richtig. Damit gilt tatsächlich $L = L(M')$. Da M' offenbar eine pp-TM ist, folgt $L \in \text{BPP}$. \square

Dieses negative Resultat führt dazu, dass das Modell für *Nicht-interaktives Zero Knowledge* mit einer zusätzlichen *Setup Voraussetzung* definiert wird (nach [BSMP91]): mit einem von beiden Parteien verwendbaren und als zufällig vorausgesetzten *Referenz-Zufallsstring* σ . Da im nicht-interaktiven Sonderfall von interaktiven Beweissystemen der Verifier V das Verhalten des Provers P nicht beeinflussen kann, kann die Definition so weit vereinfacht werden, dass der nötige Simulator M nicht spezifisch für einen besonderen Verifier existieren muss, sondern gleich den allgemeinen Fall abdeckt.

Besonders interessiert sind wir an Beweissystemen für NP-Sprachen, daher betrachten wir hier eine Definition, in der der Prover P zwar polynomiell in der Laufzeit beschränkt ist, jedoch dafür einen Zeugen w aus der Zeugenmenge $R_L(x)$ für ein Element $x \in L$ bekommt. Die Definition ist dann wie folgt:

Definition 4.13 (Nicht-interaktives Zero Knowledge, NIZK).

1. Ein Paar (P, V) aus einer pp-TM P und einer P-TM V ist ein nicht-interaktives Beweissystem (NI-Beweissystem) für $L \in \text{NP}$, falls gilt:

Completeness: Für jedes $x \in L$ und $w \in R_L(x)$ gilt:

$$\Pr_{\sigma, P}[V(x, \sigma, P(x, w, \sigma)) = 1] \geq 2/3$$

Soundness: Für jedes $x \notin L$ und jede TM B gilt:

$$\Pr_{\sigma}[V(x, \sigma, B(x, \sigma)) = 1] \leq 1/3$$

Dabei wird $\sigma \in_R \{0, 1\}^{\text{poly}(|x|)}$ gewählt und heißt der Referenz-Zufallsstring.

2. Ein NI-Beweissystem (P, V) hat die Zero Knowledge Eigenschaft, falls es ein Polynom $p(\cdot)$ und eine pp-TM M gibt, für welche die beiden folgenden Ensembles nicht-uniform ununterscheidbar sind:

- $\{(x, U_{p(|x|)}, P(x, w, U_{p(|x|)}))\}_{x \in L, w \in R_L(x)}$, also ein Tupel aus Eingabe x , Referenz-Zufallsstring $U_{p(|x|)}$, und der Prover-Ausgabe auf diesen beiden Argumenten und einem Zeugen $w \in R_L(x)$, und
- $\{M(x)\}_{x \in L, w \in R_L(x)}$, also die Ausgabe von M bei Eingabe x .

Wir bemerken, dass die *nicht-interaktive* Zero Knowledge Eigenschaft bei paralleler Komposition erhalten bleibt, denn der Simulator M kann dann auch einfach parallel ausgeführt werden, um eine ununterscheidbare Simulation zu erzeugen. (Voraussetzung ist, dass jede Instanz „frische“ Zufallsbits σ benutzt.) Hieraus folgt unter anderem, dass die Fehlerwahrscheinlichkeit bei positiven und auch negativen Eingaben von $1/3$ auf exponentiell klein gesenkt werden kann, indem polynomiell viele parallele Instanzen des Beweissystems gestartet werden.

Es ist bekannt, dass sich NIZK Beweissysteme für alle Sprachen in NP konstruieren lassen, wenn Einwegpermutationen existieren. Dabei kann der Prover P effizient implementiert werden (wie in unserer Definition verlangt), falls die Voraussetzung auf Trapdoor-Einwegpermutationen erweitert wird. Ohne dieses Objekt formal zu definieren, können wir Trapdoor-Einwegfunktionen beschreiben als Einwegfunktionen, für die bei der Konstruktion der Funktion eine zusätzliche Information entsteht, die das effiziente Invertieren erlaubt. (Daher werden hier genaugenommen Familien anstelle einer Funktion betrachtet.) In diesem Falle muss P dann die verwendete Funktion nicht selbst umkehren können, sondern benutzt die „Falltür“ dazu. Wir fassen das Resultat zusammen, welches obendrein eine perfekte Completeness garantiert:

Theorem 4.14 ([KP98], [FLS00], [GOS05] et al.). *Falls Trapdoor-Einwegpermutationen existieren, so existiert zu jeder NP-Sprache ein nicht-interaktives Zero Knowledge Beweissystem mit perfekter Completeness.*

4.4 Zaps

Für praktische Anwendungen kann die *Setup Voraussetzung* der NIZK Beweissysteme ein Hindernis sein, etwa wenn beide Teilnehmer keinen Zugang zu einer gemeinsamen Zufallsquelle haben. Um dennoch ein Beweissystem mit Geheimhaltungseigenschaft jedoch möglichst wenig Interaktion für jede Sprache in NP zu erhalten, schlugen Dwork und Naor in [DN00] sogenannte *Zaps* vor. Ein Zap ist ein interaktives public-coin Beweissystem in 2 Runden (d.h. mit 2 ausgetauschten Nachrichten), welches die Eigenschaft der Zeugen-Ununterscheidbarkeit hat. Außerdem ist die Soundness-Bedingung adaptiv, d.h. gilt sogar wenn das zu beweisende x erst *nach* der ersten Nachricht gewählt wird. Wir fassen dies konkret noch einmal in folgender Definition zusammen:

Definition 4.15 (Zap, nach [DN00]). *Ein Zap für eine Sprache $L \in \text{NP}$ ist ein Paar (P, V) von pp-ITMs, die ein 2-Runden Protokoll implementieren, für das gilt:*

Eingabe: *Die gemeinsame Eingabe ist 1^n (d.h. die unäre Länge von x), und P bekommt $x \in L_n$ und $w \in R_L(x)$ als private Eingabe.*

1. Runde: *V sendet an P einen String $\rho \in_R \{0, 1\}^{\text{poly}(n)}$. (Diesen Teil bezeichnen wir mit V_1 .)*

2. Runde: *P sendet an V das Wort x und außerdem eine Antwort $\pi = \pi(x, w, \rho)$. (Diese ZV hängt auch von den Zufallsentscheidungen von P ab, denn P ist probabilistisch.)*

Akzeptanz: *V entscheidet die Akzeptanz deterministisch, d.h. in Abhängigkeit von (x, ρ, π) , ohne erneute Zufallsentscheidungen. (Diesen deterministischen Teil bezeichnen wir mit V_2 .)*

Completeness: *Für jedes $x \in L$ und $w \in R_L(x)$ gilt:*

$$\Pr_{\rho, P}[V_2(x, \rho, \pi) = 1] \geq 2/3$$

Soundness: *Es gilt⁵:*

$$\Pr_{\rho}[\exists x' \in \overline{L}_n, \exists \pi' : V_2(x', \rho, \pi') = 1] \leq 1/3$$

⁵Dabei schreiben wir \overline{L}_n für $\{0, 1\}^n \setminus L$

Zeugen-Ununterscheidbarkeit: Für alle Folgen $\{w_x^1\}_{x \in L}$ und $\{w_x^2\}_{x \in L}$ von Zeugen $w_x^1, w_x^2 \in R_L(x)$ und $\{\rho_x\}_{x \in L}$ von Zufallsstrings sind die folgenden Ensembles nicht-uniform ununterscheidbar:

$$\{\pi(x, w_x^1, \rho_x)\}_{x \in L} \text{ und } \{\pi(x, w_x^2, \rho_x)\}_{x \in L}$$

Um die Soundness-Bedingung zu nutzen, an der V gelegen ist, wird V also den String ρ zufällig wählen. In obiger Definition ist diese Bedingung besonders stark, denn die Eingabe darf ein Angreifer (auf der Seite von P) nach ρ , also *adaptiv* wählen. Ähnliches gilt für die Bedingung der Zeugen-Ununterscheidbarkeit: Diese muss auch gelten, wenn ρ durch einen Angreifer (auf der Seite von V) adaptiv, d.h. nicht zufällig gewählt ist, insbesondere also auch in Abhängigkeit von x .

Ein wichtiges Resultat von [DN00] ist nun, dass sich für jede Sprache $L \in \text{NP}$ ein Zap aus einem nicht-interaktiven Zero Knowledge Beweissystem für L konstruieren lässt:

Theorem 4.16 ([DN00]). Sei $L \in \text{NP}$. Falls L ein NIZK Beweissystem besitzt, so existiert auch ein Zap für L .

Wir bemerken, dass die Soundness im zugrunde gelegten NIZK Beweissystem *nicht-adaptiv* ist, also eine schwächere Form.

Beweis. Für $L \in \text{NP}$ ist ein NIZK vorausgesetzt und soll in ein Zap transformiert werden. Die Aufgabe im Beweis besteht darin, die Setup Voraussetzung in Form des Referenz-Zufallsstrings σ im NIZK für $L \in \text{NP}$ zu entfernen, wozu jedoch ein zusätzlicher Schritt der Interaktion benutzt werden darf. Dabei muss sich der Ersatz für σ durch Beiträge von beiden Parteien ergeben: Würde nur V den String σ wählen, so wäre ein böswilliger V^* möglicherweise in der Lage, die Wahl derart zu treffen, dass Informationen über den Zeugen w offenbar werden, und so die Zeugen-Ununterscheidbarkeit verletzt. Würde jedoch nur P den String σ wählen, könnte dies zur Verletzung der Soundness-Eigenschaft führen, wenn P geschickt ein σ so wählt, um für ein $x \notin L$ den Verifier vom Gegenteil überzeugen zu können.

Dieses Problem besteht auch dann, wenn beide Parteien einen String wählen – sagen wir, V wählt B und P wählt C – und σ dann als Kombination von beiden gesetzt würde: $\sigma := B \oplus C$. Da nämlich P seinen String C erst nach Erhalt von B wählen muss (um anschließend das NIZK zu simulieren), kann P damit den Wert von σ dennoch beliebig bestimmen.

Der Ausweg hierzu ist, das Protokoll in vielen Instanzen *parallel* ablaufen zu lassen. Für jede Instanz j wählt V ein neues B_j , jedoch wählt P nur

einmalig ein C . Diese werden dann jeweils zu $\sigma_j := B_j \oplus C$ verknüpft und für die Simulation des gegebenen NIZK Beweissystems verwendet.

Zur Vereinfachung der formalen Notation übernehmen wir einige Bezeichnungen von [DN00]: Die Verteilung der Nachrichten, die der Prover des NIZK Beweissystems bei Eingabe x , Zeugen w und Referenz-Zufallsstring σ generiert, sei $\tilde{P}N(x, w, \sigma)$. Sei außerdem \tilde{V} der Verifier des NIZK Beweissystems. Für $x \in L$, $w \in R_L(x)$ und zufälliges σ ist also $\pi \in_R \tilde{P}N(x, w, \sigma)$ eine Nachricht, die mit hoher Wahrscheinlichkeit zu $\tilde{V}(x, \sigma, \pi) = 1$ führt. Sei für Eingabe x außerdem l die Anzahl der Zufallsbits, die im NIZK verwendet werden. Wir bezeichnen mit m die Anzahl der parallelen Instanzen des NIZK die wir im Zap aufrufen wollen und mit $k = m \cdot l$ die Anzahl der Zufallsbits, die dazu generiert werden müssen. (Den Wert von m legen wir später fest.) Wir geben nun das Protokoll an:

Protokoll 4.17 (Zap-Konstruktion nach [DN00]).

V (Schritt V_1): Wähle $\rho = b_1 \dots b_k \in_R \{0, 1\}^k$ und sende ρ an P .

P: Interpretiere ρ als $B_1 \dots B_m$ mit $B_j \in \{0, 1\}^l$ für alle j . Wähle dann $C \in_R \{0, 1\}^l$ und für alle j :

$$\pi_j \in_R \tilde{P}N(x, w, \sigma_j) \text{ mit } \sigma_j := B_j \oplus C$$

Sende schließlich x , C und $\pi = (\pi_1, \dots, \pi_m)$ an V .

V (Schritt V_2): Setze ebenfalls $\sigma_j := B_j \oplus C$ und akzeptiere, falls:

$$\forall j = 1, \dots, m : \tilde{V}(x, \sigma_j, \pi_j) = 1$$

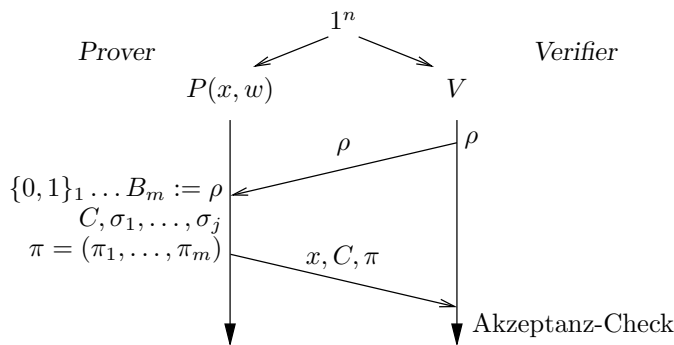


Abbildung 4.4: Zap-Konstruktion aus Protokoll 4.17: Der NIZK-Prover wird m mal parallel instanziiert.

Wir müssen nun die drei Eigenschaften *Completeness*, *Soundness* und *Zeugen-Ununterscheidbarkeit* von Protokoll 4.17 nachweisen.

Completeness: Falls $x \in L$ Eingabe des Protokolls ist, und sich P und V an das Protokoll halten, so sind alle σ_j gleichverteilt und unabhängig aus $\{0, 1\}^l$ gewählt. Für jedes $\pi_j \in_{\mathbb{R}} \tilde{PN}(x, w, \sigma_j)$ gilt damit:

$$\Pr[\tilde{V}(x, \sigma_j, \pi_j) = 1] \geq 1 - \epsilon$$

Laut Definition ist zwar $\epsilon = 1/3$, jedoch kann die Fehlerwahrscheinlichkeit sogar als exponentiell klein angenommen werden, d.h. $\epsilon = 1/2^{|x|}$. Die Fehlerwahrscheinlichkeit für jeden „Block“ j beträgt damit nur ϵ und für hinreichend langes $|x|$ klappt damit der Test in V_2 auch mit einer Wahrscheinlichkeit der notwendigen $2/3$.

Hat das zugrunde liegende NIZK perfekte Completeness (d.h. $\epsilon = 0$), so hat das resultierende Zap Protokoll diese ebenfalls.

Soundness: Sei q die Fehlerschranke für die Soundness des NIZK Beweissystems:

$$\forall x \notin L : \Pr_{\sigma}[\exists \pi' : \tilde{V}(x, \sigma, \pi') = 1] \leq q$$

Wir möchten zeigen, dass für $\delta = 1/3$ gilt:

$$\Pr_{B_1, \dots, B_m}[\exists x \in \overline{L_n}, \exists (C, \pi_1, \dots, \pi_m) : \underbrace{\forall j : \tilde{V}(x, B_j \oplus C, \pi_j) = 1}_{V_2(x, B_1 \dots B_m, (C, \pi))=1}] \leq \delta$$

Diese Wahrscheinlichkeit kann man wie folgt umschreiben und umformen:

$$\begin{aligned} & \Pr_{B_1, \dots, B_m}[\exists x \in \overline{L_n}, \exists C : (\forall j = 1, \dots, m : \exists \pi_j : \tilde{V}(x, B_j \oplus C, \pi_j) = 1)] \\ &= \Pr_{B_1, \dots, B_m} \left[\bigvee_{\substack{x \in \overline{L_n} \\ C \in \{0,1\}^l}} \bigwedge_j \exists \pi_j : \tilde{V}(x, B_j \oplus C, \pi_j) = 1 \right] \\ &\leq \sum_{\substack{x \in \overline{L_n} \\ C \in \{0,1\}^l}} \Pr_{B_1, \dots, B_m} \left[\bigwedge_j \exists \pi_j : \tilde{V}(x, B_j \oplus C, \pi_j) = 1 \right] \\ &= 2^l \sum_{x \in \overline{L_n}} \Pr_{B_1, \dots, B_m} \left[\bigwedge_j \exists \pi_j : \tilde{V}(x, B_j, \pi_j) = 1 \right] \\ &= 2^l \sum_{x \in \overline{L_n}} \prod_{j=1}^m \underbrace{\Pr_{B_j}[\exists \pi_j : \tilde{V}(x, B_j, \pi_j) = 1]}_{\leq q} \\ &\leq 2^{l+n} q^m \quad \text{da } |\overline{L_n}| \leq 2^n \end{aligned}$$

Die gesuchte Wahrscheinlichkeit ist also beschränkt durch $2^{l+n}q^m$. Wird m hinreichend groß gewählt, also so groß dass $q^m < \delta/2^{l+n}$, wird damit δ unterschritten. (Insbesondere bleibt m damit polynomiell in $|x|$.)

Zeugen-Ununterscheidbarkeit: Zum Zwecke eines indirekten Beweises seien $\{w_x^1\}_{x \in L}$ und $\{w_x^2\}_{x \in L}$ zwei Folgen von Zeugen $w_x^1, w_x^2 \in R_L(x)$ und $\{\rho_x\}_{x \in L}$ eine Folge von Zufallsstrings, für welche die Ausgaben von P aus Protokoll 4.17 nicht-uniform unterscheidbar sind. Fixieren wir zur Vereinfachung für den Rest des Beweises ein x und damit $w = w_x^1$, $w' = w_x^2$ und $\rho = \rho_x$, und schreiben wir wieder $\rho = B_1 \dots B_m$, so geht es also um die Unterscheidbarkeit von:

$$\begin{aligned} &(\pi_1, \dots, \pi_m) \text{ mit } \forall i : \pi_i \in_{\mathbb{R}} \tilde{P}N(x, w, B_j \oplus C) \text{ und} \\ &(\pi'_1, \dots, \pi'_m) \text{ mit } \forall i : \pi'_i \in_{\mathbb{R}} \tilde{P}N(x, w', B_j \oplus C) \text{ für ein } C \in_{\mathbb{R}} \{0, 1\}^l. \end{aligned}$$

Aus der Unterscheidbarkeit dieser Tupel folgt nach einem Hybrid-Argument die Unterscheidbarkeit zweier Hybride

$$h^{(j-1)} = (\pi_1, \dots, \pi_{j-1}, \pi'_j, \dots, \pi'_m) \text{ und } h^{(j)} = (\pi_1, \dots, \pi_j, \pi'_{j+1}, \dots, \pi'_m)$$

für ein $1 \leq j \leq m$.

Sei also D ein polynomieller Schaltkreis, der $h^{(j-1)}$ und $h^{(j)}$ unterscheidet. Wir werden D nun verwenden, um einen Unterscheider für $\tilde{P}N(x, \cdot, \sigma)$ zu konstruieren, also den Zeugen-Eingang des Provers vom NIZK auf dem das Zap basiert. Da dieser Prover die Zero Knowledge Eigenschaft hat, führt dies zum Widerspruch.

Zunächst konstruieren wir das zu unterscheidende Objekt: Wir wählen ein τ gleichverteilt zufällig aus $\{0, 1\}^l$ und $u \in_{\mathbb{R}} \{w, w'\}$. Für diesen Zeugen u lassen wir einen Beweis durch das NIZK Beweissystem erzeugen: $\pi \in_{\mathbb{R}} \tilde{P}N(x, u, \tau)$.

Nun werden wir aus π feststellen, ob $u = w$ oder $u = w'$ gewählt war, was die Zeugen-Ununterscheidbarkeit des NIZK verletzt. Dazu setzen wir $C := \tau \oplus B_j$. (Beachte, dass wir ein fixiertes j betrachten.) Damit ist $B_j \oplus C$ gleichverteilt zufällig aus $\{0, 1\}^l$. Für $i < j$ wählen wir nun $\tilde{\pi}_i \in_{\mathbb{R}} \tilde{P}N(x, w, B_j \oplus C)$, und für $i > j$ wählen wir $\tilde{\pi}_i \in_{\mathbb{R}} \tilde{P}N(x, w', B_j \oplus C)$. Mit $\tilde{\pi}_j := \pi$ haben wir nun also eine Folge $(\tilde{\pi}_1, \dots, \tilde{\pi}_m)$, die in ihrer Verteilung entweder $h^{(j)}$ entspricht (falls $u = w$ gewählt war), oder aber $h^{(j-1)}$ (falls $u = w'$ gewählt war). Wir können D für diese Unterscheidung nutzen und damit den Wert von u mit nicht-vernachlässigbarer Wahrscheinlichkeit erraten.

□

4.5 Derandomisierung von Zaps

Wir hatten nun also konstruktiv gesehen, dass die Existenz von NIKZ Beweissystemen die Existenz von Zaps impliziert. (In [DN00] zeigen die Autoren sogar, dass auch die Umkehrung gilt, falls Einwegfunktionen gegen Schaltkreise existieren.) Reihen wir die Implikationen aneinander, also Theoreme 4.14 und 4.16, so haben wir damit unter der Voraussetzung von Falltür-Funktionen WI-Beweissysteme für jede NP-Sprache in nur 2 Runden ohne jegliche Setup Voraussetzung. In diesem letzten Abschnitt wird es nun darum gehen, auch diese letzte Interaktion nach dem Resultat in [BOV07] mittels eines Hitting Set Generators zu beseitigen und somit ein nicht-interaktives Beweissystem für jede NP-Sprache zu erhalten, welches die Zeugen-Ununterscheidbarkeit erfüllt.

Dazu ist folgende Beobachtung zentral. Nach Definition 4.15 gilt für den deterministischen Teil V_2 des Verifiers:

$$\Pr_\rho[\exists x' \in \overline{L_n}, \exists \pi' : V_2(x', \rho, \pi') = 1] \leq 1/3$$

Dies lässt sich abschwächen in:

$$\forall x' \in \overline{L_n} : \Pr_\rho[\exists \pi' : V_2(x', \rho, \pi') = 1] \leq 1/3$$

(Dies ist die „nicht-adaptive“ Variante der Soundness, bei der x' unabhängig von ρ gewählt werden muss. Dies ist offenbar eine schwächere Bedingung, jedoch soll das nicht-interaktive System, welches wir erhalten, auch nur eine solche nicht-adaptive Soundness erfüllen.)

Das heißt also, dass für jedes $x' \in \overline{L_n}$ und den überwiegenden Anteil aller ρ kein π' existiert mit $V_2(x', \rho, \pi') = 1$. Solche ρ nennen wir *sound bezüglich x* . Die Bedingung, dass ρ sound bezüglich x' ist, kann auch wie folgt formuliert werden:

$$\rho \text{ ist sound bezüglich } x' \iff \forall \pi' : V_2(x', \rho, \pi') = 0$$

Dies kann offenbar mittels eines *co-nichtdeterministischen* Tests in polynomieller Zeit getestet werden. Sei $p(\cdot)$ ein diesen Test in der Laufzeit beschränkendes Polynom. Dann gibt es für jedes $x' \in \overline{L_n}$ einen co-nichtdeterministischen Schaltkreis $C_{x'}$ von höchstens Größe $k \cdot p(|x|)^2$ für ein $k > 0$, der eine Eingabe ρ genau dann akzeptiert, wenn ρ sound bezüglich x' ist. Dieser Schaltkreis akzeptiert damit den überwiegenden Teil seiner Eingaben⁶, insbesondere also mindestens 1/2.

⁶Zwar ist der Anteil hier sogar mindestens 2/3, jedoch wird die Konstruktion gleich allgemein für jede Soundness-Fehlerschranke $\delta < 1/2$ betrachtet, sodass wir hier nur von einer Akzeptanz von mindestens 1/2 der Eingaben ausgehen.

Wie schon in Kapitel 3 zur Derandomisierung des Bit Commitment Protokolls, können wir nun also einen 1/2-Hitting Set Generator einsetzen, um ein passendes ρ zu generieren. (In der Tat wird der HSG viele Elemente generieren, jedoch hat mindestens eines die gewünschte Eigenschaft.) Dies ersetzt damit die erste Nachricht im Zap, denn dort sendet V an P ein zufällig gewähltes ρ , um P daran zu hindern, eines zu wählen, welches die Soundness-Bedingung verletzt. Diese Eigenschaft von ρ garantiert uns stattdessen der Hitting Set Generator, der in diesem Fall einer *gegen co-nichtdeterministische Schaltkreise* sein muss.

Seien also P und V die Teilnehmer des Zap, wobei V_2 wieder der deterministische, zweite Teil des Verifiers ist. Sei $l(\cdot)$ die Anzahl der Zufallsbits die V_1 sendet und $p(\cdot)$ und k wie oben. Sei außerdem H ein 1/2-Hitting Set Generator gegen co-nichtdeterministische Schaltkreise und $h(\cdot)$ die (resultierende) polynomielle Größe seiner Ausgabemenge. Unser derandomisiertes Protokoll funktioniert nun wie folgt:

Protokoll 4.18 (Zap derandomisiert, nach [BOV07]).

Eingabe: Die gemeinsame Eingabe ist $x \in L$, die private Eingabe von P ist ein $w \in R_L(x)$.

- P':**
1. Rufe $H(1^{l(n)}, 1^{k \cdot p(n)^2})$, sei $(\rho_1, \dots, \rho_{h(n)})$ dessen Ausgabe.
 2. Rufe P mit x und privater Eingabe w auf jedem ρ_i auf. Die Antwort sei jeweils π_i .
 3. Sende $\pi = (\pi_1, \dots, \pi_{h(n)})$ an V' .
- V':**
1. Rufe $H(1^{l(n)}, 1^{k \cdot p(n)^2})$, sei $(\rho_1, \dots, \rho_{h(n)})$ dessen Ausgabe.
 2. Rufe V_2 mit (x, ρ_i, π_i) für jedes i auf.
 3. Akzeptiere, falls V_2 für jedes i akzeptiert.

Wir haben nun also ein Protokoll mit nur einer Runde, und zudem ist V' deterministisch. Außerdem erfüllt es die Eigenschaften eines NP-Beweissystems, mit einer Geheimhaltungs-Eigenschaft d.h.:

Theorem 4.19. Protokoll 4.18 hat die folgenden Eigenschaften:

1. Perfekte Completeness, d.h.

$$\forall x \in L, w \in R_L : \langle P'(w), V' \rangle(x) \text{ akzeptiert.}$$

2. Perfekte Soundness, d.h.

$$\forall x \notin L, w' : \langle P'(w'), V' \rangle(x) \text{ akzeptiert nicht.}$$

3. Zeugen-Ununterscheidbarkeit (siehe Definition 4.9)

Beweis. Wir untersuchen alle drei Eigenschaften:

Perfekte Completeness: Die perfekte Completeness wird direkt von der des Zap Protokolls geerbt.

Perfekte Soundness: Sei $x \notin L$. Wie in der Einleitung beschrieben gibt es einen Schaltkreis C_x , welcher testet, ob ein ihm übergebener String ρ sound bezüglich x ist. Wegen der Soundness des Zap Protokolls muss C_x damit mindestens die Hälfte seiner Eingaben akzeptieren. Folglich gibt es mindestens ein ρ_i in der Ausgabe des 1/2-HSG H , für welches unabhängig von π_i der spätere Aufruf von V_2 mit Eingabe (x, ρ_i, π_i) zu Nicht-Akzeptanz führt. Mithin akzeptiert V' nicht.

Zeugen-Ununterscheidbarkeit: Das Zap Protokoll vererbt auch die Zeugen-Ununterscheidbarkeit, da dessen Komponenten polynomiell oft parallel aufgerufen werden und die Zeugen-Ununterscheidbarkeit unter paralleler Komposition abgeschlossen ist.

□

Wir fassen die Erkenntnisse aus den vergangenen Abschnitten zusammen:

Korollar 4.20. *Falls es 1/2-Hitting Set Generatoren gegen co-nichtdeterministische Schaltkreise und Trapdoor-Einwegfunktionen gibt, so hat jede Sprache $L \in \text{NP}$ ein NP-Beweissystem mit der Eigenschaft der Zeugen-Ununterscheidbarkeit.*

Dieses Resultat ist eine Neuerung dahingehend, dass es das erste NP-Beweissystem mit Geheimhaltungs-Eigenschaft ist, siehe Abbildung 4.5. Eine Anwendung werden wir uns im nächsten Kapitel ansehen.

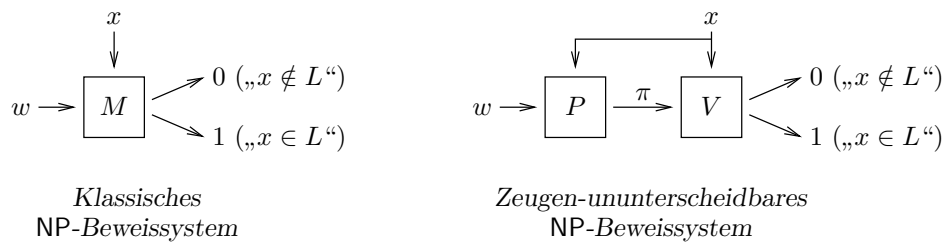


Abbildung 4.5: Vergleich NP-Beweissysteme ohne (links) und mit (rechts) Geheimhaltungs-Eigenschaft (z.B. Zeugen-Ununterscheidbarkeit). Durch einen „zwischen geschalteten“ Prover P wird der verwendete Zeuge w vor der entscheidenden Maschine geschützt.

Kapitel 5

Oblivious Transfer

Den letzten Typus von Protokollen, die wir uns näher ansehen, sind die *Oblivious Transfer* Protokolle – was man übersetzen kann mit „nicht wahrnehmbare Übertragung“. Es geht hier darum, dass zwischen zwei Parteien Daten übertragen werden sollen, wobei der Sender nicht erkennen – also „wahrnehmen“ – darf, ob der Empfänger sie tatsächlich bekam, bzw. welche.

In der Literatur werden dabei zwei Formen unterschieden. In der ursprünglichen und von Michael Rabin in [Rab81] vorgestellten Form wird in der Tat nur ein Bit b übertragen, wobei der Sender S sicherstellen will, dass der Empfänger R dieses Bit nur mit einer Wahrscheinlichkeit von $1/2$ erhält. Auf der anderen Seite möchte R vor S geheim halten, ob er b tatsächlich erhielt. Es wurde gezeigt (durch Kilian in [Kil88]), dass eine Vielzahl kryptographischer Primitive aus einem solchen Protokoll ohne weitere Annahmen konstruiert werden können, u.a. Zero Knowledge Beweissysteme für jede Sprache in NP.

Die andere praktische Variante, die von Even, Goldreich und Lempel in [EGL85] vorgeschlagen wurde, unterscheidet sich nur leicht: Hier hat der Sender S zwei Bits b_1 und b_2 und möchte eines davon (welches R auswählen darf) an R übertragen – jedoch über das andere keine Information offenbaren. Dem Empfänger R hingegen ist daran gelegen, dass S nicht erkennt, welches der beiden Bits er erhielt. Auf diese Variante, die auch *1-aus-2 Oblivious Transfer*¹ (oder kurz *1-2 OT*) genannt wird, werden wir uns im folgenden beziehen. In der Tat sind, was die Existenz angeht, beide Varianten äquivalent [Cre87].

¹Verallgemeinert wird auch *1-aus- n Oblivious Transfer* betrachtet. Dies beschreibt das Szenario, in dem ein Zugriff auf genau ein Element aus einer Datenbank mit n Elementen erfolgt, wobei die Datenbank selbst nicht erkennen darf, welches Datum abgefragt wurde.

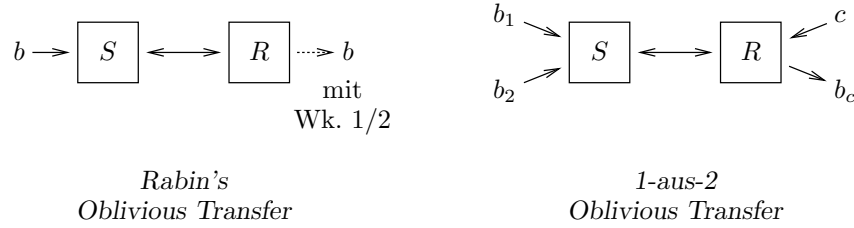


Abbildung 5.1: Vergleich der beiden Varianten von Oblivious Transfer.

5.1 1-aus-2 Oblivious Transfer

Zur Veranschaulichung geben wir auch hier eine kurze, diesmal [BSW06] entnommene Story:

Eine Industriespionin hat zwei Geheimnisse zu verkaufen. Ein potentieller Käufer interessiert sich für eines der beiden, möchte jedoch der unzuverlässigen Spionin nicht verraten, für welches. Allerdings möchte er auch nur den Preis für ein Geheimnis bezahlen, daher möchte die Spionin ihm nicht einfach beide Geheimnisse überlassen. Sie benötigen also ein Verfahren, bei dem der Käufer genau eines der beiden Geheimnisse wählen kann und daraufhin erfährt – aber die Verkäuferin nicht weiß, welches.

Als nächstes geben wir eine formale Definition von 1-2 Oblivious Transfer an. Dabei halten wir uns an die Notation von [BOV07] und schreiben daher $\text{output}_{M_0}(M_0(y_0), M_1(y_1), 1^n)$ für die Ausgabe von M_0 nach Interaktion mit M_1 bei gemeinsamer Eingabe 1^n und privaten Eingaben y_0 bzw. y_1 . Analog für die Ausgabe von M_1 .

Definition 5.1 (1-2 OT nach [BOV07]). *Ein Paar von pp-ITMs (S, R) ist ein 1-2 Oblivious Transfer Protokoll, falls folgende Eigenschaften erfüllt sind:*

Completeness: *Für alle $b_1, b_2 \in \{0, 1\}$ und $c \in \{1, 2\}$ gilt:*

$$\Pr_{S,R}[\text{output}_R(S(b_1, b_2), R(c), 1^n) = b_c] \geq 2/3$$

R-Privacy: *Für alle pp-ITMs S^* sind die folgenden Ensembles nicht-uniform ununterscheidbar:*

$$\{\text{output}_{S^*}(S^*, R(1), 1^n)\}_{n \in \mathbb{N}} \text{ und } \{\text{output}_{S^*}(S^*, R(2), 1^n)\}_{n \in \mathbb{N}}$$

S-Privacy: *Für alle pp-ITMs R^* gilt:*

- Sei $\Delta_1^b(n)$ statistische Differenz² von $\{\text{output}_{R^*}(S(0, b), R^*, 1^n)\}_{n \in \mathbb{N}}$ und $\{\text{output}_{R^*}(S(1, b), R^*, 1^n)\}_{n \in \mathbb{N}}$ für alle $b \in \{0, 1\}$.
- Sei $\Delta_2^b(n)$ statistische Differenz von $\{\text{output}_{R^*}(S(b, 0), R^*, 1^n)\}_{n \in \mathbb{N}}$ und $\{\text{output}_{R^*}(S(b, 1), R^*, 1^n)\}_{n \in \mathbb{N}}$ für alle $b \in \{0, 1\}$.

Dann ist $\tilde{\Delta}(n) := \min_i \max_b \Delta_i^b(n)$ vernachlässigbar.

Die einleitende, intuitive Formulierung wird dabei durch diese Definition direkt widerspiegelt: Die *Completeness* Bedingung stellt sicher, dass R auch tatsächlich das geforderte Bit erhält. Durch die *R-Privacy* Bedingung wird kein wie auch immer vom Protokoll abweichender Sender S^* herausfinden können, welches der beiden Bits R angefordert hat. Umgekehrt schützt die *S-Privacy* Bedingung mindestens eines der beiden Bits von S vor einem böswilligen R^* : Über eines der beiden erhält R^* keinerlei Informationen (jedoch ist nicht sichergestellt, dass R^* das von ihm „gewählte“ Bit b_c erhält und nicht vielleicht stattdessen das andere).

5.2 Dwork und Naor's Protokoll

In [DN00] schlugen Dwork und Naor folgendes 1-2 OT Protokoll vor, welches auf Zaps (siehe Abschnitt 4.4) aufbaut. Es läuft in 3 Runden, wobei die ersten zwei Runden einem Zap entsprechen. Als zusätzliche Voraussetzung wird die *Quadratische-Reste-Annahme*³ benötigt.

Protokoll 5.2. 3-Runden 1-2 OT nach [DN00]

Eingabe: Gemeinsame Eingabe ist der Sicherheits-Parameter 1^n . Der Sender S bekommt außerdem zwei Bits $b_1, b_2 \in \{0, 1\}$ und der Empfänger R ein $c \in \{1, 2\}$ als private Eingabe. (Schreibe \bar{c} für $3 - c$.)

S (Schritt S_1): Wähle $\rho \in_{\mathcal{R}} \{0, 1\}^{\text{poly}(n)}$ und sende ρ an R .

R (Schritt R_1): Wähle ein N mit n Bits als Produkt zweier hinreichend großer Primzahlen p und q . Wähle außerdem y_1 und y_2 mit $y_c \in_{\mathcal{R}} \text{QNR}(N) = \mathbb{Z}_N^* \setminus \text{QR}(N)$ und $y_{\bar{c}} \in_{\mathcal{R}} \text{QR}(N)$, beide mit Jacobi-Symbol⁴ 1 mit hinreichender Qualität gemäß der QRA.

²vgl. Definition 1.4 auf Seite 9

³Es wird angenommen, dass für zusammengesetzte n und $a \in \mathbb{Z}_n^*$ mit Jacobi-Symbol $J(a, n) = 1$ (die günstig gewählt sind, also hinreichend lang etc.) nicht effizient bestimmt werden kann, ob a quadratischer Rest modulo n ist ($a \in \text{QR}(n)$). Diese Annahme ist die *Quadratische-Reste-Annahme (QRA)*.

⁴Das Jacobi-Symbol $J(a, n)$ für $a \in \mathbb{Z}_n^*$ ist ein Objekt aus der Zahlentheorie mit wichtigen Anwendungen für die Kryptographie. Im Falle n prim definiert man es als 1, falls a quadratischer Rest modulo n ist, d.h. falls es $b \in \mathbb{Z}_n^*$ gibt mit $b^2 \equiv_n a$. Andernfalls ist $J(a, n) = -1$. Für zusammengesetzte $n = p_1^{\alpha_1} \cdot \dots \cdot p_k^{\alpha_k}$ setzt man $J(a, n) := J(a, p_1)^{\alpha_1} \cdot \dots \cdot J(a, p_k)^{\alpha_k}$. Das Jacobi-Symbol ist effizient berechenbar, jedoch ist sein Wert 1 nur notwendig für quadratische Reste, aber nicht hinreichend (bei zusammengesetzten n).

Sende N , y_1 und y_2 zusammen mit einem (Zap-)Beweis π (mittels ρ) für folgende NP-Aussage an S :

$$y_1 \in \text{QR}(N) \vee y_2 \in \text{QR}(N)$$

(Ein NP-Zeuge hierfür ist jeweils die Wurzel des Elementes, welches quadratischer Rest ist.)

S (Schritt S_2): Bei gültigem Beweis π wähle $x_1, x_2 \in_R \mathbb{Z}_N^*$ und sende an R :

$$Z := \left\{ \underbrace{(y_1^{b_1} \cdot x_1^2) \bmod N}_{z_1}, \underbrace{(y_2^{b_2} \cdot x_2^2) \bmod N}_{z_2} \right\}$$

R (Schritt R_2): Bestimme die Anzahl der quadratischen Reste modulo N in Z und gib $\sigma \in \{0, 1\}$ aus mit:

$$\sigma := \begin{cases} 0, & \text{falls } |Z \cap \text{QR}(N)| = 2 \\ 1, & \text{sonst} \end{cases}$$

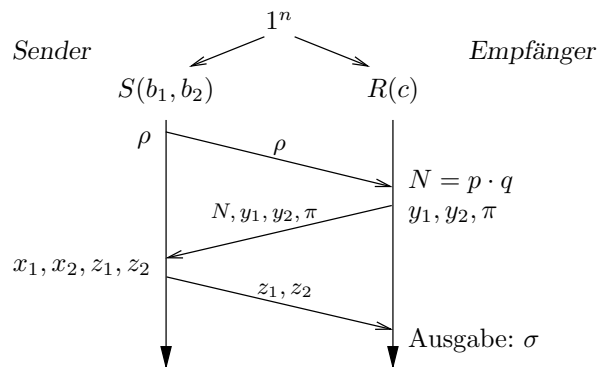


Abbildung 5.2: Illustration des 3-Runden Protokolls 5.2.

Theorem 5.3. *Unter den Voraussetzungen der Quadratische-Reste-Annahme und der Existenz von Trapdoor-Einwegfunktionen ist obiges Protokoll 5.2 ein 1-aus-2 Oblivious Transfer Protokoll.*

Beweis (nach [DN00]). Es sind die drei Eigenschaften *Completeness*, *R-Privacy* und *S-Privacy* zu untersuchen.

Completeness: Zunächst bemerken wir, dass R den Modulus N erzeugt hat und damit insbesondere dessen Zerlegung $N = p \cdot q$ in zwei Primzahlen kennt. Damit ist R in der Lage, effizient quadratische Reste und Nicht-Reste in \mathbb{Z}_N^* zu identifizieren.

Falls sich beide Parteien an das Protokoll halten, gilt:

$$y_c \in \text{QNR}(N) \wedge y_{\bar{c}} \in \text{QR}(N)$$

Daraus folgt für die in S_2 gesendeten Werte:

$$\begin{aligned} z_c &= y_c^{b_c} \cdot x_c^2 \bmod N \in \text{QR}(N) \iff b_c = 0 \\ z_{\bar{c}} &= y_{\bar{c}}^{b_{\bar{c}}} \cdot x_{\bar{c}}^2 \bmod N \in \text{QR}(N) \iff b_{\bar{c}} \in \{0, 1\} \end{aligned}$$

Da damit $z_{\bar{c}} \in \text{QR}(N)$ ohnehin gilt, kann R so den Wert von b_c bestimmen, indem Z auf die Anzahl von quadratischen Resten überprüft wird: Enthält nämlich Z zwei quadratische Reste, so muss $b_c = 0$ gelten, andernfalls (d.h. nur ein quadratischer Rest) gilt $b_c = 1$. Der in R_2 bestimmte Wert σ stimmt somit mit b_c überein.

S-Privacy: Wir nehmen nun an, ein Angreifer R^* weicht in der Rolle von R beliebig vom Protokoll ab. Falls R^* das Paar (y_1, y_2) mit $\forall i : y_i \in \text{QNR}(N)$ generiert hat, so wird mit überwiegender Wahrscheinlichkeit über ρ sein Zap-Beweis π in Schritt S_2 aufgrund der Soundness⁵ des Zaps⁶ nicht akzeptiert, wodurch S abbricht und R^* weder über b_1 noch b_2 etwas lernt. Mit nur vernachlässigbarer Wahrscheinlichkeit (über ρ) wird also S in Schritt S_2 fortsetzen, falls $\forall i : y_i \in \text{QNR}(N)$.

Betrachten wir nun den anderen Fall, also dass einer der y_i quadratischer Rest ist. Sei dafür $y_d \in \text{QR}(N)$ für ein $d \in \{1, 2\}$, dann ist damit auch $z_d \in \text{QR}(N)$ unabhängig vom Wert von b_d . Dies ist der Fall, weil $\text{QR}(N)$ eine Untergruppe in \mathbb{Z}_N^* bildet und somit z_d für beide Werte von b_d Produkt zweier quadratischer Reste ist und in $\text{QR}(N)$ insbesondere gleichverteilt. R^* kann also hier nichts über b_d lernen, woraus die Behauptung folgt. (Falls sogar beide y_i quadratische Reste sind, lernt R^* nach dem selben Argument nichts über beide Bits.)

⁵Wir bemerken hier, dass die Soundness adaptiv sein muss, da die zu beweisende Aussage potentiell erst nach dem Zufallsstring ρ gewählt wird und daher an diesen angepasst sein könnte.

⁶Für „überwiegende“ Wahrscheinlichkeit müssen wir o.B.d.A. voraussetzen, dass ein Zap aus der ursprünglichen Definition 4.15 hinreichend oft parallel ausgeführt wird, um eine vernachlässigbare Fehlerschranke für die Soundness zu erhalten.

R-Privacy: Sei nun umgekehrt S^* ein Angreifer in der Rolle von S . Da sich R an das Protokoll hält und damit N , y_1 und y_2 in hinreichender Qualität erzeugt, ist S^* nicht in der Lage zu erkennen, welches der beiden y_i der quadratische Rest ist. Insbesondere kann S^* nicht die folgenden Verteilungen von (N, y_1, y_2, ρ, π) unterscheiden:

1. $y_1 \in_{\mathbb{R}} \text{QR}(N)$, $y_2 \in_{\mathbb{R}} \text{QNR}(N)$ mit $J(y_2, N) = 1$, und der Zeuge im Zap-Beweis zur Konstruktion von π war $\sqrt{y_1}$,
2. $y_1, y_2 \in_{\mathbb{R}} \text{QR}(N)$, und der Zeuge im Zap-Beweis zur Konstruktion von π war $\sqrt{y_1}$,
3. $y_1, y_2 \in_{\mathbb{R}} \text{QR}(N)$, und der Zeuge im Zap-Beweis zur Konstruktion von π war $\sqrt{y_2}$,
4. $y_2 \in_{\mathbb{R}} \text{QR}(N)$, $y_1 \in_{\mathbb{R}} \text{QNR}(N)$ mit $J(y_1, N) = 1$, und der Zeuge im Zap-Beweis zur Konstruktion von π war $\sqrt{y_2}$.

Dabei sind die Verteilungen aus 2. und 3. durch die WI-Eigenschaft des Zaps ununterscheidbar, und 1. und 2. (bzw. 3. und 4.) aufgrund der QRA: Ist ein $y =: y_2$ mit $J(y, N) = 1$ gegeben, so könnte hierzu ein $x \in_{\mathbb{R}} \mathbb{Z}_N^*$ quadriert werden, um $y_1 \in \text{QR}(N)$ zu erhalten und π mitsamt ρ zufällig zu generieren. Mit einem Unterscheider für 1. und 2. könnte damit $y \stackrel{?}{\in} \text{QR}(N)$ mit nicht-vernachlässigbarer Erfolgswahrscheinlichkeit entschieden werden. (Analog 3. und 4.)

Zusammen sind folglich die Verteilungen aus 1. und 4. ununterscheidbar. Damit kann S^* jedoch nicht herausfinden, über welches der beiden Bits b_1 und b_2 der Empfänger R Informationen erhält.

□

5.3 Derandomisierung von Dwork und Naor's Protokoll

In Kapitel 4 hatten wir gesehen, wie ein Zap derandomisiert werden kann. Dies kann nun direkt verwendet werden, um das 1-2 OT Protokoll ebenfalls um eine Runde zu verkürzen, indem direkt ein NP-Beweis mit Zeugen-Ununterscheidbarkeit geführt wird. Es fällt dadurch die erste Runde weg, in der S an R den Zufallsstring ρ schicken muss, ganz analog der Derandomisierung in Abschnitt 4.5.

Es ergibt sich aus Theorem 5.3:

Theorem 5.4 ([BOV07]). *Falls es 1/2-Hitting Set Generatoren gegen nichtdeterministische Schaltkreise gibt und die Quadratische-Reste-Annahme gilt, so gibt es ein 1-aus-2 Oblivious Transfer Protokoll in 2 Runden.*

Dabei wird die Voraussetzung von Trapdoor-Einwegfunktionen für Zaps nicht benötigt, denn sie wird von der QRA impliziert.

Die Reduzierung auf 2 Runden ist die maximal mögliche Reduzierung, denn für 1-aus-2 Oblivious Transfer ist es im Gegensatz zu Bit Commitment (Kapitel 3) oder WI-Beweissysteme (Kapitel 4) nicht möglich, ein nicht-interaktives Protokoll anzugeben:

- Überträgt nur R an S eine Nachricht, so kann R das Bit b_c nicht erhalten und die *Completeness* ist verletzt.
- Überträgt nur S an R eine Nachricht, so muss R in der Lage sein, aus dieser das Bit b_c für beide Möglichkeiten von $c \in \{1, 2\}$ zu erhalten damit die *Completeness* erfüllt ist, denn S hat keine Information von R erhalten um die gesendete Nachricht in irgendeiner Weise anzupassen. Allerdings verletzt dies damit die *S-Privacy* Bedingung.

Es bleibt anzumerken, dass dies nicht die erste Konstruktion eines 1-2 OT Protokolles in 2 Runden ist. Es wird jedoch keine Setup Annahme getroffen, so werden beispielsweise keine *Public Keys* verwendet, wie in einigen anderen Protokollen üblich. (Lässt man diese zu, kann aus Protokoll 5.2 ohnehin die erste Runde entfernt werden, denn ρ kann Teil des Public Keys sein.)

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit haben wir uns damit beschäftigt, die Interaktion in ausgewählten kryptographischen Protokollen mittels Derandomisierung zu reduzieren. Die eingesetzte Technik, ein Hitting Set Generator als spezielle Form eines „nicht-kryptographischen“ Pseudozufallsgenerators vom NW-Typ, hat ihren Ursprung in der Komplexitätstheorie, und findet damit ein neues Anwendungsfeld in der Kryptographie. In allen drei Fällen führten die Konstruktionen zur Reduktion auf das Minimum der möglichen Interaktion für die betrachteten Protokolle. Dafür wird allerdings eine zusätzliche, nicht-kryptographische (also eher schwächere) Annahme getroffen: die Existenz eines $1/2$ -Hitting Set Generators gegen uniforme bzw. nicht-uniforme co-nichtdeterministische Beobachter.

Alle drei derandomisierten Protokolle haben Merkmale gemeinsam, welche die Derandomisierung ermöglichten: In einer Runde wird ausschließlich ein Zufallsstring übertragen, der mit hoher Wahrscheinlichkeit eine feste Eigenschaft haben soll. Diese Eigenschaft soll sicherstellen, dass die Gegenseite nicht „schummeln“ kann: Beim Bit Commitment in Kapitel 3 sollen Kollisionen vermieden werden, um die Binding Eigenschaft zu sichern. Bei den Zaps aus den Kapiteln 4 und 5 soll ein Brechen der Soundness verhindert werden, indem der generierte Beweis „passend“ zur gesendeten Zufallszahl generiert werden muss und damit zumeist nur positive Instanzen beweisen kann.

Diese *feste Eigenschaft* kann jeweils mit einem „einfachen“ Test überprüft werden: Hier reichen co-nichtdeterministische Polynomialzeit-Tests aus (uniform für das Bit Commitment, nicht-uniform für die Zaps). Dies war die Voraussetzung, damit ein Hitting Set Generator stets eine für diesen Test positive Eingabe liefert, also *deterministisch* ein Element mit der gewünschten Eigenschaft. Genutzt werden konnte dies, indem die ursprünglichen Protokolle dann für jedes vom Hitting Set Generator ausgegebene Element *parallel* aufgerufen wurden, was das Senden der Zufallsbits ersetzt und die Interaktion

verringert. (Die Nachrichtengröße steigt hierdurch allerdings polynomiell an.) Die zweite notwendige Eigenschaft für die Anwendung der hier vorgestellten Derandomisierung ist also die *Abgeschlossenheit* der gewünschten kryptographischen Eigenschaften *unter (polynomieller) paralleler Komposition*.

Eine natürliche Fortführung der Arbeit wäre nun, weitere Protokolle darauf zu untersuchen, ob sie die obigen beiden Eigenschaften zur Anwendung der vorgestellten Derandomisierungs-Technik erfüllen:

1. Übertragung einer Zufallszahl, die eine feste Eigenschaft erfüllen muss, für die ein „einfacher“ Test existiert, und
2. Abgeschlossenheit der gewünschten Eigenschaft unter paralleler Komposition.

Es ist Gegenstand aktueller Forschung, weitere Anwendungen mit diesen Eigenschaften zu finden, und zu untersuchen, wie Derandomisierungs-Techniken auch bei diesen zur Optimierung des Protokolls verwendet werden können.

In aktuellen Untersuchungen steht insbesondere auch die Rundenanzahl für Zero Knowledge Protokolle im Mittelpunkt. So ist für Statistisches Zero Knowledge bekannt, dass für NP-Probleme solche Beweissysteme mit konstanter Rundenanzahl existieren und die Menge der zu übertragenden Daten wird untersucht (z.B. [SV08]). Für Perfect Zero Knowledge ist dies nicht bekannt, ist jedoch auch aktueller Forschungsgegenstand (z.B. [Mal08]). Eine interessante Frage ist hier zum Beispiel, ob *öffentliche* Zufallsbits (engl. *public coins*) sich in konstante Rundenzahlen verwandeln lassen. Eine weitere Bedeutung haben diese Untersuchungen auch im Zusammenhang der Separierung der entstehenden Komplexitätsklassen.

Literaturverzeichnis

- [ACR98] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. A new general derandomization method. *J. ACM*, 45(1):179–213, 1998.
- [Bab85] L. Babai. Trading group theory for randomness. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429, New York, NY, USA, 1985. ACM.
- [Blu82] Manuel Blum. Coin flipping by phone. *24th IEEE Computer Conference (CompCon)*, pages 133–137, 1982.
- [BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.
- [BOV07] Boaz Barak, Shien Jin Ong, and Salil Vadhan. Derandomization in Cryptography. *SIAM Journal on Computing*, 37(2):380–400, 2007.
- [BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive Zero-Knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.
- [BSW06] Albrecht Beutelspacher, Jörg Schwenk, and Klaus-Dieter Wolfenstetter. *Moderne Verfahren der Kryptographie. Von RSA zu Zero-Knowledge*. Vieweg, 2006.
- [Cre87] Claude Crepeau. Equivalence Between Two Flavours of Oblivious Transfers. In *CRYPTO*, pages 350–354, 1987.
- [DN00] Cynthia Dwork and Moni Naor. Zaps and Their Applications. In *In 41st FOCS*, pages 283–293. IEEE, 2000.
- [EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.
- [FGM⁺89] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, pages 429–442,

- Greenwich, Connecticut, 1989. Advances in Computing Research, vol. 5, JAI Press.
- [FLS00] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions. *SIAM J. Comput.*, 29(1):1–28, 2000.
- [FS90] U. Feige and A. Shamir. Witness indistinguishable and Witness Hiding Protocols. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, New York, NY, USA, 1990. ACM.
- [GK96] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.
- [GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, New York, NY, USA, 1985. ACM.
- [GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.
- [GO94] Oded Goldreich and Yair Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.
- [Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2001.
- [Gol02] Oded Goldreich. Zero-knowledge twenty years after its invention, 2002.
- [GOS05] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect Non-Interactive Zero Knowledge for NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.
- [GSTS04] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3-4):85–130, 2004.

- [GVW00] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. Simplified derandomization of BPP using a hitting set generator. *Electronic Colloquium on Computational Complexity (ECCC)*, (004), 2000.
- [HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [ISW99] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Near-Optimal Conversion of Hardness into Pseudo-Randomness. In *IEEE Symposium on Foundations of Computer Science*, pages 181–190, 1999.
- [IW97] Russell Impagliazzo and Avi Wigderson. $P=BPP$ unless E has sub-exponential circuits: Derandomizing the XOR Lemma, 1997.
- [Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31, New York, NY, USA, 1988. ACM.
- [KP98] Joe Kilian and Erez Petrank. An Efficient Noninteractive Zero-Knowledge Proof System for NP with General Assumptions. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 11(1):1–27, Winter 1998.
- [Mal08] Lior Malka. Instance-Dependent Commitment Schemes and the Round Complexity of Perfect Zero-Knowledge Proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 2008.
- [MV99] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin Games Using Hitting Sets. In *IEEE Symposium on Foundations of Computer Science*, pages 71–80, 1999.
- [Nao91] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 4(2):151–158, 1991.
- [NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.
- [QGB89] J. J. Quisquater, L. Guillo, and T. Berson. How to explain zero knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lec-*

- ture Notes in Computer Science*, pages 628–631. Springer-Verlag, 1989.
- [Rab81] Michael O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.
- [Sav72] J.E. Savage. Computational work and time of finite machines. *Journal of the ACM*, 19:660–674, 1972.
- [Sch95] Bruce Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.
- [SV08] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *To appear in: Journal of the ACM*, 2008.
- [Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.