# Zero Knowledge Beweissysteme mit konstanten Runden

Dominik Meyer
<dmeyer@federationhq.de>

## Zusammenfassung

Diese Ausarbeitung befaßt sich mit zero-knowledge Beweisen mit konstanter Rundenzahl. Sie basiert zum größten Teil auf den Arbeiten von Goldreich die er in seinem Buch [2] veröffentlicht hat. Diese zero-knowledge Beweise spielen in der modernen Kryptographie eine immer größere Rolle. Deshalb ist es sinnvoll sich mit diesen effizienten Implementierungen zu befassen.
Diese Ausarbeitung wurde von mir im Rahmen eines Seminars über Kryptographie am Institut für Informatik der Christian Albrechts Universität zu Kiel erstellt.

## 1. EINLEITUNG

### 1.1 Überblick

Der Anwendungsbereich von zero-knowledge Beweissystemen, in der modernen Kryptographie, sind Unterprotokolle von kryptographsichen Protokollen oder selbständige Authentifizierungsprotokolle. Dabei ist der wichtigste Punkt, daß bei dieser Kommunikation kein zusätzliches Wissen vermittelt wird.
Betrachten wir dafür zwei Parteien, einen *Prover* und einen *Verifier*. Der Prover versucht den Verifier davon zu überzeugen, daß er im Besitz einer bestimmten Information ist, ohne diesem die Information mitzuteilen. Im allgemeinen geht es darum, , daß der Prover für eine gemeinsame Eingabe $x$ den Verifier davon überzeugt, daß $x$ in einer Sprache $L$ enthalten ist.
In dieser Ausarbeitung wird eine Einführung in die zero-knowledge Beweise gegeben und dann gezeigt, wie sich aus diesen Protokolle mit konstanter Rundenzahl konstruieren lassen. Diese lassen sich dann effizient in andere Protokolle einbinden.

### 1.2 Aufbau

Nach dieser kurzen Einleitung werden in Kapitel 2 die Beschreibungen und Definitionen vorgestellt, die für das Verstehen von zero-knowledge Beweissystemen mit konstanter Rundenzahl nötig sind. Dazu gehören auch die Themen Commitment-Schemen, Ununterscheidbarkeit von Wahrscheinlichkeitsverteilungen und Einwegfunktionen. In Kapitel 3 erfolgt dann der Einstig in die zero-knowledge Beweissysteme. Dort wird beschrieben wie diese aufgebaut sind und wie sie funktionieren. In Kapitel 4 wird gezeigt, wie sich aus „schwachen" zero-knowledge Beweisen, Protokolle konstruieren lassen, die mit konstanten Rundenzahl auskommen, aber trotzdem perfekt zero-knowledge sind.

### 1.3 Literatur

Die Ausarbeitung stützt sich zum größten Teil auf das Buch von Oded Goldreich [2]. Um Definitionen aber besser verstehen zu können, habe ich auf die Diplomarbeit von Thomas Schwarze [5] zurückgegriffen und habe mich bei der Angabe von Definitionen von ihm leiten lassen. Ebenfalls in die Ausarbeitung sind Informationen aus einem neueren Paper von Oded Goldreich eingeflossen [3]. Für den Beweis der zero-knowledge Eigenschaft in Kapitel 4 habe ich ein älteres Paper von Oded Goldreich verwendet. In [4] beschreibt er genauer, wie der Beweis der zero-knowledge Eigenschaft durchgeführt wird.

## 2. EINFÜHRENDE DEFINITIONEN

Dieses Kapitel dient dazu die grundlegenden Definition, die zum Verständnis von zero-knowledge Beweisen benötigt werden, anzugeben. Begonnen wird mit der Frage, was im Kontext von zero-knowledge Beweisen eigentlich „Wissen" und „Nichtwissen" bedeutet.

### 2.1 Knowledge und Zero-Knowledge

In der gesamten Ausarbeitung werden häufig die Begriffe *knowledge* und *zero-knowledge* verwenden. Was ist unter diesen Begriffen eigentlich zu verstehen ?
In dieser Arbeit bezeichnet der Begriff „Knowledge" nur Informationen, die ein Verifier bei einem Beweis erhält, die er selbst nicht in polynomieller Zeit hätte berechnen können. Dabei geht es nicht direkt um den vorgeschriebenen Verifier, sondern um einen Verifier, der versucht, den Prover zu täuschen.
*Zero-knowledge* ist dabei die (relative) Sicherheit dafür, daß ein Verifier mit keinem (berechenbaren) Verfahren und mit keiner Täuschungshandlung weitere Informationen über die Behauptung hinaus erlangen kann.

### 2.2 Unterscheidbarkeit von Wahrscheinlichkeitsverteilungen

In den weiteren Kapiteln, vor allem bei den Definitionen um zero-knowledge, werden wir oft über die Ununterscheidbarkeit von Wahrscheinlichkeitsverteilungen argumentieren. Dies liegt daran, daß die Teilnehmer an einem Beweissystem in der Regel probalistische Turingmaschinen sind und man deshalb die Ausgabe solcher TM nur mit einer Wahrscheinlichkeitsverteilung beschreiben kann. Um jetzt Familien von Wahrscheinlichkeitsverteilungen unterscheiden zu können benötigen wir als erstes den Begriff der *vernachlässigbaren Funktion.*

**Definition 2.1 vernachlässigbare Funktion**
Eine Funktion $\mu : \mathbf{N} \rightarrow [0,1]$ heißt vernachlässigbar, wenn gilt:

$$\forall k \in \mathbf{N}, \exists n_k \in \mathbf{N}, \forall n \geq n_k : \mu(n) \leq \frac{1}{n^k}$$

Mit dieser Definition können wir nun die Definition der Ununterscheidbarkeit angeben.

**Definition 2.2 Ununterscheidbarkeit**
Sei $L \subseteq \{0,1\}^*$ eine Sprache, $U = \{U_x\}_{x \in L}$ und $V = \{V_x\}_{x \in L}$ Familien von Wahrscheinlichkeitsverteilungen, eine für jedes $x \in L$.

1. Die Familien $U$ und $V$ heißen **berechenbar ununterscheidbar**, wenn es eine vernachlässigbare Funktion $\mu(\cdot)$ gibt, so daß für alle probalistischen polynomiell-beschränkten Turingmaschinen A gilt:
$$|Pr(A(U_x) = 1) - Pr(A(V_x) = 1)| \leq \mu(|x|)$$
2. Die Familien $U$ und $V$ heißen **statistisch ununterscheidbar**, wenn es eine vernachlässigbare Funktion $\mu(\cdot)$ gibt, so daß für alle (nicht nur polynomiell beschränkten) Turingmaschinen $B$ gilt:
$$|Pr(B(U_x) = 1) - Pr(B(V_x) = 1)| \leq \mu(|x|)$$
3. Die Familien $U$ und $V$ heißen **perfekt ununterscheidbar** oder auch identisch, wenn $U = V$ gilt.

## 2.3 Interaktive Systeme

Die zero-knowledge Beweissystemte setzen sich aus interaktiven Systemen zusammen. Interaktive Systeme bestehen aus zwei Parteien, die mit einander kommunizieren. Ich werde diese Kommunikation auch als Protokoll bezeichnen. Die an diesem Protokoll beteiligten Systeme sind die *interaktiven Systeme*. Jedes dieser Systeme wird von einer interaktiven Mehrbandturingmaschine dargestellt, die in einer bestimmten Weise miteinander verknüpft sind und gemeinsame Berechnungen durchführen können. Deshalb wird dieses Unterkapitel auch mit der Beschreibung beginnen, was in dieser Ausarbeitung als Turingmaschine bezeichnet wird.

**Definition 2.3 Turingmaschine in diesem Kontext**
Eine **Turingmaschine** (im weiteren auch **TM** genannt) in diesem Kontext ist eine Mehrbandturingmaschine, welche folgende Bänder zusätzlich zu ihrem Arbeitsband besitzt:

1. ein Eingabeband, das nur gelesen werden kann
2. ein Ausgabeband, auf das nur geschrieben werden kann

Das Eingabeband enthält die Starteingabe für die TM. Wenn die TM hält steht auf dem Ausgabeband, die Ausgabe des Algorithmus.

Wie in fast allen Bereichen der heutigen Kryptographie wird auch bei den zero-knowledge Beweissystemen auf Zufallswerte gesetzt. Dies ist der Grund für die erste Erweiterung der Turingmaschine zu einer *probalistischen Turingmaschine*. Sie ermöglicht die Verwendung von zufälligen Bits innerhalb einer Berechnung einer Turingmaschine.

**Definition 2.4 probalistische TM**
Eine probalistische Turingmaschine ist eine Mehrbandturingmaschine, die ein weiteres Eingabeband besitzt. Dieses Eingabeband ist beim Start der TM mit zufälligen Bits $\delta \in \{0,1\}$ initialisiert. Dabei sind die einzelnen Werte unabhängig und gleichverteilt über dem gesamten Band.
Dieses Band ist nach links beschränkt und nach rechts unbeschränkt. Beim Lesen eines Bit von dem Band wird der Lesekopf einen Schritt nach rechts geschoben.

Die Entscheidungen dieser TM hängen jetzt nicht nur von Ihren Berechnungen ab, sondern auch von den Werten die vom Zufallsband gelesen werden. Das $\delta \in \{0,1\}$ ist die Wahrscheinlichkeit für eine Ausführung $P = \frac{1}{2}$.
Jetzt ist verständlich, was unter einer probalistischen Turingmaschine zu verstehen ist. Aber wie kommunizieren nun diese beiden TM ? Zur Zeit hat eine TM nur die Möglichkeit, mit der Außenwelt, über ihr Eingabe- und Ausgabeband zu kommunizieren, aber nicht mit einer weiteren Turingmaschine. Deshalb wird nun die Definition 2.3 zu einer *interaktiven Turingmaschine* erweitert.

**Definition 2.5 interaktive TM**
Ein interaktive Turingmaschine ist eine probalistische TM, die folgende weiteren Bänder besitzt:

1. Ein Empfangsband, von welchem nur gelesen werden kann
2. Ein Sendeband, auf welches nur geschrieben werden kann
3. Ein Switch-Band, von welchem gelesen und auf welches geschrieben werden kann

Jede interaktive TM besitzt eine Identität $\lambda \in \{0,1\}$ und kann verschiedene Zustände haben. Ein TM heißt:

**aktiv** falls auf dem Switch Band ihre Identität steht

**idle** falls auf dem Switch Band nicht ihre Identität steht.

Wenn eine TM idle ist, verändert sie ihren Zustand nicht.

Die TM besitzt nun weitere Kommunikationswege zur Außenwelt. Das Empfangs- und das Sendeband bieten die Möglichkeit Daten nach außen zu versenden und Eingaben von Außen zu erhalten. Das Switchband ist eine einfache Möglichkeit, zwei an einer Kommunikation teilnehmenden Turingmaschinen gleichzuschalten. Nun wird ein *interaktiven Systems* eingeführt. Unter diesem Begriff verstehen wir ein Paar von interaktiven Turingmaschinen, welche auf eine bestimmte Art zusammengeschlossen sind.

**Definition 2.6 interaktives System**
Ein interaktives System besteht aus zwei interaktiven TM's $A, B$.
$A$ und $B$ sind verbunden wenn gilt:

- $A$ und $B$ haben unterschiedliche Identitäten
- Beide haben die gleiche Eingabe auf ihrem Eingabeband
- Beide teilen sich das **gleiche** Switch-Band
- Das Sendeband von A ist das Empfangsband von B und umgekehrt.

Wenn eine TM hält, während auf dem Switch-Band ihre Identität steht, dann halten beide TM.

Ein interaktives System ist also eine bidirektionale Verbindung von zwei Turingmaschinen. Wenn wir zu den *interaktiven Beweisystemen* kommen, wird eine dieser Maschinen die Aufgabe des *Provers* und die anderen TM die Aufgabe des *Verifiers* übernehmen.
In einem interaktiven System bekommen immer beide Kommunikationspartner die identische Eingabe. Später wird eine Möglichkeit beschrieben, wie eine Turingmaschine eine weitere Hilfseingabe bekommen kann.
Um jetzt die Kommunikation eines solchen interaktiven Systems analysieren zu können, muß noch geklärt werden, was eine *Berechnung eines interaktiven Systems* ist.

**Definition 2.7 Berechnung eines Interaktiven Systems**
Die gemeinsame Berechnung eines Interaktiven Systems besteht aus einem Paar von Konfigurationen. Diese Konfigurationen ergeben sich aus den lokalen Konfigurationen jeder einzelnen TM. In jedem dieser Paare ist immer eine der Maschinen im Zustand **aktiv** und die andere im Zustand **idle**.

Für das nächste Unterkapitel benötigen wir noch eine Notation, die die Ausgabe einer interaktiven Turingmaschine beschreibt und die Definition der Komplexität einer interaktiven TM. Dies Notation benötigen wir, da alle Ausgaben der TM von ihrem Zufallsband abhängen. Wollen wir etwas über die Ausgabe einer Turingmaschine sagen, müssen wir natürlich über Wahrscheinlichkeiten argumentieren.
Seien $A, B$ ein interaktives System von TMs und nehmen wir an, daß jede Interaktion zwischen $A$ und $B$, für jede gemeinsame Eingabe, nach endlich vielen Schritten beendet ist.

**Notation:**
Wir beschreiben im weiteren Verlauf mit $< A, B > (x)$ die zufällige Ausgabe von $B$ nach der Interaktion mit $A$ über die gemeinsame Eingabe $x$, wenn die zufällige Eingabe für jede der Maschinen unabhängig und gleichverteilt ausgewählt wird.

**Definition 2.8 Komplexität einer interaktiven TM**
Wir sagen, eine interaktive TM $A$ hat Zeitkomplexität $t : \mathbf{N} \to \mathbf{N}$ wenn für jede interaktive TM $B$ und jede Eingabe $x$ gilt:
Wenn $A$ mit $B$ interagiert auf einer gemeinsamen Eingabe $x$, dann hält $A$ (egal was auf den Zufallsbändern von $A$ und $B$ steht) innerhalb von $t(|x|)$ Schritten.

## 2.4 interaktive Beweissysteme

Nun kommen wir zu den interaktiven Beweissystemen. Zero-Knowledge Beweissysteme sind genau solche Beweissysteme, nur daß sie zusätzlich die zero-knowledge Eigenschaft erfüllen.
Ein interaktiver Beweis ist kein statischer Beweis, wie wir ihn aus der Mathematik her kennen, sondern ein zufallsgesteuerte Kommunikation zwischen zwei Partnern dem *Verifier* und dem *Prover*. Wie im vorherigen Kapitel beschrieben, findet die Kommunikation in einem interaktiven System statt und die Kommunikationspartner sind interaktive Turingmaschinen nach Definition 2.3. In dieser Interaktion versucht der Prover den Verifier von einer bestimmten Annahme zu überzeugen. Genau bedeutet dies, daß der Prover versucht zu beweisen, daß die gemeinsame Eingabe des interaktiven Systems in einer Sprache $L$ liegt.

**Definition 2.9 interaktives Beweissystem**
Seien $P$ und $V$ interaktive TM. Das Paar $(P, V)$ heißt **interaktives Beweissystem für eine Sprache** $L$, falls $V$ polynomiell ist und folgende Bedingungen gelten:

**Vollständigkeit** Für jedes $x \in L$

$$Pr(< P, V > (x) = 1) \geq \frac{2}{3}$$

**Zuverlässigkeit** Für jedes $x \notin L$ und jede interaktive TM $B$

$$Pr(< P, B > (x) = 1) \leq \frac{1}{3}$$

Die Klasse für die es interaktive Beweissysteme gibt heißt **IP**, in ihr liegen alle Sprachen für die es ein solches gibt.

Diese Definition setzt feste Grenzen für Vollständigkeit und Zuverlässigkeit. In der nächsten Definition werden diese Grenzen aufgehoben. Es ist zu bemerken, daß die Zuverlässigkeitsbedingung für alle möglichen Prover gelten muß, die Vollständigkeitsbedingung aber nur für die vorbestimmten Prover.Da es bei realen Systemen oft nicht realistisch ist perfekte Sicherheit zu fordern, ist es sinnvoll eine Fehlerwahrscheinlichkeit einzuführen. Diese läßt sich aber durch wiederholtes Ausführen eines interaktiven Beweises weiter verringern.
Jetzt ist noch die Frage zu klären, für welche Sprachen es ein solches interaktives Beweissystem gibt?

**Beispiel 2.10**
Jede Sprache in **NP** besitzt ein interaktives Beweissystem.
Sei dazu $L \in \mathbf{NP}$ und $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ eine Zeugenrelation für die Sprache $L$. Dies bedeutet, daß $R_L$ in polynomieller Zeit bestimmbar ist und
$L = \{x : \exists y \; so, dass \; |y| = poly(x) \wedge (x, y) \in R_L\}$ gilt. Dann besteht ein interaktives Beweissystem für die Sprache $L$ aus

einem Prover, der für eine Eingabe $x \in L$ einen Zeugen $y$ sendet und aus einem Verifier, der $y$ empfängt und 1 ausgibt wenn $|y| = poly(x)$ und $(x, y) \in R_L$. Sonst gibt der Verifier 0 aus. Dies ist effizient möglich, da die Klasse **NP** so definiert ist.

Es ist offensichtlich, daß der Verifier immer 1 ausgibt, wenn er mit dem vorbestimmten Prover interagiert, da dieser immer einen „echten" Zeugen sendet. Außerdem wird der Verifier immer 0 ausgeben, wenn er mit einem anderen Prover interagiert, da der Verifier nur Elemente der Sprache $L$ erkennt. In diesem Beweissystem sind beide Parteien deterministisch. Keiner benutzt bisher Werte von seinem Zufallsband. Man kann sich **NP** auch als Klasse von Sprachen vorstellen, deren Interaktion unidirektional ist. Die Kommunikation geht nur vom Prover zum Verifier. Es ist also festzustellen das **NP** $\subseteq$ **IP**
Es ist wichtig zu erwähnen, daß das obige Beispiel nicht die zero-knowledge Eigenschaft erfüllt, da der Prover den Zeugen $y$ bekannt gibt.
Da Beweissysteme mit den bisherigen Einschränkungen für Vollständigkeit und Zuverlässigkeit in der Praxis kaum eingesetzt werden können, werden nun diese Restriktionen aufgehoben.

## 2.5 Allgemeine interaktive Beweissysteme

In diesem Unterkapitel werden die konstanten Grenzen für Vollständigkeit und Zuverlässigkeit aufgehoben und durch Funktionen ersetzt.

**Definition 2.11 verallgemeinerte interaktive Beweissysteme**
Seien $c, s : \mathbf{N} \to \mathbf{N}$ Funktionen für die gilt: $c(n) > s(n) + \frac{1}{p(n)}$ für ein Polynom $p(\cdot)$.
Ein interaktives System $(P, V)$ heißt dann **verallgemeinertes interaktives Beweissystem** für die Sprache $L$ mit Vollständigkeitsgrenze $c(\cdot)$ und Zuverlässigkeitsgrenze $s(\cdot)$ wenn gilt:

**veränderte Vollständigkeit** Für jedes $x \in L$

$$Pr(< P, V > (x) = 1) \geq c(|x|)$$

**veränderte Zuverlässigkeit** Für jedes $x \notin L$ und jede interaktive TM $B$

$$Pr(< B, V > (x) = 1) \leq s(|x|)$$

Die Funktion $g(\cdot)$ , wobei $g(n) \stackrel{\text{def}}{=} c(n) - s(n)$ heißt „Akzeptanz Lücke" (engl. acceptance gap) und die Funktion $e(\cdot)$, wobei $e(n) \stackrel{\text{def}}{=} max\{1 - c(n), s(n)\}$ heißt Fehlerwahrscheinlichkeit von $(P, V)$

Da in der Regel zero-knowledge Beweissysteme als Unterprotokolle in anderen kryptographischen Protokollen eingesetzt werden, muß jetzt noch eine Möglichkeit geschaffen werden, wie Informationen von diesen in das Beweissystem gelangen kann. Dazu wird die interaktive TM durch ein weiteres Band erweitert. Auf diesem Band können dann Hilfseingaben dem Prover oder dem Verifier zur Verfügung gestellt werden.

**Definition 2.12 interaktive TM mit Zusatzeingabe**
Sei $M$ eine interaktive TM wie in Definition 2.3. $M$ ist dann eine interaktive TM mit Zusatzband, wenn sie ein zusätzliches Leseband besitzt. Es wird „Auxiliary Band" genannt und der Inhalt dieses Bandes „Auxiliary Eingabe" oder „Hilfseingabe".

Durch diese Definition besitzt eine TM eine weitere Eingabe. Dies muß dann auch in den anderen Definitionen, die auf den interaktiven TM aufbauen, geändert werden. Ich gebe dies nur für das „interaktive Beweissystem" an. Die anderen Definitionen folgen analog.

**Definition 2.13 interaktives Beweissystem mit Zusatzeingabe**
Ein Paar $(P, V)$ heißt **interaktives Beweissystem** für eine Sprache $L$, wenn $V$ polynomiell ist und die folgenden Bedingungen gelten:

**Vollständigkeit** Für jedes $x \in L$ existiert eine Zeichenkette $y$ so, daß für jedes $z \in \{0,1\}^*$ gilt:

$$P_r(< P(y), V(z) > (x) = 1) \geq \frac{2}{3}$$

**Zuverlässigkeit** Für jedes $x \notin L$ und jede interaktive TM $B$ und jedes $y, z \in \{0,1\}^*$ gilt:

$$P_r(< P(y), V(z) > (x) = 1) \leq \frac{1}{3}$$

Da diese Beweissystem wieder mit Konstanten Grenzen definiert ist, kann man äquivalent zur der Definition des „allgemeinen interaktiven Beweissystems" auch ein solches für diese Definition erstellen. An dieser Stelle wird darauf verzichtet, da nur die Wahrscheinlichkeiten für Vollständigkeit und Zuverlässigkeit ausgetauscht werden müssen.
Als nächstes wird eine „Ansicht" auf die abgelaufene Interaktion definiert. Diese ist für jede der beteiligten Maschinen unterschiedlich.

**Definition 2.14** $view_A(\cdot)$
Sei $(A, B)$ ein interaktives Beweissystem. Dann bezeichnet $view_A(A(\cdot), B(\cdot))$ oder kurz $view_A^B(\cdot)$ eine Zufallsvariable, die den Inhalt des Empfangsbandes von $A$ und des von $A$ benutzen Teil des Zufallsbandes der Kommunikation mit $B$ enthält. Entsprechendes beschreibt $view_B(\cdot)$ die Sicht der Maschine $B$.

## 2.6 Einweg Funktionen

Einweg Funktionen sind wichtige Elemente der modernen Kryptographie. Sie basieren darauf, daß bestimmte Funktionen leicht zu berechnen sind, aber praktisch nicht zu invertieren sind. Das „praktisch" bezieht sich dabei auf den Einsatz in der Kryptographie. Das bedeutet, daß das berechnen der Funktion $y = f(x)$ eine Komplexität von $O(n \log n)$ haben darf, die von $x = f^{-1}(y)$ jedoch $O(2^n)$. Es ist jedoch wichtig zu beachten, daß die ganze Überlegung von Einweg-Funktionen nur Sinn macht, wenn **P** $\neq$ **NP** gilt. Bisher existiert kein Beweis dafür, daß es Einweg-Funktionen wirklich gibt, aber es gibt Funktionen wie die Multiplikation zweier großer Primzahlen mit dem Problem der Primfaktorzerlegung oder Restklassenringe mit dem Problem des diskreten Logarithmusses, die vermutlich Einweg-Funktionen sind.

## 2.7 Bit-Commitment-Schemen

Folgende Informationen habe ich aus [5, Schwarze] entnommen.
Commitment-Schemen sind das digitale Analogon zu versiegelten, undurchsichtigen Briefumschlägen, bei denen sichergestellt ist, daß die Inhalte wirklich von dem Absender stammen und nach dem Versiegeln nicht mehr verändert werden können. Ein Bit-Commitment-Schema ist das Basisverfahren, wie ein einzelnes Bit eines Strings in einem Commitment-Schema verarbeitet wird. Ein Commitment-Scheme besteht aus zwei Phasen:

1. *Festlegungsphase*
2. *Öffnungsphase*

In der Festlegungsphase wird ein Bit $\delta \in \{0,1\}$ von dem sendenden Teilnehmer $S$ so festgelegt, daß es hinterher nicht mehr verändert werden kann, während $\delta$ für jeden anderen nicht sichtbar ist. Das veröffentlichte Paket wird dabei *Commitment* genannt. In der Öffnungsphase wird das festgelegte Bit $\delta$ dem empfangenen Teilnehmer $E$ bekannt gegeben, wobei durch das Bit-Commitment-Schema garantiert ist, daß nur das von $S$ ursprünglich festgelegte Bit $\delta$ und nicht $1-\delta$ herauskommen kann.
Commitment-Schemen haben zwei wesentliche Eigenschaften:

- Eindeutigkeit (engl. binding property): Der Wert kann nach der Festlegungsphase vom Sender nicht mehr verändert werden
- Geheimhaltung (engl. hiding property) Vor der Öffnungsphase lernt der Empfänger nichts über den Wert von $\delta$

Aus diesen Informationen läßt sich nun ein Bit-Commitment-Schema formal definieren.

**Definition 2.15 Bit-Commitment-Schemen**
Sei $(S,E)$ ein interaktives Beweissystem ($S$ für Sender und $E$ für Empfänger) und $\delta \in \{0,1\}$ das nur $S$ bekannte Bit. $(S,E)$ ist ein **Bit-Commitment-Scheme**, wenn gilt:

- *Geheimhaltung (engli. hiding)*:
  Der Empfänger $E$ kann vor der Öffnung ein 0-Commitment nicht von einem 1-Commitment unterscheiden, selbst wenn er von dem Protokoll abweicht. Sei $E^*$ eine beliebige (möglicherweise täuschende) Turing Maschine. Dann sind nach der Feststellungphase $\{view_{E^*}((S(\delta=0),E^*)(n))\}_{n\in\mathbf{N}}$ und $\{view_{E^*}((S(\delta=1),E^*)(n))\}_{n\in\mathbf{N}}$berechenbar ununterscheidbar.
- *Festlegung oder Eindeutigkeit (engl. binding)*:
  Seien $v \in \{0,1\}$ das nur $S$ bekannte Bit und $n \in \mathbf{N}$ beliebig. Nach der Festlegungsphase sei $view_E((S(v),E)(n))$ ein mögliches $v$-Commitment. Dann darf $view_E((S(v),E)(n))$ berechenbar resp. statistisch nicht gleichzeitig ein mögliches 0-Commitment und ein mögliches 1-Commitment sein.

Die gemeinsame Eingabe $n$ kann hierbei als Sicherheitsparameter interpretiert werden.

Sollen mehr als ein Bit in einem Commitment verschlüsselt werden, kann ein Bit-Commitment mehrfach hintereinander ausgeführt werden. Bei der sequentiellen Ausführung bleibt die Geheimhaltung gewahrt bei paralleler Ausführung im Allgmeinen nicht. Letzteres wird bei [2] und [1] gezeigt.

# 3. ZERO-KNOWLEDGE BEWEISSYSTEME

In diesem Kapitel werden die Zero-Knowledge Beweissysteme eingeführt. Was zero-knowedge bedeutet wurde bereits in Kapitel 2 beschrieben. Dabei teilen wir diese Beweissysteme in zwei Klassen auf:

- perfekt Zero-Knowledge Beweissysteme
- computational Zero-Knowledge Beweissysteme

Diese Unterscheidung ist wichtig, da bei einem Beweissystem mit der berechenbar zero-knowledge Eigenschaft nicht sicher ist, daß durch ein Nachbearbeiten der Kommunikation, nach dem eigentlichen Beweis, nicht doch zusätzliche Informationen abgeleitet werden können. Dies ist bei Beweissystemen mit der perfekt zero-knowledge Eigenschaft ausgeschlossen. Diese haben aber den Nachteil, daß sie aufwendiger zu implementieren sind. Die Definition von *zero-knowledge* baut auf einem Simulator auf, der die Kommunikation zwischen belibigen Verifiern und dem Prover simultiert. Kann dann nicht zwischen der Ausgabe des Simulators und der Ausgabe des Beweissystems unterschieden werden, dann wurde kein weiteres Wissen aus der Kommunikation gelernt. Daraus wird dann folgende Definition.

**Definition 3.1 perfekt zero knowledge**
Sei $(P,V)$ ein interaktives Beweissystem für eine Sprache $L$. $(P,V)$ ist **perfekt zero knowledge** für jede probalistische polynomielle interaktive TM $V^*$, wenn es einen probalistischen polynomiellen Algorithmus $M^*$ gibt, so daß für jedes $x \in L$ gilt:

1. Mit maximaler Wahrscheinlichkeit von $\frac{1}{2}$, für die Eingabe $x$ gibt der Algorithmus $M^*$ ein spezielles Symbol ($\perp$) aus. ($Pr(M^*(x) = \perp) \leq \frac{1}{2}$)
2. Sei $m^*(x)$ eine Zufallsvariable, die die Verteilung von $M^*(x)$ bedingt durch $M^*(x) \neq \perp$ beschreibt. ($Pr(m*(x) = \alpha) = Pr(M^*(x) = \alpha) | M^*(x) \neq \perp$) für jedes $\alpha \in \{0,1\}^*$). Dann sind die folgenden Zufallsvariablen identisch verteilt:
   - $< P, V^* > (x)$, die Ausgabe der interaktiven TM $V^*$
   - $m^*(x)$, die Ausgabe des Algorithmusses $M^*$ wenn nicht $\perp$ ausgegeben wird

Die Maschine $M^*$ heißt ein **perfekter Simulator** für die Interaktion von $V^*$ mit $P$.

Da aber nicht immer eine perfekte Sicherheit gefordert wird, sondern nur eine Sicherheit während der Kommunikation,

gibt es eine weitere Definition für *berechenbar zero-knowledge*.

**Definition 3.2 berechenbar zero-knowledge**
Sei $(P,V)$ ein interaktives Beweissystem für eine Sprache $L$. Dann heißt $(P,V)$ **berechenbar zero-knowledge** oder auch nur **zero-knowledge**, wenn für jede probalistische polynomielle TM $V^*$ ein probalistischer polynomieller Algorithmus existiert, so daß folgende Mengen berechenbar ununterscheidbar sind:

1. $\{< P,V^* > (x)\}_{x\in L}$ (die Ausgabe von $V^*$ nach der Interaktion mit $P$ für die gemeinsame Eingabe $x$
2. $\{M^*(x)\}_{x\in L}$ (die Ausgabe des Algorithmus $M^*$ für Eingabe $x$

Den Algorithmus $M^*$ nennt man einen Simulator für die Interaktion von $V^*$ mit $P$.

Nach diesen beiden Definitionen ist noch zu erwähnen, daß zero-knowledge eigentlich eine Eigenschaft des Provers ist und nicht des gesamten Beweissystems. Um im nächsten Kapitel mit den Beweissystemen mit konstanter Rundenzahl anfangen zu können benötigen wir noch eine Erweiterung der Definition von zero-knowledge. In der Praxis benötigt ein Prover eine zusätzliche Eingabe um dem Verifier etwas zu beweisen, wie den Zeugen aus dem Beispiel aus Kapitel 2. Bei dem Beweissystem zur Graphen 3-Färbung wäre dies eine vorhandene Färbung des Graphen. Um diese Hilfseingabe ordentlich angeben zu können erweitern wir die vorherigen Definitionen in folgender Weise:

**Definition 3.3 zero-knowledge**
Sei $(P,V)$ ein interaktives Beweissystem mit Zusatzeingabe für eine Sprache $L$. Wir bezeichnen mit $P_L(x)$ die Menge von Zeichenreihen, für die die Vollständigkeitsbedingung bezüglich $x \in L$ gilt. (d.h. $\forall z \in \{0,1\}^* Pr(< P(y),V(z) > (x) = 1) \geq \frac{2}{3}$).
$(P,V)$ heißt **zero-knowledge bezüglich einer Zusatzeingabe** wenn für jede probalistische polynomielle interaktive TM $V^*$ ein probalistischer polynomieller Algorithmus $M^*$ existiert, so daß die folgenden Familien berechenbar ununterscheidbar sind:

1. $\{< P(y),V^*(z) > (x)\}_{x\in L, y\in P_L, z\in\{0,1\}^*}$
2. $\{M^*(x,z)\}_{x\in L, z\in\{0,1\}^*}$

Das bedeutet, daß für jeden probalistischen Algorithmus, der polynomiell zur Länge der ersten Eingabe ist, für jedes Polynom $p(\cdot)$, alle $x \in L$ die lang genug sind,alle $y \in P_L$ und alle $z \in \{0,1\}^*$ gilt:
$|Pr(D(x,z,< P(y),V^*(z) > (x)) = 1) -$
$Pr(D(x,z,M^*(x,z)) = 1)| < \frac{1}{p(|x|)}$

In dieser Definition war $y$ eine zusätzliche Eingabe für den Prover und $z$ eine zusätzliche Eingabe für den Verifier.

# 4. ZERO-KNOWLEDGE BEWEISE MIT KONSTANTER RUNDENZAHL

In diesem Kapitel geht es um die Konstruktion von Beweissystemen mit konstanter Rundenzahl. Diese Beweissysteme werden im weiteren Verlauf des Kapitels auch Protokolle genannt. Die Konstruktion solcher Protokolle ist sehr wichtig, denn in der Praxis müssen diese effizient sein und zusätzlich eine vernachlässigbare Fehlerwahrscheinlichkeit aufweisen. Aus diesem Grund wird die Eigenschaft der *Rundeneffizienz* eingeführt.

**Definition 4.1 rundeneffizient**
Ein zero-knowledge Beweissystem ist **rundeneffizient** wenn gilt:

1. Das Beweissystem kommt mit konstanter Rundenzahl aus
2. die Fehlerwahrscheinlichkeit des Beweissystems ist vernachlässigbar

Mit Runde wird eine Kommunikationsschritt zwischen einem Prover und einem Verifier bezeichnet. Bei schwachen zero-knowledge Beweisen hat man oft eine konstante Rundenzahl, aber dann eine Fehlerwahrscheinlichkeit von z.B. $\frac{1}{3}$. Diese lassen sich dann in der Realität nicht einsetzen. Im Verlauf dieses Kapitels wird dann gezeigt, wie die Fehlerwahrscheinlichkeit verringert werden kann, ohne die Runden-Komplexität zu vergrößern und die zero-knowledge Eigenschaft zu behalten. Diese Konstruktion basiert auf den Arbeiten [2] und [4]. Um jetzt mit der Konstruktion zu beginnen wird zunächst das Protokoll aus Konstruktion 4.4.6 von [2] betrachtet (siehe dazu Listing 1).

Die Programme für den Verifier und den Prover können in probalistischer polynomieller Zeit implementiert werden. Das gelingt dem Prover durch seine zusätzliche Hilfseingabe. Der Beweis, daß dieses Protokoll zero-knowledge ist, wird in [2, Goldreich] Kapitel 4.4 gegeben. Das Protokoll hat konstante Anzahl von Runden, aber die Fehlerwahrscheinlichkeit ist zu hoch. Goldreich schlägt deshalb vor, das Protokoll oft genug zu wiederholen um die Fehlerwahrscheinlichkeit zu reduzieren. Dies führt dann aber zu einem Beweissystem mit nicht konstanter Rundenzahl. Ein weiterer Vorschlag wäre, das Protokoll parallel ablaufen zu lassen. Dann kann aber nicht mit Sicherheit gesagt werden, ob die zero-knowledge Eigenschaft erhalten bleibt.
Deshalb wird jetzt das Protokoll aus Listing 1 so verändert, daß es zwei verschiedene Commitment-Schemen verwendet. Ein Commitment Schema mit perfekter Sicherheit und eins mit berechenbarer Sicherheit. Die Konstruktion dieses Protokolls ist in Listing 2 zu sehen. In diesem Protokoll werden, nicht wie in Listing 1, nur eine Kante des Graphen überprüft, sondern es wird eine Anfrage nach mehreren Kanten gestellt. Dies ermöglicht dann den Erhalt der Rundenkomplexität und die Verringerung der Fehlerwahrscheinlichkeit. In dem Protokoll bedeutet:

$P_{m,s}(E)$ Das Commitment mit perfekter Sicherheit von dem Wert $E$ nach dem Empfang einer initialen Nachricht $m$ und einem Zufallsstring $s$.

$C_s(E)$ Das Commitment mit berechenbarer Sicherheit von dem Wert $E$ mit dem Zufallsstring $s$.

Das erste was nach dieser Konstruktion geklärt werden muß ist, ob Listing 2 weiterhin ein interaktiver Beweis für G3F ist.

**Listing 1: Protokoll für Graphen 3-Färbung**

In dieser Konstruktion wird mit $C_s(\delta)$
die Festlegung des Senders in einem
Commitment−Scheme mit Zufallswert $s$
und Wert $\delta$ bezeichnet.
Als Commitment−Scheme wird irgendein one−way
Interaktion Commitment−Scheme ausgewählt.

Common Input:
Ein einfacher (3−färbbarer) Graph $G = (V, E)$.
Sei $n = |V|$ und $V = \{1, \cdots, n\}$

Auxiliary Input für den Prover:
Eine 3−Färbung von $G$
bezeichnet mit $\psi$

1. Schritt des Provers:
Der Prover wählt eine zufällige Permutation $\pi$
über $\{1, 2, 3\}$ und setzt $\phi(v) = \pi(\psi(v))$
für jedes $v \in V$. Dann benutzt der Prover ein
Commitment−Scheme um sich auf eine Farbe für
jeden Knoten festzulegen. Genauer: Der Prover
wählt gleichmäßig und unabhängig
$s_1, \cdots, s_n \in \{0,1\}^n$, berechnet
$c_i = C_{s_i}(\psi(i))$ für jedes $i \in V$ und
sendet $c_1, \cdots, c_n$ zum Verifier.

1. Schritt des Verifiers:
Der Verifier wählt gleichmäßig eine Kante
$(u, v) \in E$ und sendet diese an den
Prover

Motivierende Bemerkung:
Der Verifier fragt nach den Farben von $u$ und $v$

2. Schritt des Provers:
o.B.d.A. nehmen wir an, daß der Prover vom
Verifier
die Kante $(u, v)$ empfangen hat. Jetzt benutzt
der Prover die Öffnungsphase des Commitment−
Schemes
dazu, die Farben der Knoten $u$ und $v$ zu
veröffentlichen.
Genauer: der Prover sendet $s_u, \psi(u))$ und
$s_v, \psi(v))$ zum Verifier

2. Schritt des Verifiers:
Der Verifier überprüft, ob die gesendeten
Werte zu den
Commitments passen und ob sie unterschiedlich
sind.
Genauer: Nach dem Empfang von $(s, \delta)$ und $(s', \tau)$
überprüft der Verifier ob
$c_u = C_s(\delta), c_i = C_{s'}(\tau), \delta \neq \tau$ und ob beide
in $\{1, 2, 3\}$ liegen.
Wenn alle diese Bedingungen gelten, dann
akzeptiert der
Verifier, ansonsten verwirft er.

**Listing 2: runden-effizientes Protokoll für G3F**

Common Input:
Ein einfacher 3−färbbarer Graph $G = (V, E)$
Sei $n = |V|, t = n \cdot |E|$
Auxiliary Input to the Prover:
Eine 3−Färbung von $G$ benannt $\psi$
Provers preliminary step (P0):
Der Prover legt sich in der Festlegungsphase
des Commitment Schemes mit perfekter
Sicherheit fest und sendet an den Verifier
eine Nachricht $m$
Verifiers preliminary step (V0):
Der Verifier wählt unabhängig und
gleichverteilt eine Sequenz von
$t$ Kanten, $\overline{E} = ((u_1, v_1), \cdots, (u_t, v_t)) \in E^t$
Dann sendet er dem Prover ein
Commitment zu diesen Kanten.
Genauer: Der Verifier wählt gleichverteilt
$\overline{s} \in \{0,1\}^n$ und sendet $P_{m,\overline{s}}(\overline{E})$ zum Prover
Motivating Remark:
An diesem Punkt ist der Verifier auf eine
Sequenz von Kanten festgelegt. Diese
Festlegung ist von perfekter Sicherheit.
Provers Step (P1):
Der Prover wählt unabhängig und
gleichverteilt $t$ Permutationen
$\pi$ über $\{1, 2, 3\}$
und setzt $\phi_j(\nu) = \pi_j(\psi(v))$ für jedes
$v \in V$ und $1 \leq j \leq t$
Nun benutzt der Prover ein
commitment−scheme mit
berechenbarer Sicherheit und legt sich
auf Farben für jeden Knoten fest.
Genauer: Der Prover wählt unabhängig und
gleichverteilt $s_{1,1}, \cdots, s_{n,t} \in \{0,1\}^n$
und berechnet $c_{i,j}(\psi_j(i))$ für jedes
$i \in V$ und $1 \leq j \leq t$ und sendet
$c_{1,1}, \cdots, c_{n,t}$ zum Verifier.
Verifiers Step (V1):
Der Verifier veröffentlicht die Sequenz
$\overline{E} = ((u_1, v_1), \cdots, (u_t, v_t))$
Genauer: Der Verifier sendet $(\overline{s}, \overline{E})$
Motivating Remark:
Ab diesem Punkt ist die gesamte Festlegung des
Verifiers veröffentlicht. Er erwartet nun
für jedes $j$ die Farben die ihm von der
t−ten Färbung der Knoten $u_j$ und $v_j$
zugewiesen wurden.
Provers Step (P2):
Nun überprüft der Prover, ob die eingegangenen
Nachrichten zu dem Commitment des Verifiers
passen. Wenn dies nicht der Fall ist,
beendet der Prover sich sofort. Wie bezeichnen
die Sequenz der $t$ Kanten, die gerade
veröffentlicht worden sind mit
$(u_1, v_1), \cdots, (u_t, v_t)$. Nun benutzt
der Prover die Veröffentlichungsphase des
Commitment Schemes mit berechenbarer
Sicherheit für jedes $j$ die j−te Färbung
von Knoten $u_j$ und $v_j$. Der Prover sendet
folgendes Quadrupel von Knoten:
$(s_{u_1,1}; \psi_1(u_1), s_{t_1,1}; \psi_1(v_1)), \cdots$
$(s_{u_t,t}; \psi_t(u_t), s_{t_t,t}; \psi_t(v_t))$
Verifiers Step (V2):
Der Verifier überprüft nun, ob jedes $j$ in dem
j−ten Quadrupel aus einer richtigen
Veröffentlichung des Commitments besteht.
Genauer: Beim Empfangen von $(s_1, \psi_1, s'_1, \tau_1)$ bis
$(s_t, \psi_t, s'_t, \tau_t)$ überprüft
der Verifier ob jedes $j$ gilt das:
$c_{u_j,j} = C_{s_j}(\delta_j), c_{t_j,j} = C_{s'_j}(\tau_k)$
und $\delta_j \neq \tau_j$ und beide sind
in $\{1, 2, 3\}$. Wenn alle
Bedingungen erfüllt sind dann akzeptiert
der Verifier sonst nicht.

**Satz 4.2**
Das Protokoll in Listing 2 ist ein interaktiver Beweis für G3F.

*Beweis.* Es ist also zu zeigen:

1. Wenn die gemeinsame Eingabe $G = (V, E)$ ein 3-färbbarer Graph ist, dann akzeptiert der Verifier immer, wenn er mit dem Prover interagiert
2. Wenn die Eingabe $G = (V, E)$ kein 3-färbbarer Graph ist, dann lehnt der Verifier mit mindestens einer Wahrscheinlichkeit von $\frac{1}{2}$ ab, egal mit welcher Maschine er interagiert.

Also:

1. Bei Eingabe eines 3-färbbaren Graphen akzeptiert der Verifier, wenn er mit dem vorgesehenen Prover kommuniziert. Dies gilt nach der Konstruktion des Protokolls, da der Prover dann auf die Anfrage des Verifiers mit einer korrekten Kantenfärbung antworten kann.
2. Angenommen die Eingabe war kein 3-färbbarer Graph. Dann enthält das Commitment, das der Prover in Schritt P1 an den Verifier sendet, mindestens eine nicht erlaubte Färbung von einer Kante. Da der Verifier seine Festlegung in einem Commitment mit perfekter Sicherheit verschickt hat, ist dem Prover in Schritt 1 nicht bekannt, nach welchen Kanten der Verifier fragen wird. Für ihn ist also jede Möglichkeit gleich wahrscheinlich. Deshalb ist die Wahrscheinlichkeit, daß die Festgelegten Kanten und die festgelegte Färbung übereinstimmen maximal:
$$\left(1 - \frac{1}{|E|}\right)^t \approx e^{-n} < 2^{-n}$$

Das bedeutet, daß Listing 2 ein interaktives Beweissystem für G3F ist. □

Nun bleibt noch der Nachweis zu erbringen, daß Listing 2 die zero-knowledge Eigenschaft besitzt.
Den gesamten Beweis anzugeben würde den Rahmen dieser Ausarbeitung sprengen, deshalb beschränke ich mich darauf einen Simulator für das Protokoll anzugeben und dann die Schritte zu zeigen, die nötig sind, um den Beweis auf die standard Argumente zu reduzieren, die z.B. in Kapitel 4.4 von [2] benutzt werden, um für Listing 1 die zero-knowledge Eigenschaft zu beweisen.

**Satz 4.3**
Das Protokoll in Listing 2 erfüllt die zero-knowledge Eigenschaft.

*Beweis.* Dies wird dadurch gezeigt, daß ich einen Simulator für jeden möglichen $V^*$ angebe, der die Kommunikation von Prover und Verifier simuliert. Ich werde erst eine komplizierteren Simulator $M^*$ angeben, dann aber zeigen, daß es reicht, mit einem deutlich vereinfachtem Simulator weiterzuarbeiten.

**Listing 3: Simulator $M^*$**

$M^*$
−−−−−−−−

Start:
$M^*$ wählt eine Belegung des Zufallbandes $r$ für $V^*$.
Dann bestimmt der Simulator die Commitment Nachricht vom Verifier, indem er $V^*$ mit $G$ und $r$ startet. Nach dem Senden der Commitment Initialisierungs Message m erhält der Simulator die Commitment Nachricht CM.

Schritt 1:Die angefragten Kanten speichern:
$M^*$ generiert eine Folge von $n \cdot t$ zufälligen Commitments zu dummy Werten (alle Werte sind 1). und übergibt diese an $V^*$. Wenn $V^*$ nun eine Folge von $t$ Kanten $(u_1, v_1), \cdots, (u_t, v_t)$ veröffentlicht, die zu den Commitments passen, speichert der Simulator diese und geht über zum nächsten Schritt. Sind die Kanten keine gültige Veröffentlichung des Commitments CM, dann hält der Simulator und gibt seine jetzige Sicht aus.
Dies ist im Einklang mit Schritt P2 des Provers im Protokoll.

Schritt 1.5: Abschätzung der Wahrscheinlichkeit $q(G, r)$
In diesem Schritt wird der vorherige Schritt solange ausgeführt bis eine feste ( polynomiell in $|G|$) Anzahl von gleichen Veröffentlichungen gezählt wurde. Daraus wird dann eine Abschätzung für $q(G, r)$ berechnet. Dies ist die Wahrscheinlichkeit, daß in einem weiteren Aufruf von $V^*$, das gleiche Commitment noch ein Mal gesendet wird.

Schritt 2:Interaktion generieren, die zu den Angefragten Kanten paßt
Der Simulator generiert nun eine Folge von $n \cdot t$ Commitments $c_{1,1}, \cdots c_{n,t}$ so daß für jedes $j = 1, \cdots t$ gilt: $c_{u_j,j}$ und $c_{t_j,j}$ sind zufällige commitments zu Werten aus $\{1, 2, 3\}$ und alle anderen $c_{i,j}$ sind zufällige commitments zu dummy Werten (alle Werte sind 1) Die diesem Commitment zugrunde liegenden Werte nennen wir pseudo Färbung. Der Simulator startet $V^*$ neu und übergibt diese Commitments an $V^*$. Wenn $V^*$ die in Schritt 1 gespeicherten Werte als Veröffentlichung zurückgibt, dann kann $M^*$ die Simulation mit einer ,,echten'' Interaktion beenden. Wenn nicht wird Schritt 2 wiederholt bis die Werte übereinstimmen oder der Schritt maximal $poly(|G|)/q(G, r)$ durchgeführt wurden. Wird kein passendes Commitment gefunden gibt der Simulator ein Time−Out Symbol aus.

Im Schritt 1.5 des Simulators kann, durch eine geeignete Wahl eines großgenügenden Polynoms, sichergestellt werden, daß ,mit übergroßer Wahrscheinlichkeit $(1-2^{-poly(|G|)})$, die Annäherung in einem konstanten Faktor um $q(G, r)$ liegt.

Aus der maximalen Laufzeit von Schritt 2 und $q(G,r)$ kann man die maximale Laufzeit des Simulator bestimmen. Diese ist $q(G,r) \cdot \frac{poly(|G|)}{q(G,r)} = poly(|G|)$. Das bedeutet, daß der Simulator eine erwartete polynomielle Laufzeit besitzt.

Nun wird ein vereinfachter Simulator angegeben. Bei diesem können wir aber nicht genau sagen, ob er polynomielle Laufzeit hat. Dieses Problem wird aber dadurch gelöst, daß wir die Wahrscheinlichkeit, daß der originale Simulator das „time-out" Zeichen ausgibt, begrenzen und uns so auf den vereinfachten Algorithmus beschränken können.

**Listing 4: Simulator $M^{**}$**

$M^{**}$
————————

Start:
$M^*$ wählt eine Belegung des Zufallbandes $r$ für $V^*$. Dann bestimmt der Simulator die Commitment Nachricht vom Verifier, indem er $V^*$ mit $G$ und $r$ startet. Nach dem Senden der Commitment Initialisierungs Message m erhält der Simulator die Commitment Nachricht CM.

Schritt 1:die angefragten Kanten speichern
$M^*$ generiert eine Folge von $n \cdot t$ zufälligen Commitments zu dummy Werten (alle Werte sind 1). und übergibt diese an $V^*$. Wenn $V^*$ nun eine Folge von $t$ Kanten $(u_1, v_1), \cdots, (u_t, v_t)$ veröffentlicht, die zu den Commitments passen, speichert der Simulator diese und geht über zum nächsten Schritt. Sind die Kanten keine gültige Veröffentlichung des Commitments CM, dann hält der Simulator und gibt seine jetzige Sicht aus. Dies ist im Einklang mit Schritt P2 des Provers im Protokoll.

Schritt 2:Interaktion generieren, die zu den Angefragten Kanten paßt
Der Simulator generiert nun eine Folge von $n \cdot t$ Commitments $c_{1,1}, \cdots c_{n,t}$ so daß für jedes $j = 1, \cdots t$ gilt: $c_{u_j,j}$ und $c_{t_j,j}$ sind zufällige commitments zu Werten aus $\{1,2,3\}$ und alle anderen $c_{i,j}$ sind zufällige commitments zu dummy Werten (alle Werte sind 1) Die diesem Commitment zugrunde liegenden Werte nennen wir pseudo Färbung. Der Simulator startet $V^*$ neu und übergibt diese Commitments an $V^*$. Wenn $V^*$ die in Schritt 1 gespeicherten Werte als Veröffentlichung zurückgibt, dann kann $M^*$ die Simulation mit einer ,,echten'' Interaktion beenden. Wenn nicht wird Schritt 2 wiederholt bis die Werte übereinstimmen.

Ich betrachte jetzt das Verhalten des vereinfachten Simulators im allgemeinen Fall. Es wird mit $q(G,r)$ die Wahrscheinlichkeit, daß $V^*$ bei Eingabe $G$ und Zufallsband $r$ das richtige Commitment aus Schritt V0, nach dem Empfang eines Commitments für eine beliebige pseudo Färbung aus Schritt 1, veröffentlicht wird. Sei ähnlich $p(G,r)$ die Wahrscheinlichkeit, daß $V^*$ das richtige Commitment aus Schritt V0, nach dem Empfang eines Commitments für eine beliebige pseudo Färbung aus Schritt 2, veröffentlicht wird.
Der Unterschied zwischen $q(G,r)$ und $p(G,r)$ ist vernachlässigbar gering. Es wird festgestellt, daß der Simulator Schritt 2 mit Wahrscheinlichkeit $q \stackrel{\text{def}}{=} (G,r)$ und jede Ausführung von Schritt 2 wird mit einer Wahrscheinlichkeit von $p = p(G,r)$ erfolgreich beendet. Also ist die erwartete Anzahl von Ausführung von Schritt 2 $q \cdot \frac{1}{p}$.Hier erkennt man nun daß Problem. Es kann nicht sichergestellt werden, daß $\frac{q}{p}$ annähernd 1 ist oder durch ein Polynom zur Eingabe beschränkt ist. Wähle als Beispiel $p = 2^{-n}$ und $q = 2^{-n/2}$. Dann ist der Unterschied zwischen $p$ und $q$ vernachlässigbar aber $\frac{q}{p}$ ist nicht durch $poly(n)$ beschränkt.
Die Unterschiede zwischen den beiden Simulatoren ist offensichtlich. Der original Simulator versucht durch seinen Zwischenschritt 1.5, daß $q(G,r)$ zu bestimmen, um dann Schritt 2 in der Anzahl der Ausführungen begrenzen zu können. Wenn jetzt gezeigt wird, daß die Wahrscheinlichkeit, daß der originale Simulator diese Schranke überschreitet, vernachlässigbar ist, dann reicht es die zero-knowledge Eigenschaft für den vereinfachten Algorithmus zu zeigen.

**Lemma:**
Die Wahrscheinlichkeit, daß der originale Simulator $M^*$ ,das „time-out" Symbol ausgibt, ist eine vernachlässigbare Funktion von $|G|$.

*Beweis.* Sei $\triangle(G,r)$ die Wahrscheinlichkeit, daß der original Simulator $M^*$ ,bei der Eingabe von eionem Grpahen $G$ und eines Zufallsbandes $r$ , das „time-out" Symbol ausgibt. Dann gilt:

$$\begin{aligned} \triangle(G,r) &\stackrel{\text{def}}{=} q(G,r) \cdot \left(1 - (1-p(G,r))^{\frac{poly(|G|)}{q(G,r)}}\right) \\ &= q(G,r) \cdot (1-p(G,r))^{\frac{poly(|G|)}{q(G,r)}} \end{aligned}$$

Ich nehme jetzt an, daß $\triangle(G,r)$ keine vernachlässigbare Funktion ist.
Dann existiert ein Polynom $P(\cdot)$, eine unendliche Folge von Graphen $\{G_n\}$ und eine unendliche Anzahl von Zufallsbändern $\{r_n\}$, so daß $\triangle(G_n, r_n) > \frac{1}{P(n)}$. Daraus folgt, daß für jedes dieser $n$ gilt:

$$q(G_n, r_n) > \frac{1}{P(n)}$$

. Dann gibt es zwei Fälle:

1. Für unendlich viele dieser $n$ gilt $p(G_n, r_n) \geq q(G_n, r_n)$. In diesem Fall erhalten wir

$$\begin{aligned} \triangle(G_n, r_n) &\leq (1 - p(G_n, r_n))^{\frac{poly(|G_n|}{q(G_n, r_n)}} \\ &\leq \left(1 - \frac{q(G_n, r_n)}{2}\right)^{\frac{poly(|G_n|}{q(G_n, r_n)}} \\ &< 2^{\frac{-poly(|G_m|)}{2}} \end{aligned}$$

   Dies widerspricht unserer Annahme, daß $\triangle(G_n, r_n) > \frac{1}{poly(n)}$.

2. Für unendliche viele dieser $n$ gilt $p(G_n, r_n) < q(G_n, r_n)$ Daraus folgt, daß $|q(G_n, r_n) - p(G_n, r_n)| > \frac{P(n)}{2}$. Dies

führt aber zu einem Widerspruch in der berechenbaren Sicherheit des vom Prover verwendeten Commitment Schemas.

Aus diesen beiden Widersprüchen folgt, daß die Behauptung gilt. □

Nun fehlt noch zu zeigen, daß die Ausgabe des Simulators $M^{**}$ und die des Beweissystems berechenbar ununterscheidbar sind.

**Lemma:**
Die Familien $\{M^{**}(G)\}_{G\in G3F}$ und $\{(P,V^*)(G)\}_{G\in G3F}$ sind berechenbar ununterscheidbar.

*Beweis.* Der Beweis wird wieder über Widerspruch durchgeführt.
Es wird angenommen, daß die beiden Familien von einem Algorithmus $A$ mit Lückee$(G)$ $(e(G) = |Pr(A(M^{**}(G)) = 1) - Pr(A((P,V^*)(G)) = 1)|)$ unterscheidbar sind. Und es gilt $e(G) \geq \frac{1}{P(|G|)}$ für ein Polynom $P(\cdot)$ und eine endliche Folge von Graphen $\{G_n : n \in S$. Es wir nun ein Prädikat $R$ definiert, so daß $R(y) = 1$ für $y$ ein Mitschnitt einer Interaktion in der der Verifier das Commitment aus V0 richtig veröffentlicht. Im anderen Fall gilt $R(y) = 0$.Nun sind wieder zwei Fälle zu betrachten:

1. Für eine unendliche Anzahl von diesen $n$ gilt:
   $Pr(R((P,V^*)(G_n)) = 1) \geq \frac{e(G_n)}{3}$
   Für diese $n$ wird dann Schritt 2 (von $M^{**}$) maximal $\frac{3}{e(G)} \leq 3P(|G_n|)$ oft durchgeführt. Mit einer Wahrscheinlichkeit von $\frac{1}{2}P(|G_n|)$ wird von $M^*$ Schritt 2 öfter als $T(|G|) \stackrel{\text{def}}{=} 6P(|G_n|)^2$ ausgeführt. Schränkt man nun die Ausführungen von Schritt 2 in $M^*$ auf $T(|G_n|)$ ein, dann ist die Ausgabe von der Ausgabe von $M^**$ maximal $\frac{1}{2}P(|G_n|)$ entfernt. Daraus folgt, daß A immer noch die Ausgabe von $M^*$ eingeschränkt auf $T(|G_n|)$ mit einer Lückevon $e(G_n) - \frac{1}{2}P(|G_n|)$ unterscheiden kann. Von hier ab, können die gleichen Argumente angewendet werden, wie sie Goldreich in Kapitel 4.4 von [2] verwendet, um einen Widerspruch herbeizuführen.

2. Für unendlich viele dieser $n$ gilt:
   $Pr(R((P,V^*)(G_n)) = 1) < \frac{e(G_n)}{3}$
   Die Wahrscheinlichkeit, daß für diese $G_n$ , die Interaktion von $V^*$ mit dem realen Prover in Schritt V1 enhält, ist $1 - \frac{e(g_n)}{2}$. Man kann weitere zwei Fälle unterscheiden:

   (a) In diesem Fall nehmen wir an, daß der Simulator in Schritt S1 mit maximaler Wahrscheinlichkeit von $1 - \frac{e(G_n)}{2}$. Also gibt es eine Lücke von mindestens $\frac{e(G_n)}{6}$ zwischen der Wahrscheinlichkeit, daß $V^*$ das Commitment richtig veröffentlicht wurde, bei einer Interaktion mit dem Prover und der Wahrscheinlichkeit, daß $V^*$ das Commitment richtig veröffentlicht bei einer Interaktion mit dem Simulator. In diesem Fall wird jetzt $V^*$ als Unterscheider für das Commitment von dummy Werte von Commitments von einer Färbung benutzt. Dies ist ein Widerspruch zu der berechenbaren Sicherheit des verwendeten Commitment Schemas

   (b) In diesem Fall nehmen wir an, daß der Simulator in Schritt S1 mit einer Wahrscheinlichkeit von mindestens $1 - \frac{e(G_n)}{2}$ anhält. Das bedeutet, daß die simulierte Interaktion und die reale Interaktion mit dieser Wahrscheinlichkeit suspendieren. Der Algorithmus $A$ muß also solche Suspendierungen mit mindestens einer Lücke von $\frac{e(G_n)}{2}$ unterscheiden. Daraus folgt, daß $A$ zwischen Commitments von dummy Werten und Commitments von echten Färbungen unterscheidet. Dies ist wieder ein Widerspruch zu der berechenbaren Sicherheit des verwendeten Commitment Schemas.

Aus diesen Widersprüchen folgt nun die Behauptung. □

Aus den beiden Lemmata folgt, daß Listing 2 ein rundeneffizienter zero-knowledge Beweis für Graphen 3-Färbung ist. □

Damit wurde nun gezeigt, wie man aus einem „schwachen" zero-knowledge Beweissystem ein rundeneffizientes zero-knowledge Beweissystem macht. Zum Abschluß ist noch zu bemerken, daß solche zero-knowledge Beweise für alle Sprachen aus **NP** existieren.

# 5. AUSBLICK

Die hier vorgestellten Konstruktionen von zero-knowledge Beweisen mit konstanter Rundenzahl lassen sich noch weiter umkonstruieren, so daß Commitment Schemen mit berechenbarer Sicherheit verwendet werden können. Dies ist in [2] und [4] nachzulesen.

# 6. LITERATURVERZEICHNIS


[1] Brassard, Gilles, Claude Crepeau, and Moti Yung. *Constant Round perfekt zero-knowledge computationally convincing protocols.* Theoretical Computer Science, 1991.

[2] Oded Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

[3] Oded Goldreich. Zero-knowledge twenty years after its invention. Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, Mar 2004.

[4] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. Journal of Cryptology, 1996, Mar 1995.

[5] Thomas Schwarze. Zero-knowledge arguments. Diplomarbeit, FernUniversität -Gesamthochschule-Hagen, Fachbereich Informatik, sep 2003.