SEMINAR "INTERAKTIVE BEWEISSYSTEME"

BJÖRN KARGE — ZERO KNOWLEDGE (1)

Seminarleitung: Prof. J. Köbler, O. Beyersdorff

Zusammenfassung. In interaktiven Zero-Knowledge Beweissysteme ist Zugriff auf Wissen des Provers, welches nicht zur Lösung des betrachteten Entscheidungsproblems notwendig ist, unmöglich. Neben einer formalen Einführung des Zero-Knowledge-Begriffs wird ein Zero-Knowledge-Protokoll für Graphen-Isomorphie (ISO) gezeigt und die Abgeschlossenheit von ZK-Beweissystemen unter paralleler und sequentieller Komposition untersucht.

Inhaltsverzeichnis

1. Einführung

Diese Ausarbeitung mag etwas trocken sein, da gerade die Formalisierung des Zero-Knowledge-Begriffes einer ihrer Schwerpunkte ist. Eine sehr anschauliche Vermittlung des Zero-Knowledge-Begriffes wird in [NNR99] gegeben, etwas anspruchsvoller, aber immer noch gut verständlich, geht es bei [May] zu.

Bisher haben wir interaktive Beweissysteme mit der Zielstellung untersucht, herauszufinden, *ob*, bzw. *welche* Sprachen mit ihnen entscheidbar sind. Im folgenden soll auch dem "*wie*" Beachtung geschenkt und überprüft werden, welche Protokolle und welches Verhalten des Provers es ermöglichen, daß der Prover ausschließlich für die Lösung des betrachteten Entscheidungsproblems herangezogen werden kann, d.h., nicht zu anderen Zwecken "mißbraucht" wird.

Deshalb ist es zunächst notwendig, zu bestimmen, worin der “Mißbrauch” eines Provers bestehen könnte, und auf welche Weise man diesen mit Hilfe einer formalen Definition für eine gewisse Klasse von Provern ausschließen kann. Dies erfordert die Betrachtung der Ausgabe eines Interaktiven Beweissystems als vom Eingabewort abhängige Zufallsvariable, weshalb einige Konstrukte eingeführt werden, mit denen sich diese Zufallsvariablen besser handhaben lassen. Nachdem der Begriff des Zero-Knowledge-Beweissystems dann formal eingeführt wurde, wird das Verhalten solcher Beweissysteme unter sequentieller und paralleler Komposition untersucht, und eine Menge von Zero-Knowledge-Beweissystemen eingeführt, die unter sequentieller Komposition abgeschlossen ist.

1.1. **Knowledge vs. Information.**

Zunächst ist es wichtig den Begriff Knowledge näher zu bestimmen. Wir betrachten zunächst folgendes

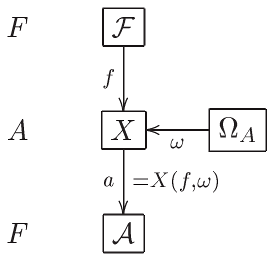
Beispiel. A und F vereinbaren folgendes Protokoll: F übersendet A eine natürliche Zahl n und A antwortet, ob n seiner Meinung nach eine Primzahl ist.

Falls A die Antwort durch ablesen in einer Tabelle ermittelt, so ist die Wahrscheinlichkeit, daß n eine Primzahl ist, falls A mit “ja” antwortet, sehr groß (A könnte schließlich Fehler beim ablesen machen). Falls A die Frage jedoch per Münzwurf (von dem F nichts weiß) entscheidet, so erhält F (anhand der “ja”-“nein”-Verteilung) höchstens Informationen über die Beschaffenheit der Münze (von der F nichts weiß). Im ersten Fall erlangt F **Wissen** darüber, ob n Primzahl ist, im zweiten Fall erhält F lediglich **Daten** bzw. **Information**.

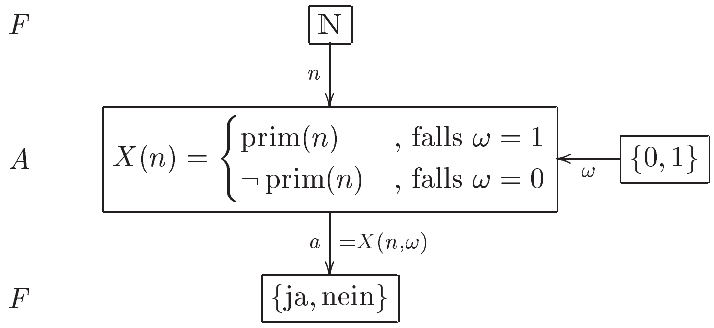
Wir verallgemeinern dieses Beispiel wie folgt: zwei Parteien F, A mögen über einen Kommunikationskanal miteinander in Verbindung stehen. F sendet eine Frage $f \in \mathcal{F}$ über diesen Kanal an A, und A antwortet F daraufhin mit dem Wert $a \in \mathcal{A}$. Auf dieser Ebene ist es für F aufgrund der Vereinbarungen noch nicht möglich, zu entscheiden, nach welchen Kriterien A antwortet, ob sich also irgend ein “Sinn” hinter den Antworten von A verbirgt. Die Antworten von A können deshalb lediglich als Daten bzw. “freie” Information betrachtet werden.

$$(1) \quad F: \quad f \longrightarrow A \qquad\qquad (2) \quad a = a(f, ?) \qquad\qquad (3) \quad F \longleftarrow a \quad : A$$

Setzt man der Einfachheit halber voraus, daß A gedächtnislos ist, sich also nie an evtl. vorher stattgefundene “Unterhaltungen” erinnern kann, ist es möglich, die Antwort von A als diskrete Zufallsvariable $X : \mathcal{F} \times \Omega_A \to \mathcal{A}$ aufzufassen, wobei Ω_A die Menge der möglichen “internen” Entscheidungen von A beschreibt. Falls F nun Kenntnis über X hat, kann F mit der Antwort etwas anfangen, da F dann weiß, *wie stark X mit anderen, für F interessanten Verteilungen korreliert*:



Man beachte, daß die Wahl von ω dabei, unter anderem, auch von f abhängen kann. Das obenstehende "Primzahlbeispiel" kann man dann z.B. wie folgt modellieren:



wobei $\omega = 0$, falls A sich beim ablesen irrt. Wir nehmen nun an, A irrt sich bei einer gewissen Primzahl seltener als bei allen anderen, weil es sich um die vierstellige Zahlenkombination ihres Bankschließfaches handelt. Sofern wir über dieses Wissen verfügen, könnten wir theoretisch durch ausreichende Geduld und mehrfache Überprüfung aller vierstelligen Primzahlen mit hoher Sicherheit diese Zahlenkombination in Erfahrung bringen.

1.2. **Was soll Zero Knowledge?**

Wendet man obige Sichtweise auf Interaktive Beweissysteme an so nehmen Prover und Verifier die Rollen von A bzw. F ein. Allerdings verhindert das Gedächtnis des Provers eine einfache Modellierung von X als Zufallsvariable. Ziel ist es, trotzdem eine Klasse von Provern zu definieren, die neben dem im Protokoll vereinbarten, kein weiteres Wissen an den Verifier weiterreichen. Gelingt dies, so erhält man eine Klasse von interaktiven Beweissystemen, in denen ein Verifier statistisch von einem Sachverhalt überzeugt werden kann, ohne weitere, evtl. schwer zu berechnende Details an ihn weiterzugeben.

Praktische Anwendungen dafür sind vor allem im Bereich der Kryptographie zu finden, man stelle sich, beispielsweise, anonyme und zugleich zugangsbeschränkte Systeme wie gewisse Online-Transaktionsdienste oder Online-Wahllokale vor, in denen man (als Prover) lediglich zeigen muß, daß man zugangsberechtigt ist, jedoch keine weiteren Informationen über seine Identität (oder irgendetwas sonst) preisgeben möchte. In diesem Falle wäre die Anonymität selbst gegenüber "boshaften" Lauschern am Kommunikationskanal gewährleistet. Ein weiteres, höchst unerfreuliches aber sehr anschauliches Anwendungsgebiet von Zero-Knowledge-Protokollen sind Lösegeldverbrechen. Hier wird durch Videos oder Fotos bewiesen, daß sich bestimmte Personen in der Gewalt der Erpresser befinden, sowie Aufschluß über deren Zustand gegeben. Der Aufenthaltsort läßt sich daraus in der Regel jedoch nicht ableiten.

Um den Begriff Zero-Knowledge-Protokoll formal zu fassen wird die Menge der *mit Hilfe des Provers* zu lösenden Probleme auf Entscheidungsprobleme reduziert und gefordert, daß *jeder* Verifier ohne Prover simuliert werden kann, falls die Lösung des Entscheidungsproblems bekannt ist. Da es sich ansonsten beim Verifier um eine probabilistische, polynomialzeitbeschränkte TURING-Maschine ($\mathcal{P}$-PTM) handelt, ist jedoch davon auszugehen, daß es zu jedem Eingabewort verschiedene mögliche

Ausgabeworte geben kann, und man formuliert schließlich, daß die Wahrscheinlichkeitsverteilung der Ausgaben zu einem gegebenen Eingabewort im Falle einer positiven Antwort auf das Entscheidungsproblem nicht von der der Simulation unterscheidbar sein soll.

2. Berechnungsmässige Ununterscheidbarkeit

2.1. **Definition.** Sei M eine Menge, $f : M \to \mathbb{R}$ eine Abbildung. Dann heißt [1]

$$\operatorname{supp} f := \overline{M \setminus \ker f} = \overline{\{m \in M | f(m) \neq 0\}} = \overline{f^{-1}(\mathbb{R}^*)}$$

der ***Träger*** oder ***Support*** von f. ◁

2.2. *Bemerkung.*

(1) $f^{-1}(\mathbb{R}^*)$ ist endlich $\iff$ $\operatorname{supp} f$ ist endlich
(2) $\operatorname{supp} f$ endlich $\implies$ $\operatorname{supp} f = f^{-1}(\mathbb{R}^*)$ ◁

Wir beschäftigen uns im folgenden nur mit endlichen Trägern, können deshalb also immer Gebrauch von der zweiten Beobachtung machen und die Sache mit dem topologischen Abschluß getrost vergessen; wir benötigen diese Definition im wesentlichen für die elegante Formulierung der folgenden

2.3. **Definition.** Sei I eine Indexmenge, Ω eine Menge von Elementarereignissen. Eine Folge $(p_i)_{i \in I}$ von Wahrscheinlichkeitsverteilungen $p_i : \Omega \to [\![0, 1]\!]$ mit *endlichem* Träger nennen wir ***Folge diskreter Wahrscheinlichkeitsverteilungen mit endlichem Träger***, bzw. ***FDW*** oder auch (engl.) ***probability ensemble***. ◁

Insbesondere für Sprachen L kann man sich vorstellen, daß so für FDW $(p_w)_{w \in L}$ zu jedem Wort $w \in L$ eine minimale endliche Teilmenge $\Omega_w \subseteq \Omega$ mit

$$\sum_{\omega \in \Omega \setminus \Omega_w} p_w(\omega) = 0$$

definiert ist ($\Omega_w = p_w(\operatorname{supp} p_w)$). Außerdem kann man mit einer *FDW* die Wahrscheinlichkeiten der verschiedenen Ausgaben einer PTM M bei gegebener Eingabe w beschreiben:

$$L \ni w \longrightarrow \boxed{M} \longrightarrow \omega \in_{p_w} \Omega_w \subseteq \Omega$$

2.4. **Definition.** Sei Σ ein Alphabet, $L \subseteq \Sigma^\star$ eine Sprache und $(p_w)_{w \in L}$ FDW. (p_w) heißt ***FDW mit polynomiell beschränkten Trägern***, oder ***$\mathcal{P}$-FDW*** falls es ein Polynom $r \in \mathbb{R}[X]$ so gibt, daß für alle $w \in L$ gilt: $\max\{|v| \big| v \in \operatorname{supp} p_w\} < r(|w|)$. ◁

Inhaltlich sichern wir durch diese Definition, daß Worte in $\operatorname{supp} p_x$ polynomiell durch in der Länge von x beschränkt sind. Dies wird dadurch motiviert, daß die Ausgaben eine $\mathcal{P}$-PTM auch polynomiell durch die Länge ihrer Eingabe beschränkt ist. Dem entsprechend kann man sich eine $\mathcal{P}$-FDW als eine Folge vorstellen, die für eine $\mathcal{P}$-PTM angibt, mit welcher Wahrscheinlichkeit eine Ausgabe ω bei Eingabe x erfolgt.

[1] $\overline{S}$ bezeichnet den *topologische Abschluß* der Menge S, also die kleinste abgeschlossene ("berandete") Menge, die S enthält. Z.B. gilt (bezüglich $\mathbb{R}$): $\overline{]\!]0, 1[\![} = [\![0, 1]\!]$, $\overline{\mathbb{N}} = \mathbb{N}$, $\overline{\mathbb{Q}} = \mathbb{R}$, $\overline{\mathbb{R} \setminus \mathbb{Q}} = \mathbb{R}$.

Es soll nun definiert werden, was es heißt, daß zwei FDW (p_w), (q_w) nicht unterscheidbar sind. Inhaltlich soll erreicht werden, daß in diesem Falle kein probabilistischer polynomialzeitbeschränkter Entscheider A (also $\mathcal{P}$-PTM mit Ausgabe 0 oder 1) die p_w und q_w bei hinreichender Länge von w mit Sicherheit unterscheiden kann:

2.5. Definition. Sei Σ ein Alphabet und $L \subseteq \Sigma^\star$ eine Sprache. Zwei $\mathcal{P}$-FDW $(p_w)_{w\in L}$, $(q_w)_{w\in L}$ über derselben Menge Ω von Elementarereignissen heißen ***berechnungsmäßig ununterscheidbar*** oder ***polynomiell ununterscheidbar***, falls es für alle probabilistischen polynomialzeitbeschränkten Entscheider A und alle Polynome $r \in \mathbb{R}[X]$ stets ein $l = l\big(A, r\big) \in \mathbb{N}$ so gibt, daß für alle $w \in L$ mit $|w| > l$ gilt:

$$\forall x \in_{p_w} \Omega, y \in_{q_w} \Omega : \big|\Pr\left[A(w,x) = 1\right] - \Pr\left[A(w,y) = 1\right]\big| \leq \frac{1}{r(|w|)}$$

◁

Alternativ kann man für Folgen von Zufallsvariablen (P_w), (Q_w) mit den Verteilungen p_w für P_w bzw. q_w für Q_w $(w \in L)$ den letzten Teil der Definition (ab "gilt:") auch äquivalent als

$$\big|\Pr\left[A(w,P_w) = 1\right] - \Pr\left[A(w,Q_w) = 1\right]\big| \leq \frac{1}{r(|w|)}$$

formulieren.

Anschaulich bedeutet berechnungsmäßige Ununterscheidbarkeit also, daß die Abstände der Verteilungen so gering sind, daß auch durch polynomiell viele Experimente $A(w,x)$ und $A(w,y)$ diese Differenz nicht ausreichend "verstärkt" werden kann (*probability amplification*). Zum Beweis sei $p \in \mathbb{R}[X]$ ein Polynom und wir setzen

$$\delta = \delta(x,y,w) := \big|\Pr\left[A(w,x) = 1\right] - \Pr\left[A(w,y) = 1\right]\big|.$$

Mit der Bernoulli-Ungleichung[2] erhält man: $1-(1-\delta)^{p(|w|)} \leq 1-(1-\delta p(|w|)) = \delta p(|w|)$. Da es definitionsgemäß für jedes Polynom r ein $w \in L$ mit $\forall x, y \in \Omega : \delta \leq r(|w|)^{-1}$ gibt, bleibt die Wahrscheinlichkeit für ausreichend große w auch nach polynomiell vielen Ausführungen des Experiments vernachlässigbar gering.

3. Zero-Knowledge Beweissysteme

Sei im weiteren Sei Σ ein Alphabet, $L \subseteq \Sigma^\star$ eine Sprache und $(P, \diamond)$ ein Interaktives Beweissystem mit dem Verifier $\diamond$ und Eingabeworten aus $\Sigma^\star$, das das L entscheidet. [3]

Wie bereits erwähnt, ist es unser Ziel, eine Klasse von interaktiven Beweissystemen zu bestimmen, in denen der Prover kein anderes Wissen als das vereinbarte mitteilen kann.

Da ein interaktives Beweissystem (IBS) die Aufgabe hat, zu einer gegebenen Sprache L und einem Eingabewort w zu entscheiden, ob $w \in L$ oder $w \notin L$, können wir die Funktion des Provers auf das Lösen genau dieses Entscheidungsproblems reduzieren. Da der Verifier eines interaktiven Beweissystems eine probabilistische $\mathcal{P}$-TM ist, ist die Ausgabe eines Beweissystems nicht durch das Eingabewort determiniert,

[2] $(1+x)^n \geq 1+nx$ für $x \geq -1$ und $n \in \mathbb{N} \setminus \{0\}$

[3] Der unkonventionelle Name soll darauf hindeuten, daß dieser Verifier bis auf die Existenz hier eine untergeordnete Rolle spielt

man kann also für jedes fixierte Eingabewort w die Ausgabe des IBS als Zufallsvariable auffassen:

3.1. **Definition.** Sei $w \in \Sigma^\star$. Sei ferner (P, V) ein interaktives Beweissystem und M eine $\mathcal{P}$-PTM, beide mit Eingabeworten aus $\Sigma^\star$.

Mit $(P, V)(w)$ sei dann die Zufallsvariable bezeichnet, die die möglichen Ausgaben des Beweissystems bei Eingabe w beschreibt. Analog bezeichne $M(w)$ die Zufallsvariable, die die möglichen Ausgaben von M bei Eingabe w beschreibt. $\triangleleft$

$(P, \diamond)$ soll als Zero-Knowledge bezeichnet werden, falls es zu *jedem* Verifier V (einschließlich $\diamond$) für dieses Beweissystem im Falle $w \in L$ einen Simulator, d.h. eine $\mathcal{P}$-PTM M_V gibt, für den die Verteilung der möglichen Ausgaben, $M_V(w)$, von der Verteilung der möglichen Ausgaben von (P, V), d.h. von $(P, V)(w)$ nicht zu unterscheiden ist.

Die Beschränkung auf $w \in L$ rührt evtl. daher, daß in der Praxis der Fall $w \notin L$ dem Versuch "unberechtigter" Authentifizierung nahe kommt und so nicht "schutzwürdig" ist. Außerdem braucht der Simulator so nicht mit dieser Information parametrisiert zu werden. Man kann aus der Existenz eines ZK-IBS für L aber nicht ohne weiteres auf die Existenz eines ZK-IBS für die co-Sprache $\overline{L}$ schließen!

Nun besteht die Aufgabe des Provers nicht nur darin, zu entscheiden, ob $w \in L$, sondern auch darin dies dem Verifier glaubhaft zu machen. Da diese, durch das Protokoll definierte Aufgabe ebenfalls mehr als polynomielle Laufzeit (auf Seiten des Provers) in Anspruch nehmen kann, sind Situationen denkbar, in denen M_V die nötige Rechenkapazität des Provers fehlt. Darum erlauben wir dem Simulator unter Umständen, mit beschränkter Wahrscheinlichkeit zu "versagen" und in diesem Falle ein spezielles Symbol, $\perp$, auszugeben, um das wir die Statistik "bereinigen".

3.2. **Definition.** $(P, \diamond)$ heißt ***perfekt Zero-Knowledge***, falls für jeden Verifier V, mit dem (P, V) ebenfalls ein L entscheidendes IBS mit Eingabeworten aus $\Sigma^\star$ ist, eine probabilistische $\mathcal{P}$-TM M_V und ein $t \in [\![0, 1[\![$ mit

(1) $\forall w \in L : \Pr[M_V(w) = \perp] \leq t$

(2) $\forall w \in L\ \forall \omega \in \Omega : \Pr[(P, V)(w) = \omega] = \Pr[M_V(w) = \omega | \omega \neq \perp]$

existieren. [4] $\triangleleft$

3.3. **Definition.** $(P, \diamond)$ heißt ***berechnungsmäßig Zero-Knowledge***, oder einfach ***Zero-Knowledge***, falls es zu jedem Verifier V, mit dem (P, V) ebenfalls ein L entscheidendes IBS mit Eingabeworten aus $\Sigma^\star$ ist, eine probabilistische $\mathcal{P}$-TM M_V gibt, sodaß die $\mathcal{P}$-FDW $((P, V)(w))_{w \in L}$ und $(M_V(w))_{w \in L}$ polynomiell ununterscheidbar sind. $\triangleleft$

In der letzten Definition wurde darauf verzichtet, dem Simulator das "kontrollierte Versagen" zu ermöglichen. Eine solche Möglichkeit führt jedoch zu einer äquivalenten Definition: Sei nämlich M_V ein Simulator, der mit Wahrscheinlichkeit $0 \leq t < 1$ "versagen" darf, sowie M'_V ein Simulator, der M_V $p(|w|)$ polynomiell oft ($p(|w|)$-mal) ausführt. M'_V gebe $\perp$ aus, falls M_V in jedem Durchlauf $\perp$ ausgegeben hat, sonst jedoch die Ausgabe der ersten erfolgreichen Ausführung von M_V. Man erhält so $\Pr[M'_V(w) = \perp] \leq t^{p(|w|)}$. Sei M''_V schließlich

[4] Die entsprechende Definition in [Gol95] fixiert $t = \frac{1}{2}$, die Äquivalenz zur hier angegebenen Variante ist aber sehr leicht zu zeigen (vgl. Bemerkung zu Def. 3.3).

ein Simulator, der ein beliebig fixiertes $\omega_\perp \in \Omega$ ausgebe, falls $M'_V(w) = \perp$ und sonst die Ausgabe von M'_V übernehme. Da $0 \leq t < 1$ wird $t^{p(|w|)}$ exponentiell klein, $(M''_V(w))$ ist also polynomiell ununterscheidbar von $((P,V)(w))$. Die vollständige Formulierung (insbesondere des letzten Schrittes) sei dem interessierten Leser überlassen.

Der Vollständigkeit halber sei schließlich noch folgende Definition angegeben:

3.4. **Definition.** $(P,\diamond)$ heißt ***statistisch Zero-Knowledge***, falls für jedes Polynom $r \in \mathbb{R}[X]$ und jeden Verifier V, mit dem (P,V) ebenfalls ein L entscheidendes IBS mit Eingabeworten aus $\Sigma^\star$ ist, ein $l \in \mathbb{N}$ und eine probabilistische $\mathcal{P}$-TM M_V mit

$$\forall w \in L, |w| > l : \sum_{\omega\in\Omega} \left| \Pr\left[(P,V)(w) = \omega\right] - \Pr\left[M_V(w) = \omega\right] \right| \leq \frac{1}{r(|w|)}$$

existieren. ◁

3.5. *Bemerkung.*

(1) Die Zero-Knowledge-Eigenschaft bezieht sich in jeder dieser Definitionen ausschließlich auf den Prover und das Protokoll, da die angeführten Bedingungen für alle möglichen Verifier erfüllt sein müssen. Wenn deshalb im folgenden von einem ***Zero-Knowledge-Prover*** (analog für alle anderen Sorten von Zero-Knowledge) für eine Sprache L die Rede ist, soll damit ein Prover gemeint sein, der mit dem von ihm bedienten Protokoll und wenigstens einem Verifier zu einem ZK-IBS wird.

(2) Nicht jede interaktive $\mathcal{P}$-PTM kommt als Verifier V in Betracht, da (P,V) stets ein IBS, also widerspruchsfrei und vollständig sein muß.

(3) Der Simulator M_V kann im Falle $w \notin L$ ausgeben, was er möchte, da die Verteilung der Ausgabe nur für den Fall $w \in L$ betrachtet wird. ◁

Ein alternative Formulierung von Zero-Knowledge bezieht sich nicht auf die Ausgabe ω des Verifiers, sondern auf die vom Prover erhaltenen Nachrichten und den Inhalt des Zufallsbandes (mit dem der Verifier “würfelt”). Wir führen dazu folgendes ein:

3.6. **Definition.** Sei V ein Verifier, mit dem (P,V) ebenfalls ein L entscheidendes IBS mit Eingabeworten aus $\Sigma^\star$ ist. Das Paar $\mathrm{view}_V^P(w) = (\zeta, \rho)$ beschreibe mit ζ die von V verwendeten Zufallsbits und mit ρ den von V *gelesenen* Inhalt der Nachrichten, die V bei Eingabe w während der Interaktion mit P erhalten hat. ◁

Offenkundig ist $(P,V)(w)$ durch $\mathrm{view}_V^P(w)$ und w eindeutig festgelegt (und läßt sich in Polynomialzeit rekonstuieren), was zu folgender Definition führt:

3.7. **Satz.** $(P,\diamond)$ *ist genau dann* ***Zero-Knowledge****, falls es zu jedem Verifier V, mit dem (P,V) ebenfalls ein L entscheidendes IBS mit Eingabeworten aus $\Sigma^\star$ ist, eine probabilistische $\mathcal{P}$-TM M_V gibt, sodaß die $\mathcal{P}$-FDW $(\mathrm{view}_V^P(w))_{w\in L}$ und $(M_V(w))_{w\in L}$ polynomiell ununterscheidbar sind.* ◁

Beweis. 3.7 $\Longrightarrow$ 3.3: Da durch Simulation der Sitzung die Ausgabe von V in deterministischer Polynomialzeit aus $\mathrm{view}_V^P(w)$ und w berechnet werden kann, führt die polynomielle Ununterscheidbarkeid von $(\mathrm{view}_V^P(w))$ und $(M_V(w))$ auch zur polynomiellen Ununterscheidbarkeid der aus $\mathrm{view}_V^P(w)$ bzw. $M_V(w)$ berechneten Ausgaben (andernfalls könnte man ja auf diese Weise einen Polynomialzeit-Unterscheider für $(\mathrm{view}_V^P(w))$ und $(M_V(w))$ konstruieren).

3.3 $\Longrightarrow$ 3.7 ist ebenfalls sehr einfach: Sei V ein Verifier, der die Bedingungen von Definition 3.3 erfüllt. Offensichtlich existiert dann ein, die gleichen Bedingungen erfüllender, Verifier V' der view_V^P ausgibt, also $(P, V')(w) = \mathrm{view}_V^P(w)$. V' hat definitionsgemäß einen passenden Simulator M'_V der nach Konstruktion die in Definition 3.7 an M_V gestellten Bedingungen erfüllt (denn $(P, V')(w) = \mathrm{view}_V^P(w)$). □

Analoge Sätze und Beweise ergeben sich auch für die anderen Varianten von Zero-Knowledge. Dieser Abschnitt soll mit der Definition einiger, auf dem Zero-Knowledge-Begriff basierender Komplexitätsklassen beendet werden:

3.8. **Definition.** Wir bezeichnen mit $\mathcal{PZK}/\mathcal{CZK}/\mathcal{SZK}$ die Mengen von Sprachen, für die es Beweissysteme gibt, die perfekt/berechnungsmäßig/statistisch Zero-Knowledge sind. Außerdem setzen wir $\mathcal{ZK} = \mathcal{CZK}$. ◁

3.9. **Korollar.** $\mathcal{BPP} \subseteq \mathcal{PZK} \subseteq \mathcal{SZK} \subseteq \mathcal{CZK} \subseteq \mathcal{IP}$. □

Bei Existenz so genannter nicht-uniformer Einwegfunktionen folgt sogar $\mathcal{CZK} = \mathcal{IP}$ (vgl. [Gol95]).

4. Graphenisomorphie (ISO) ist Zero-Knowledge

4.1. **Satz.** ISO $\in \mathcal{PZK}$ ◁

Wir beweisen diesen Satz konstruktiv durch Angabe eines Zero-Knowledge-Protokolls. Das Eingabewort w beschreibe dabei zwei Graphen $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$. (P, V) soll nun probabilistisch entscheiden, ob $G_1 \cong G_2$. O.B.d.A. können wir dabei davon ausgehen, daß $m = |E_1| = |E_2|$ und $n = |V_1| = |V_2|$, da alle anderen Fälle in Polynomialzeit entscheidbar sind. Da eine entsprechende Umbenennungen effizient durchführbar ist, sei außerdem von $V_1 = V_2 = \{1, ..., n\}$ ausgegangen. In Abb.1 wird das unter diesen Voraussetzungen gewählte Protokoll angegeben. [5]

Zum besseren Verständnis dieses Protokolls dient dabei möglicherweise folgende Skizze für H_1 im Falle $G_1 \cong G_2$:

$$\begin{tikzcd} G_1 \arrow[rr, "\varphi"] \arrow[dr, "\pi_1"'] & \circlearrowleft & G_2 \arrow[dl, "\pi_1 \circ \varphi^{-1}"] \\ & H_1 & \end{tikzcd} \qquad \psi_1 = \begin{cases} \pi_1 & , k_1 = 1 \\ \pi_1 \circ \varphi^{-1} & , k_1 = 2 \end{cases}$$

[5] Im Unterschied zu dem in [Gol99] angegebenen Protokoll werden durch dieses Protokoll alle an V gesendeten Objekte "doppelt" angefertigt. Dies hat zwei Gründe: (1.) Zum Nachweis der Widerspruchsfreiheit wird im Falle $w \notin L$ eine Akzeptanzwahrscheinlichkeit von weniger als einem Drittel gefordert. Bei der "einfach-Version" erhält man jedoch (wie wir sehen werden) den Wert $\frac{1}{2}$. (2.) Obwohl intuitiv klar ist, daß man diesen Wert durch zweifache Ausführung auf $\frac{1}{4}$ verringern kann, nehme ich an, daß eine solche Argumentation den aufmerksamen Leser irritieren würde: der Verifier würde sich nämlich, formal betrachtet, so nicht an das Protokoll halten. Darüberhinaus sind die bislang eingeführten Formen von Zero-Knowledge unter paralleler oder sequentieller Komposition nicht abgeschlossen, und die Hilfsmittel, mit denen eine solche Vorgehensweise hier dennoch zu rechtfertigen wäre, werden aber erst in Abschnitt 6 eingeführt.

(1) V sendet $w = (G_1, G_2)$ an P
(2) P sendet zwei zu G_1 isomorphe Graphen H_1, H_2 an V, die wie folgt bestimmt werden ($j \in \{1,2\}$):
   i) P wählt zufällig eine Permutation $\pi_j \in \mathfrak{S}_n$
   ii) $H_j := \pi_j(G_1) := \big(V_1, \{\pi_j(e) | e \in E_1\}\big)$
(3) V wählt zufällig $k_1, k_2 \in \{1,2\}$ und sendet k_1, k_2 an V
(4) P sendet Permutationen ψ_1, ψ_2 an V, die so bestimmt werden ($j \in \{1,2\}$):
   - falls $G_{k_j} \cong H_j$, ist ψ_j ein Isomorphismus mit $\psi_j(G_{k_j}) = H_j$
   - andernfalls wählt P ψ_j beliebig aus $\mathfrak{S}_n$
(5) V gibt 1 aus, falls $\psi_1(G_{k_1}) = H_1$ und $\psi_2(G_{k_2}) = H_2$, sowie 0 sonst.

Abbildung 1. Zero-Knowledge-Protokoll für ISO

Zum Nachweis, daß dadurch tatsächlich ein perfekt-Zero-Knowledge IBS gegeben ist, müssen wir Vollständigkeit und Widerspruchsfreiheit zeigen, sowie, daß das so entstandene IBS perfekt Zero-Knowledge ist.
Vollständigkeit: Bei Einhaltung des Protokolls ist die Akzeptanzwahrscheinlichkeit für jedes $w \in L$ mindestens $\frac{2}{3}$.

Beweis. Da P aufgrund seiner unbeschränkten Rechenzeit im Falle $w \in L$, (d.h., $G_1 \cong G_2$) stets den Isomorphismus φ mit $\varphi(G_1) = G_2$ bestimmen und so Isomorphismen ψ_1, ψ_2 mit $\psi_j(G_{k_j}) = H_j$ $(j = 1, 2)$ angeben kann ist die Akzeptanzwahrscheinlichkeit hier sogar 1. □

Widerspruchsfreiheit: Bei Einhaltung des Protokolls durch den Verifier (also unabhängig vom gewählten Prover) ist die Akzeptanzwahrscheinlichkeit für jedes $w \notin L$ höchstens $\frac{1}{3}$.

Beweis. Sei $w \notin L$, (d.h., $G_1 \ncong G_2$). Da wir die Akzeptanzwahrscheinlichkeit maximieren wollen, können wir annehmen, daß $H_j \cong G_1$ oder $H_j \cong G_2$ $(j = 1, 2)$. Sei nämlich o.B.d.A. $H_1 \ncong G_1$ und $H_1 \ncong G_2$. Dann kann es kein k_1, ψ_1 mit $\psi_1(G_{k_1}) = H_1$ geben, V würde also stets verwerfen (0 ausgeben). Sei deshalb $H_j \cong G_{l_j}$ mit $l_j \in \{1,2\}$ $(j = 1, 2)$.

V akzeptiert offensichtlich genau dann, wenn $k_j = l_j$ für $j = 1, 2$, da sonst für ein $j \in \{1,2\}$ wegen $G_{l_j} \cong H_j$ folgt, daß $G_{k_j} \ncong H_j$ und somit wegen der Transitivität von $\cong$ kein Isomorphismus ψ_j mit $\psi_j(G_{k_j}) = H_j \cong G_{l_j}$ existiert, weshalb V verwerfen würde. Da die Wahl der k_j unabhängig von der Wahl der l_j erfolgt (denn V hält sich an das Protokoll), gilt $(j = 1, 2)$:

$$\begin{aligned}
\Pr[k_j = l_j] =& \Pr[k_j = 1 | l_j = 1] + \Pr[k_j = 2 | l_j = 2] \\
=& \Pr[k_j = 1] \cdot \Pr[l_j = 1] + \Pr[k_j = 2] \cdot \Pr[l_j = 2] \\
=& \frac{1}{2}\big(\Pr[l_j = 1] + \Pr[l_j = 2]\big) = \frac{1}{2}
\end{aligned}$$

Da die Wahl der k_j auch unabhängig voneinander erfolgt, prüft man ähnlich nach:

$$\Pr[k_1 = l_1 \wedge k_2 = l_2] \underset{(!)}{=} \Pr[k_1 = l_1] \cdot \Pr[k_2 = l_2] = \frac{1}{4}$$

□

perfekt Zero-Knowledge: Seien P, V entsprechend der view-Variante von Definition 3.2 gewählt und fixiert. Wir setzen $t = \frac{3}{4}$ und konstruieren einen Simulator M_V für view_V^P:

(1) M_V initialisiert das Zufallsband des Verifiers mit ausreichend vielen Zufallsbits ζ und wählt zufällig $l_1, l_2 \in \{1, 2\}$.
(2) M_V simuliert V und erhält $w = (G_1, G_2)$ auf dem Kommunikationsband.
(3) M_V wählt zufällig $\pi_1, \pi_2 \in \mathfrak{S}_n$
(4) M_V wählt $H_1 := \pi_1(G_{l_1})$ und $H_2 := \pi_2(G_{l_2})$
(5) M_V schreibt H_1, H_2 auf das Kommunikationsband
(6) M_V simuliert V bis zum Eintreffen von k_1, k_2.
(7) Falls $k_1 \neq l_1$ oder $k_2 \neq l_2$, gibt M_V das Zeichen $\perp$ aus, und terminiert.
(8) Andernfalls setzt M_V $\psi_1 := \pi_1$ und $\psi_2 := \pi_2$ (der Verifier würde also stets π_1 und π_2 als Isomorphismen erhalten).
(9) M_V gibt $(\zeta, \rho) = \big(\zeta, (H_1, H_2, \psi_1, \psi_2)\big)$ aus und terminiert.

Sei $w = (G_1, G_2)$ mit $G_1 \cong G_2$ fixiert. Ähnlich wie im Beweis der Widerspruchsfreiheit zeigt man: $\Pr[k_1 = l_1 \wedge k_2 = l_2] = \frac{1}{4}$, d.h., $\Pr[M_V(w) = \perp] = 1 - \frac{1}{4} = \frac{3}{4} \leq t$.

Sei nun $k_1 = l_1$ und $k_2 = l_2$. Offensichtlich würde V in dieser Situation stets akzeptieren. Wir müssen nur noch zeigen, daß $M_V(w)$ und $\mathrm{view}_V^P(w)$ die gleiche Verteilung haben.

Für den Inhalt des Zufallsbandes ist dies klar.

Da π_1 und π_2 gleichverteilt aus $\mathfrak{S}_n$ gewählt wurden und $G_1 \cong G_2$, müssen wir zeigen, daß es egal ist, ob G_1 oder G_2 zu H_j permutiert werden (denn im Originalprotokoll wurde stets G_1 genommen), also, daß für fixierte φ, G_1, H_j gilt: $\Pr[\pi_j(G_1) = H_j] = \Pr[\pi_j(\varphi(G_1)) = H_j]$. Als letztes muß gezeigt werden, daß für wieder fixiertes φ die Permutationen π_j und $\pi_j \circ \varphi^{-1}$ gleichverteilt sind, und somit die gleiche Verteilung haben, damit klar ist, daß auch die unterschiedliche Wahl der ψ_j sich nicht auf deren Verteilung auswirkt. Insgesamt ist das Grund genug für folgende Beobachtung:

4.2. **Lemma.** *Seien $n \in \mathbb{N}$ und $\varphi, \sigma \in \mathfrak{S}_n$ beliebig fixiert. Sei ferner zufällig und gleichverteilt $\pi \in \mathfrak{S}_n$ gewählt. Dann gilt:* $\Pr[\pi \circ \varphi = \sigma] = \Pr[\pi = \sigma \circ \varphi^{-}1] = \frac{1}{n!} = \Pr[\pi = \sigma]$.

□

Damit folgen direkt die obige Aussagen, und da w, P, V keinen weiteren Einschränkungen unterlagen, erhalten wir insgesamt: ISO $\in \mathcal{PZK}$.

5. Alle Zero-Knowledge $\mathcal{NP}$-Beweissysteme haben $\mathcal{BPP}$-Sprachen

Eine interessante Frage ist, ob es auch für "schwächere" Beweissysteme sinnvoll ist, Zero-Knowledge-Protokolle zu untersuchen. Dies soll in diesem Abschnitt insbesondere für $\mathcal{NP}$-Beweissysteme beantwortet werden.

5.1. **Definition.** Sei $L \subseteq \Sigma^\star$ und $R_L \subseteq \Sigma^\star \times \Sigma^\star$ mit $w \in L \iff \exists z \in \Sigma^\star : (w, z) \in R_L$. Dies induziert eine Abbildung

$$v : \Sigma^\star \times \Sigma^\star \to \{0, 1\}, \qquad v(w, z) = \begin{cases} 1 & \text{, falls } (w, z) \in R_L \\ 0 & \text{, sonst} \end{cases}.$$

Falls v in Polynomialzeit berechnet werden kann, heißt R_L $\mathcal{NP}$***-Relation*** von L, v heißt $\mathcal{NP}$***-Beweissystem*** (und die Menge aller L mit $\mathcal{NP}$-Beweissystemen heißt $\mathcal{NP}$). ◁

Wir müssen nun eine Klasse von interaktiven Beweissystemen konstruieren, die äquivalent zu $\mathcal{NP}$-Beweissystemen ist, um den Zero-Knowledge-Begriff darauf anwenden zu können.

5.2. **Definition.** Ein IBS (P, V) heißt ***interaktives*** $\mathcal{NP}$***-Beweissystem*** ($\mathcal{NP}$-IBS), falls

(1) Weder P noch V Zufallsbits zur Verfügung stehen [6]
(2) P in der Zeitkomplexität auf $\mathcal{NP}$ beschränkt ist ◁

Die Äquivalenz von $\mathcal{NP}$-IBS und $\mathcal{NP}$-Beweissystemen ist dabei offensichtlich. Außerdem mache man sich klar, daß $\mathcal{NP}$-IBS fehlerfrei entscheiden *müssen*, d.h., $\Pr[(P,V) = 1 | w \in L] = 1$ und $\Pr[(P,V) = 1 | w \notin L] = 0$.

5.3. **Satz.** *Die von Zero-Knowledge-$\mathcal{NP}$-IBS entschiedenen Sprachen liegen in $\mathcal{BPP}$.* ◁

Beweis. Wir erweitern den Simulator für ein Zero-Knowledge-$\mathcal{NP}$-IBS so, daß er ebenfalls die gegebene Sprache entscheidet: Sei (P, V) ein Zero-Knowledge-$\mathcal{NP}$-IBS mit Ausgabe 0 oder 1, sowie $L \subseteq \Sigma^\star$ eine Sprache, die (P, V) entscheidet. Definitionsgemäß gibt es nun einen Simulator M_V für view_V^P, für den die Verteilung von $M_V(w)$ im Falle $w \in L$ nicht von der Verteilung von $\mathrm{view}_V^P(w)$ zu unterscheiden ist.

Wir nehmen nun an, daß die Ausgabe von $M_V(w)$ zur Simulation der Ausgabe des Provers verwendet wird. Im Falle $w \in L$ wird V ebenfalls mit einer nicht signifikant von 1 abweichenden Wahrscheinlichkeit akzeptieren (sonst hätte man hier einen Def. 3.3 widersprechenden Unterscheider). Da V außerdem für $w \notin L$ nie akzeptiert und M_V eine $\mathcal{P}$-PTM ist, folgt $L \in \mathcal{BPP}$. □

Aus der Struktur des Beweises kann man erkennen, daß diese Eigenschaft für alle Beweissysteme mit einseitigem Fehler gilt:

5.4. **Korollar.** *Sei $L \subseteq \Sigma^\star$ eine Sprache, (P, V) ein Zero-Knowledge-IBS mit Ausgabe* 0 *oder* 1 *und* $\Pr[(P,V)(w) = 1 | w \notin L] = 0$, *sowie* $\Pr[(P,V)(w) = 1 | w \in L] \geq \frac{2}{3}$. *Dann ist $L \in \mathcal{BPP}$.* ◁

Beweis. Analog 5.3. □

5.5. **Korollar.** *Sei $L \subseteq \Sigma^\star$ eine Sprache, (P, V) ein ZK-IBS mit Ausgabe* 0 *oder* 1 *und öffentlichem Zufallsband, das L entscheidet. Falls als Prover $\mathcal{P}$-PTM gestattet sind, ist $L \in \mathcal{BPP}$.* ◁

Beweis. Wie bei 5.3 konstruieren wir einen Simulator M_V für view_V^P, der offensichtlich in Polynomialzeit deterministisch zu einem (evtl. “unehrlichen”) Prover Q umfunktioniert werden kann. Aufgrund der Widerspruchsfreiheit liegt auch die Akzeptanzwahrscheinlichkeit von $(Q, V)(w)$ für $w \notin L$ unterhalb $\frac{1}{3}$ und, nach Konstruktion, für $w \in L$ sehr dicht bei der von $(P, V)(w)$. Also $L \in \mathcal{BPP}$. □

[6] Wir müssen dem Verifier die Möglichkeit, zu Würfeln nehmen, solange $\mathcal{BPP} \subseteq \mathcal{NP}$ unklar ist.

Zero-Knowledge-Protokolle für solche Beweissysteme sind also überflüssig, da die betroffenen Sprachen auch ohne Prover entschieden werden können (falls der Verifier mächtig genug (d.h., in $\mathcal{BPP}$) ist). Im Falle der $\mathcal{NP}$-IBS ist der Verifier jedoch "nur" eine $\mathcal{P}$-TM und da $\mathcal{BPP} \supseteq \mathcal{P}$, wäre es im Falle $\mathcal{P} \neq \mathcal{NP}$ möglich, daß man mit ZK-$\mathcal{NP}$-IBS immer noch mehr als mit einer einfachen $\mathcal{P}$-TM entscheiden kann.

6. Zusammengesetzte Zero-Knowledge-Beweissysteme

Abschließend soll das Thema behandelt werden, ob die Nacheinanderausführung von Zero-Knowledge-Protokollen wieder Zero-Knowledge-Protokolle erzeugt. Problematisch ist hier, daß der Verifier z.B. Informationen aus vorhergehenden Sitzungen nutzen könnte.

6.1. Nacheinanderausführung von $\mathcal{ZK}$-Protokollen.

Um ein formal einfach zu handhabendes Beispiel zu konstruieren, führen wir den folgenden Begriff ein:

6.1. Definition. Sei $r \in \mathbb{Z}[X]$ ein Polynom mit $x > 0 \implies r(x) > 0$. Dann heißt eine Folge $(S_n)_{n \in \mathbb{N}}$ nichtleerer Mengen mit $S_n \subseteq \{0,1\}^{r(n)}$ ***$\mathcal{P}$-evasiv***, falls für alle $\mathcal{P}$-PTM A, $0 \leq c \in \mathbb{R}$, ausreichend große $n \geq n_0(A, c)$ und $x \in \{0,1\}^n$ gilt:

$$\Pr[A(x) \in S_n] < \frac{1}{n^c}.$$

◁

Zu einer $\mathcal{P}$-evasiven Folge gibt es also keine probabilistische $\mathcal{P}$-TM, die mit signifikanter Wahrscheinlichkeit Elemente von unendlich vielen S_n erraten kann.

6.2. Satz (Goldreich, Krawczyk; 1992). *Es gibt eine $\mathcal{P}$-evasive Folge $(S_n)_{n \in \mathbb{N}}$ pseudozufälliger Mengen für $r(n) = 4n$ und eine TM A mit $A(1^n) = S_n$.* ◁

Beweis: [GK93]. □

Der Begriff der Pseudozufälligkeit wird in [GK93] und [GK94] genauer besprochen, für unsere Zwecke soll die Vorstellung ausreichen, daß sich eine pseudozufällige Menge nicht in polynomieller Zeit von einer Menge zufällig ausgewählter Elemente unterscheiden läßt.

6.3. Korollar. *$\mathcal{ZK}$ ist unter sequentieller Komposition nicht abgeschlossen.* ◁

Beweis. Sei S_n eine wie durch Satz 6.2 zugesicherte $\mathcal{P}$-evasive Folge sowie $F : \{0,1\}^\star \to \{0,1\}$ eine Funktion mit $L_F := F^{-1}(1) \notin \mathcal{BPP}$. Wir betrachten nun folgendes Protokoll für $L = \{0,1\}^\star$ bei der gemeinsamen Eingabe $w \in \{0,1\}^\star$:

(1) V sendet eine zufälligen String $s \in \{0,1\}^{4|w|}$ an P
(2) Falls $s \in S_{|w|}$ sendet P den Wert $F(w)$ an V
(3) Falls $s \notin S_{|w|}$ sendet P einen zufällig aus $S_{|w|}$ gewählten String s_0 an V
(4) V akzeptiert, unabhängig von der Antwort von P

Es handelt sich dabei tatsächltich um ein (etwas umständliches) Zero-Knowledge-Protokoll: Vollständigkeit und Widerspruchsfreiheit sind offensichtlich. Die Zero-Knowledge-Eigenschaft folgt aus der $\mathcal{P}$-Evasivität von $S_{|w|}$: Die Wahrscheinlichkeit, daß $s \in S_{|w|}$ ist vernachlässigbar klein! Ein Simulator braucht den Fall (2) also nicht zu beachten und kann zufällig ein Element aus $S_{|w|}$ wählen. Da die S_n jedoch pseudozufällige Teilmengen aus $\{0,1\}^{4n}$ sind, und s_0 zufällig und gleichverteilt aus $S_{|w|}$ gewählt wird, kann dieser Schritt durch eine zufällige und gleichverteilte Wahl von s_0 aus $\{0,1\}^{4|w|}$ simuliert werden, wiederum ohne die resultierende Verteilung signifikant zu ändern (vgl. [GK94]). Damit gibt es einen Simulator für view_V^P und wir sind fertig.

Wird dieses Protokoll jedoch von ein- und derselben $\mathcal{P}$-PTM M zweimal ausgeführt, so könnte ein ("unehrlicher") Verifier das im ersten Durchlauf erhaltene s_0 bekannt geben und M könnte mit diesem Wissen einen zweiten (ebenfalls "unehrlichen") Verifier konstruieren, der anstelle des zufälligen s eben jenes s_0 an den Prover, der stets die gleiche Folge (S_n) verwendet, überträgt und M die Antwort des Provers bekanntgibt, womit M aber $w \in L_F$ entscheiden kann. ↯

□

6.2. Erweiterte $\mathcal{ZK}$-Beweissysteme.

Das Problem bei zusammengesetzten Zero-Knowledge-IBS liegt offensichtlich darin, daß Wissen aus vorhergehenden Sitzungen dazu verwendet werden kann, das Protokoll in Folgesitzungen "auszuhebeln". Aus diesem Grund werden wir diese Informationen "kanalisieren", um sie so kontrollierbar zu machen.

Zu diesem Zweck erhalten sowohl Prover als auch Verifier ein zusätzliches Eingabeband (engl. *auxiliary input*), welches Informationen zusätzlich zum gemeinsamen Eingabeband enthält. Die Komplexität wird dabei allerdings nach wie vor abhängig von der Länge der Eingabe w auf dem gemeinsamen Eingabeband betrachtet. Das bedeutet insbesondere, daß der Verifier nicht genug Zeit haben kann, um seine Zusatzeingabe vollständig auszuwerten.

6.4. Definition. Sei (P, V) ein Paar interaktiver probabilistischer TURING-Maschinen mit zusätzlichen Eingabebändern und gemeinsamer Eingabe w, wobei die Laufzeit von V polynomiell durch $|w|$ beschränkt ist. (P, V) heißt ***Interaktives Beweissystem mit Zusatzeingabe*** für $L \subseteq \Sigma^\star$, falls

(1) $\forall w \in L\ \exists x \in \Sigma^\star\ \forall y \in \Sigma^\star : \Pr[(P(x), V(y))(w) = 1] \geq \frac{2}{3}$
(2) $\forall w \notin L\ \exists x \in \Sigma^\star\ \forall y \in \Sigma^\star : \Pr[(P(x), V(y))(w) = 1] \leq \frac{1}{3}$ ◁

Diese Definition unterscheidet sich zunächst nicht grundlegend von der bisherigen Definition interaktiver Beweissysteme, da man stets einen Prover P_x bzw. Verifier V_y konstruieren kann, der diese Zusatzeingabe zu Beginn seiner Ausführung selbst erzeugt, und sich danach wie $P(x)$ bzw. $V(y)$ verhält.

6.5. Definition. Sei $L \subseteq \Sigma^\star$ eine Sprache und (P, V) ein IBS mit Zusatzeingabe, daß L entscheidet. Dann setzen wir für $w \in L$:

$$P_L(w) = \left\{ x \in \Sigma^\star \,\middle|\, \forall y \in \Sigma^\star : \Pr[(P(x), V(y))(w) = 1] \geq \frac{2}{3} \right\}$$

◁

$P_L(w)$ bezeichnet also alle Zusatzeingaben x des Provers, mit denen $(P(x), V(y))$ für jedes y das Wort w richtig entscheidet (d.h., mit einer Wahrscheinlichkeit von mehr als $\frac{2}{3}$ akzeptiert). Insbesondere ist $\bigcap_{w \in L} P_L(w)$ die Menge aller Zusatzeingaben x, mit denen $(P(x), V(y))$ für jedes y vollständig bzgl. L ist.

Mit diesen Hilfsmitteln können wir nun eine Zero-Knowledge-Definition formulieren, die die kontrollierte Nacheinanderausführung von Zero-Knowledge-Protokollen erlaubt.

6.6. Definition. Sei $L \subseteq \Sigma^\star$ eine Sprache und $(P, \diamond)$ ein IBS mit Zusatzeingabe und Ausgabe 0 oder 1, das L entscheidet. $(P, \diamond)$ heißt ***Zero-Knowledge unter Berücksichtigung von Zusatzeingaben*** (oder kurz ***ZK****), falls zu jeder interaktiven $\mathcal{P}$-PTM V mit Zusatzeingabe, mit der (P, V) ein IBS mit Zusatzeingabe und Ausgabe 0 oder 1 ist, das L entscheidet, eine $\mathcal{P}$-PTM M mit Zusatzeingabe existiert, sodaß die folgenden $\mathcal{P}$-FDW berechnungsmäßig ununterscheidbar sind:

- $\big((P(x), V(y))(w)\big)_{(x,y,w) \in P_L \times \Sigma^\star \times L}$
- $\big((M(y))(w)\big)_{(y,w) \in \Sigma^\star \times L}$ ◁

Da es sich bei den angegebenen Folgen nicht um $\mathcal{P}$-FDW handelt, wie sie hier definiert wurden, da die Indizes keine Worte aus $\Sigma^\star$ sind, muß dies präzisiert werden. Gemeint ist: Für jede $\mathcal{P}$-PTM A mit Ausgabe 0 oder 1 und jedes Polynom $r \in \mathbb{R}[X]$ gibt es ein $l = l(A, r) \in \mathbb{N}$ so, daß für jedes $w \in L$ mit $|w| > l$ und alle $x \in P_L(w)$, $y \in \Sigma^\star$ gilt:

$$\Big|\Pr\left[A(w, y, (P(x), V(y))(w))\right] - \Pr\left[A(w, y, (M(y))(w))\right]\Big| \le \frac{1}{r(|w|)}.$$

Dies unterscheidet sich nun erheblich von der vorher gegebenen Definition von Zero-Knowledge, da der Schwellwert $r(|w|)^{-1}$ unabhängig von den Zusatzeingaben ist, und A ausschließlich Informationen über die Zusatzeingabe von V erhält. Man kann hier also nicht mehr von $P(x)$ zu einem parameterlosen Prover P_x, der sich wie $P(x)$ verhält, übergehen, da P bereits fixiert und u.U. in die Bestimmung von l eingegangen ist. Gleiches gilt für den Verifier.

6.3. Nacheinanderausführung von erweiterten $\mathcal{ZK}$-Protokollen.

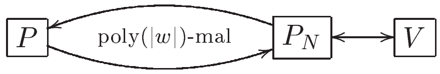
Wir sind nun in der Lage, Bedingungen zu formulieren, unter denen die Nacheinanderausführung von Zero-Knowledge-Protokollen wieder Zero-Knowledge ist.

6.7. Satz ("Sequential Composition Lemma" [Gol95])**.** *Sei*

- $L \subseteq \Sigma^\star$ *eine Sprache*
- P *ein ZK***-Prover für* L *(vgl. Bemerkung 3.5)*
- *die Interaktion mit* P *beliebig oft ausgeführbar, wobei eventuelle Zufallsbits von* P *jedoch bei jedem Durchlauf unabhängig von anderen Durchläufen sind*
- *das Protokoll von* P *dergestalt, daß* P *auschließlich polynomiell lange Antworten gibt, von denen die letzte ein spezielles, sonst nie verwendetes Zeichen* ■ *("Beweisende") am Schluß enthalte*
- $N \in \mathbb{Z}[X]$ *ein Polynom mit* $x > 0 \implies N(x) > 0$

Sei P_N *nun ein Prover, der bei Eingabe* w *das Protokoll von* P $N(|w|)$*-mal durch wiederholte Ausführung von* P *verlängert. Dann ist* P_N *ebenfalls ZK***.* ◁

Die Funktionsweise eines IBS mit P_N sei durch folgende Skizze illustriert:



Inhalt dieses Satzes ist also, daß die Interaktion mit *einem* fixierten ZK*-Prover von einer interaktiven $\mathcal{P}$-PTM polynomiell häufig wiederholt werden kann, ohne daß der Prover mehr als $w \in L$ verraten würde. Die Zusatzeingabe von P ist dabei allerdings für die Dauer der Ausführung von P_N fixiert, und wird deshalb nicht weiter betrachtet.

Im nun folgenden Beweis wird sich die Möglichkeit des zusätzlichen Eingabebandes des Verifiers zunutze gemacht: Die Kommunikation jeder interaktiven $\mathcal{P}$-PTM V mit Zusatzeingabe mit P_N, wird, synchron zum Erscheinen von ■ auf dem Kommunikationsband, in Phasen zerlegt, welche, jede für sich, durch eine interaktive $\mathcal{P}$-PTM V' mit Zusatzeingabe z simuliert werden können. V' simuliert V mit der Startkonfiguration und Zusatzeingabe y und gibt bei Erscheinen von ■ die dann aktuelle Konfiguration von V aus. Falls V terminiert gibt V' die Ausgabe von V aus. Durch “feedback” der Endkonfigurationen jeden Schrittes von V' an sich selbst erhält man schließlich eine vollständige Simulation von $(P_N, V(y))(w)$ (einzig das “bootstrapping” von V' bedarf noch besonderer Behandlung, aus diesem Grund wird die Maschine V'_b eingeführt). Aus den definitionsgemäß existierenden nichtinteraktiven Simulatoren für V'_b und V' wird schließlich ein Simulator M_V für V konstruiert, der Definition 6.6 erfüllt.

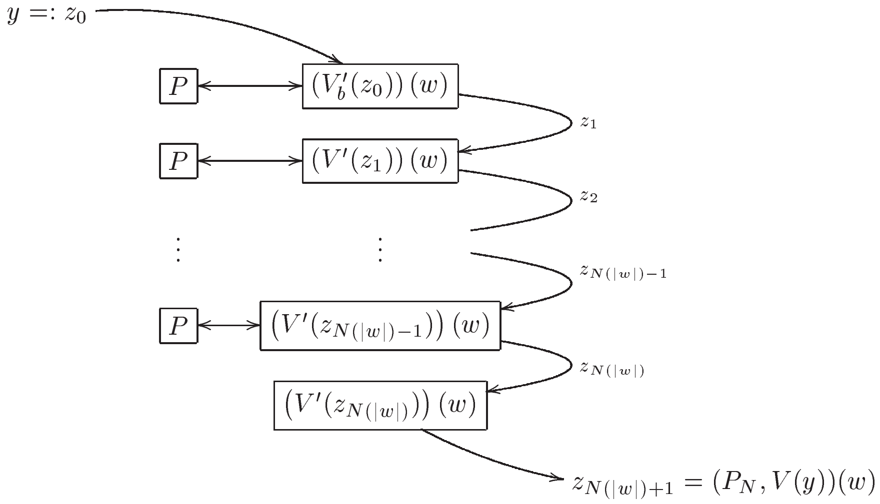


Abbildung 2. Zum Beweis des Sequential Composition Lemmas

Beweis von Satz 6.7. Wir nehmen im folgenden $N(|w|) > 1$ an, um Trivialitäten aus dem Wege zu gehen. Zunächst widmen wir uns der Konstruktion von V' und V'_b. Dazu bauen wir als erstes V zu einem Verifier W um, der als erstes den benötigten Inhalt sämtlicher schreibgeschützter Eingabebänder (w, y, ζ) [7] auf sein Arbeitsband, um diese Informationen bei Bedarf stets von dort zu lesen. Gleiches gelte für den Inhalt der Nachrichten von P_N (wobei die letzte Nachricht vollständig gelesen werden muß). Die Existenz und die erreichbare polynomielle Laufzeit eines solchen W (bzgl. der Laufzeit von V) sind unmittelbar einsichtig.

[7] ζ bezeichne hier wieder die von V verwendeten Zufallsbits

V' V' arbeite nun bei Zusatzeingabe z wie folgt:

(1) V' dekodiert einen Konfiguration von W und dabei insbesondere einen (von V und w abhängigen) polynomiell langen Inhalt des Arbeitsbandes von W
(2) Die Aktionen von W werden ausgehend von der in (1) dekodierten Konfiguration ausgeführt, bis W das Zeichen ■ vom Kommunikationsband liest oder bis W terminiert hätte, je nach dem welcher Zustand eher eintritt.
(3) Falls W terminiert hätte, terminiert auch V'. Andernfalls kodiert V' die "virtuelle" Konfiguration von W auf dem Ausgabeband.

Das Verhalten von (2) ist leicht durch eine eventuelle Umbenennung von Zuständen von W (zur "Einpassung" von W in V') und eine Anpassung der Zustandsübergangsfunktion von W und zu erreichen. V'_b erhält man durch "vorschalten" des Initialisierungsteils von W vor V' und deaktivieren des Teils von V', der Schritt (1) ausführt.

V'_b

Offensichtlich kodiert z_i (vgl. Abb. 2) für $1 \leq i \leq N(|w|)$ den Inhalt des Arbeitsbandes von W nach der i-ten Interaktion mit P, und da W nach Voraussetzung genau $N(|w|)$-mal mit P interagiert terminiert $\left(V'\left(z_{N(|w|)}\right)\right)(w)$, und zwar ohne weiteren Zugriff auf das Kommunikationsband.

Nach Voraussetzung existieren nun Simulatoren M', M'_b für V' bzw. V'_b. Dabei sei daran erinnert, daß es sich bei den z_i um die Zusatzeingaben von V'_b bzw. V' handelt, weshalb die Ausgaben der Simulatoren bei Eingabe z_i im Falle $w \in L$ *fast immer* die korrekte Beschreibung z_{i+1} des Arbeitsbandes und von Zustandsbezeichnungen von W nach der $(i+1)$-ten Interaktion mit P_N sind. Wir gehen deshalb im folgenden davon aus , daß die Ausgaben der Simulatoren immer gültige Konfigurationen von W in dem Sinne sind, daß M' ausgehend von einer solchen Konfiguration nach polynomiell vielen Schritten terminiert. [8]

!

M_V Wir konstruieren $(M_V(y))(w)$ nun analog zur oben beschriebenen Zerlegung von (P_N, V): zuerst führt M_V den Simulator M'_b aus, um danach $N(|w|)$-mal M' auszuführen, jedesmal mit der Ausgabe der vorherigen Iteration als Zusatzeingabe.

$M_V^{(i)}$ Es bleibt zu zeigen, daß die Ausgabe von M_V für $w \in L$ ununterscheidbar von (P_N, V) ist. Dazu führen wir zuerst $(M_V^{(i)}(z))(w)$ durch Angabe des folgenden Algorithmus ein:

Eingabe: $i \in \mathbb{N}$
$z, w \in \Sigma^\star$
 	
Ausgabe: $\omega \in \Sigma^\star$
	`ω := z;`
	`IF i = 0 THEN`
	`    ω := (M'_b(ω))(w);`
	
	`FOR k := max(i,1) TO N(|w|) DO`
	`    ω := (M'(ω))(w);`

Offensichtlich führt $M_V^{(i)}$ stets die letzten $N(|w|) - i + 1$ Iterationen von M_V aus.

H_i Wir setzen außerdem $(H_i(y))(w) = (M_V^{(i)}(z_i))(w)$ und erhalten folgende Identitäten

$$M_V^{(0)} = H_0 = M_V \qquad (M_V^{(N(|w|)+1)}(z))(w) = z \qquad H_{N(|w|)+1} = (P_N, V)$$

[8] Das ist z.B. erfüllt, wenn die Simulatoren nie Ausgaben erzeugen, die nicht auch von V' bzw. V'_b kommen könnten (wenn auch evtl. mit anderer Wahrscheinlichkeit).

H_i ist eine interaktive $\mathcal{P}$-PTM mit Zusatzeingabe, die die ersten i Iterationen mit Hilfe der Verifier V' und V'_b absolviert und für die letzten $N(|w|)-i+1$ Iterationen die nicht-interaktiven Simulatoren verwendet:

$$(H_i(y))(w) : \overset{z_0=y}{\underset{w}{\rightarrow}} \underbrace{V'_b \overset{z_1}{\underset{w}{\rightarrow}} V' \overset{z_2}{\underset{w}{\rightarrow}} V' \overset{z_3}{\underset{w}{\rightarrow}} \ldots \overset{z_{i-1}}{\underset{w}{\rightarrow}} V'}_{i\text{-mal}} \overset{z_i}{\underset{w}{\rightarrow}} \underbrace{M' \overset{z_{i+1}}{\underset{w}{\rightarrow}} M' \overset{z_{i+2}}{\underset{w}{\rightarrow}} \ldots \overset{z_{N(|w|)}}{\underset{w}{\rightarrow}} M'}_{(N(|w|)-i+1)\text{-mal}} \rightarrow \text{Ausgabe}$$

Die Idee ist nun, zu zeigen, daß die Ausgabe der Nachbarn H_i und H_{i+1} für alle $0 \leq i < N(|w|)$ polynomiell ununterscheidbar ist, und somit auch H_0 und $H_{N(|w|)+1}$. Dazu betrachten wir die folgenden Gleichungen ($1 \leq i \leq N(|w|)$):

$$\begin{aligned}
(H_0(y))(w) &\underset{\text{def}}{=} (M_V^{(0)}(z_0))(w) &&= \left(M_V^{(1)}\left((M'_b(z_0))(w)\right)\right)(w) \\
(H_1(y))(w) &\underset{\text{def}}{=} (M_V^{(1)}(z_1))(w) &&= \left(M_V^{(1)}\left((P, V'_b(z_0))(w)\right)\right)(w) \\
(H_i(y))(w) &\underset{\text{def}}{=} (M_V^{(i)}(z_i))(w) &&= \left(M_V^{(i+1)}\left((M'(z_i))(w)\right)\right)(w) \\
(H_{i+1}(y))(w) &\underset{\text{def}}{=} (M_V^{(i+1)}(z_{i+1}))(w) &&= \left(M_V^{(i+1)}\left((P, V'(z_i))(w)\right)\right)(w)
\end{aligned}$$

Da $M'_b(z_0)$ und $(P, V'_b(z_0))$ sowie $M'(z_i)$ und $(P, V'(z_i))$ polynomiell ununterscheidbar sind, gilt dies offenkundig auch für H_i und H_{i+1} $(0 \leq i \leq N(|w|)+1)$, da man sonst mit Hilfe von eines solchen $M_V^{(i)}$ einen Unterscheider A für $M'_b(z_0)$ und $(P, V'_b(z_0))$ oder $M'(z_i)$ und $(P, V'(z_i))$ konstruieren könnte. [9] □

6.4. **Simultane Ausführung von $\mathcal{ZK}$-Protokollen.**

Die strenge Formulierung von Satz 6.7 mag bereits darauf hindeuten, daß die dort getroffene Aussage nicht gilt, wenn ein Verifier mit mehreren Provern interagiert. Das Hauptproblem bei paralleler Komposition besteht darin, daß sich der Prover nicht mehr auf die geringe Rechenkraft des Verifiers verlassen kann - schließlich kann dieser ja mit einem zweiten Prover kommunizieren, der eine Lösung für ein Problem bereithält, mit deren Hilfe es dem Verifier möglich wird, den ersten Prover zu “überlisten”. Es ist also weniger die Gleichzeitigkeit als der Zugriff auf weitere Prover, der die Zero-Knowledge-Eigenschaft bei Parallelausführung zerstört. Anschaulich mag man sich die simultane Ausführung von Zero-Knowledge-Protokollen vielleicht wie ein Kreuzverhör der Prover vorstellen.

Zur Konstruktion eines Beispiels ist es notwendig, den Begriff der $\mathcal{P}$-Evasivität so zu erweitern, daß auch $\mathcal{P}$-PTM mit Zusatzeingabe Schwierigkeiten haben, Elemente aus den S_i zu erraten. Das Problem ist nämlich, daß eine $\mathcal{P}$-PTM mit Zusatzeingabe ein solches Element als Zusatzeingabe erhalten und einfach ausgeben kann. Die Existenz solcher $\mathcal{P}/\text{poly}$-evasiven Folgen wird in [GK94] gezeigt, und wir wollen uns einfach darauf verlassen, da der hier ohnehin schon arg strapazierte Rahmen sonst engültig gesprengt würde. Für die vollständige Erklärung der im angegebenen

[9] Ich muß an dieser Stelle (zu meiner Schande?) gestehen, daß ich die Notwendigkeit des in [Gol95] angeführten Nachtrags bzgl. der Konstruktion von A nicht wirklich verstanden habe. Sollte der hier angegebene Beweis also unvollständig erscheinen, so wird der geneigte Leser dort vielleicht fündig.

Beispiel verwendeten Notation (ich meine damit das $S_j^{(n)}$) sei der Leser auf [GK94] verwiesen, für unsere Zwecke reicht die Vorstellung einer hinreichend starken $\mathcal{P}$-evasiven Folge.

P_1	V	P_2
$i \in_p \{1, ..., 2^{\lvert w \rvert}\} \rightarrow$		
	$j \in_p \{1, ..., 2^{\lvert w \rvert}\} \rightarrow$	$\leftarrow r \in_p S_j^{(\lvert w \rvert)}$
	$\leftarrow s \in_p \{0,1\}^{4\lvert w \rvert}$	
`IF` $s \in S_i^{(\lvert w \rvert)}$ `THEN` $F(w) \rightarrow$		

Es ist klar, daß ein "unehrlicher" Verifier, der $i = j$ und $r = s$ setzt, $w \in L_F$ entscheidet. Desweiteren ist P_2 Zero-Knowledge, da auch $\mathcal{P}/\,$poly-evasive Folgen pseudozufällig sind, weshalb die durch sie hervorgerufenen Verteilungen polynomiell ununterscheidbar von einer gleichverteilten Auswahl sind. Damit ist die Simulation von (P_2, V) trivial und (P_1, V) entspricht im Wesentlichen dem im Beweis zu Satz **6.3** angegebenen Protokoll.

Leider ist bis jetzt keine vernünftige Definition für Zero-Knowldege "aufgetaucht", die die Klasse $\mathcal{ZK}$ so erweitert oder einschränkt, daß sie unter Parallelausführung abgeschlossen ist. Als eine praktische Folge dieser Nicht-Abgeschlossenheit kann man sich eine gewisse Schwäche von Zero-Knowledge-Protokollen für so genannte "Man-In-The-Middle"-Attacken vorstellen, obwohl dafür natürlich ein passender zweiter Prover vorhanden sein muß.

7. "Disclaimer"

Ich empfehle eine kritische Auseinandersetzung mit diesem Papier, da so einiges "selbstgestrickt" ist. Fragen, Hinweise und Korrekturen nehme ich natürlich gerne unter `mailto:karge@informatik.hu-berlin.de` entgegen.

Literatur

[GK93] Oded Goldreich und Hugo Krawczyk, *Sparse pseudorandom distributions*, Random Structures & Algorithms **3** (1993), Nr. 2.

[GK94] Oded Goldreich und Hugo Krawczyk, *On the composition of zero-knowledge proof systems*, `http://www.wisdom.weizmann.ac.il/~oded` (1994(?)).

[Gol95] Oded Goldreich, *Foundations of cryptography (fragments of a book)*, `http://www.wisdom.weizmann.ac.il/~oded` (1995).

[Gol99] Oded Goldreich, *Introduction to complexity theory - lecture notes*, `http://www.wisdom.weizmann.ac.il/~oded` (1999).

[May] Tim May, *Zero knowledge proofs*, `http://www.cpsr.org/cpsr/privacy/crypto/tools/docs/zero-knowledge-proofs.txt`.

[NNR99] Moni Naor, Yael Naor, und Omer Reingold, *Applied kid cryptography or how to convince your children you are not cheating*, `http://www.wisdom.weizmann.ac.il/~naor/onpub.html` (1999).

E-mail address: `karge@informatik.hu-berlin.de`