Institute of Information Security

University of Stuttgart
Universitätsstr. 38
D–70569 Stuttgart

Masterarbeit

Nicht-interaktive Zero-Knowledge Beweise von Wissen mittels Fiat-Shamir Transformation

Julian Liedtke

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Ralf Küsters
Betreuer/in:	Prof. Dr. Ralf Küsters
Beginn am:	October 15, 2017
Beendet am:	March 15, 2018
CR-Nummer:	I.7.2

Kurzfassung

Σ-Protokolle sind sehr effiziente Beweise von Wissen. Leider weisen sie nur die Special Honest Verifier Zero-Knowledge Eigenschaft auf, welche schwächer als die Zero-Knowledge Eigenschaft ist [HL10]. Das liegt daran, dass bei Honest Verifier Zero-Knowledge nur für ehrliche Verifizierer ein Simulator existieren muss, während bei Zero-Knowledge auch für bösartige Verifizierer, das sind Verifizierer, die sich möglicherweise nicht an das Protokoll halten, eine erfolgreiche Simulationen verlangt werden.

Eine Möglichkeit, Σ-Protokolle in Zero-Knowledge Protokolle umzuwandeln, besteht in der Fiat-Shamir Transformation. Dabei entsteht nicht nur ein Zero-Knowledge Beweis von Wissen, sondern auch ein nicht-interaktives Beweissystem.

Die Idee der Fiat-Shamir Transformation besteht darin, dass der Beweiser die Challenge mittels einer Hashfunktion aus dem gemeinsamen Eingabewort und dem Commitment berechnet.

Trotz der aktiven Verwendung der Fiat-Shamir Transformation in der Praxis wurde erst 2012 in Arbeiten von Bernhard, Pereira und Warinschi [WS12] sowie Faust, Kohlweiss, Marson, und Venturi [GN12] der Versuch eines Beweises der Korrektheit erbracht. Der Beweis der ersten Arbeit wird in dieser Masterarbeit ausformuliert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Zero-Knowledge Beweise sind Protokolle für zwei Parteien, einem Beweiser und einem Verifizierer. Sie ermöglichen es dem Beweiser, dem Verifizierer eine Aussage zu beweisen, ohne dass dieser etwas neben dem Wahrheitsgehalt der Aussage lernt.

Ein anschauliches Beispiel für ein Zero-Knowledge Beweis findet sich in [Qui+89]. In diesem Beispiel gibt es eine Höhle, dessen Weg sich nach einiger Zeit in zwei aufteilt. Diese beiden Wege sind über eine Tür miteinander verbunden, die nur mit einem geheimen Wort geöffnet werden kann.

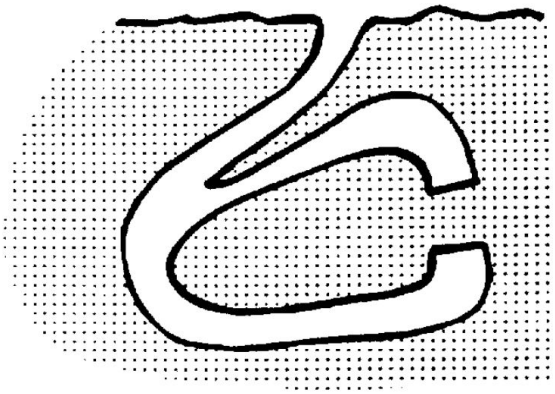

Abbildung 1.1: Die Höhle [Qui+89]

Eine Person, genannt Alice, möchte einer weiteren Person, Bob, beweisen, dass sie das geheime Wort kennt, ohne dieses zu verraten.

Dafür stellen sich beide Parteien zunächst an den Eingang der Höhle. Dann betritt Alice diese und folgt einem der beiden Pfade - welchen bleibt ihr überlassen. Bob kann vom Eingang aus nicht erkennen, welchen Pfad Alice gewählt hat.

Anschließend betritt Bob die Höhle und läuft bis zur Gabelung. Er entscheidet sich nun für einen der beiden Pfade und ruft diesen in die Höhle. Alices Aufgabe ist nun durch diesen Pfad zu Bob zu gelangen. Schafft sie dies, ist sie dem Ziel, Bob zu beweisen, dass sie das geheime Wort kennt, einen Schritt näher gekommen.

Hat Alice den Pfad gewählt den auch Bob gewählt hat, benötigt sie das geheime Wort nicht. Nur wenn sie sich für den anderen Pfad entschieden hat, muss sie durch die Tür gelangen,

um auf den richtigen Pfad zu gelangen. Das bedeutet, dass eine bösartige Alice, die nicht in Kenntnis des geheimen Wortes ist, nur zu 50% in einem Durchlauf dieses Protokolls Bob überzeugen kann.

Dieses Vorgehen ist ein Zero-Knowledge Beweis von Wissen. Denn egal wie oft Alice und Bob den Ablauf wiederholen, Alice wird, sofern sie das geheime Wort kennt, stets den von Bob gewählten Pfad einschlagen können. Bob erfährt währenddessen nichts über das Geheimnis.

Dieses Protokoll weist eine wichtige Eigenschaft auf: Nur Bob wird überzeugt, dass Alice das Geheimnis kennt. Wenn ein Beobachter den Ablauf beobachtet, wird er nicht überzeugt sein, dass Alice tatsächlich das Geheimnis kennt. Das liegt daran, dass sich Alice und Bob im Voraus abgesprochen haben können, welchen Pfad Bob wählen wird. Mit diesem Wissen kann Alice stets den von Bob gewählten Pfad wählen ohne Kenntnis des Geheimnisses.

Das bedeutet: Um eine Partei zu überzeugen, muss man den Beweis mit dieser als Verifizierer ausführen.

Eine weitere, wichtige Eigenschaft von Zero-Knowledge Beweisen ist ihre Komplexität. Je mehr Interaktion zwischen den Parteien nötig ist, umso teurer ist die Ausführung. Besonders effiziente Beweise sind Σ-Protokolle. Diese bestehen aus drei Schritten. Zunächst sendet der Beweiser (Alice) ein Commitment an den Verifizierer (Bob). Im obigen Beispiel entspricht das der anfänglichen Wahl des Pfades von Alice. Anschließend schickt der Verifizierer eine Challenge an den Beweiser. Das ist die Wahl des Pfades von Bob. Zuletzt antwortet der Beweiser mit einer Response. Das entspricht der Rückkehr von Alice zur Gabelung. Daraufhin entscheidet der Verifizierer, ob er den erbrachten Beweis akzeptiert.

Das obige Beispiel entspricht folglich dem Ablauf eines Σ-Protokolls. Ein Nachteil der Σ-Protokolle besteht jedoch darin, dass sie nur gegenüber ehrlichen Verifizierern die Zero-Knowledge Eigenschaft aufweisen. Das bedeutet, wählt der Verifizierer die Challenge nicht wie vom Protokoll vorgegeben (in der Regel wird die Challenge zufällig gleichverteilt bestimmt), sondern auf eine andere Weise, kann er unter Umständen etwas über das Geheimnis lernen.

Auf der anderen Seite sind Σ-Protokolle Beweise von Wissen. Das bedeutet, diese geben dem Beweiser die Möglichkeit zu beweisen, dass er eine bestimmte Information besitzt, ohne diese selbst zu verraten.

Zero-Knowledge Beweise werden unter Anderem bei elektronischen Wahlen eingesetzt. Ein Server, die alle abgegebenen Stimmen zusammenzählt und das Ergebnis bekannt gibt, muss verifizieren, dass er das korrekte Ergebnis berechnet hat, ohne dass er die Stimmen der Wähler bekannt gibt.

Damit davon jeder Wähler überzeugt wird, muss der Server ein Zero-Knowledge Beweis mit jedem Wähler einzeln ausführen. Je nach Anzahl der Stimmberechtigten kann dies eine große Menge an nötigen Beweisen sein.

Um die Anzahl an nötigen Beweisen zu optimieren können sogenannte nicht-interaktive Zero-Knowledge Beweise eingesetzt werden. Diese ermöglichen es, dem Beweiser ohne Interaktion

mit einem Verifizierer, einen Beweis für eine Aussage zu erzeugen. Der Beweis kann von jedem, der Aussage des Beweiser verifizieren möchte, überprüft werden.

Ein Verfahren, um Σ-Protokolle in Zero-Knowledge Beweise zu überführen ist die Fiat-Shamir Transformation. Diese erzeugt nicht nur Zero-Knowledge Beweise, sondern wandelt das Beweissystem gleichzeitig in einen nicht-interaktiven Beweis von Wissen um.

Diese Arbeit stellt die Fiat-Shamir Transformation vor und beweist, dass die Konstruktion korrekt ist.

Diese Arbeit ist wie folgt aufgebaut:

Kapitel 2 – Grundlagen beinhaltet grundlegende Definitionen, gibt einen Einblick in Beweissysteme und stellt die Fiat-Shamir Transformation vor.

Kapitel 3 – Aktueller Forschungsstand beschreibt den aktuellen Forschungsstand beim Beweis der Korrektheit der Fiat-Shamir Transformation.

Kapitel 4 – Eigenschaften der Strong Fiat-Shamir Transformation zeigt die Lücken im Beweis des vorherigen Kapitels auf und schließt diese.

Kapitel 5 – Zusammenfassung und Ausblick: fasst die Arbeit zusammen und gibt einen Ausblick.

2 Grundlagen

Dieses Kapitel legt den Grundstein für die Arbeit. Zunächst werden die für Beweissysteme benötigte Komponenten wie Algorithmen, Sprachen und Random Oracles in Abschnitt 2.1 vorgestellt. In Abschnitt 2.2 wird auf nicht-interaktive Beweissysteme eingegangen. Diese zeichnen sich dadurch aus, dass keine interaktive Kommunikation zwischen dem Beweiser und dem Verifizierer, stattfindet. Ausgehend von dieser Art von Beweissystemen werden die Eigenschaften der Systeme definiert. Abschnitt 2.3 behandelt Σ-Protokolle. Diese sind der Ausgangspunkt in der Umwandlung mittels der Fiat-Shamir Transformation. Ein Beispiel für ein Σ-Protokoll findet sich in Abschnitt 2.4. Auf die Fiat-Shamir Transformation wird in Abschnitt 2.5 eingegangen.

2.1 Definitionen

Dieser Abschnitt definiert grundlegende Komponenten für Beweissysteme. In Abschnitt 2.1.1 werden Algorithmen und ihre Laufzeiten definiert. Anschließend werden in Abschnitt 2.1.2 spezielle Arten von Funktionen vorgestellt, die verwendet werden, um die Sicherheitseigenschaften der Beweissysteme zu definieren.

Die Sprachen, die von Beweissystemen verwendet werden können, sind in Abschnitt 2.1.3 vorgestellt. In Abschnitt 2.1.4 werden Random Oracles vorgestellt, welche ein wichtiger Bestandteil nicht-interaktiver Beweissysteme sind. Schlussendlich wird ein Forking Lemma in Abschnitt 2.1.5 vorgestellt. Dieses wird im Beweis der Eigenschaften der Fiat-Shamir Transformation eingesetzt.

2.1.1 Algorithmen

Dieser Abschnitt stellt Algorithmen und deren Eigenschaften, wie zum Beispiel ihre Laufzeit, vor.

Definition 2.1.1 (Randomisierter Algorithmus)**.** *Ein randomisierter Algorithmus ist ein Algorithmus, der Zugriff auf ein Band hat, das mit zufälligen Bits beschrieben ist. Für eine Ausführung werden diese Bits zufällig gleichverteilt gewählt.*

Bemerkung. Ein Bit von diesem Band zu lesen benötigt einen Schritt. Ein Algorithmus, der eine Laufzeit von t hat, kann folglich maximal t Bits von diesem Band lesen.

Definition 2.1.2 (Elementauswahl). *Mit $s \xleftarrow{\$} S$ wird eine Operation bezeichnet, bei der ein Element s zufällig gleichverteilt aus S gezogen wird. Wenn $A(x \in \mathcal{X}) : \mathcal{Y}$ ein randomisierter Algorithmus ist, bezeichnet $A(x_1, ...; \rho)$ die Ausgabe auf Eingabe $x_1, ...$ mit Random Bits ρ. Weiterhin bedeutet $y \xleftarrow{\$} A(x_1, ...)$, dass ρ gleichverteilt gewählt und $y = A(x_1, ...; \rho)$ gesetzt wird.*

Definition 2.1.3 (Laufzeit). *Sei $A(x \in \mathcal{X}) : \mathcal{Y}$ ein probabilistischer Algorithmus. Die Funktion $t_A(x; \alpha) : \mathcal{X} \times \{0,1\}^* \to \mathbb{N} \cup \{\infty\}$ bezeichne die Anzahl an Schritten, die A auf Eingabe $x \in \mathcal{X}$ mit Random Bits α läuft. Wenn $A(x; \alpha)$ nicht terminiert, ist $t_A(x; \alpha) := \infty$.*

Bemerkung. Ist aus dem Kontext klar, um welchen Algorithmus es sich handelt, wird statt $t_A(x; \alpha)$ auch $t(x; \alpha)$ geschrieben.

Definition 2.1.4 (Wahrscheinlichkeitsraum für randomisierte Algorithmen). *Sei $A(x \in \mathcal{X}) : \mathcal{Y}$ ein Algorithmus und sei $x \in \mathcal{X}$. Sei $\Omega = \{0,1\}^\omega$ die Menge aller unendlichen Bitstrings. Für einen endlichen Bitstring $\beta \in \{0,1\}^*$ sei $\bar{\beta} = \{\alpha \in \Omega | \beta$ ist Prefix von $\alpha\}$ die Menge aller unendlichen Bitstrings, die mit β beginnen.*

Sei $e = \{\bar{\beta} | \beta \in \{0,1\}^*\}$ und $A = \sigma(e)$ die σ-Algebra über Ω, die von e erzeugt wird.

Dann ist (Ω, A, P) der Wahrscheinlichkeitsraum über Ω mit

$$P(\bar{\beta}) = \frac{1}{2^{|\beta|}}$$

Definition 2.1.5 (Zufallsvariable für die Laufzeit). *Sei $A(x \in \mathcal{X}) : \mathcal{Y}$ ein Algorithmus, sei $x \in \mathcal{X}$ und der Wahrscheinlichkeitsraum (Ω, A, P) gemäß Definition 2.1.4 gegeben. Dann ist $t(x) : \Omega \to \mathbb{N} \cup \{\infty\}, \alpha \mapsto t(x, \alpha)$ eine Zufallsvariable, die die Laufzeit von A auf Eingabe x beschreibt.*

Definition 2.1.6 (Polynomialzeitbeschränkter Algorithmus). *Ein Algorithmus $A(x \in \mathcal{X}) : \mathcal{Y}$ ist polynomialzeitbeschränkt, falls ein Polynom p existiert, sodass*

$$\exists n_0 \forall x \in \mathcal{X} \text{ mit } |x| > n \forall \alpha : t(x, \alpha) \leq p(|x|)$$

gilt.

Definition 2.1.7 (Schwach polynomialzeitbeschränkter Algorithmus). *Ein Algorithmus $A(x \in \mathcal{X}) : \mathcal{Y}$ ist schwach polynomialzeitbeschränkt, falls Polynome p, q existieren, sodass*

$$\exists n_0 \forall x \in \mathcal{X} \text{ mit } |x| > n : Pr[t(x) > p(|x|)] < \frac{1}{q(|x|)}$$

gilt.

Bemerkung. Mit anderen Worten ist die Wahrscheinlichkeit, dass die Laufzeit von A nicht durch p beschränkt ist, vernachlässigbar.

2.1.2 Vernachlässigbare und überwältigende Funktionen

In diesem Abschnitt werden vernachlässigbare und überwältigende Funktionen definiert, welche in den Definitionen der Sicherheitseigenschaften der Beweissysteme Verwendung finden.

Definition 2.1.8 (Vernachlässigbare Funktion)**.** *Eine Funktion $v : \mathbb{N} \to \mathbb{R}_{\geq 0}$ ist vernachlässigbar (engl. negligible), falls*

$$\forall p \, \exists n_0 \in \mathbb{N} \, \forall n > n_0 : v(n) < \frac{1}{p(n)}$$

für p als positives Polynom gilt.

Ist der Wertebereich von v eine Sprache L, wird die Länge der jeweiligen Worte verwendet.

Definition 2.1.9 (Vernachlässigbare Funktion für Sprachen)**.** *Sei L eine Sprache. Eine Funktion $v : L \to \mathbb{R}_{\geq 0}$ ist vernachlässigbar, falls*

$$\forall p \, \exists n_0 \in \mathbb{N} \, \forall w \in L \textit{ mit } |w| > n_0 : v(|w|) < \frac{1}{p(|w|)}$$

für p als positives Polynom gilt.

Definition 2.1.10 (Überwältigende Funktion)**.** *Eine Funktion $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ ist überwältigend (engl. overwhelming), falls die Funktion $1 - f(n)$ vernachlässigbar ist.*

Definition 2.1.11 (Superpolynomielle Funktion)**.** *Eine Funktion $e : \mathbb{N} \to \mathbb{R}_{\geq 0}$ ist superpolynomiell, falls*

$$\forall p \, \exists n_0 \in \mathbb{N} \, \forall n > n_0 : e(n) > p(n)$$

für p als positives Polynom gilt.

2.1.3 Relationen und dazugehörige Sprachen

Im Folgenden werden Relationen und Sprachen vorgestellt, die in Beweissystemen eingesetzt werden können.

Definition 2.1.12 (Effizient entscheidbare Relation)**.** *Eine Relation $R \in \mathcal{P}(\{0,1\}^* \times \{0,1\}^*)$ ist effizient entscheidbar, wenn ein deterministischer Algorithmus existiert, der in polynomieller Zeit auf Eingabe x und y entscheidet, ob $(x, y) \in R$ gilt.*

Bemerkung. Endliche Relationen, d.h. Relationen die endlich viele Tupel enthalten, sind effizient entscheidbar. Das liegt daran, dass endliche Relationen eine endliche Sprache beschreiben und diese regulär sind.

Beispiel 2.1.1 (R_0)**.** *$R_0 = \{(0,0)\}$ ist effizient entscheidbar.*

Neben effizient entscheidbaren Relationen gibt es auch welche, die eine Sprache in NP beschreiben.

Definition 2.1.13 (NP-Relation)**.** *Eine Relation $R \in \mathcal{P}(\{0,1\}^* \times \{0,1\}^*)$ ist eine NP-Relation, wenn ein Polynom p und ein deterministischer Algorithmus existiert, der in polynomieller Zeit auf Eingabe x und y entscheidet, ob $(x, y) \in R$ gilt. Zusätzlich ist y nicht zu groß: $|y| \leq p(|x|)$.*

Bemerkung. Jede effizient entscheidbare Relation ist eine NP-Relation.

Beispiel 2.1.2 (R_0)**.** *R_0 (siehe Beispiel 2.1.1) ist eine NP-Relation.*

Definition 2.1.14 (Sprache zu einer Relation)**.** *Sei R eine effizient entscheidbare Relation und sei p ein Polynom. Dann ist*

$$L_R = \{y \mid \exists w : |w| \leq p(|y|) \wedge (y, w) \in R\}.$$

Bemerkung. Das zweite Element des Tupels, kann als *Zeuge* (engl. *witness*) von y angesehen werden.

Beispiel 2.1.3 (L_{R_0})**.** *Für die Relation R_0 und das Polynom $p(x) = 42$ ist $L_{R_0} = \{0\}$.*

Beispiel 2.1.4. *Für ein geeignetes Polynom p ist zu der Relation R der Graph 3-Färbbarkeit $L_R = \{G | G$ ist 3-Färbbar$\}$.*

Für eine Sprache L ist es für einige x in der Regel trivial zu entscheiden, ob $x \in L$ gilt. Beispielsweise kann für die Sprache *Graph 3-Färbbarkeit* trivialerweise jedes x entschieden werden, das keinen Graph kodiert.

Deshalb wird eine in Polynomialzeit entscheidbare Obermenge $\Lambda \supseteq L$ betrachtet, die alle „nicht-trivialen“ Elemente enthält.

Definition 2.1.15 (Obermenge Λ zu einer Sprache L)**.** *Sei L eine Sprache und $\Lambda \supseteq L$ effizient entscheidbar. Dann ist Λ effizient entscheidbare Obermenge von L.*

2.1.4 Random Oracle Model

Damit nicht-interaktive Beweissysteme für nicht-triviale Sprachen eingesetzt werden können, müssen zusätzliche Voraussetzungen erfüllt sein. Eine mögliche Voraussetzung kann ein gemeinsames Random Oracle sein.

Im Grund genommen ist ein Random Oracle eine perfekte Hashfunktion. Das bedeutet, der Beweiser kann einen Hashwert zu einer Eingabe anfordern, und der Verifizierer kann die Korrektheit überprüfen, indem er den Hashwert zu dieser Eingabe ebenfalls mittels des Random Oracles berechnet.

Perfekte Hashfunktion bedeutet, dass die Hashwerte zufällig gleichverteilt aus der Bildmenge des Random Oralces gewählt werden.

Im Random Oracle Model wird eine Hashfunktion $\mathcal{X} \to \mathcal{Y}$ durch ein Random Oracle modelliert.

Definition 2.1.16 (Random Oracle)**.** *Ein Random Oracle ist ein Algorithmus $\mathcal{H}(\mathcal{X}) : \mathcal{Y}$, der sich wie folgt verhält:*

- *Jeder kann eine Anfrage $x \in \mathcal{X}$ an das Orakel stellen*
- *Das Orakel hält eine Menge M aus Anfragen mit zugehörigen Antworten, um konsistent antworten zu können.*
- *Eine Anfrage $x \in \mathcal{X}$ wird vom Orakel wie folgt beantwortet:*
  - *Wenn die Menge M ein Paar (x, y) enthält, so wird y ausgegeben.*
  - *Wenn dies nicht der Fall ist, bestimmt das Orakel einen Wert y aus $\mathcal{Y}$ zufällig gleichverteilt. Anschließend fügt das Orakel das Paar (x, y) der Menge M hinzu und gibt y aus. Das bedeutet, jede weitere Anfrage von x wird mit y beantwortet.*

Für die später vorgestellten Beweise ist ein simuliertes Orakel hilfreich. Diesem werden die ersten n Antworten vorgegeben. Das bedeutet, diese ersten n verschiedenen Anfragen werden nicht durch Würfeln, sondern mittels den vorgegeben Werten, beantwortet. Dies ermöglicht es, eine Ausführung eines Protokolls zu wiederholen, indem das Random Oracle die gleichen Ergebnisse liefert.

Definition 2.1.17 (Simuliertes Orakel)**.** *Ein simuliertes Orakel ist ein Algorithmus $\mathcal{H}_{y_1,\ldots,y_n}(\mathcal{X}) : \mathcal{Y}$, der n Eingaben $y_1, \ldots, y_n \in \mathcal{Y}$ erhält. Ein simuliertes Orakel verhält sich in allen Belangen wie ein Random Oracle, bis auf die ersten n voneinander verschiedenen Anfragen. Bei diesen wählt das simulierte Orakel den nächsten, noch nicht verwendeten, Wert aus der Eingabemenge und verwendet diesen als Antwort.*

2.1.5 Forking Lemma

Das in diesem Abschnitt vorgestellte Forking Lemma stammt von M. Bellare und G. Neven [BN06]. Es wird im Beweis der Sicherheitseigenschaften der Fiat-Shamir Transformation verwendet.

Lemma 2.1 (Forking Lemma). Sei $q \in \mathbb{N} \geq 1$ und sei H eine Menge der Mächtigkeit $h := |H| \geq 2$. Sei A ein randomisierter Algorithmus, welcher auf den Eingaben $x, h_1 \in H, ..., h_q \in H$ ein Paar (j, σ) mit $j \in \mathbb{N}$ ausgibt. Die Ausgabe j wird verwendet um anzeigen, ob A die Eingabe akzeptiert hat. Mit σ wird eine sogenannte *Nebenausgabe* bezeichnet.

Sei IG ein probabilistischer Algorithmus, der ein x als mögliche Eingabe für A ausgibt. Dieser Algorithmus IG wird als Input Generator bezeichnet.

Sei *acc* die Wahrscheinlichkeit, dass A im folgenden Experiment akzeptiert ($j \geq 1$):

$$x \xleftarrow{\$} \text{IG}; h_1, ..., h_q \xleftarrow{\$} H; (j, \sigma) \xleftarrow{\$} A(x, h_1, ..., h_q)$$

Der zu A gehörende Forking Algorithmus F_A ist der randomisierte Algorithmus, der x als Input erhält und wie in Algorithmus 2.1 beschrieben ist.

Algorithmus 2.1 Forking Algorithmus $F_A(x)$

```
1  Pick coins ρ for A uniform at random
2  h_1, ..., h_q ←$ H
3  (j, σ) ← A(x, h_1, ..., h_q; ρ)
4  if j = 0 then
5  |  return (0, ϵ, ϵ)
6  end
7  h'_j, ..., h'_q ←$ H
8  (j', σ') ← A(x, h_1, ..., h_{j-1}, h'_j, ..., h'_q; ρ)
9  if (j = j' and h_j ≠ h'_j) then
10 |  return (1, σ, σ')
11 else
12 |  return (0, ϵ, ϵ)
13 end
```

Sei

$$frk = Pr[b = 1 | x \xleftarrow{\$} \text{IG}; (b, \sigma, \sigma') \xleftarrow{\$} F_A(x)]$$

Dann ist

$$frk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{h} \right)$$

zu zeigen

2.2 Nicht-interaktive Beweissysteme

Nicht-interaktive Beweissysteme zeichnen sich dadurch aus, dass keine interaktive Kommunikation zwischem dem Beweiser und dem Verifizierer stattfindet. Das bedeutet, der Beweiser gibt lediglich einen Beweis für das zu zeigende Wort aus und sendet dieses an den Verfizierer. Es findet kein weiterer Nachrichtenaustausch zwischen den beiden Parteien statt.

Dieser Abschnitt gibt zunächst eine Definition für nicht-interaktive Beweissysteme an und gibt anschließend die Sicherheitseigenschaften dieser Systeme an.

Die Definitionen in diesem Kapitel stammen aus [BPW12], wenn nicht anders genannt. Erweiterungen sind in **blau** hervorgehoben.

Definition 2.2.1 (Nicht-interaktives Beweissystem)**.** *Ein nicht-interaktives Beweissystem für eine Sprache $L_R \subseteq \{0,1\}^*$ ist ein Paar (P, V) aus zwei **probabilistischen polynomialzeit-beschränkten [Cor+17]** Algorithmen, Beweiser P und Verifizierer V, das die Eigenschaften Vollständigkeit (siehe Definition 2.2.2) und Korrektheit (siehe Definition 2.2.3) erfüllt. Die Algorithmen verhalten sich wie folgt:*

- *$P(y, w)$ erhält als Input ein Wort y und einen Zeugen w mit $(y, w) \in R$ und gibt einen Beweis π aus.*
- *$V(y, \pi)$ erhält das Wort y und einen Beweis π. Der Verifizierer entscheidet, ob er den erbrachten Beweis akzeptiert oder nicht, indem er 0 oder 1 ausgibt.*

Bemerkung. Nicht-interaktiv bedeutet, dass der Verifizierer den Beweis vom Beweiser ohne vorherige Interaktion der Algorithmen erhält.

Ein Beweissystem (P, V) für die Sprache L_R ist vollständig (engl. *complete*), falls der Beweiser für die gemeinsame Eingabe $y \in L$ stets einen Beweis ausgeben kann, der von V akzeptiert wird.

Definition 2.2.2 (Vollständigkeit eines Beweissystems). *Sei (P, V) ein Beweissystem. Das Beweissystem ist vollständig, falls*

$$\forall (y, w) \in R : Pr[V(y, P(y, w)) = 1] = 1$$

gilt.

Definition 2.2.3 (Korrektheit eines Beweissystems). *Sei (P, V) ein Beweissystem und B ein polynomialzeitbeschränkter Algorithmus. Das Beweissystem ist korrekt, falls*

$$\forall y \notin L_R \exists \text{ vernachlässigbare Funktion } v : Pr[V(1^\eta, y, B(1^\eta, y)) = 1] \leq v(\eta)$$

gilt.

Bemerkung. Der Beweiser B unterliegt keinen Einschränkungen.

Nach der Vorlesung Vorlesung Security & Privacy, Sommersemester 2016/17, existieren nicht-interaktive Zero-Knowledge Beweissysteme nur für *triviale* Sprachen. Um nicht-interaktive Zero-Knowledge Beweissysteme auch für *nicht-triviale* Sprachen zu erhalten, sind weitere Annahmen nötig. Eine Möglichkeit besteht darin, dass Beweiser und Verifizierer Zugriff auf ein Random Oracle haben.

Definition 2.2.4 (Nicht-interaktives Beweissystem mit Orakel). *Sei $\mathcal{H}(\mathcal{X}) : \mathcal{Y}$ ein Orakel. Dann ist $(P^{\mathcal{H}}, V^{\mathcal{H}})$ ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$, wenn die Eigenschaften eines nicht-interaktiven Beweissystems erfüllt sind. Zudem haben sowohl der Beweiser, wie auch der Verifizierer, Zugriff auf das Orakel $\mathcal{H}$.*

Bemerkung. Da sowohl der Beweiser als auch der Verifizierer Zugriff auf das Orakel haben, kann der Verifizierer den Beweiser überprüfen. Wenn zum Beispiel der Beweiser in einem Beweis behauptet, dass er auf eine Anfrage $\mathcal{H}(x)$ die Antwort y erhalten hat, so kann dies vom Verifizierer durch eine Anfrage $\mathcal{H}(x)$ verifiziert werden.

Definition 2.2.5 (Simulator im Random Oracle Model). *Ein Simulator S im Random Oracle Model mit einer Hashfunktion $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ ist ein **probabilistischer, polynomialzeitbeschränkter** Algorithmus, der zwei Methoden besitzt:*

- *$\mathcal{H}(s)$. Eine Hashanfrage behandelt S wie ein Random Oracle. Das heißt, S speichert eine Menge M aus Frage/Antwort Paaren. Für wiederholte Fragen $q \in X$ antwortet S konsistent, d.h. es wird die passende, gespeicherte Antwort r mit $(q, r) \in M$ ausgegeben. Für neue Fragen $q' \in X$ bestimmt S eine Antwort r' gleichverteilt aus Y, fügt (q', r') zu M hinzu und gibt r' zurück.*

- ***Simulate***$(y \in \Lambda)$. *Der Simulator gibt einen Beweis* π *aus, sodass* $V(y, \pi) = 1$ *gilt, sofern der Verifizierer den Simulator für seine Anfragen an das Random Oracle nutzt. Der Simulator kann während einer* ***Simulate*** *Anfrage Frage/Antwort Paare zu seiner Orakelmenge* M *hinzufügen.*

Für die Zero-Knowledge Eigenschaft werden verschiedene Szenarien betrachtet. Diese Szenarien werden im Folgenden definiert.

Definition 2.2.6 (Szenario)**.** *Sei* $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ *ein Random Oracle. Sei* $(P^{\mathcal{H}}, V^{\mathcal{H}})$ *ein nicht-interaktive Beweissystem für die NP-Relation* $R \subseteq \Lambda$*. Ein Szenario ist ein Algorithmus, der folgenden Schnittstellen besitzt:*

- $\mathcal{H}(x)$*, eine Hashanfrage.*
- *Prove*(y, w)*, eine Anfrage für einen Beweis* π *von* y*.*

Definition 2.2.7 (Szenario 1, echter Durchlauf)**.** *Sei* $\mathcal{H}$ *ein Random Oracle. Sei* $(P^{\mathcal{H}}, V^{\mathcal{H}})$ *ein nicht-interaktives Beweissystem für die NP-Relation* $R \subseteq \Lambda$*. In Szenario 1 werden die beiden Methoden wie folgt implementiert.*

- $\mathcal{H}(x)$ *wird an das Random Oracle weitergeleitet und dessen Ausgabe wird zurückgegeben.*
- *Für Prove*(y, w) *wird zunächst geprüft, ob* $(y, w) \in R$ *gilt. Wenn ja, wird die Ausgabe von* $P^{\mathcal{H}}(y, w)$ *ausgegeben. Wenn* $(y, w) \notin R$ *gilt, wird* $\perp$ *ausgegeben.*

Definition 2.2.8 (Szenario 2, Simulation)**.** *Sei* $\mathcal{H}$ *ein Random Oracle. Sei* $(P, V) = sFS(\Sigma)$ *ein Beweissystem für die NP-Relation* $R \subseteq \Lambda$*.*

Sei $S^{\mathcal{H}}$ *ein Simulator gemäß Definition 2.2.5. Für Szenario 2 werden die beiden Methoden wie folgt implementiert.*

- $\mathcal{H}(x)$ *wird an* S *weitergeleitet.*
- *Für Prove*(y, w) *wird zunächst geprüft, ob* $(y, w) \in R$ *gilt. Wenn ja, wird* ***Simulate***(y) *von* S *aufgerufen und dessen Ausgabe ausgegeben, sonst wird* $\perp$ *ausgegeben.*

Bemerkung. Während in Szenario 1 der Beweiser den Zeugen erhält, bekommt der Simulator in Szenario 2 lediglich das zu beweisende Wort.

Definition 2.2.9 (Unterscheider)**.** *Sei* $U(S, 1^\eta) : \{1, 2\}$ *ein Algorithmus, der als Eingabe ein Szenario und einen Sicherheitsparameter erhält. Dann ist* U *ein Unterscheider.*

Bemerkung. Die Notation $U(S1, 1^\eta)$ bzw. $U(S2, 1^\eta)$ bedeutet, dass U mit Szenario 1 bzw. Szenario 2 kommuniziert und 1^η der Sicherheitsparameter ist.

Definition 2.2.10 (Zero-Knowledge). *Sei (P, V) ein Beweissystem.*

Das Beweissystem ist Zero-Knowledge, falls ein Simulator S existiert, sodass kein Unterscheider U mit polynomialzeitbeschränkter Laufzeit Szenario 1 von Szenario 2 mit nicht-vernachlässigbarer Wahrscheinlichkeit unterscheiden kann. D.h. es gilt für alle polynomialzeitbeschränkten U:

$$\exists \textit{ vernachlässigbare Funktion } v : |Pr[U(S1, 1^\eta) = 1] - Pr[U(S2, 1^\eta) = 1]| \leq v(\eta)$$

Im Folgenden wird ein bösartiger Beweiser betrachtet. Dieser versucht Beweise für Wörter $y \in \Lambda \setminus L$ zu erzeugen. Dafür kann er simulierte Beweise einsehen.

Definition 2.2.11 (Bösartiger Beweiser). *Sei (P, V) ein Beweissystem. Ein bösartiger Beweiser ist ein Algorithmus $\mathcal{A}^{\mathcal{H},\textbf{Simulate}}(1^\eta) : (y_i, \pi_i)_{i\in\{1,\dots,n\}}$, welcher zwei Algorithmen erwartet:*

1. *$\mathcal{H}$ - ein Hashorakel: $\mathcal{X} \to \mathcal{Y}$.*
2. ***Simulate** - ein Simulationsorakel. $\mathcal{A}$ kann für $y \in \Lambda$ mittels **Simulate**(y) einen Beweis π anfordern. Für π gilt $V(y, \pi) = 1$.*

Der bösartige Beweiser $\mathcal{A}^{\mathcal{H},\textbf{Simulate}}$ gibt ein Vektor von Paaren $(y_i, \pi_i)_{i\in\{1,\dots,n\}}$ aus, wobei π_i der zu y_i gehörige erzeugte Beweis ist.

Definition 2.2.12 (Simulation Sound Extractability). *Sei $(P^\mathcal{H}, V^\mathcal{H})$ ein Zero-Knowledge Beweissystem. Sei S der Simulator.*

Sei weiterhin $\mathcal{A}^{\mathcal{H},S}(1^\eta)$ ein polynomialzeitbeschränkter Angreifer, der ein Hashorakel $\mathcal{H}$ und einen Simulator S erwartet.

Im Folgenden wird das Sicherheitsspiel für Simulation Sound Extractability angegeben. Das Spiel ist in Algorithmus 2.2 dargestellt.

1. ***Initial Run.** Es wird eine Instanz von $\mathcal{A}^{\mathcal{H},S}$ gestartet, wobei die Random Bits zufällig gleichverteilt gewählt werden. Die gestartete Instanz gibt einen Vektor $(y_i, \pi_i)_{i\in\{1,\dots,n\}}$ an Paaren bestehend aus Wort und Beweis aus.*

   In den folgenden Fällen gewinnt K das Spiel sofort:

   - *Der Vektor ist leer, d.h. es wurden keine Beweise ausgegeben.*
   - *Für ein Paar (y, π) des Vektors ist π die Ausgabe einer **Simulate**(y) Anfrage an den Simulator.*
   - *Ein Paar (y, π) des Vektors wird nicht von $V^\mathcal{H}$ verifiziert.*

2. ***Extraction***. *Das Spiel startet eine Instanz vom Extraktor K. Die Laufzeit des Extraktors hängt polynomiell von der Laufzeit des Beweisers ab. Dieser erhält das Transkript aller Anfragen[1] von $\mathcal{A}^{\mathcal{H},S}$ aus dem initalen Run und den Ausgabevektor (y_i, π_i) von $\mathcal{A}^{\mathcal{H},S}$. Der Extraktor darf beliebig oft eine Anfrage der Form* ***invoke*** *starten. Bei einem* ***invoke*** *Aufruf startet das Spiel eine neue Instanz von $\mathcal{A}$ mit den gleichen Random Bits wie beim initalen Run. Alle Anfragen von $\mathcal{A}^{\mathcal{H},S}$ werden nun nicht mehr vom Random Oracle oder dem Simulator beantwortet. Die Anfragen werden an K übergeben, dieser kann die Antworten bestimmen, die $\mathcal{A}^{\mathcal{H},S}$ erhält.*

3. ***Evaluation***. *Wenn K einen Vektor von Zeugen w_i ausgeben kann, der zu den Wörtern y aus dem initalen Run passt, d.h. $\forall i$ gilt $(y_i, w_i) \in R$, dann gewinnt der Extraktor das Spiel.*

Das Beweissystem ist Simulation Sound Extractable, wenn ein probabilistischer polynomialzeitbeschränkter Extraktor K existiert, der das Sicherheitsspiel mit nicht-vernachlässigbarer Wahrscheinlichkeit gewinnt:

$$\exists \text{ Polynom } p \forall n_0 \exists \eta > n_0 : Pr[SSE_{\mathcal{A},K}(1^\eta) = 1] > \frac{1}{p(\eta)}$$

2.3 Σ-Protokolle

Σ-Protokolle sind interaktive Beweissysteme. Das bedeutet, zwischen Beweiser und Verifizierer können, im Gegensatz zu nicht-interaktiven Beweissystemen, mehrere Nachrichten ausgetauscht werden. Je mehr Nachrichten ausgetauscht werden müssen, umso aufwändiger gestaltet sich eine Ausführung. Σ-Protokolle sind besonders effiziente interaktive Beweissysteme. Bei ihnen werden genau drei Nachrichten ausgetauscht.

Definition 2.3.1 (Σ-Protokoll)**.** *Sei R eine NP-Relation und $\Lambda \supseteq L_R$ in Polynomialzeit entscheidbar. Das Paar (P, V) ist ein Σ-Protokoll für L_R mit Commitmentraum M_α, Challengeraum M_β und Responseraum M_γ, falls die folgenden Punkte erfüllt sind:*

- *(P, V) hat die Form gemäß Abbildung 2.1.*
- *P und V sind ppt ITMs.*
- *$V(y, \alpha, \beta, \gamma)$ ist ein deterministischer Algorithmus.*
- *Die folgenden Eigenschaften sind erfüllt:*
  - *Korrektheit (siehe Definition 2.3.2)*

[1] Der Begriff Transkript wird in Abschnitt 4.3.2 definiert. Mit Transkript ist sämtliche Kommunikation zwischen dem bösartigen Beweiser und den Orakeln gemeint.

Algorithmus 2.2 Sicherheitsspiel der Simulation Sound Extractability: $SSE_{\mathcal{A},K}$

Input: Securityparameter 1^η
Proofer $\mathcal{A}^{\mathcal{H},S}(1^\eta)$
Extraktor $K^{\mathcal{A}}(1^\eta)$
1. Initial Run
$P = (y_i, \pi_i)_{i\in\{1,\dots,n\}} \xleftarrow{\$} \mathcal{A}^{\mathcal{H},S}$
$p \leftarrow$ number of steps taken by A
if $n = 0$ **then**
  return 1
end
if $\exists(y,\pi) \in P \exists$ ***Simulate***(y) *query with reponse* π *in transkript of run of* $\mathcal{A}$ **then**
  return 1
end
if $\exists(y,\pi) \in P : V^{\mathcal{H}}(y,\pi) = 0$ **then**
  return 1
end
2. Extraction
$(w_i)_{i\in\{1,\dots,m\}} \xleftarrow{\$} K(1^{p(\eta)}, P)$
3. Evaluation
if $n \neq m$ **then**
  return 0
end
if $\exists i \in \{1,\dots,n\} : (y_i, w_i) \notin R$ **then**
  return 0
end
return 1

- *Special Soundness (siehe Definition 2.3.3)*
- *Special Honest Verifier Zero-Knowledge (siehe Definition 2.3.4)*

Bemerkung. Der Verifizierer geht in einem Σ-Protokoll gemäß Algorithmus 2.3 vor.

Definition 2.3.2 (Vollständigkeit von Σ-Protokollen)**.** *Ein Σ-Protokoll (P, V) ist vollständig bezüglich L_R, falls*

$$\forall(y,w) \in R : Pr[\langle P(w), V\rangle(y) = 1] = 1.$$

gilt.

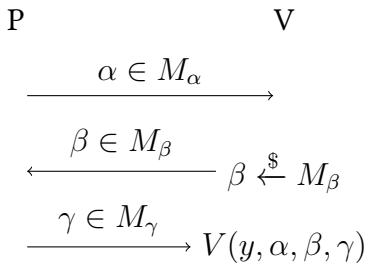

Abbildung 2.1: Aufbau von Σ-Protokollen

Algorithmus 2.3 Verifizierer eines Σ-Protokolls

Input: x
if $x \notin \Lambda$ **then**
  return 0
end
receive commitment α
$\beta \xleftarrow{\$} M_\beta$, send β
receive response γ
return $V(y, \alpha, \beta, \gamma)$

Definition 2.3.3 (Special Soundness)**.** *Seien ein Wort y und zwei akzeptierte Beweise $\pi_1 = (\alpha, \beta, \gamma)$, $\pi_2 = (\alpha, \beta', \gamma')$ mit $\beta \neq \beta'$ gegeben.*

*Ein Σ-Protokoll (P, V) ist Special Sound, falls ein **deterministischer polynomialzeitbeschränkter** Algorithmus **Extract** existiert, der stets auf einer oben beschriebenen Eingabe (y, π_1, π_2) einen Zeugen w für y ausgeben kann, sodass $(y, w) \in R$ gilt.*

Definition 2.3.4 (Special Honest Verifier Zero-Knowledge)**.** *Ein Σ-Protokoll (P, V) ist Special Honest Verifier Zero-Knowledge, falls ein **probabilistischer, polynomialzeitbeschränkter** Algorithmus **Simulate** existiert, der auf Eingabe von $y \in \Lambda$, einer Challenge β und Response γ ein Commitment α ausgibt, sodass $V(y, \alpha, \beta, \gamma) = 1$ gilt.*

*Weiterhin ist das Tripel (α, β, γ), sofern y **aus** L und β und γ zufällig gleichverteilt aus M_β bzw. M_γ gewählt worden sind, zu der Ausführung von Beweiser und Verifier identisch verteilt.*

Bemerkung. **Simulate** verwendet kein Random Oracle.

2.4 Beispiel für ein Σ-Protokoll

Im Folgenden wird ein Beispiel für ein Σ-Protokoll angegeben.

Beispiel 2.4.1. *Sei $L = L_{R_0}$ (siehe Beispiel 2.1.3) mit $R_L = R_0$ (siehe Beispiel 2.1.2) und $\Lambda = \{0,1\}$. Sei η ein Sicherheitsparameter. Im Folgenden wird ein Beweissystem (P,V) für L angegeben:*

- $M_\alpha = M_\beta = \{0,...,2^\eta\}$
- $M_\gamma = \{0\}$
- (P,V) *hat die Form gemäß Abbildung 2.2.*

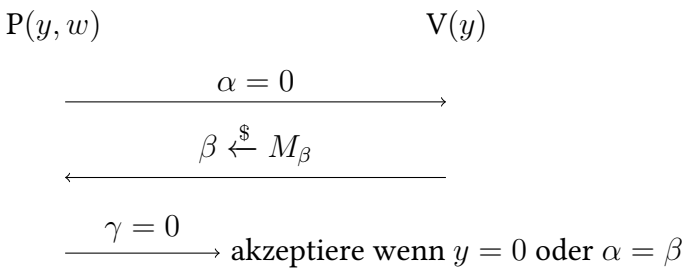

Abbildung 2.2: Beispiel für ein Σ-Protokoll

Behauptung: Es handelt sich hierbei um ein Σ-Protokoll. Dafür müssen die Eigenschaften Vollständigkeit, Special Soundness und Special Honest Verifier Zero-Knowledge erfüllt sein. Diese Eigenschaften werden nun einzeln gezeigt.

Lemma 2.2 (Vollständigkeit). Das gegebene Protokoll erfüllt die Eigenschaft Vollständigkeit.

Beweis. Der Verifizierer akzeptiert auf Grund der Konstruktion jedes $y \in L$. Es gilt:

$$\forall (y,w) \in R : Pr[\langle P(w), V\rangle(y) = 1] = 1.$$

□

Lemma 2.3 (Special Soundness). Das gegebene Protokoll erfüllt die Eigenschaft Special Soundness.

Beweis. Sei **Extract** gegen durch Algorithmus 2.4. Dieser Algorithmus ist deterministisch und polynomialzeitbeschränkt.

Dieser Algorithmus ist ein Extraktor gemäß der Definition von Special Soundness. Um diese Aussage zu zeigen, seien $\pi_1 = (\alpha, \beta, \gamma)$, $\pi_2 = (\alpha, \beta', \gamma')$ mit $\beta \neq \beta'$ und Verify$(y, \pi_1) = 1$ und Verify$(y, \pi_2) = 1$ gegeben. Da für $y \in \Lambda \setminus L$ ein Beweis akzeptiert wird, wenn $\alpha = \beta$ gilt, folgt wegen $\beta \neq \beta' : \alpha \neq \alpha'$. Damit kann y nur 0 sein, womit 0 ein zulässiger Zeuge für y ist. Genau dies gibt der Simulator aus.

□

Algorithmus 2.4 Extraktor für das Beispiel eines Σ-Protokolls

Input: $y \in \Lambda$
$\pi_1 = (\alpha, \beta, \gamma)$ mit Verify$(y, \pi_1) = 1$
$\pi_2 = (\alpha, \beta', \gamma')$ mit Verify$(y, \pi_2) = 1$ und $\beta \neq \beta'$

```
1 return 0
```

Lemma 2.4 (Special Honest Verifier Zero-Knowledge). Das gegebene Protokoll erfüllt die Eigenschaft Special Honest Verifier Zero-Knowledge.

Beweis. Sei S durch Algorithmus 2.5 gegeben. Dieser Algorithmus ist polynomialzeitbeschränkt.

Algorithmus 2.5 Simulator für das Beispiel eines Σ-Protokolls

Input: $y \in \Lambda, \beta \in \{0, ..., 2^\eta\}, \gamma \in \{0\}$

```
1 if y = 0 then
2 |   return 0
3 else
4 |   return β
5 end
```

Dieser Algorithmus ist ein Simulator gemäß der Special Honest Verifier Zero-Knowledge Eigenschaft. Um dies zu zeigen, wird eine beliebige Eingabe y, β, γ betrachtet. Es gibt zwei mögliche Fälle:

- $y = 0$. Der Algorithmus gibt folglich 0 aus. Es gilt Verify$(0, 0, \cdot, \cdot) = 1$. Da sowohl der Algorithmus als auch P das Commitment stets auf 0 setzen, sind die erbrachten Beweise identisch verteilt, sofern die Eingaben β und γ gleichverteilt aus den entsprechenden Mengen gewählt worden sind.
- $y \neq 0$. Der Algorithmus gibt β aus. Somit entspricht das Commitment der Challenge und es gilt Verify$(\cdot, \beta, \beta, \cdot) = 1$.

□

Es sind alle nötigen Eigenschaften gezeigt, damit ist das gegebene Beweissystem (P, V) ein Σ-Protokoll.

2.5 Fiat-Shamir Transformation

Mittels der Fiat-Shamit Transformation kann ein Σ-Protokoll in ein nicht-interaktives Beweissystem umgewandelt werden. Dabei wird Challenge des Verifiers durch eine Hashfunktion ersetzt, die auf das Commitment α (und y) angewendet wird. Beweiser und Verifizierer des Σ-Protokolls werden dazu passend modifiziert.

Wenn nicht anders genannt stammen die Definitionen in diesem Kapitel aus [BPW12]. Erweiterungen sind in **blau** hervorgehoben.

Definition 2.5.1 (Weak Fiat-Shamir Transformation). *Sei $\Sigma = (P_\Sigma, V_\Sigma)$ ein Σ-Protokoll **für eine Sprache** $L \subseteq \Lambda$. Sei $\mathcal{H} : \Lambda \times M_\alpha \to M_\beta$ eine Hash-Funktion. Die Weak Fiat-Shamit Transformation $wFS_h(\Sigma) = (P, V)$ ist ein Beweissystem mit dem Aufbau gemäß Abbildung 2.3, wobei $V(\mathcal{H}, y, \alpha, \beta, \gamma)$ in Algorithmus 2.6 gegeben ist.*

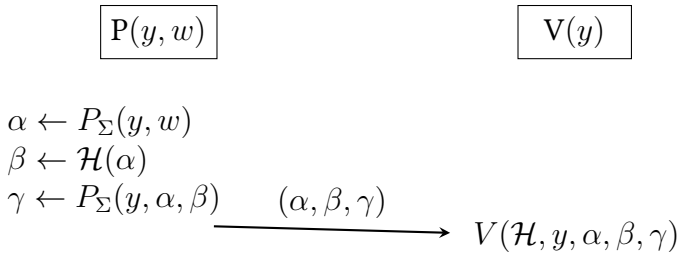

Abbildung 2.3: Weak Fiat-Shamir Transformation

Bemerkung. Die Hashfunktion wird nur auf α angewendet. Diese Variante ist unsicher. Insbesondere ist Korrektheit nicht gegeben [BPW12].

Algorithmus 2.6 Verifizierer in der Weak Fiat-Shamir Transformation: $V(\mathcal{H}, y, \alpha, \beta, \gamma)$

```
1 if H(α) ≠ β then
2 |   return 0
3 end
4 return V_Σ(y, α, β, γ)
```

Definition 2.5.2 (Strong Fiat-Shamir Transformation). *Sei $\Sigma = (P_\Sigma, V_\Sigma)$ ein Σ-Protokoll **für eine Sprache** $L \subseteq \Lambda$. Sei $\mathcal{H} : \Lambda \times M_\alpha \to M_\beta$ eine Hash-Funktion. Die Strong Fiat-Shamit Transformation $sFS_h(\Sigma) = (P, V)$ ist ein Beweissystem mit dem Aufbau gemäß Abbildung 2.3, wobei $V(\mathcal{H}, y, \alpha, \beta, \gamma)$ in Algorithmus 2.7 gegeben ist.*

Bemerkung. Die einzige Änderung zur Weak Fiat-Shamir Transformation ist, dass die Hashfunktion auf α und y angewendet wird.

Bemerkung. Eine Eigenschaft von Prove, die später benötigt wird, ist die Reihenfolge bei der Bestimmung von α, β, γ. Diese Werte werden in genau dieser Reihenfolge vom Beweiser festgelegt.

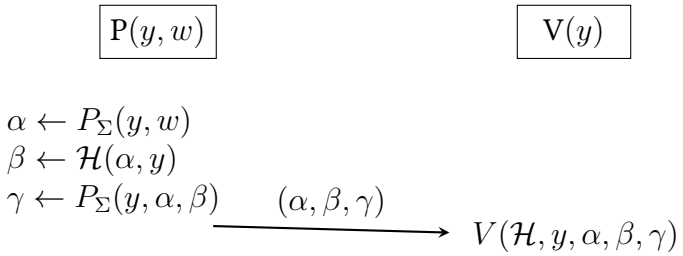

Abbildung 2.4: Strong Fiat-Shamir Transformation

Algorithmus 2.7 Verifizierer in der Strong Fiat-Shamir Transformation: $V(\mathcal{H}, y, \alpha, \beta, \gamma)$

```
1 if H(α, y) ≠ β then
2 |   return 0
3 end
4 return V_Σ(y, α, β, γ)
```

3 Aktueller Forschungsstand

Das folgende Theorem aus [BPW12] stellt den aktuellen Forschungsstand zu Eigenschaften der Fiat-Shamir Transformation dar. Anschließend werden die Beweise, wie sie in diesem Paper dargestellt sind, beschrieben.

Theorem 1 (Theorem 1 des Papers)**.** *Sei Σ ein Σ-Protokoll mit einem im Sicherheitsparameter superpolynomiell großen Challengeraum, das die Eigenschaften Special Soundness und Special Honest Verifier Zero-Knowledge erfüllt. Dann ist sFS(Σ) Zero-Knowledge und Simulation Sound Extractable in Bezug auf **schwach** polynomialzeitbeschränkte Angreifer.*

Bemerkung. Im Paper wird von erwarteten polynomialzeitbeschränkten Angreifern statt von schwach Polynomialzeitbeschränkten gesprochen.

3.1 Beweis der Zero-Knowledge Eigenschaft

Im Paper wird ein Simulator S für die Zero-Knowledge Eigenschaft wie folgt angegeben. Der Simulator verwendet den Algorithmus **Simulate** aus der Special Honest Verifier Zero-Knowledge Eigenschaft. Zusätzlich speichert S sämtliche Hashanfragen mit den zugehörigen Hashwerten in einer Menge M ab. Damit kann der Simulator Hashanfragen stets konsistent beantworten. Die beiden Methoden werden vom Simulator wie folgt abgearbeitet.

- $\mathcal{H}(s)$. Wenn die Hashanfrage s in M bereits vorhanden ist, antwortet S konsistent. Für noch nicht vorhandene s wird wie bei einem Random Oracle ein entsprechend zufälliger Wert r gezogen. S fügt das Paar (s, r) der Menge M hinzu und gibt r aus.
- **Proof** ist in Algorithmus 3.1 beschrieben.

Algorithmus 3.1 Proof

Input: $y \in \Lambda$
1 $\beta \xleftarrow{\$} M_\beta, \gamma \xleftarrow{\$} M_\gamma$
2 $\alpha \leftarrow$ **simulate**(y, β, γ)
3 add $((\alpha, y), \beta)$ to M
4 return $(y, \alpha, \beta, \gamma)$

An dieser Stelle endet der Beweis zur Zero-Knowledge Eigenschaft im Paper.

Bemerkung. In Schritt 3 überschreibt der Simulator ein möglicherweise bereits vorhandenes Paar $((\alpha, y), \beta')$ mit $\beta' \neq \beta$.

3.2 Beweis der Simulation Sound Extractable Eigenschaft

In der folgenden Definition wird der Extraktor beschrieben, der im Paper als Beweis angegeben wird.

Definition 3.2.1 (Extraktor K des Papers)**.** *Der Algorithmus K erhält als Input das Transkript und den Vektor an Wörtern mit Beweisen (y_i, π_i) von der Ausführung von $A^{\mathcal{H},S}$.*

*Für **invoke** Anfragen konstruiert K eine Orakelliste. Diese Orakelliste beinhaltet alle Orakelanfragen mit den zugehörigen Antworten gemäß dem Transrkipt.*

K iteriert über alle Elemente im Vektor. Sei (y, π) mit $\pi = (\alpha, \beta, \gamma)$ das aktuell betrachtete Paar. K geht wie folgt vor:

1. *K prüft, ob im Transkript eine Orakelanfrage mit Eingabe (α, y) vorhanden ist.*

   Der Beweis wurde von $V^{\mathcal{H}}$ akzeptiert. Ansonsten hat K gemäß der Definition des Spiels automatisch gewonnen. Der Verifizierer überprüft die Korrektheit des Beweises. Insbesondere überprüft er, ob die Challenge korrekt gewählt worden ist, d.h. $\mathcal{H}(\alpha, y) = \beta$ gilt. Wenn keine Orakelanfrage $\mathcal{H}(\alpha, y)$ im Transkript vorhanden ist, dann wurde die Ausgabe dieser Orkalenanfrage erst durch $V^{\mathcal{H}}$ bei der Verifizierung ermittelt.

   Wenn diese Orakelanfrage nicht im Transkript vorhanden ist, bricht der Extraktor ab. Dies passiert mit vernachlässigbarer Wahrscheinlichkeit. Das liegt daran, dass der Challengeraum superpolynomiell groß ist und der Hashwert gleichverteilt gewählt wird.

2. *K startet eine **invoke** Anfrage. K antwortet gemäß der Orakelliste bis zur Orakelanfrage (α, y). Diese Anfrage existiert, denn sonst hätte K in Schritt 1 abgebrochen. **Im Paper wird von der Orakelanfrage α gesprochen. Dies entspricht nicht der Strong Fiat-Shamir Transformation.** K wählt einen neuen Hashwert β' **gleichverteilt aus M_β** und ersetzt in der Orakelliste (y, β) durch (y, β') und beantwortet die Orakelanfrage mit β'. Anschließend simuliert K für alle weiteren Orakelanfragen ein Random Oracle gemäß der angepassten Orakelliste und beantwortet Simulationsanfragen mit dem Simulator S. **Das bedeutet, Orakelanfragen, die bereits angefragt worden sind, werden konsistent beantwortet. Die Antworten auf neue Fragen werden zufällig gleichverteilt gewählt.***

3. *Wenn die **invoke** Anfrage einen Beweis (y, α') ausgibt, sodass K den Extraktor der Special Soundness (siehe Definition 2.3.3) anwenden kann, geht K zum nächsten Paar aus dem Vektor über. **Ansonsten gibt K auf.***

4. *K gibt die mittels des Extraktors der Special Soundness gewonnen Zeugen aus.*

Die Wahrscheinlichkeit, dass in der ***invoke*** *Anfrage ein korrekter Beweis mit dem Hashwert β' ausgegeben wird, ist wegen des Forking Lemmas nicht-vernachlässigbar. Zusätzlich gilt $\beta' \neq \beta$ mit nicht-vernachlässigbarer Wahrscheinlichkeit. Damit hat K eine nicht-vernachlässigbare Gewinnwahrscheinlichkeit.*

4 Eigenschaften der Strong Fiat-Shamir Transformation

In diesem Kapitel werden die in Kapitel 3 vorgestellten Beweise analysiert und, falls nötig, erweitert.

4.1 Beweisvorbereitung

Zunächst werden einige Lemmata bewiesen, die für die Beweise als Hilfe verwendet werden.

4.1.1 Laufzeit

Lemma 4.1 (Schwache Polynomialzeitbeschränkung). Sei A ein Algorithmus mit schwacher polynomialbeschränkter Laufzeit. Dann ist die Wahrscheinlichkeit, dass A eine polynomielle Laufzeit hat, überwältigend:

$$\exists \text{ Polynom } p : Pr[t_A(x) \leq p(|x|)] \text{ ist überwältigend}$$

Beweis. Folgt direkt aus der Definition der schwachen Polynomialzeitbeschränkung und der Definition von überwältigenden Funktionen. □

Theorem 2 (Schwach polynomialzeitbeschränkte Angreifer für die Zero-Knowledge Eigenschaft)**.** *Sei Σ ein Σ-Protokoll und sei sFS(Σ) Zero-Knowledge in Bezug auf polynomialbeschränkte Angreifer.*

Dann ist sFS(Σ) Zero-Knowledge in Bezug auf schwach polynomialzeitbeschränkte Angreifer.

Beweis. Sei S der Simulator gemäß der Zero-Knowledge Eigenschaft gegenüber polynomialzeitbeschränkten Angreifern. Sei A ein schwach polynomialzeitbeschränkter Angreifer.

Im Folgenden wird gezeigt, dass der Simulator S sicher gegenüber dem Angreifer A ist. Der Algorithmus A besitzt eine schwache polynomialzeitbeschränkte Laufzeit. Mit Lemma 4.1 folgt, dass ein Polynom p existiert, sodass

$$\exists n_0 \forall x \text{ mit } |x| \geq n_0 : Pr[t_A(x) \leq p(|x|)] \text{ ist überwältigend}$$

gilt. Gemäß der Definition von überwältigenden Funktionen gilt auch, dass $Pr[t_A(x) > p(|x|)]$ vernachlässigbar ist. Das bedeutet, für alle positiven Polynome q gilt:

$$\exists n_0 \forall x \text{ mit } |x| \geq n_0 : Pr[t_A(x) > p(|x|)] \leq \frac{1}{q(n)}.$$

Im Folgenden werden zwei Behauptungen gezeigt, die die zu beweisende Aussage implizieren:

1. Wenn $A(1^\eta; \alpha)$ eine Laufzeit kleiner gleich als $p(|1^\eta|)$ besitzt, dann ist der Simulator S sicher gegenüber dieser Ausführung. Dies folgt aus der Voraussetzung, dass der Simulator sicher gegenüber polynomialzeitbeschränkten Angreifern ist.
2. Wenn $A(1^\eta; \alpha)$ eine Laufzeit größer als $p(|1^\eta|)$ aufweist, spielt es keine Rolle, wie sich der Simulator verhält.

   Angenommen, $A(1^\eta; \alpha)$ schafft es stets, die beiden Szenarien voneinander zu unterscheiden, wenn die Laufzeit größer als $p(|1^\eta|)$ ist. Wie zuvor gezeigt, ist die Wahrscheinlichkeit, dass $A(1^\eta, \alpha)$ eine Laufzeit größer als $p(|1^\eta|)$ aufweist, vernachlässigbar. Da der Simulator gegenüber den übrigen Ausführungen sicher ist, kann A nur mit vernachlässigbarer Wahrscheinlichkeit gewinnen.

Damit ist der Simulator S sicher gegenüber Angreifer A und damit sicher gegenüber schwach polynomialzeitbeschränkten Angreifern.

□

Theorem 3 (Schwach polynomialzeitbeschränkte Angreifer für die Simulation Sound Extractable Eigenschaft)**.** *Sei Σ ein Σ-Protokoll und sei sFS(Σ) Simulation Sound Extractable in Bezug auf polynomialbeschränkte Angreifer.*

Dann ist sFS(Σ) Simulation Sound Extractable in Bezug auf schwach polynomialzeitbeschränkte Angreifer.

Beweis. Sei K der Extraktor für polynomialzeitbeschränkte Angreifer. Dieser hat eine polynomialzeitbeschränkte Laufzeit. Sei $p(1^\eta)$ ein Polynom, das die Laufzeit des Extraktors nach oben hin beschränkt.

Im Sicherheitsspiel der Simulation Sound Extractability werden somit maximal $p(\eta) + 1$ viele Ausführungen eines Angreifers gestartet: ein initialer Run und maximal $p(\eta)$ viele **invoke** Anfragen.

Gemäß Lemma 4.1 existiert ein Polynom $q(\eta)$, sodass die Wahrscheinlichkeit, dass der Angreifer mit schwach polynomialzeitbeschränkter Laufzeit eine Laufzeit kleiner gleich $q(\eta)$ annimmt, überwältigend ist. Mit anderen Worten ist die Funktion $p'(\eta) := 1 - Pr[t_A(\eta) \leq q(\eta)]$ vernachlässigbar.

Damit ist auch die Wahrscheinlichkeit, dass alle $p(\eta) + 1$ Ausführungen eine Laufzeit kleiner gleich $q(\eta)$ annehmen, überwältigend:

Seien $\alpha_1, ..., \alpha_{p(\eta)+1}$ mögliche Random Bits für den Angreifer, bestehend aus den Random Bits für A sowie den Random Bits für die Bearbeitung der Orakelanfragen. Es gilt:

$$\begin{aligned} Pr[t_A(1^\eta; \alpha_i) > q(\eta) \vee ... \vee t_A(1^\eta; \alpha_{p(\eta)+1}) > q(\eta)] &\leq \sum_{i \in \{1,...,p(\eta)+1\}} Pr[t_A(1^\eta; \alpha_i) > q(\eta)] \\ &\leq \sum_{i \in \{1,...,p(\eta)+1\}} p'(\eta) \\ &= (p(\eta) + 1) \cdot p'(\eta) \end{aligned}$$

Da $p'(\eta)$ vernachlässigbar und $p(\eta) + 1$ ein Polynom ist, ist $(p(\eta) + 1) \cdot p'(\eta)$ vernachlässigbar. Damit ist die Wahrscheinlichkeit, dass der Angreifer im Sicherheitsspiel eine superpolynomielle Laufzeit annimmt, vernachlässigbar.

Ein Extraktor, der gegenüber polynomialzeitbeschränkten Angreifern eine nicht-vernachlässige Gewinnchance besitzt, weist somit auch gegen schwach polynomialzeitbeschränkte Angreifer eine nicht-vernachlässigbare Gewinnchance auf.

□

4.1.2 Vernachlässigbare Funktionen

Lemma 4.2 (Vernachlässigbare Funktion). Sei $e(n)$ eine superpolynomielle Funktion und $p(n)$ ein Polynom. Dann ist $f(n) := \frac{p(n)}{e(n)}$ vernachlässigbar.

Beweis. Beweis per Widerspruch.

Angenommen, f ist nicht-vernachlässigbar. Dann existiert ein Polynom q mit

$$\begin{aligned} &\forall n_0 \exists n > n_0 : f(n) > \frac{1}{q(n)} \\ \Leftrightarrow\ &\forall n_0 \exists n > n_0 : \frac{p(n)}{e(n)} > \frac{1}{q(n)} \\ \Leftrightarrow\ &\forall n_0 \exists n > n_0 : \frac{p(n)q(n)}{e(n)} > 1 \end{aligned}$$

Die Funktion $p'(n) := p(n) \cdot q(n)$ ist nach Definition von p und q ein Polynom.

$$\Leftrightarrow \forall n_0 \exists n > n_0 : \frac{p'(n)}{e(n)} > 1 \lightning$$

□

Lemma 4.3. Sei f eine nicht-vernachlässigbare Funktion und p ein Polynom. Dann ist die Funktion $g(n) := \frac{f(n)}{p(n)}$ nicht-vernachlässigbar.

Beweis. Sei f eine nicht-vernachlässigbare Funktion und p ein Polynom. Angenommen, $\frac{f(n)}{p(n)}$ ist vernachlässigbar. Dann gilt:

$$\forall \text{ positive Polynome } p' \exists n_0 \forall n > n_0 : \frac{f(n)}{p(n)} < \frac{1}{p'(n)}$$

Da f nicht-vernachlässigbar ist, gilt:

$$\exists \text{ positives Polynom } q \forall n_0 \exists n > n_0 : f(n) \geq \frac{1}{q(n)}$$

Im Folgenden wird das positive Polynom $p'(n) := p(n)q(n)$ betrachtet. Da $\frac{f(n)}{p(n)}$ vernachlässigbar ist, gilt:

$$\begin{aligned} \forall n_0 \exists n > n_0 : & \frac{f(n)}{p(n)} < \frac{1}{p'(n)} = \frac{1}{p(n)q(n)} \\ & \Leftrightarrow f(n) < \frac{1}{q(n)} \end{aligned}$$

Dies ist ein Widerspruch dazu, dass f nicht-vernachlässigbar ist $\lightning$.

□

Lemma 4.4. Sei f eine nicht-vernachlässigbare Funktion und g eine vernachlässigbare Funktion. Dann ist die Funktion $h(n) := f(n) - g(n)$ nicht-vernachlässigbar.

Beweis. Sei f eine nicht-vernachlässigbare Funktion und g eine vernachlässigbare Funktion. Angenommen, die Funktion $h(n) := f(n) - g(n)$ ist vernachlässigbar. Da die Menge der vernachlässigbaren Funktion abgeschlossen gegenüber Addition ist, ist die Funktion $h'(n) := h(n) + g(n)$ vernachlässigbar. Es gilt:

$$h'(n) = h(n) + g(n) := f(n) - g(n) + g(n) = f(n)$$

Nach Voraussetzung ist die Funktion f jedoch nicht-vernachlässigbar $\lightning$.

□

Lemma 4.5. Sei $f(n)$ eine nicht-vernachlässigbare Funktion. Dann ist $f(n)^2$ eine nicht-vernachlässigbare Funktion.

Beweis. Es wird gezeigt, dass wenn f eine vernachlässigbare Funktion ist, dann ist $\sqrt{f}$ ebenfalls eine vernachlässigbare Funktion. Daraus folgt die zu zeigende Aussage, denn wenn $f(n)^2$ vernachlässigbar ist, dann ist damit auch $\sqrt{f(n)^2} = f(n)$ eine vernachlässigbare Funktion.

Sei f eine vernachlässigbare Funktion. Dann gilt:

$$\forall \text{ positive Polynome } p \exists n_0 \forall n > n_0 : f(n) < \frac{1}{p(n)}$$

Insbesondere gilt damit für das Polynom $q(n) = 1$:

$$\exists n_0' \forall n > n_0' : f(n) < \frac{1}{q(n)} = 1$$

Sei nun p ein beliebiges, positives Polynom. Sei n_0 so, dass gilt:

$$\forall n > n_0 : f(n) < \frac{1}{p(n)}$$

Da f vernachlässigbar ist, existiert so ein n_0.

Sei nun $n_0'' = \max\{n_0, n_0'\}$. Es gilt:

$$\forall n > n_0'' : f(n)^2 < f(n),$$

da $f(n) < 1$ für alle $n > n_0''$. Es folgt:

$$\forall n > n_0'' : f(n)^2 < f(n) < \frac{1}{p(n)}$$

Damit ist $f(n)^2$ vernachlässigbar.

□

Lemma 4.6 (Forking Algorithmus Wahrscheinlichkeit). Sei $a(n) : \mathbb{N} \to [0,1]$ eine nicht-vernachlässige Funktion, $q(n)$ ein positives Polynom und $h(n)$ eine superpolynomielle Funktion. Dann ist die Funktion

$$f(n) := a\left(\frac{a}{q} - \frac{a}{h}\right)$$

nicht-vernachlässigbar.

Beweis. Sei $a(n) : \mathbb{N} \to [0,1]$ eine nicht-vernachlässige Funktion, $q(n)$ ein positives Polynom und $h(n)$ eine superpolynomielle Funktion. Sei

$$f(n) := a(n)\left(\frac{a(n)}{q(n)} - \frac{1}{h(n)}\right).$$

Es gilt:

$$\begin{aligned} f(n) &= \frac{a(n)^2}{q(n)} - \frac{a(n)}{h(n)} \\ &\geq \frac{a(n)^2}{q(n)} - \frac{1}{h(n)}, \end{aligned}$$

da a nach $[0,1]$ abbildet. Weiterhin gilt:

- Da a nach Voraussetzung nicht-vernachlässigbar ist, ist a^2 nach Lemma 4.5 ebenfalls nicht-vernachlässigbar. Da q polynomiell ist, folgt schließlich mittels Lemma 4.3, dass $\frac{a(n)^2}{q(n)}$ nicht-vernachlässigbar ist.
- Da h eine superpolynomielle Funktion ist, ist $\frac{1}{h(n)}$ vernachlässigbar.

Mittels Lemma 4.4 folgt daraus, dass f nicht-vernachlässigbar ist.

□

Lemma 4.7. Sei X ein Ereignis mit nicht-vernachlässigbarer Erfolgswahrscheinlichkeit $s(\eta)$. Wiederholt man das Experiment $q(\eta)^2$ mal, wobei q ein Polynom ist, dann ist die Wahrscheinlichkeit, mindestens einen Erfolg zu haben, überwältigend.

Beweis.

$$Pr[q(\eta) \text{ mal Misserfolg}] = (1 - s(\eta))^{q(\eta)}$$

Da s nicht vernachlässigbar ist, existiert ein Polynom p, sodass gilt:

$$\forall n_0 \exists n > n_0 : s(n) \geq \frac{1}{p(n)}$$

Damit folgt:

$$(1 - s(\eta))^{q(\eta)} \leq \left(1 - \frac{1}{p(\eta)}\right)^{(q(\eta)^2)} = \left(\frac{1}{e}\right)^{q(\eta)}$$

wobei letzteres vernachlässigbar ist. Damit ist die Wahrscheinlichkeit, keinen Erfolg zu haben, vernachlässigbar. Daraus folgt, dass die Wahrscheinlichkeit, mindestens einen Erfolg zu haben, überwältigend ist.

□

4.2 Vollständigkeit

Lemma 4.8. Sei Σ ein Σ-Protokoll und sei $(P, V) = \text{sFS}(\Sigma)$. Dann ist (P, V) vollständig.

Beweis. Sei (P_Σ, V_Σ) ein Σ-Protokoll für die Sprache L_R und sei $(P, V) = \text{sFS}_\Sigma(P_\Sigma, V_\Sigma)$.

Da (P_Σ, V_Σ) ein Σ-Protokoll ist, erfüllt es die Eigenschaft der Vollständigkeit für Σ-Protokolle. Es gilt:

$$\forall (y, w) \in R : Pr[\langle P_\Sigma(w), V_\Sigma \rangle(y) = 1] = 1$$

Der Verifizierer V_Σ ist gemäß der Definition von Σ-Protokollen wie Algorithmus 2.3 aufgebaut. Auf Grund der Konstruktion des Verifizierers kommt für die Challenge jeder mögliche Wert aus M_β in Frage. Damit gilt:

$$\forall \beta \in M_\beta \forall (y, w) \in R : Pr[\langle P_\Sigma(w), V_\Sigma \rangle(y) = 1 | \text{ Challenge ist } \beta] = 1$$

Im Beweissystem (P, V) wird die Challenge vom Hashorakel gewählt. Dieses Hashorakel wählt seine Antwort auf eine bisher ungesehen Anfrage zufällig gleichverteilt aus M_β. Da P mittels P_Σ einen Beweis erzeugt, welcher als Challenge ein Wort aus M_β verwendet, folgt

$$\forall (y, w) \in R : Pr[\langle P(w), V \rangle(y) = 1] = 1.$$

Damit ist die Vollständigkeit des Beweissystems gezeigt.

□

4.3 Zero-Knowledge

Zunächst wird aufgezeigt, warum der gegebene Beweis nicht hinreichend ist. Anschließend werden die Lücken geschlossen.

4.3.1 Ein Gegenbeispiel zum bisherigen Beweis

Sei (P_Σ, V_Σ) das Σ-Protokoll aus Beispiel 2.4.1. Sei $(P, V) = \text{sFS}_\mathcal{H}(P_\Sigma, V_\Sigma)$ mit einer Hashfunktion $\mathcal{H} : \Lambda \times M_\alpha \to M_\beta$.

Lemma 4.9. Der Simulator S, der im Paper beschrieben wird, ist kein Simulator für die Zero-Knowledge Eigenschaft von (P, V).

Beweis. Algorithmus 4.1 beschreibt einen Unterscheider U.

Dieser Unterscheider kann die beiden Szenarien mit nicht-vernachlässigbarer Wahrscheinlichkeit voneinander unterscheiden:

- Szenario 1. Auf die Anfrage Proof$(0, 0)$ in Zeile 2 wird stets $(0, x, 0)$ zurückgeben. Es gilt $V_\Sigma(0, 0, x, 0) = 1$. Daraus folgt:

$$Pr[U(S1, 1^\eta) = 1] = 1$$

Algorithmus 4.1 Unterscheider für das Σ-Protokoll aus Beispiel 2.4.1

```
1 x ← H(0, 0)
2 (α, β, γ) ← Proof(0, 0)
3 x' ← H(0, 0)
4 if x ≠ x' or x ≠ β or V(0, α, β, γ) = 0 then
5 |   return 2
6 else
7 |   return 1
8 end
```

- Szenario 2. In Zeile 2 wird der Simulator S die Challenge gleichverteilt aus M_β wählen. Wenn diese Challenge nicht äquivalent zu x aus Zeile 1 ist, gibt es zwei Möglichkeiten:

  1. Der Simulator lässt das Random Oracle unverändert. In diesem Fall gibt der Unterscheider 2 wegen $x \neq \beta$ aus.
  2. Da der Simulator die Kontrolle über das Random Oracle hat, kann dieser den Eintrag bezüglich x im Random Oracle überschreiben. In diesem Fall gibt der Unterscheider 2 wegen $x \neq x'$ aus.

  Die einzige Möglichkeit für den Simulator, dass U nicht 2 ausgibt, besteht folglich darin, dass x als Challenge gewählt wird:

$$Pr[U(S2, 1^\eta) = 1] = Pr[S \text{ wählt Challenge } x] = \frac{1}{|M_\beta|}$$

Es gilt $|Pr[U(S1, 1^\eta) = 1] - Pr[U(S2, 1^\eta) = 1]| = |1 - \frac{1}{|M_\beta|}|$, was nicht-vernachlässigbar ist, da $|M_\beta|$ superpolynomiell ist. Damit ist S kein Simulator für die Zero-Knowledge Eigenschaft bezüglich diesem Unterscheider. □

Was ist nun das Problem des Simulators?

Angenommen, der Simulator versucht eine Anfrage Proof(y, w) mit $(y, w) \in R$ zu beantworten.

Sei dafür α das Commitment, das die **simulate**(y, β, γ) Anfrage ausgibt. Sei ferner $((\alpha, y), \beta') \in M$ mit $\beta' \neq \beta$. Der Hashwert von (α, y) ist folglich bereits mit einem anderen Wert als β gesetzt.

Der Simulator hat nun zwei Möglichkeiten für Schritt 3:

1. Er kann das Paar $((\alpha, y), \beta')$ in M überschreiben. In diesem Fall antwortet der Simulator nicht konsistent bezüglich des Random Oracles. Somit ist diese Ausgabe von der vom Beweiser unterscheidbar, da der Beweiser stets konsistent antwortet.
2. Er kann M unmodifiziert lassen. In diesem Fall ist der Simulator nicht in der Lage einen Beweis auszugeben, der vom Verifizierer akzeptiert wird. Das liegt daran, dass der Hashwert in dem Beweis (α, β, γ) für y nicht stimmt, da $\mathcal{H}(\alpha, y) = \beta'$ mit $\beta' \neq \beta$ gilt. Auch in diesem Fall ist die Ausgabe von der vom Beweiser unterscheidbar, da der Beweiser stets Beweise ausgibt, die vom Verifizierer akzeptiert werden.

Beide Fälle ermöglichen es dem Unterscheider zwischen dem Simulator und Beweiser zu differenzieren. Folglich muss die Wahrscheinlichkeit, in diesen Fällen zu landen, vernachlässigbar sein. Dafür muss sichergestellt werden, dass die Wahrscheinlichkeit vernachlässigbar ist, dass der Simulator mittels **simulate** ein Commitment erhält, das zusammen mit y schon in M eingetragen ist.

Ein möglicher Unterscheider kann versuchen, möglichst viele Hashwerte durch Aufrufe von $\mathcal{H}(\cdot, y)$ zu setzen, um die Wahrscheinlichkeit zu maximieren, die obige Situation herbeizuführen.

Um über diese Thesen nähere Aussagen treffen zu können, wird im Folgenden die Größe des Commitmentraumes untersucht. Je kleiner dieser, in Relation zur Laufzeit des Unterscheiders, ist, umso wahrscheinlicher ist es, dass der Unterscheider die obige Situation durch Setzen der Hashwerte herbeiführen kann.

4.3.1.1 Die Größe des Commitmentraumes

Wie klein kann der Commitmentraum eines Σ-Protokolls sein? Es wird bewiesen, dass der Commitmentraum für ein Σ-Protokoll mindestens so groß sein muss wie der Challengeraum.

Theorem 4. *Sei (P, V) ein Σ-Protokoll für eine Sprache $L \subsetneq \Lambda$. Dann gilt $|M_\alpha| \geq |M_\beta|$.*

Beweis. Angenommen, (P, V) ist ein Σ-Protokoll mit $|M_\alpha| < |M_\beta|$.

Das Σ-Protokoll erfüllt per Definition die Special Honest Verifier Zero-Knowledge Eigenschaft. Das bedeutet, es existiert ein Algorithmus **Simulate**, der bei Eingabe $y \in \Lambda$, einer Challenge β und einer Response γ ein Commitment α ausgibt, sodass $V(y, \alpha, \beta, \gamma) = 1$ gilt.

Wegen $L \subsetneq \Lambda$ existiert ein $y \in \Lambda \setminus L$. Sei $\gamma \in M_\gamma$ beliebig. Für dieses γ wird die folgende Menge betrachtet:

$$N = \cup_{\beta \in M_\beta} \{(\textbf{Simulate}(y, \beta, \gamma), \beta, \gamma)\}.$$

Diese Menge besitzt verschiedene Eigenschaften:

1. Es gilt $\forall \pi \in N : V(y, \pi) = 1$, wegen der Eigenschaften von **Simulate**.
2. Zu jedem $\beta \in M_\beta$ existiert genau ein Beweis in N, der diese Challenge β verwendet.
3. Wegen $|M_\alpha| < |M_\beta|$ existiert ein $\alpha \in M_\alpha$, sodass α in mindestens zwei Beweisen als Commitment verwendet wird.

Sei $\alpha \in M_\alpha$ ein solches Element aus Eigenschaft 3. Dann existieren für y die Beweise (α, β, γ) und $(\alpha, \beta', \gamma')$, mit $\beta \neq \beta'$ wegen Eigenschaft 2. Da wegen Eigenschaft 1 beide Beweise vom Verifizierer akzeptiert werden, kann mittels des Algorithmus der Special Soundness Eigenschaft ein Zeuge w aus den beiden Beweisen extrahiert werden, sodass $(y, w) \in R$ gilt. Damit ist $y \in L$. $\lightning$

□

4.3.1.2 Die Wahl des Commitments

Es ist zwar gezeigt worden, dass der Commitmentraum mindestens so groß ist wie der Challengeraum, dennoch ist nicht sichergestellt, dass der Simulator die Chance hat, eine bisher unbenutzte Eingabe für die Hashfunktion zu finden.

Wählt der Beweiser in einem Σ-Protokoll das Commitment nicht aus dem kompletten Commitmentraum, sondern nur aus einer sehr kleinen Teilmenge, dann können wie zuvor gezeigt, die Hashwerte für diese Teilmenge belegt werden, und der Simulator scheitert.

4.3.2 Notation

4.3.2.1 Wahrscheinlichkeitsraum

Definition 4.3.1 (Ausführung von Unterscheider und Szenario)**.** *Sei U ein Unterscheider, S ein Szenario (Szenario 1 oder 2) und seien α, β Random Bit Strings. $\langle U^\alpha, S^\beta \rangle$ bezeichne eine Ausführung von U mit S, wobei U Random Bits α und S Random Bits β erhalten und α, β gleichverteilt gewählt werden.*

Definition 4.3.2 (Zugrundeliegender Wahrscheinlichkeitsraum)**.** *Sei U ein Unterscheider und S ein Szenario. Der zugrundeliegende Wahrscheinlichkeitsraum für $\langle U, S \rangle$ ist eine Gleichverteilung auf $\Omega = \{0,1\}^* \times \{0,1\}^*$.*

Definition 4.3.3 (Wahrscheinlichkeitsraum für eine Proof Anfrage in Szenario 2)**.** *Der Wahrscheinlichkeitsraum für eine Proof Anfrage in Szenario 2 ist*

$$\Omega = M_\beta \times M_\gamma \times \{0,1\}^{t_{Simulate}}$$

Wobei $t_{Simulate}$ die Laufzeit des Simulators aus der Special Honest Verifier Zero-Knowledge Eigenschaft ist.

4.3.2.2 Vorangegangene Kommunikation

Definition 4.3.4 (Transkript einer Anfrage)**.** *Sei U ein Algorithmus, der eine Anfrage an ein Szenario S stellt. Ein Transkript dieser Anfrage ist ein Tupel (X, q, r) mit den folgenden Bestandteilen:*

- *$X \in \{\mathcal{H}, Proof\}$ gibt die angefragte Methode an*
- *q ist die Eingabe für die Methode*
- *r ist die Antwort des Szenarios*

Bemerkung. Die Methode Proof entspricht in Szenario 1 der Methode Proof und in Szenario 2 der Methode Simulate.

Definition 4.3.5 (Transkript)**.** *Sei $\langle U^\alpha, S^\beta \rangle$ eine Ausführung von einem Unterscheider U und einem Szenario S mit Random Bits α bzw. β. Sei n die Anzahl an Anfragen, die U während dieser Ausführung an S stellt.*

Dann ist das Transkript der Ausführung $m := (X_i, q_i, r_i)_{i \in \{1,\dots,n\}}$ eine Liste von Transkripten von Anfragen, wobei (X_i, q_i, r_i) die i-te Anfrage von U an S ist.

Weiterhin bezeichne $|m| := n$ die Anzahl der Elemente in m und $m_i := (X_i, q_i, r_i)$ bezeichne das Transkript der i-ten Anfrage von U.

Definition 4.3.6 (Transkript einer Ausführung)**.** *Sei $\langle U^\alpha, S^\beta \rangle$ eine Ausführung von einem Unterscheider U und einem Szenario S mit Random Bits α bzw. β. Die Notation $m \leftarrow_t \langle U^\alpha, S^\beta \rangle$ bezeichne, dass m das Transkript zugewiesen wird, das während der Ausführung entsteht.*

Definition 4.3.7 (Anfang eines Transkriptes)**.** *Sei $\langle U^\alpha, S^\beta \rangle$ eine Ausführung von einem Unterscheider U und einem Szenario S mit Random Bits α bzw. β. Die Notation $m \leftarrow_t^n \langle U^\alpha, S^\beta \rangle$ bezeichne, dass m das Transkript der ersten n Anfragen zugewiesen wird, die während der Ausführung entstehen:*

$$m \leftarrow_t^n \langle U^\alpha, S^\beta \rangle := m'_1, ..., m'_n \text{ mit } m' \leftarrow_t \langle U^\alpha, S^\beta \rangle,$$

also die ersten n Transkripte von Anfragen in m' und m identisch sind.

Definition 4.3.8 (Ereignis für ein Transkript)**.** *Sei U ein Unterscheider, S ein Szenario und m ein Transkript der Länge n.*

Dann ist $E_m := \{(\alpha, \beta) \mid m' \leftarrow_t^n \langle U^\alpha, S^\beta \rangle, m = m'\}$ das Ereignis über alle Random Bits, bei denen mittels Ausführung ein Transkript m' erzeugt wird, dessen erste n Einträge mit m übereinstimmen.

Definition 4.3.9 (Ereignis für ein Transkript mit Anfrage)**.** *Sei U ein Unterscheider, S ein Szenario und m ein Transkript der Länge n. Ferner sei $Y \in \{\mathcal{H}, Proof\}$.*

Dann ist $E_{m,Y,q} := \{(\alpha, \beta) \mid m' \leftarrow_t^{n+1} \langle U^\alpha, S^\beta \rangle, \forall i \in \{1, ..., n\} : m_i = m'_i, m'_{n+1} = (Y, q, \cdot)\}$ das Ereignis über alle Random Bits, bei bei denen mittels Ausführung ein Transkript m' erzeugt wird, dessen erste n Einträge mit m übereinstimmen und die $n+1$-te Anfrage $Y(q)$ ist.

4.3.2.3 Zufallsvariablen für Transkripte und Szenarien

Definition 4.3.10 (Zufallsvariable für eine Anfrage)**.** *Sei $E_{m,Y,q}$ ein Ereignis gemäß Definition 4.3.9 mit $|m| = n$. Im Folgenden wird der Wahrscheinlichkeitsraum für die Ausgabe von $Y(q)$ mit vorausgegangener Kommunikation m betrachtet, der durch $Pr'[x] = Pr[x \mid E_{m,Y,q}]$ indiziert wird.*

Sei $X \in \{1, 2\}$ das betrachtete Szenario.

Die Zufallsvariable $SX_{Y,q,m}$ fragt nach der Ausgabe der $n+1$-ten Anfrage. Diese Zufallsvariable ist wie folgt aufgebaut:

- *$X \in \{1, 2\}$ gibt das Szenario an*
- *$Y \in \{\mathcal{H}, Proof\}$ gibt die angefragte Methode an*
- *q ist der Input für die Anfrage*

Diese Zufallsvariable ist wie folgt definiert:

$$SX_{Y,q,m}(\alpha, \beta) := r, \text{ sodass } m' \leftarrow_t \langle U^\alpha, S^\beta \rangle, m'_{n+1} = (Y, q, r)$$

Bemerkung. Die Methode Proof entspricht in Szenario 1 der Methode Proof und in Szenario 2 der Methode Simulate.

Bemerkung. Da als Wahrscheinlichkeitsraum die bedingte Wahrscheinlichkeit unter $E_{m,Y,q}$ betrachtet wird, gilt $(\alpha, \beta) \in E_{m,Y,q}$. Damit ist das r stets definiert.

Bemerkung. Sei m ein Transkript mit $|m| = n$ und $m' := m_1, ..., m_n, (Y, q, r)$. Es gilt:

$$Pr[SX_{Y,q,m} = r] = Pr[E_{m'} \mid E_{m,Y,q}]$$

4.3.2.4 Zufallsvariablen für Proof Anfragen

Wie in Abschnitt 2.5 aufgezeigt wurde, wird in Szenario 1 zunächst das Commitment, anschließend die Challenge und schlussendlich die Response gewählt. Damit kann die Zufallsvariable $S1_{\text{Proof},q,m}(\alpha, \beta)$ weiter zerlegt werden.

Definition 4.3.11 (Zufallsvariable für das Commitment)**.** *Sei m ein Transkript der Länge n und (y, w) eine Eingabe für eine Proof Anfrage. Für $S1_{\textit{Proof},(y,w),m}$ sei $S1_{(y,w),m}$ die Zufallsvariable, die das Commitment beschreibt, das in $S1$ für den Beweis gewählt wird. Das entspricht dem Schritt $\alpha \leftarrow \textbf{\textit{Prove}}_{\Sigma}(y, w)$ in der Definition der Strong Fiat-Shamir Transformation:*

$$S1_{(y,w),m}(\alpha', \beta') = \begin{cases} \alpha & , S1_{\textit{Proof},(y,w),m}(\alpha', \beta') = (\alpha, \cdot, \cdot) \\ \perp & , \textit{ sonst} \end{cases}$$

Bemerkung. Der *sonst*-Zweig wird nur eingeschlagen, wenn eine Simulation von einem $y \notin L$ erfragt wird. Denn dort wird nach Definition vom Szenario $\perp$ ausgegeben. Für $y \in L$ wird der Beweiser in Szenario 1 stets einen Beweis ausgeben.

Definition 4.3.12 (Zufallsvariable für die Challenge)**.** *Sei m ein Transkript der Länge n, (y, w) eine Eingabe für eine Simulate Anfrage und α ein Commitment. Es wird der Wahrscheinlichkeitsraum unter der bedingten Wahrscheinlichkeit, dass das Transkript m erzeugt und für die darauf folgende Proof(y, w) Anfrage das Commitment α verwendet wird, betrachtet.*

Für $S1_{\textit{Proof},(y,w),m}$ sei $S1^{\alpha}_{(y,w),m}$ die Zufallsvariable, die die Challenge beschreibt, die in $S1$ für den Beweis gewählt wird. Das entspricht dem Schritt $\beta \leftarrow \mathcal{H}(\alpha, y)$ in der Definition der Strong Fiat-Shamir Transformation:

$$S1^{\alpha}_{(y,w),m}(\alpha', \beta') = \begin{cases} \beta & , S1_{\textit{Proof},(y,w),m}(\alpha', \beta') = (\alpha, \beta, \cdot) \\ \perp & , \textit{ sonst} \end{cases}$$

Definition 4.3.13 (Zufallsvariable für die Response)**.** *Sei m ein Transkript der Länge n, (y, w) eine Eingabe für eine Simulate Anfrage und $\alpha \in M_{\alpha}$ und $\beta \in M_{\beta}$. Es wird der Wahrscheinlichkeitsraum unter der bedingten Wahrscheinlichkeit, dass das Transkript m erzeugt und für die darauf folgende Proof(y, w) Anfrage das Commitment α und die Challenge β verwendet werden, betrachtet.*

Für $S1_{\textit{Proof},(y,w),m}$ sei $S1^{\alpha,\beta}_{(y,w),m}$ die Zufallsvariable, die die Response beschreibt, die in $S1$ für den Beweis gewählt wird. Das entspricht dem Schritt $\gamma \leftarrow \textbf{\textit{Prove}}_{\Sigma}(y, w, \alpha, \beta)$ in der Definition der Strong Fiat-Shamir Transformation:

$$S1^{\alpha,\beta}_{(y,w),m}(\alpha', \beta') = \begin{cases} \gamma & , S1_{\textit{Proof},(y,w),m}(\alpha', \beta') = (\alpha, \beta, \gamma) \\ \perp & , \textit{ sonst} \end{cases}$$

4.3.2.5 Unbenutze Hashwerte

Definition 4.3.14 (Gesetzte Hashwerte)**.** *Für eine Zufallsvariable $SX_{Proof,(y,w),m}$, d.h. einer Anfrage der Form Proof(y, w), sei $A_{y,m}$ die Menge aller Commitments $\alpha \in M_\alpha$, für die $\mathcal{H}(\alpha, y)$ unmittelbar vor der Anfrage gesetzt ist. Das bedeutet, im Random Oracle ist ein Paar $((\alpha, y), \cdot)$ gespeichert.*

Bemerkung. Für ein gegebenes m besteht $A_{y,m}$ für alle y stets aus den gleichen Elementen, ohne Abhängigkeit vom Szenario (für die zuvor beschriebenen Szenarien). Das liegt daran, dass aus m alle im Random Oracle gesesetzten Paare eindeutig hervorgehen.

Definition 4.3.15 (Indikatorvariable für Verwendung gesetzte Hashwerte)**.** *Sei m ein Transkript. Die Zufallsvariable B_m ist eine Indikatorvariable, die anzeigt, ob im Transkript ein Beweis (α, β, γ) für y mit $\alpha \in A_{y,m'}$ vorhanden ist. Dabei bezeichne m' das Transkript bis zu dieser Beweisanfrage.*

$$B_m = \begin{cases} 1 & , \exists(\textit{Proof}, (y, w), (\alpha, \beta, \gamma)) \in m : \alpha \in A_{y,m'} \\ 0 & , \textit{ sonst} \end{cases}$$

4.3.3 Beweis der Zero-Knowledge Eigenschaft

Lemma 4.10 (Größe von $A_{y,m}$). Für eine Proof$(y, \cdot)$ Anfrage mit vorausgegangener Kommunikation m gilt $|A_{y,m}| \leq |m|$.

Beweis. Sei $(X, q, r) \in m$ ein Transkript einer Anfrage. Dieses Transkript kann auf zwei verschiedene Weisen aufgebaut sein:

- $X = \mathcal{H}$. Hierbei handelt es sich um eine Hashanfrage für q mit Antwort r, das bedeutet, q ist der Hashwert r zugeordnet. Dieses Paar wird im Random Oracle gespeichert, sofern es nicht bereits dort vorhanden ist.
- $X =$ Proof. Hierbei handelt es sich um eine Simulationsanfrage für $q = (y, w)$. Sofern $(y, w) \in R$ gilt, ist $r = (\alpha, \beta, \cdot)$ ein Beweis für y (ansonsten wird $\perp$ ausgegeben). Auf Grund der Konstruktion der Strong Fiat-Shamir Transformation ist β der Hashwert von (α, y), d.h. das Paar $((\alpha, y), \beta)$ wird im Random Oracle gespeichert, sofern es nicht bereits dort vorhanden ist.

Damit kann jedes Element in m maximal einen Hashwert setzen. Daraus folgt die zu zeigende Aussage. □

Lemma 4.11. Die Ausgabe einer Beweisanfrage an Szenario 2 ist unabhängig von der vorausgegangenen Kommunikation, es gilt $\forall m \forall m' \forall \pi = (\alpha, \beta, \gamma) \forall y \forall w$:

$$Pr[S2_{\text{Proof},(y,w),m} = \pi] = Pr[S2_{\text{Proof},(y,w),m'} = \pi]$$

Beweis. Das Lemma gilt auf Grund der Konstruktion von Szenario 2. Dort wird eine Simulate(y) Anfrage ohne Einfluss vorausgegangener Kommunikation abgehandelt: Zunächst wählt der Simulator Challenge und Response gleichverteilt aus den jeweiligen Mengen. Anschließend wird mittels des Simulators aus der Special Honest Verifier Zero-Knowledge Eigenschaft ein Commitment bestimmt. Insbesondere wird kein Hashwert vom Random Oracle angefragt, sondern höchstens ein bereits gesetztes überschrieben.

Folglich ist eine Proof(y, w) Anfrage in Szenario 2 unabhängig von vorausgegangener Kommunikation. □

Lemma 4.12. Für Anfragen der Form Proof(y, w) gilt:

$$\forall m \forall \pi = (\alpha, \beta, \gamma) : Pr[S2_{\text{Proof},(y,w),m} = \pi] = Pr[S1_{\text{Proof},(y,w),\emptyset} = \pi]$$

Beweis. Für $(y, w) \notin R$ wird in beiden Fällen $\perp$ ausgegeben. Sei also $(y, w) \in R$ und sei (P, V) das zugrundeliegende Beweissystem. Es gilt wegen Lemma 4.11:

$$\forall m \forall \pi : Pr[S2_{\text{Proof},(y,w),m} = \pi] = Pr[S2_{\text{Proof},(y,w),\emptyset} = \pi]$$

Zunächst wählt der Simulator Challenge β und Response γ gleichverteilt aus den jeweiligen Mengen. Anschließend wird mittels des Simulators aus der Special Honest Verifier Zero-Knowledge Eigenschaft ein Commitment α bestimmt. Auf Grund der Konstruktion dieses Simulators ist der erbrachte Beweis (α, β, γ) identisch verteilt zu einer Ausführung von (P, V) auf Eingabe (y, w) (siehe Definition 2.3.4):

$$... = Pr[\langle P(w), V \rangle(y) = \pi]$$

Bei dieser Ausführung wählt der Verifizierer die Challenge gleichverteilt aus M_β, da es sich um ein Σ-Protokoll handelt. Dies ist äquivalent zu einem Orakelaufruf an ein Orakel $M_\alpha \times \Lambda \to M_\beta$, das noch keine Hashwerte gespeichert hat. Denn dort wird die Antwort ebenfalls gleichverteilt aus M_β gewählt. Dies entspricht einer Ausführung von Szenario 1 auf eine Proof(y, w) Anfrage ohne vorausgegangene Kommunikation. Dabei werden die Methoden des Beweisers der Reihe nach aufgerufen und die Challenge mittels des Random Oracles ermittelt. Es folgt:

$$... = Pr[S1_{\text{Proof},(y,w),\emptyset} = \pi]$$

□

Lemma 4.13 (Wahl des Commitments). Es gilt:

$$\forall (y,w) \in R \, \forall m \forall \alpha \in M_\alpha : Pr[S1_{(y,w),m} = \alpha] = \frac{1}{|M_\alpha|} = Pr[S2_{\text{Proof},(y,w),m} = (\alpha, \cdot, \cdot)]$$

Beweis. Der Beweis wird in zwei Teilen erbracht.

- $\forall (y,w) \in R \, \forall m \forall \alpha \in M_\alpha : Pr[S1_{(y,w),m} = \alpha] = \frac{1}{|M_\alpha|}$. In Szenario 1 wählt P das Commitment stets gleichverteilt aus M_α. Damit ist diese Aussage gezeigt.
- $\forall (y,w) \in R \, \forall m \forall \alpha \in M_\alpha : Pr[S2_{\text{Proof},(y,w),m}() = \alpha] = Pr[S1_{(y,w),m} = \alpha]$. Wegen Lemma 4.12 und dem bereits erbrachten Beweisstück gilt:

$$\begin{aligned} \forall (y,w) \in R \, \forall m \forall \alpha \in M_\alpha : Pr[S2_{\text{Proof},(y,w),m} = (\alpha, \cdot, \cdot)] \\ = Pr[S1_{\text{Proof},(y,w),\emptyset} = (\alpha, \cdot, \cdot)] \\ = Pr[S1_{(y,w),m} = \alpha] = \frac{1}{|M_\alpha|} \end{aligned}$$

□

Lemma 4.14 (Identische Verteilung der Hashwerte in beiden Szenarien). In beiden Szenarien ist die Ausgabe der Hashanfragen stets dieselbe, sofern die vorausgegangene Kommunikation identisch ist:

$$\forall x \forall m \forall y \in M_\beta : Pr[S1_{\mathcal{H},x,m} = y] = Pr[S2_{\mathcal{H},x,m} = y]$$

Beweis. Es gibt zwei Fälle zu unterscheiden.

- x ist bereits ein Hashwert zugeordnet. Beide Szenarien speichern zuvor gesetzte Hashwerte ab und geben diese konsistent zurück. Der Hashwert ist durch m eindeutig festgelegt. Folglich ist die Ausgabe die Gleiche.

- x ist noch kein Hashwert zugeordnet. Da beide Szenarien den Hashwert stets gleichverteilt aus M_β wählen, ohne vorausgegangene Kommunikation einzubeziehen, sind die Ausgaben identisch verteilt:

$$\forall x \text{ ohne bisher zugeordnetem Hashwert } \forall m \forall y \in M_\beta : \\ Pr[S1_{\mathcal{H},x,m} = y] = \frac{1}{|M_\beta|} = Pr[S2_{\mathcal{H},x,m} = y]$$

□

Lemma 4.15 (Identische Verteilung von Beweisen mit unbenutzten Commitments). Sei m ein Transkript und $(y, w) \in R$. Es gilt:

$$\forall \pi = (\alpha \notin A_{y,m}, \beta \in M_\beta, \gamma \in M_\gamma) : Pr[S1_{\text{Proof},(y,w),m} = \pi] = Pr[S2_{\text{Proof},(y,w),m} = \pi]$$

Beweis. Sei m ein Transkript und $(y, w) \in R$. Sei $\alpha \notin A_{y,m}$ für die Zufallsvariablen $SX_{\text{Proof},(y,w),m}$ mit $X \in \{1, 2\}$.

Nach Lemma 4.12 gilt:

$$Pr[S2_{\text{Proof},(y,w),m} = \pi] = Pr[S1_{\text{Proof},(y,w),\emptyset} = \pi]$$

womit zu zeigen ist:

$$\forall \pi = (\alpha, \beta, \gamma) \text{ mit } a \notin A_{y,m} : Pr[S1_{\text{Proof},(y,w),\emptyset} = \pi] = Pr[S1_{\text{Proof},(y,w),m} = \pi]$$

Es gilt:

- $\forall \alpha \in M_\alpha : Pr[S1_{(y,w),m} = \alpha] = Pr[S1_{(y,w),\emptyset} = \alpha]$, da das Commitment in Szenario 1 stets gleichverteilt aus M_α gewählt wird.
- Für $\alpha \notin A_{y,m}$ wird die Challenge vom Random Oracle gleichverteilt aus M_β gewählt, da nach Definition von $A_{y,m}$ kein Paar (α, y) gespeichert ist. Damit gilt:

$$\forall \alpha \notin A_{y,m} \forall \beta \in M_\beta : Pr[S1^\alpha_{(y,w),m} = \beta] = Pr[S1^\alpha_{(y,w),\emptyset} = \beta]$$

- Die Response wird von $S1$ mittels α, β und y, ohne Einfluss vorangegangener Kommunikation, bestimmt:

$$\forall\alpha \in M_\alpha \forall\beta \in M_\beta \forall\gamma \in M_\gamma : Pr[S1^{\alpha,\beta}_{(y,w),m} = \gamma] = Pr[S1^{\alpha,\beta}_{(y,w),\emptyset} = \gamma]$$

Da für jeden Schritt gilt, dass, unter der Voraussetzung, dass alle bisherigen Schritte die gleichen Ergebnisse geliefert haben, das nächste Ergebnis mit identischer Verteilung bestimmt wird, folgt die zu zeigende Aussage. □

Lemma 4.16 (Verwendung gesetzter Hashwerte in Szenario 1). Sei $\Sigma = (P, V)$ ein Σ-Protokoll mit einem im Sicherheitsparameter superpolynomiell großen Challengeraum und den Eigenschaften Special Soundness und Special Honest Verifier Zero-Knowledge. Zudem wählt P das Commitment gleichverteilt aus der gesamten Commitmentmenge. Sei U ein polynomialzeitbeschränkter Unterscheider.

Dann ist die Wahrscheinlichkeit, dass für ein Transkript m von U und Szenario 1 $B_m = 1$ gilt, vernachlässigbar.

Beweis. Sei U ein polynomialzeitbeschränkter Unterscheider mit Laufzeit $t_U(\eta)$ für den Sicherheitsparameter η und sei $m \xleftarrow{\$} \langle U, S1\rangle$.

Es gilt

$$Pr[B_m = 1] = Pr[\exists(\text{Proof}, (y, w), (\alpha, \beta, \gamma)) \in m : \alpha \in A_{y,m'}],$$

wobei m' der Anfang des Transkriptes m bis zur gegebenen Proof Anfrage bezeichne.

In Szenario 1 wird das Commitment für einen Beweis stets gleichverteilt aus M_α gewählt. Zudem können maximal $t_U(\eta)$ viele Proof Anfragen gestellt werden. Es folgt:

$$\ldots = \sum_{i=1}^{t_U(\eta)} \sum_{\substack{\pi=(\alpha,\cdot,\cdot),\\ \alpha\in A_{y_i,m_i}}} Pr[S1_{\text{Proof},(y_i,w_i),m_i} = \pi] = \sum_{i=1}^{t_U(\eta)} \sum_{\alpha\in A_{y_i,m_i}} Pr[S1_{(y_i,w_i),m_i} = \alpha].$$

Lemma 4.13 gibt an, wie wahrscheinlich ein Beweis mit einem bestimmten Commitment ist:

$$\ldots = \sum_{i=1}^{t_U(\eta)} \sum_{\alpha_i\in A_{y_i,m_i}} \frac{1}{|M_\alpha|} = \frac{1}{|M_\alpha|} \sum_{i=1}^{t_U(\eta)} \sum_{\alpha\in A_{y_i,m_i}} 1 = \frac{1}{|M_\alpha|} \sum_{i=1}^{t_U(\eta)} |A_{y_i,m_i}|$$

Die Mächtigkeit von A_{y_i,m_i} wird in Lemma 4.10 abgeschätzt:

$$\begin{aligned}
\ldots \leq \frac{1}{|M_\alpha|} \sum_{i=1}^{t_U(\eta)} i - 1 = \frac{1}{|M_\alpha|} \sum_{i=0}^{t_U(\eta)-1} i = \frac{1}{|M_\alpha|} \sum_{i=1}^{t_U(\eta)-1} i &= \frac{1}{|M_\alpha|} \frac{(t_U(\eta)-1)\cdot(t_U(\eta)-1+1)}{2} \\
&= \frac{1}{|M_\alpha|} \frac{(t_U(\eta)-1)\cdot t_U(\eta)}{2} \\
&= \frac{1}{|M_\alpha|} \frac{t_U(\eta)^2 - t_U(\eta)}{2} \\
&\leq \frac{t_U(\eta)^2}{2|M_\alpha|}
\end{aligned}$$

Da $t_U(\eta)^2$ polynomiell und $2|M_\alpha|$ superpolynomiell ist, folgt mittels Lemma 4.2, dass $\frac{t_U(\eta)^2}{2|M_\alpha|}$ vernachlässigbar ist. Aus $Pr[B_m = 1] \leq \frac{t_U(\eta)^2}{2|M_\alpha|}$ folgt damit die zu zeigende Aussage. □

Lemma 4.17 (Verwendung gesetzter Hashwerte in Szenario 2). Sei $\Sigma = (P, V)$ ein Σ-Protokoll mit einem im Sicherheitsparameter superpolynomiell großen Challengeraum und den Eigenschaften Special Soundness und Special Honest Verifier Zero-Knowledge. Zudem wählt P das Commitment gleichverteilt aus der gesamten Commitmentmenge. Sei U ein polynomialzeitbeschränkter Unterscheider.

Dann ist die Wahrscheinlichkeit, dass für ein Transkript m von U und Szenario 2 $B_m = 1$ gilt, vernachlässigbar.

Beweis. Sei U ein polynomialzeitbeschränkter Unterscheider mit Laufzeit $t_U(\eta)$ für den Sicherheitsparameter η und sei $m \xleftarrow{\$} \langle U, S2 \rangle$.

Es gilt

$$Pr[B_m = 1] = Pr[\exists(\text{Proof}, (y, w), (\alpha, \beta, \gamma)) \in m : \alpha \in A_{y,m'}],$$

Seien $(SX_{\text{Proof},(y_i,w_i),m_i})_{i\in\{1,\ldots,t_U(\eta)\}}$ alle Zufallsvariablen für sämtliche Proof$(\cdot)$ Anfragen von U. Auf Grund der Laufzeit von U können dies maximal $t_U(\eta)$ viele sein. Es ist zu zeigen:

$$\left(\sum_{i=1}^{t_U(\eta)} \sum_{\substack{\pi=(\alpha,\cdot,\cdot),\\ \alpha\in A_{y_i,m_i}}} Pr[S1_{\text{Proof},(y_i,w_i),m_i} = \pi] \right) \text{ ist vernachlässigbar.}$$

Es werden nur Beweise mit einem bestimmten Commitment betrachtet. Weitere Eigenschaften der Beweise werden nicht benötigt. Daher kann Lemma 4.13 angewendet werden, um zu zeigen, dass die obige Formel für beide Szenarien identisch ist:

$$\sum_{i=1}^{t_U(\eta)} \sum_{\substack{\pi=(\alpha,\cdot,\cdot),\\ \alpha\in A_{y_i,m_i}}} Pr[S2_{\text{Proof},(y_i,w_i),m_i} = \pi] = \sum_{i=1}^{t_U(\eta)} \sum_{\substack{\pi=(\alpha,\cdot,\cdot),\\ \alpha\in A_{y_i,m_i}}} Pr[S1_{\text{Proof},(y_i,w_i),m_i} = \pi],$$

wobei Letzteres gemäß des Beweises von Lemma 4.16 vernachlässigbar ist. Daraus folgt die zu zeigende Aussage. □

Um die Zero-Knowledge Eigenschaft zu beweisen muss gezeigt werden, dass ein Simulator gemäß Definition 2.2.10 existiert.

Der Simulator kann über zwei Methoden aufgerufen werden:

- $\mathcal{H}(s)$, eine Hashanfrage
- Prove$(y \in \Lambda, w)$, eine Anfrage für einen Beweis π von y mit Verify$(y, \pi) = 1$.

Dieser Simulator muss sich so verhalten, dass nicht unterschieden werden kann, ob die Anfragen vom Simulator oder von einem echten Beweiser beantwortet werden.

Es wird vorausgesetzt, dass der Beweiser das Commitment gleichverteilt wählt.

Theorem 5. *Sei $\Sigma = (P, V)$ ein Σ-Protokoll mit einem im Sicherheitsparameter superpolynomiell großen Challengeraum und den Eigenschaften Special Soundness und Special Honest Verifier Zero-Knowledge. Zudem wählt P das Commitment gleichverteilt aus der gesamten Commitmentmenge. Dann ist sFS(Σ) Zero-Knowledge.*

Beweis. Es ist zu zeigen:

$$\forall U \exists v : |Pr[\langle U, S1\rangle(1^\eta) = 1] - Pr[\langle U, S2\rangle(1^\eta) = 1] \leq v(\eta),$$

wobei U polynomialzeitbeschränkt und v eine vernachlässigbare Funktion ist.

Im Folgenden werden alle erfolgreichen Ausführungen betrachtet. Diese Ausführungen erzeugen jeweils ein Transkript:

$$Pr[\langle U, S1\rangle(1^\eta) = 1] = \sum_m Pr[\langle U, S1\rangle(1^\eta) = 1 | E_m] \cdot Pr[E_m].$$

Die Transkripte lassen sich nach der Indikatorvariable B_m sortieren:

$$... = \sum_{m:B_m=1} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m] + \sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m],$$

also in jene Ausführungen, bei denen für eine Beweisanfrage ein Commitment genutzt wird, das bereits zuvor gesetzt worden ist, und in Ausführungen, in denen dies nicht auftritt. Ersterer Fall ist gemäß Lemma 4.16 vernachlässigbar, weshalb eine vernachlässigbare Funktion $v_{S1}^{B=1}(\eta)$ existiert, sodass gilt:

$$... \leq v_{S1}^{B=1}(\eta) + \sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m].$$

Nun wird die Summe für $B_m = 0$ weiter untersucht.

Dafür lässt sich das Transkript in die einzelnen Anfragen unterteilen. Dafür bezeichnen im folgenden die Transkripte $m_1, ..., m_{|m|}$ die Aufteilung des akzeptierten Transkriptes m dahingehend, dass m_{i+1} das Transkript m_i um eine Anfrage erweitert. Das Transkript m_i ist das Transkript der ersten Anfrage und $m_{|m|} = m$ ist das gesamte Transkript.

$$\begin{aligned}
&\sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m] \\
&= \sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m|E_{m_{|m|-1}}] \cdot Pr[E_{m_{|m|-1}}] \\
&= ... \\
&= \sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m|E_{m_{|m|-1}}] \cdot Pr[E_{m_{|m|-1}}] \cdot ... \cdot Pr[E_{m_2}|E_{m_1}] \cdot Pr[E_{m_1}]
\end{aligned}$$

Nun wird gezeigt, dass jede einzelne Anfrage in den beiden Szenarien die gleiche Wahrscheinlichkeitsverteilung besitzt. Sei (X, q, r) das Transkript der i-ten Anfrage. Es wird eine Fallunterscheidung nach der angefragten Methode durchgeführt.

- $X = \mathcal{H}$. Nach Lemma 4.14 gilt:

$$\forall x \forall m_{i-1} \forall y \in M_\beta : Pr[S1_{\mathcal{H},x,m_{i-1}} = y] = Pr[S2_{\mathcal{H},x,m_{i-1}} = y]$$

- $X =$ Proof. Wegen $B_m = 0$ werden nur Beweise $\pi = (\alpha, \cdot, \cdot)$ mit $\alpha \notin A_{y,m_i}$ betrachtet. Für diese gilt mittels Lemma 4.15:

$$\forall m_{i-1} : Pr[S1_{\text{Proof},(y,w),m_{i-1}} = \pi] = Pr[S2_{\text{Proof},(y,w),m_{i-1}} = \pi]$$

Damit können die einzelnen Transkripte der Anfragen von Szenario 1 in Szenario 2 überführt werden, ohne dass sich die Wahrscheinlichkeiten ändern:

$$\begin{aligned}
&\sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m|E_{m_{|m|-1}}] \cdot Pr[E_{m_{|m|-1}}] \cdot ... \cdot Pr[E_{m_2}|E_{m_1}] \cdot Pr[E_{m_1}] \\
&= \sum_{m:B_m=0} Pr[\langle U, S2\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m|E_{m_{|m|-1}}] \cdot Pr[E_{m_{|m|-1}}] \cdot ... \cdot Pr[E_{m_2}|E_{m_1}] \cdot Pr[E_{m_1}]
\end{aligned}$$

Damit gilt insgesamt:

$$\begin{aligned}
Pr[\langle U, S1\rangle(1^\eta) = 1] &\leq v_{S1}^{B=1}(\eta) + \sum_{m:B_m=0} Pr[\langle U, S1\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m] \\
&= v_{S1}^{B=1}(\eta) + \sum_{m:B_m=0} Pr[\langle U, S2\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m]
\end{aligned}$$

Die Erfolgswahrscheinlichkeit in Szenario 2 kann wie folgt aufgeschrieben werden:

$$\begin{aligned}
&Pr[\langle U, S2\rangle(1^\eta) = 1] \\
&= \sum_{m:B_m=1} Pr[\langle U, S2\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m] + \sum_{m:B_m=0} Pr[\langle U, S2\rangle(1^\eta) = 1|E_m] \cdot Pr[E_m],
\end{aligned}$$

wobei die erste Summe gemäß Lemma 4.16 vernachlässigbar ist, also durch eine vernachlässigbare Funktion $v_{S2}^{B=1}(\eta)$ nach oben hin beschränkt ist.

Dies ermöglicht es, die obige Formel umzuschreiben:

$$Pr[\langle U, S1\rangle(1^\eta) = 1] \leq v_{S1}^{B=1}(\eta) + Pr[\langle U, S2\rangle(1^\eta) = 1] - v_{S2}^{B=1}(\eta)$$

Daraus folgt:

$$|Pr[\langle U, S1\rangle(1^\eta) = 1] - Pr[\langle U, S2\rangle(1^\eta) = 1]| \leq |v_{S1}^{B=1}(\eta) + v_{S2}^{B=1}(\eta)|,$$

wobei der rechte Term vernachlässigbar ist. Folglich ist das Prootkoll Zero-Knowledge gegenüber polynomialzeitbeschränkten Angreifern und damit mittles Theorem 2 Zero-Knowledge gegenüber schwach polynomialzeitbeschränkten Angreifern. □

4.4 Simulation Sound Extractable

Für die Simulation Sound Extractability Eigenschaft ist zu zeigen, dass ein Extraktor existiert, der mit nicht-vernachlässigbarer Wahrscheinlichkeit das Sicherheitsspiel in Definition 2.2.12 gewinnt.

Es gibt zwei Möglichkeiten, das Sicherheitsspiel gegen den Extraktor aus Definition 3.2.1 zu gewinnen:

1. Der Angreifer gibt einen Beweis $\pi = (\alpha, \beta, \gamma)$ für ein Wort y mit den folgenden Eigenschaften aus:

   - π ist nicht Ausgabe einer **Simulate**(y) Anfrage
   - $V^{\mathcal{H}}(y, \pi) = 1$
   - Es wurde keine Orakelanfrage (α, y) gestellt

   In diesem Fall gibt K auf Grund seiner Konstruktion, wegen der letzten Bedingung, auf.

2. Der Angreifer gibt einen Beweis aus, zu dem der Extraktor keinen Zeugen extrahieren kann.

Das folgende Lemma zeigt, dass die Wahrscheinlichkeit, über den ersten Fall zu gewinnen, vernachlässigbar ist.

Lemma 4.18. Gibt ein bösartiger Beweiser einen Beweis (α, β, γ) für ein Wort y aus und hat keine Orakelanfrage (α, y) gestellt, dann ist die Wahrscheinlichkeit, dass der Beweis verifiziert wird, vernachlässigbar.

Beweis. Sei (y, π) ein Paar aus Wort y und einem Beweis $\pi = (\alpha, \beta, \gamma)$, das ein bösartiger Beweiser ausgegeben hat, ohne eine Orakelanfrage (α, y) gestellt zu haben.

Der Verifizierer prüft nach Konstruktion der Strong Fiat-Shamir Transformation, ob das Orakel auf die Frage (α, y) mit β antwortet.

Da jedoch keine Orakelanfrage (α, y) im Transkript existiert, wurde dieser Hashwert erst während der Verifizierung durch $V^{\mathcal{H}}$ festgelegt.

Es gilt

$$Pr[\mathcal{H}(\alpha, y) = \beta] = \frac{1}{|M_\beta|},$$

da die Ausgabe vom Random Oracel zufällig gleichverteilt gewählt wird. Da M_β gemäß des zu zeigenden Theorems superpolynomiell groß in Bezug auf den Sicherheitsparameter ist, ist diese Wahrscheinlichkeit vernachlässigbar.

□

Im Folgenden wird die zweite Gewinnmöglichkeit untersucht.

4.4.1 Angreifer, die maximal einen Beweis ausgeben

Zunächst werden Angreifer betrachtet, die maximal einen Beweis ausgeben.

Lemma 4.19. Sei A ein Angreifer, der maximal einen Beweis ausgibt. Sei $p_A(\eta)$ die Laufzeit von A. Sei $succ$ die Wahrscheinlichkeit, dass der Extraktor K das Sicherheitsspiel gegenüber A gewinnt.

Dann gilt

$$succ \geq \frac{1}{2 \cdot p_A(\eta)}.$$

Beweis. Sei $p_A(\eta)$ die Laufzeit des Angreifers A und $q(\eta)$ die Laufzeit des Simulators.

Algorithmus 4.2 zeigt eine Ausführung des Sicherheitsspiels für den Extraktor K aus Definition 3.2.1:

- Die Zeilen 1 bis 6 entsprechen dem **Initial Run**.
  - In den ersten beiden Zeilen werden die Random Bits für den Angreifer, das Random Oracle und den Simulator festgelegt.
  - In Zeile drei wird der Angreifer für den Initialen Run ausgeführt.
  - Die Zeilen 4 bis 18 von A' nehmen die Überprüfungen des Initialen Run vor: In Zeile 4 wird geprüft, ob ein Beweis ausgegeben wurde. In Zeile 7 wird überprüft, ob der erbrachte Beweis eine Ausgabe einer **Simulate** Anfrage ist. In Zeile 10 wird überprüft, ob der erbrachte Beweis verifiziert wird. In Zeile 13 wird überprüft, ob die Challenge mittels einer Anfrage an das Random Oracle bestimmt wurde.

    Schlägt eine dieser Prüfungen fehl, wird das Spiel sofort beendet. In diesem Fall wird im Algorithmus $p_A(\eta) + 1$ als Akzeptanz für das Forking Lemma ausgegeben. In allen Fällen, bis auf den letzten, hat der Extrator das Spiel direkt gewonnen. Im letzten Fall bricht der Extraktor auf Grund seiner Konstruktion ab. Gemäß Lemma 4.18 geschieht dies mit vernachlässigbarer Wahrscheinlichkeit. Damit gewinnt der Extraktor mit überwältigender Wahrscheinlichkeit, wenn im Initialen Run $p_A(\eta) + 1$ als Akzeptanz ausgegeben wird.

Algorithmus 4.2 Sicherheitsspiel mit Extraktor für einen Angreifer, der maximal einen Beweis ausgibt

Input : Securityparameter 1^η

1 $h_1, ..., h_{p_A(\eta)+1} \xleftarrow{\$} M_\beta$
2 Select Random Bits $\alpha' = \alpha, s_1, ..., s_{p_A(\eta)} \xleftarrow{\$} \{0,1\}^{p_A(\eta)} \times \{0,1\}^{q(\eta)} \times ... \times \{0,1\}^{q(\eta)}$
3 $(j, \pi) \leftarrow A'(1^\eta, h_1, ..., h_{p_A(\eta)+1}; \alpha')$
4 **if** $j = 0$ **then**
5 | return $(0, \epsilon, \epsilon)$
6 **end**
7 $h'_j, ..., h'_{p_A(\eta)+1} \xleftarrow{\$} M_\beta$
8 $(j', \pi') \leftarrow A'(1^\eta, h_1, ..., h_{j-1}, h'_j, ..., h'_{p_A(\eta)+1}; \alpha')$
9 **if** $j = j'$ *and* $h_j \neq h'_j$ **then**
10 | return $(1, \pi, \pi')$
11 **else**
12 | return $(0, \epsilon, \epsilon)$
13 **end**

A'**:**

1 $\alpha, s_1, ..., s_{p_A(\eta)} \leftarrow \alpha'$
2 $(y, \pi = (\alpha, \beta, \gamma)) \leftarrow A(\mathcal{H}'_{h_1,...,h_{p_A(\eta)+1}}, S_{s_1,...,s_{p_A(\eta)}}, 1^\eta; \alpha)$
3 $M =$ hashqueries with answers $(q_j, r_j)_{j \in \{1,...,n\}}$ from transkript
4 **if** $V^{\mathcal{H}'_{h_1,...,h_{p_A(\eta)+1}}}(y, \pi) = 0$ **then**
5 | return $(p_A(\eta) + 1, \epsilon)$
6 **end**
7 **if** $\exists$***Simulate***(y) *query with response* π *in transkript* **then**
8 | return $(p_A(\eta) + 1, \epsilon)$
9 **end**
10 **if** $((\alpha, y), \beta) \in M$ **then**
11 | $j \leftarrow \min_j : (q_j, r_j) \in M = ((\alpha, y), \beta)$
12 | return (j, π)
13 **else**
14 | return $(p_A(\eta) + 1, \epsilon)$
15 **end**

- Die Zeilen 7 bis 13 entsprechen dem **Extraction** Schritt. Der Angreifer wird mit den gleichen Random Bits wie im Initialen Run gestartet, die Hashwerte werden ab inklusive der Challenge des Beweises neu gewürfelt.

Dieser Aufbau entspricht ebenfalls dem Aufbau des Forking Lemmas. Die Wahl von j entspricht der Wahl der Orakelanfrage, dessen Antwort als Challenge gewählt wird. Daraus folgt mittels Lemma 2.1:

$$frk(\eta) \geq acc(\eta) \cdot \left(\frac{acc(\eta)}{p_A(\eta)} - \frac{1}{|M_\beta|} \right).$$

Auf Grund der Konstruktion des Algorithmus gilt $acc(\eta) = 1$. Damit folgt:

$$frk(\eta) \geq \left(\frac{1}{p_A(\eta)} - \frac{1}{|M_\beta|} \right). \tag{4.1}$$

Gemäß der Definition der Strong Fiat-Shamir Transformation ist eine Orakelanfrage der Form (α, y), wobei α ein mögliches Commitment und y ein zu beweisendes Wort ist. Folglich kann die Antwort dieser Orakelanfrage nur genutzt werden, um y zu beweisen. Verwendet der Angreifer die gleiche Orakelanfrage für die Beweise, existieren zwei Beweise für y, die beide verifiziert worden sind. Damit ein Zeuge mittels der Special Soundness extrahiert werden kann, müssen die Challenges der Beweise verschieden sein. Tritt das Ereignis zu frk mit $j < p_A(\eta) + 1$ ein, sind die Challenges nach Definition des Forking Lemmas verschieden.

Sei $succ(\eta)$ die Wahrscheinlichkeit, dass der Extraktor K das Sicherheitsspiel gegenüber A gewinnt. Dann gilt:

$$succ(\eta) \geq frk(\eta) - v(\eta), \tag{4.2}$$

wobei $v(\eta)$ eine vernachlässigbare Funktion ist. Tritt das zu frk gehörende Ereignis ein, gibt es zwei verschiedene Möglichkeiten:

1. Es gilt $j < p_A(\eta) + 1$. In diesem Fall gewinnt K wie oben gezeigt.
2. Es gilt $j = p_A(\eta) + 1$.In diesem Fall gewinnt der Extraktor, es sei denn, der Angreifer kann einen Beweis (α, β, γ) für ein Wort y ausgeben, ohne eine Orakelanfrage (α, y) zu stellen. Dies ist gemäß Lemma 4.18 nur mit vernachlässigbarer Wahrscheinlichkeit $v(\eta)$ möglich.

Aus diesen beiden Punkten folgt Formel 4.2. In diese kann Formel 4.1 eingesetzt werden:

$$succ(\eta) \geq \frac{1}{p_A(\eta)} - \left(\frac{1}{|M_\beta|} + v(\eta) \right)$$

Wobei $\frac{1}{|M_\beta|} + v(\eta)$ vernachlässigbar ist. Damit existiert ein n_0 sodass für alle $\eta > n_0$ gilt:

$$\begin{aligned} succ(\eta) &\geq \frac{1}{p_A(\eta)} - \frac{1}{2p_A(\eta)} \\ &= \frac{1}{2 \cdot p_A(\eta)} \end{aligned}$$

□

Bemerkung. Dies gilt nur für die Strong, nicht für Weak Fiat-Shamir Transformation, da dort nur das Commitment, nicht aber das zu beweisende Wort als Orakelanfrage verwendet wird.

Lemma 4.20. Der Extraktor K aus Definition 3.2.1 gewinnt das Sicherheitsspiel gegenüber polynomialzeitbeschränkten Angreifern, die maximal einen Beweis ausgeben, mit nicht-vernachlässigbarer Wahrscheinlichkeit.

Beweis. Folgt direkt aus dem vorherigen Lemma, da $\frac{1}{2p_A(\eta)}$, mit $p_A(\eta)$ als Laufzeit von A, nicht-vernachläsigbar ist. □

4.4.2 Verbesserung der Erfolgswahrscheinlichkeit

Wie zuvor gezeigt, besitzt der Extraktor K, der im Paper beschrieben wird, eine nicht-vernachlässigbare Erfolgswahrscheinlichkeit im Sicherheitsspiel der Simulation Sound Extractability. Im Folgenden wird untersucht, ob die Gewinnwahrscheinlichkeit erhöht werden kann. Es wird gezeigt, dass ein Extraktor existiert, der mit überwältigender Wahrscheinlichkeit gewinnt.

Eine Möglichkeit, die Erfolgswahrscheinlichkeit zu erhöhen, besteht darin, den Extraktor K wiederholt auszuführen. In Algorithmus Algorithmus 4.3 ist ein Beispiel für einen solchen Extraktor gegeben.

Algorithmus 4.3 Verbesserter Extraktor K'

Input: Securityparameter 1^η, (y, π)
for *i = 0; i <* $(2\eta)^2$*; i++* **do**
  $w \xleftarrow{\$} K(1^\eta, (y, \pi))$
  if K *did not abort and outpus a witness* **then**
    output (y, w)
  end
end
abort

Wie hoch ist die Erfolgswahrscheinlichkeit des verbesserten Extraktors?

Lemma 4.21. Der Extraktor aus Algorithmus 4.3 gewinnt das Sicherheitsspiel der Simulation Sound Extractabilty mit überwältigender Wahrscheinlichkeit.

Beweis. Die Erfolgswahrscheinlichkeit für eine Ausführung des ursprünglichen Extraktors beträgt gemäß Lemma 4.19 mindestens $\frac{1}{2p_A(\eta)}$. Der Extraktor erhält gemäß der Definition des Sicherheitsspiels als Sicherheitsparameter die Anzahl an Schritte, die der Angreifer im Initalen Schritt verwendet hat. Unter der Annahme, dass die Schleifendurchläufe voneinander unabhängig sind, gilt somit

$$\begin{aligned} Pr[SSE_{A,K'} = 0] &\leq \left(1 - \frac{1}{2p_A(\eta)}\right)^{(2p_A(\eta))^2} \\ &= \left(1 - \frac{1}{2p_A(\eta)}\right)^{2p_A(\eta)\cdot 2p_A(\eta)} \\ &= \left(\left(1 - \frac{1}{2p_A(\eta)}\right)^{2p_A(\eta)}\right)^{2p_A(\eta)} \\ &= \left(\frac{1}{e}\right)^{2p_A(\eta)} \end{aligned}$$

Damit ist die Wahrscheinlichkeit, dass der verbesserte Extraktor nicht erfolgreich ist, vernachlässigbar. Folglich ist die Erfolgswahrscheinlichkeit überwältigend.

Die Voraussetzung, dass die Schleifendurchläufe voneinander unabhängig sind, trifft zu, da jeder Schleifendurchlauf für sich betrachtet einer Ausführung des Extraktors K entspricht.

□

4.5 Korrektheit

Definition 4.5.1 (Modifizierter Beweiser)**.** *Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem $(P^{\mathcal{H}}, V^{\mathcal{H}})$ mit Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$. Für eine Ausführung von $B^{\mathcal{H}}(1^\eta, y)$ sei der modifizierte Beweiser $B_y^{\mathcal{H}}(1^\eta)$ durch Algorithmus 4.4 gegeben.*

Bemerkung. Bei dem modifizierten Beweiser geht es im Wesentlichen darum, den Beweiser in eine Form zu bringen, die dem Sicherheitsspiel der Simulation Sound Extractability Eigenschaft entspricht.

Bemerkung. Der modifizierte Beweiser $B_y^{\mathcal{H}}$ verhält sich äquivalent wie der ursprüngliche Beweiser, gestartet auf Eingabe y.

Algorithmus 4.4 Modifizierter Beweiser B_y

Input: 1^η
1 return $B(1^\eta, y)$

Definition 4.5.2 (Akzeptanzwahrscheinlichkeit)**.** *Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$.*

Die Wahrscheinlichkeit $acc(\eta)$ stehe für das Ereignis, dass $B_y(\eta)$ einen Beweis $\pi = (\alpha, \beta, \gamma)$ für y ausgibt und eine Orakelanfrage (α, y) gestellt hat, dessen Antwort β ist.

Lemma 4.22. Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$ und sei y ein Wort.

Die Wahrscheinlichkeit, dass $B_y(\eta)$ einen Beweis π mit $V(1^\eta, \pi) = 1$ ausgibt, ohne dass das zu $acc(1^\eta)$ gehörende Ereignis eingetreten ist, ist vernachlässigbar.

Beweis. Folgt aus Lemma 4.18. □

Definition 4.5.3 (Extraktionswahrscheinlichkeit)**.** *Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$.*

Im Folgenden wird eine Ausführung des modifizierten Beweisers im Sicherheitsspiel der Simulation Sound Extractability betrachtet.

Die Wahrscheinlichkeit $frk(\eta)$ stehe für das Ereignis, dass $B_y(\eta)$ im Initialen Run des Sicherheitsspiel einen Beweis $\pi = (\alpha, \beta, \gamma)$ für y ausgibt und der Extraktor K aus Definition 3.2.1 im Extraktionsschritt einen Zeugen für y aus $B_y^{\mathcal{H}}(1^\eta)$ extrahieren kann.

Lemma 4.23. Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$.

Gilt $frk(\eta) > 0$, dann ist $y \in L_R$.

Beweis. Wenn die Wahrscheinlichkeit, dass ein Extraktor einen Zeugen w für ein Wort y mit $(y, w) \in R$ extrahieren kann, echt größer null ist, dann existiert solch ein Zeuge w mit $(y, w) \in R$. Damit gilt $y \in L_R$. □

Lemma 4.24. Sei $B^{\mathcal{H}}$ ein Beweiser für ein nicht-interaktives Beweissystem mit Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$. Sei $t(\eta)$ die maximale Laufzeit von $B_y(1^\eta)$.

Gilt $frk(\eta) = 0$, dann ist $acc(\eta) \leq \frac{t(\eta)}{|M_\beta|}$.

Beweis. Gemäß Lemma 4.19 gilt

$$frk(\eta) \geq acc(\eta) \cdot \left(\frac{acc(\eta)}{t(\eta)} - \frac{1}{|M_\beta|} \right).$$

Sei $frk(\eta) = 0$. Es folgt:

$$\begin{aligned} 0 &\geq acc(\eta) \cdot \left(\frac{acc(\eta)}{t(\eta)} - \frac{1}{|M_\beta|} \right) \\ \Leftrightarrow 0 &\geq \frac{acc(\eta)}{t(\eta)} - \frac{1}{|M_\beta|} \\ \Leftrightarrow \frac{1}{|M_\beta|} &\geq \frac{acc(\eta)}{t(\eta)} \\ \Leftrightarrow \frac{t(\eta)}{|M_\beta|} &\geq acc(\eta) \end{aligned}$$

□

Lemma 4.25. Sei $B^{\mathcal{H}}(1^\eta)$ ein Beweiser für ein nicht-interaktives Beweissystem mit maximaler Laufzeit $t(\eta, y)$, Hashorakel $\mathcal{H}$ und sei $y \in \Lambda$.

Gilt

$$acc(\eta) > \frac{t(\eta, y)}{|M_\beta|},$$

dann ist $y \in L_R$.

Beweis. Im Sicherheitsspiel der Simulation Sound Extractability gilt, dass acc gößer gleich $\frac{t(\eta)}{|M_\beta|}$ ist. Damit kann wegen Lemma 4.24 nicht $frk(\eta) = 0$ gelten. Draus folgt mittels Lemma 4.23 $y \in L_R$. □

Lemma 4.26. Sei $B^{\mathcal{H}}(1^\eta)$ ein polynomialzeitbeschränkter Beweiser für ein nicht-interaktives Beweissystem mit maximaler Laufzeit $t(\eta, y)$, Hashorakel $\mathcal{H}$ und sei $y \notin L$.

Dann ist $acc(\eta)$ vernachlässigbar.

Beweis. Wegen $y \notin L$ und Lemma 4.25 gilt

$$acc(\eta) \leq \frac{t(\eta, y)}{|M_\beta|}.$$

Da $t(\eta, y)$ polynomiell und $|M_\beta|$ superpolynomiell ist, folgt mittels Lemma 4.2, dass $acc(\eta)$ vernachlässigbar ist.

□

Lemma 4.27. Sei Σ ein Σ-Protokoll und sei $(P, V) = \text{sFS}(\Sigma)$. Dann ist (P, V) korrekt in Bezug auf polynomialzeitbeschränkte Beweiser.

Beweis. Sei $B^{\mathcal{H}}$ ein beliebiger, polynomialzeitbeschränkter Algorithmus, der als Beweiser für (P, V) verwendet werden kann.

Der Beweiser kann auf zwei verschiedene Weisen einen Beweis für y ausgeben:

1. B gibt einen Beweis aus, ohne dass das zu acc gehörige Ereignis eintritt. Die Wahrscheinlichkeit $w(\eta)$ für diesen Fall ist gemäß Lemma 4.22 vernachlässigbar.
2. B gibt einen Beweis aus und bei dieser Ausführung tritt das Ereignis für acc ein. Die Wahrscheinlichkeit dafür, $w'(\eta)$, ist gemäß Lemma 4.26 vernachlässigbar.

Daraus folgt

$$Pr[V(1^\eta, y, B(1^\eta, y)) = 1] \leq w(\eta) + w'(\eta),$$

wobei $w(\eta) + w'(\eta)$ vernachlässigbar ist.

□

5 Zusammenfassung und Ausblick

Diese Arbeit stellt die Fiat-Shamir Transformation vor. Diese ermöglicht es, Σ-Protokolle in nicht-interaktive Zero-Knowledge Beweise von Wissen umzuwandeln.

Die Idee der Transformation besteht darin, dass Beweiser und Verifizierer Zugriff auf eine Hashfunktion haben. Damit kann der Beweiser die Challenge bestimmen, indem er den Hashwert einer geeigneten Eingabe für die Hashfunktion bestimmt.

Während die schwache Fiat-Shamir Transformation nur das Commitment als Eingabe für die Hashfunktion verwendet, wird bei der starken Fiat-Shamir Transformation die Kombination aus zu beweisendem Wort und Commitment gehasht. Während die erste Variante unsicher ist, ist die Zweite sicher.

In dieser Arbeit wurde der Beweis geführt, dass die starke Fiat-Shamir Transformation die Eigenschaften Vollständigkeit, Zero-Knowledge, Simulation Sound Extractability und Korrektheit aufweist.

Die Simulation Sound Extractability konnte nur gegenüber Angreifern gezeigt werden, die maximal einen Beweis ausgeben. Zukünftige Forschung kann die Sicherheit auf beliebige Beweiser ausdehnen.

Die Eigenschaft Korrektheit konnte nur für Angreifer gezeigt werden, die mit überwältigender Wahrscheinlichkeit polynomialzeitbeschränkt sind. Diese Eigenschaft lässt sich eventuell auf allgemeine Angreifer ausweiten.

Literaturverzeichnis

[BN06] M. Bellare, G. Neven. „Multi-signatures in the plain public-key model and a general forking lemma“. In: *Proceedings of the 13th ACM conference on Computer and communications security*. ACM. 2006, S. 390–399 (zitiert auf S. 20).

[BPW12] D. Bernhard, O. Pereira, B. Warinschi. „How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios“. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012, S. 626–643 (zitiert auf S. 21, 30, 33).

[Cor+17] V. Cortier, C. C. Drăgan, F. Dupressoir, B. Schmidt, P.-Y. Strub, B. Warinschi. „Machine-Checked Proofs of Privacy for Electronic Voting Protocols“. In: *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE. 2017, S. 993–1008 (zitiert auf S. 21).

[GN12] S. Galbraith, M. Nandi. *Progress in Cryptology-INDOCRYPT 2012: 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011, Proceedings13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012, Proceedings*. Bd. 7668. Springer Science & Business Media, 2012 (zitiert auf S. 3).

[HL10] C. Hazay, Y. Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010 (zitiert auf S. 3).

[Qui+89] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, S. Guillou. „How to explain zero-knowledge protocols to your children“. In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, S. 628–631 (zitiert auf S. 11).

[WS12] X. Wang, K. Sako. *Advances in Cryptology–ASiACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012, Proceedings*. Bd. 7658. Springer Science & Business Media, 2012 (zitiert auf S. 3).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift