Fairer Austausch digitaler Unterschriften

Marc Fischlin

Diplomarbeit
am Fachbereich Informatik
Johann-Wolfgang-Goethe-Universität
Frankfurt am Main

Betreuer: Prof. Dr. C. P. Schnorr

16. Februar 1998

Fairer Austausch digitaler Unterschriften

Zusammenfassung

A und B möchten digitale Unterschriften auf faire Weise austauschen, d.h. A soll genau dann eine Unterschrift von B erhalten, wenn B eine Unterschrift von A erhält. Der triviale Ansatz zum Austausch zweier Unterschriften, daß A seine Unterschrift an B sendet und dann B seine Unterschrift an A schickt, ist nicht fair, da B nach Erhalt der Unterschrift von A das Protokoll vorzeitig beenden oder eine ungültige Unterschrift senden kann. Bei den bekannten praktikablen Protokollen zum fairen Austausch unterteilen die Teilnehmer die Unterschriften in kleine Blöcke aus wenigen Bits und tauschen die Blöcke dann schrittweise aus. Diese Protokolle garantieren einerseits, daß man sofort überprüfen kann, ob ein erhaltener Block korrekt ist. Andererseits geben die bereits ausgetauschten Blöcke so wenig wie möglich über den restlichen Teil der Unterschrift preis. Versucht in diesem Fall ein Teilnehmer zu betrügen, indem er beispielsweise einen falschen Wert sendet, so kann der andere Teilnehmer dies unmittelbar bemerken und stoppen. Da die noch nicht ausgetauschten Blöcke fast nichts über den übrigen Teil der Unterschrift preisgeben, hat der Betrüger höchstens einen Block mehr als der ehrliche Teilnehmer erhalten. Ist die Blockgröße hinreichend klein, kann der ehrliche Teilnehmer den Nachteil durch Raten bzw. Probieren ausgleichen.

In dieser Diplomarbeit entwickeln wir Protokolle zum fairen Austausch sogenannter Diskreter-Logarithmus-Unterschriften. Die bekannten praktikablen Protokolle zum Austausch dieses Unterschriftentyps verwenden als Sicherheitsvoraussetzung die Faktorisierungsannahme. Im Unterschied dazu beruht die Sicherheit unseres Austauschprotokolls auf der Diskreten-Logarithmus-Annahme und damit auf der des Unterschriftenverfahrens. Ferner erlauben unsere Protokoll die Herausgabe der Blöcke in beliebiger, auch vom Protokollverlauf abhängiger Reihenfolge, während die Reihenfolge bei den bisherigen Protokollen von vornherein festgelegt ist.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen Hilfsmittel als die angegebenen Quellen verwendet habe.

Marc Fischlin Frankfurt am Main, den 16. Februar 1998

Danksagung

Für wissenschaftliche Hilfe und Diskussionen bedanke ich mich bei Claus-Peter Schnorr. Ein besonderer Dank gilt Georg Schnitger für ewige Geduld hinsichtlich des Themas "Informatik-Diplomarbeit". Für "nicht-wissenschaftliche" Unterstützung bedanke ich mich bei meiner Familie und Stefanie Steidl.

Inhaltsverzeichnis

Kapitel 1

Einleitung

A und B möchten ihre Geheimnisse x und y auf faire Weise austauschen, d.h. A soll genau dann y erhalten, wenn umgekehrt B das Geheimnis x von A lernt. Bei Anwesenheit eines Notars T, dem sowohl A als auch B vertrauen, kann man dies durch ein triviales Protokoll erreichen. Dazu senden A und B ihre Geheimnisse an T, der das jeweilige Geheimnis dann und nur dann an den anderen Teilnehmer weiterreicht, wenn er beide Geheimnisse erhalten hat, und wenn beide Geheimnisse korrekt sind. Die Korrektheit der Geheimnisse ist durch ein öffentliches Prädikat verifizierbar. Ohne Notar tritt beispielsweise folgendes Problem auf: Gibt A sein Geheimnis an B und erwartet dann von B im Austausch y zu erhalten, so kann ein betrügerischer B nach Erhalt von x das Protokoll beenden oder einen falschen Wert $\tilde{y}$ schicken.

Die existierenden praktikablen Ansätze für faire Austauschprotokolle ohne vertrauenswürdigen Teilnehmer unterteilen die Geheimnisse in kleine Blöcke und tauschen abwechselnd jeweils einen Block aus. Nach Erhalt eines Blockes kann man durch das öffentliche Prädikat sofort überprüfen, ob der empfangene Block korrekt ist. Andererseits lernt man aus den bereits erhaltenen Blöcken so wenig wie möglich über die restlichen Teile des Geheimnisses. Versucht in diesem Fall ein Teilnehmer zu betrügen, so besitzt er nur einen kleinen Vorteil gegenüber dem anderen Teilnehmer.

Werden digitale Unterschriften als Geheimnisse ausgetauscht, spricht man vom fairen Austausch von Unterschriften. Solche Austauschprotokolle werden beispielsweise für die gegenseitige Unterzeichnung von Verträgen oder zum Austausch von Quittungen gegen Unterschriften verwendet. Ein korrektes Geheimnis entspricht in diesem Fall einer gültigen Unterschrift. Die bekannten effizienten Verfahren für den fairen Austausch digitaler Unterschriften erlauben lediglich die blockweise Herausgabe in einer fest vorgegebenen Reihenfolge, beginnend mit dem untersten Block, gefolgt von dem zweituntersten Block usw. Die Sicherheit dieser Verfahren beruht auf der Faktorisierungsannahme, d.h. gegeben einen zufälligen RSA-Modul $N = pq$ mit $p \neq q$ prim, kann man in Polynomialzeit nicht die Primfaktoren p, q von N bestimmen. Da spezielle zahlentheoretische Eigenschaften des Faktorisierungsproblems verwendet werden, sind diese Protokolle nicht unmittelbar auf die Diskrete-Logarithmus-Annahme (gegeben eine Primzahl p und einen Generator g einer Unterruppe G von $\mathbb{Z}_p^*$, sowie einen Wert $X \in G$, berechne in Polynomialzeit das $x \in \mathbb{Z}_{\operatorname{ord} G}$ mit $g^x = X \bmod p$) übertragbar. Im Gegensatz zu Unterschriftenschemata, die ebenfalls auf dem Faktorisierungsproblem beruhen, wie z.B. das

RSA-Unterschriftenschema, benötigen diese Verfahren daher für den fairen Austausch von Diskreten-Logarithmus-Unterschriften zusätzlich die Faktorisierungsannahme.

In dieser Diplomarbeit entwickeln wir faire Austauschprotokolle für Diskrete-Logarithmus-Unterschriften, so daß die Protokolle vollständig auf der Diskreten-Logarithmus-Annahme basieren. Folglich benötigen wir keine zusätzliche kryptographische Annahme. Weiterhin erlauben unsere Protokolle die adaptive Herausgabe der Blöcke, d.h. die Reihenfolge ist nicht festgelegt und kann vom Protokollablauf abhängen. Die Komplexitätsmerkmale (Rundenanzahl, Kommunikations- und Rechenaufwand) unserer Protokolle sind fast identisch bzw. nur unwesentlich schlechter im Vergleich zu denen der bekannten Protokolle.

Kapitel 2

Überblick: Fairer Austausch digitaler Unterschriften

Der fairer Austausch digitaler Unterschriften — oder allgemeiner: von Geheimnissen — ist eine der ersten Problemstellungen der modernen Kryptographie. Manuel Blum [B83] präsentiert 1981 ein Austauschprotokoll für die Primfaktoren p_A, q_A bzw. p_B, q_B zweier Moduln $N_A = p_A q_A$ und $N_B = p_B q_B$. Bei diesem Protokoll wird schrittweise je ein Bit der Faktoren p_A und p_B ausgetauscht. Eine Anwendung dieses Protokolls ist der faire Austausch von Nachrichten und Quittungen. Dabei soll A genau dann eine Quittung von B für eine Nachricht m erhalten, wenn B diese Nachricht erhält. Dazu verschlüsselt A die Nachricht m mit dem RSA-Verschlüsselungsverfahren [RSA78] mit Modul $N_A = p_A q_A$ und Exponent e und sendet diese Verschlüsselung an B. Umgekehrt sendet B ein Modul $N_B = p_B q_B$ an A. Dann tauschen A und B auf faire Weise die Primfaktoren aus. Nach Erhalt des Primfaktors p_A kennt B die Primfaktorzerlgung von N_A und kann damit die Nachricht m entschlüsseln. Umgekehrt wird die Kenntnis von p_B, q_B zu N_B als Quittung angesehen.

Blums Protokoll beschränkt sich allerdings auf den Austausch von Primfaktoren und eignet sich daher nicht zum Austausch digitaler Unterschriften im allgemeinen. Ein erster Lösungsansatz stammt von Shimon Even, Oded Goldreich und Abraham Lempel [EGL85]. Dieses Protokoll verwendet außer einem sicheren Verschlüsselungsschema ein $\binom{2}{1}$-*Oblivious-Transfer-Protokoll*. Durch ein solches Protokoll bekommt B einen von zwei Werten von A, so daß A nicht weiß, welchen der beiden Werte B erhalten hat, und so daß B keine Informationen über den anderen Wert erhält. Auf diesen beiden "Bausteinen" beruht folgendes Protokoll zum fairen Austausch digitaler Unterschriften: Sei C der zu unterschreibende Vertrag.

1. Beide Teilnehmer wählen jeweils $2n$ Schlüssel des Verschlüsselungsschemas. Ferner erzeugen sie n Paare $L_i =$"Dies ist die linke Hälfte der i-ten Unterschrift zu C" bzw. $R_i =$ "Dies ist die rechte Hälfte der i-ten Unterschrift zu C" und unterschreiben diese Nachrichten für $i = 1, \ldots, n$. Dann verschlüsseln sie die Unterschriften $S_A(L_i)$, $S_A(R_i)$ bzw. $S_B(L_i)$, $S_B(R_i)$ zu L_i, R_i jeweils mit einem der $2n$ Kodierschlüssel und senden die Verschlüsselungen an den anderen Teilnehmer. Die Reihenfolge, in der diese Verschlüsselungen ausgetauscht werden, ist beliebig. Per Konvention hat ein Teilnehmer eine gültige Unterschrift des anderen für C, wenn er ein gültiges Paar $(S_A(L_i), S_A(R_i))$

bzw. $(S_B(L_i), S_B(R_i))$ zu (L_i, S_i) hat.

2. Für $i = 1, \ldots, n$ wirft A jeweils eine Münze $c_i \in \{L_i, R_i\}$ und läßt sich durch ein $\binom{2}{1}$-Oblivious-Transfer-Protokoll von B den Dekodierschlüssel zu $S_B(c_i)$ geben. Durch Verwendung des Oblivious-Transfer-Protokolls weiß B nicht, welchen Schlüssel A erhält. Analog wählt B jeweils einen der beiden Dekodierschlüssel von A's Paaren für alle $i = 1, \ldots, n$.

3. A und B entschlüsseln die jeweilige Unterschrift und überprüfen die Korrektheit. Falls ein Fehler auftritt, stoppt der Teilnehmer.

4. A und B senden jeweils abwechselnd das nächste Bit von jedem der $2n$ eigenen Dekodierschlüssel. Nach jeder Runde überprüfen die Teilnehmer, daß die Bits mit den in Schritt 2 erhaltenen Dekodierschlüsseln übereinstimmen.

Wenn ein Teilnehmer für alle $i = 1, \ldots, n$ jeweils eine ungültige Unterschrift zu L_i oder R_i verschlüsselt, so wird er mit Wahrscheinlichkeit $1 - 2^{-n}$ entdeckt, da in Schritt 2 der andere Teilnehmer den Schlüssel zu dieser ungültigen Unterschrift jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ erhält. Falls ein Teilnehmer für alle $i = 1, \ldots, 2n$ jeweils ein falsches Bit in Schritt 4 sendet, dann wird dies ebenfalls mit Wahrscheinlichkeit $1 - 2^{-n}$ entdeckt, da der andere Teilnehmer einen der beiden Schlüssel mit Wahrscheinlichkeit $\frac{1}{2}$ durch das $\binom{2}{1}$-Oblivious-Transfer-Protokoll erhalten hat. Insgesamt beträgt die Betrugswahrscheinlichkeit damit höchstens $2 \cdot 2^{-n}$. Das Protokoll ist allerdings nicht effizient. Ferner muß man voraussetzen, daß es für alle Schlüssel annähernd gleich schwierig ist, aus einem bereits erhaltenen Teil des Schlüssels den Rest des Schlüssels effizient zu berechnen (*uniforme Ergänzungskomplexität der Schlüssel*). Die uniforme Ergänzungskomplexität garantiert, daß kein Teilnehmer aus einem erhaltenem Teil eines Schlüssels den Rest mit hoher Wahrscheinlichkeit erraten kann, während dies für den anderen Teilnehmer nicht möglich ist. Allgemeiner spricht man im Fall des Austauschs von Geheimnissen bzw. Unterschriften von der uniformen Ergänzungskomplexität des Geheimnisses bzw. des Unterschriftenverfahrens.

Michael Luby, Silvio Micali und Charles Rackoff [LMR83] führen die *probabilistische Methode* zum fairen Austausch zweier Bits a und b ein. Dabei wird nicht nur "atomar" Bit gegen Bit getauscht, sondern die Bits werden in "kleinere" Einheiten aufgeteilt. Luby, Micali und Rackoff betrachten Folgen von Verteilungen, so daß diese Folgen gegen das entsprechende Bit a bzw. b konvergieren. Ist beispielsweise $a = 1$, so erzeugt A eine Münze, die mit 51% den Wert 1 annimt. Dann kann B diese "verzerrte" Münze werfen, bis die so erhaltene Münzwurffolge das Bit a mit hoher Wahrscheinlichkeit bestimmt. Die Fairness-Bedingung besagt im Fall der probabilistischen Methode, daß A zu jedem Zeitpunkt das Bit b annäherend so gut raten kann, wie umgekehrt B den Wert a vorhersagen kann. In obigem Beispiel der 51%-Münze erreicht man dies nicht, sofern die verzerrten Münzen unabhängig geworfen werden. In diesem Fall kann beispielsweise ein Teilnehmer etwa gleich oft den Ausgang Kopf oder Zahl erhalten, während der andere Teilnehmer das geheime Bit aus der erhaltenen Folge mit hoher Wahrscheinlichkeit vorhersagen kann. Luby, Micali und Rackoff definieren deshalb eine sogenannte symmetrisch verzerrte Münze, die die Münzwurffolgen für beide Teilnehmer bestimmt. Der Ratevorteil der beiden Teilnehmer ist somit für jede Münzwurffolge annähernd gleich. Eine solche symmetrisch verzerrte Münze kann basierend auf der *Quadratischen Residuozitätsannahme* [GM84] realisiert werden. Informell besagt die Quadratische Residuozitätsannahme,

daß man für einen RSA-Modul $N = pq$, $p \neq q$ prim, mit $p \equiv q \equiv 3 \bmod 4$ und eine Zahl $z \in \mathbb{Z}_N^*$ mit Jacobi-Symbol $+1$ ohne Kenntnis von p, q nicht in Polynomialzeit entscheiden kann, ob z quadratischer Rest oder Nichtrest ist.

Unabhängig von Luby, Micali und Rackoff wählen auch Vazirani und Vazirani [VV83] und Tedrick [T84] die probabilistischen Methode, um Bits auszutauschen. Alle diese Protokolle haben den Nachteil, daß die Vorhersagewahrscheinlichkeit nicht einer vorgegebenen, festen Sequenz $\frac{1}{2} \leq p_1 < \cdots < p_m = 1$ folgt, sondern nur mit hoher Wahrscheinlichkeit gegen 1 konvergiert. Folglich ist die Laufzeit dieser Protokolle nur im Erwartungswert polynomiell. Dieser Nachteil wurde 1989 von Richard Cleve [C89] durch Verwendung von Markov-Ketten eliminiert. Als kryptographische Grundlage verwendet Cleve die Quadratische Residuozitätsannahme sowie die Existenz von $\binom{2}{1}$-Oblivious-Transfer-Protokollen.

Protokolle basierend auf der probabilistischen Methode sind in der Praxis nicht effizient genug. Daher zieht man i.a. bitweise orientierte Austauschprotokolle vor. Solche Protokolle lassen sich aus theoretischer Sicht trivial durch *Zero-Knowledge-Beweissysteme* [GMR85, GMR89] realisieren. Ein Zero-Knowledge-Beweissystem zwischen zwei Teilnehmern P und V für eine Sprache L hat folgende Eigenschaften:

- Wenn die gemeinsame Eingabe x in der Sprache liegt, dann kann P dies V so beweisen, daß V nicht mehr Informationen erhält, als die Tatsache, daß $x \in L$ (Zero-Knowledge-Eigenschaft).
- Wenn $x \notin L$, dann kann jeder noch so mächtige P den Prüfer V nur mit sehr kleiner Wahrscheinlichkeit überzeugen, daß x in L liegt.

Die Zero-Knowledge-Eigenschaft wird über einen sogenannten Simulator definiert, der für $x \in L$ dieselbe Verteilung der Kommunikation zwischen P und V in Polynomialzeit bereits ohne den Beweiser P erzeugen kann.

Da jede $\mathcal{NP}$-Sprache unter kryptographischen Annahmen ein Zero-Knowledge-Beweissystem besitzt [GMW86, GMW91], erhält man folgendes Austauschprotokoll für Unterschriften: Beide Teilnehmer senden dem anderen eine Verschlüsselung ihrer Unterschrift und geben abwechselnd ein Bit ihrer Unterschrift preis. Mit jedem Bit zeigen sie durch einen Zero-Knowledge-Beweis, daß das Bit ein korrekter Teil der Unterschrift ist. Als Zeuge der $\mathcal{NP}$-Sprache dienen die restlichen Bits der Unterschrift und die Münzwurfe zur Erzeugung der Verschlüsselung (inklusive des Schlüssels). Nachteil dieser Methode: Allgemeine Zero-Knowledge-Beweissystem für $\mathcal{NP}$ sind nicht praktikabel, da die Sprache erst durch Karp-Reduktion [BDG95] auf eine geeignete $\mathcal{NP}$-vollständige Sprache reduziert werden muß. Ferner sind allgemeine Zero-Knowledge-Beweissysteme bezüglich Rechen- und Kommunikationsaufwand ineffizient.

Cleve und Vazirani, Vazirani entwickeln ihrer Protokolle zunächst für die Anwesenheit eines Notars. Dieser Notar wird dann durch kryptographische Bausteine wie Oblivious-Transfer-Protokolle ersetzt. Michael Ben-Or, Oded Goldreich, Silvio Micali und Ron Rivest [BGMR90] wählen den probabilistischen Ansatz mit *passivem* Notar T, um Unterschriften auszutauschen. Ein solcher passiver Notar greift nur ein, wenn Komplikationen auftreten und er von einem der beiden Teilnehmer angerufen wird. Das Protokoll sieht wie folgt aus: Sei C der zu unterschreibende Vertrag. Beide Teilnehmer tauschen abwechselnd Nachrichten M_i der Form

"Vertrag C ist mit Wahrscheinlichkeit p_i gültig" mit zugehöriger Unterschrift aus, wobei $0 = p_0 < \cdots < p_m = 1$. In jedem Schritt überprüft der Teilnehmer vor Senden der $(i+1)$-ten Nachricht, daß die Unterschrift des anderen Teilnehmers zu Nachricht M_i gültig ist. Per Konvention hat ein Teilnehmer eine gültige Unterschrift zu C, wenn er eine gültige Unterschrift des anderen Teilenhmers zu M_m besitzt. Angenommen, ein Teilnehmer stoppt vorzeitig und hat damit eine Unterschrift für M_i, während der andere Teilnehmer nur eine Unterschrift zu M_{i-1} besitzt. Dann sendet der betrogene Teilnehmer die Nachricht M_{i-1} mit der erhaltenen Unterschrift an den Notar. Beachte, daß die Unterschrift zu M_{i-1} gültig ist, da M_i nur in diesem Fall verschickt wird. Nach Prüfen der Unterschrift zu M_{i-1} wählt T dann einen Wert ρ gleichverteilt zwischen 0 und 1. Falls $\rho > p_{i-1}$ ist, dann erklärt er den Vertragsabschluß für ungültig und sendet eine entsprechendes Zertifikat an beide Teilnehmer. Wenn $\rho \leq p_{i-1}$ ist, dann sendet er ein Zertifikat "Der Vertrag C ist bereits mit Parameter p_{i-1} gültig" an beide Teilnehmer.

Da entweder beide Teilnehmer eine gültige Unterschrift bzw. ein entsprechendes Zertifikat erhalten, oder aber keiner von beiden, ist das Verfahren prinzipiell auch für den Fall $m = 1$ geeignet, d.h. wenn nur zwei Unterschriften für Wahrscheinlichkeiten 0 und 1 ausgetauscht werden. Eine kleine Schrittweite $p_i - p_{i-1}$ garantiert allerdings, daß beide Teilnehmer einen gültigen Vertragsabschluß nur mit annähernd gleicher Wahrscheinlichkeit erzwingen können. Ist beispielsweise $p_0 = 0$, $p_1 = \frac{1}{2}$ und $p_2 = 1$, so kann der Teilnehmer, der zuerst eine gültige Unterschrift zu M_1 erhält, den Notar anrufen und mit Wahrscheinlichkeit $\frac{1}{2}$ einen gültigen Vertragsabschluß erzeugen. Dagegen kann der andere Teilnehmer zu diesem Zeitpunkt durch Anrufen des Notars noch keinen Abschluß erzwingen, da er erst eine gültige Unterschrift zu M_0 besitzt.

Bei Anwesenheit von Notaren erhält man im allgemeinen effizientere Protokolle. Matthew Franklin und Michael Reiter [FR97] geben ein Verfahren mit einem Notar an, der keine Informationen über die Geheimnisse bzw. Unterschriften erhält — sofern er mit keinem der beiden Teilnehmer zusammenarbeitet. Dagegen lernt der Notar beim Verfahren von Ben-Or, Goldreich, Micali und Rivest beide Unterschriften, wenn er angerufen wird. Das von Franklin und Reiter angegebene Protokoll setzt allerdings die aktive Teilnahme des entsprechenden Notars voraus. Die Sicherheit beruht auf *One-Way-Homomorphismen*, d.h. Homomorphismen, die leicht zu berechnen sind, aber schwierig zu invertieren. N. Asokan, Victor Shoup und Michael Waidner [ASW97] präsentieren ein praktikables Verfahren, das nur einen passiven Notar benötigt. Das Verfahren erlaubt es, Unterschriften mit homomorphen Eigenschaften mit Hilfe einer One-Way-Funktion auszutauschen. Die bekanntesten Unterschriftenverfahren erzeugen solche Signaturen. Im Fall eines Disputs lernt der Notar allerdings die Unterschriften.

Die erste adäquate formale Definition von fairen Austauschprotokollen (ohne Notar) gibt Ivan Damgård [D93, D95]. Wie bereits in den ersten Arbeiten von Blum und Even, Goldreich, Lempel reduziert Damgård faire Austauschprotokolle auf sogenannte *sichere Release-Protokolle*. Ein Release-Protokoll erlaubt die schrittweise Preisgabe des Geheimnisses eines Teilnehmers an den anderen Teilnehmer. Im Unterschied dazu geben bei Austauschprotokollen beide Teilnehmer jeweils ihr Geheimnis preis. Ein Release-Protokoll heißt sicher, wenn gilt:

1. Vollständigkeit: Wenn A und B dem Release-Protokoll folgen, dann erhält B am Ende das Geheimnis von A.

2. Korrektheit: Wenn A in einer Runde einen "falschen" Wert an B weitergibt, wird dies sofort von B entdeckt.

3. Geheimhaltung: Zu jedem Zeitpunkt der Ausführung hat B nicht mehr Informationen über das Geheimnis erhalten, als bisher ausdrücklich preisgegeben wurde.

Mit der uniformen Ergänzungskomplexität des Geheimnisses erhält man aus einem sicheren Release-Protokoll ein faires Austauschprotokoll, indem A und B jeweils abwechselnd eine Runde eines Release-Protokolls ausführen. Die Geheimhaltung wird dabei über den Zero-Knowledge-Simulatoransatz [GMR85] definiert: Es gibt einen Simulator, der die gesamte Kommunikation des Protokolls bis zur jeweiligen Runde erzeugen kann, und der als Eingabe nur den ausdrücklich preisgegebenen Teil des Geheimnisses erhält. Folglich kann B aus der Protokollausführung nicht mehr Informationen über den Rest des Geheimnisses ermitteln, als sich aus dem ausdrücklich preisgegeben Teil ergibt.

Die Struktur der bekannten Release-Protokolle sieht wie folgt aus: A hinterlegt die Unterschrift durch ein sogenanntes *Hinterlegungsprotokoll* bei B. Ein solches Protokoll erlaubt es A, an B einen Wert zu senden, so daß B keine Informationen über den hinterlegten Wert erhält, und so daß A später — in der *Aufdeckphase* — zeigen kann, welchen Wert er hinterlegt hat. Weiterhin kann A unter einer kryptographischen Annahme in der Aufdeckphase nicht erfolgreich vortäuschen, einen anderen Wert hinterlegt zu haben. Hinterlegungen lassen sich daher mit versiegelten Umschlägen vergleichen, die keine Informationen über den Inhalt preisgeben und deren Inhalt andererseits durch das Siegel nicht mehr verändert werden kann. Nachdem die Unterschrift hinterlegt wurde, zeigt A mit einem effizienten Zero-Knowledge-Beweis, daß der hinterlegte Wert eine korrekte Unterschrift darstellt. Der Zero-Knowledge-Beweis garantiert einerseits, daß A keinen falschen Wert hinterlegt hat, und andererseits, daß B wegen der Zero-Knowledge-Eigenschaft keine Informationen über die Unterschrift erhält. Schließlich wird die hinterlegte Unterschrift blockweise aufgedeckt.

Basierend auf dem Faktorisierungsproblem entwirft Damgård ein sicheres Release-Protokoll für Geheimnisse bzw. Unterschriften. Informell ist der Protokollablauf wie folgt: Damgårds Hinterlegungsprotokoll hat die zusätzliche Eigenschaft, daß man aus zwei Hinterlegungen für Werte x_1, x_2 eine Hinterlegung für $x_1 + x_2$ erhält. Nachdem die Unterschrift hinterlegt wurde, zeigt der Unterschreiber A mit der sogenannten *Cut-And-Choose-Methode*, daß der hinterlegte Wert x in einem erlaubten Bereich $[0, I)$ liegt. Dazu werden parallel K unabhängige Ausführungen des folgenden Unterprogrammes durchgeführt. In jeder dieser Ausführungen unterteilt A das Intervall $[0, I)$ in zwei Teile und hinterlegt den Teilungspunkt t_1 und den gespiegelten Teilungspunkt $t_2 = t_1 - I \in [-I, 0)$ bei B. Mit Wahrscheinlichkeit $\frac{1}{2}$ fordert B, die Teilung aufzudecken bzw. zu zeigen, daß das Geheimnis höchstens um einen Abstand I von t_1 oder t_2 entfernt liegt. Im ersten Fall deckt A die Werte t_1, t_2 auf und B überprüft, daß diese Aufdeckungen korrekt sind, und daß $t_1 = t_2 + I$ sowie $t_1 \in [0, I)$. Im anderen Fall wählt A das Bit b so, daß $t_b + x \in [0, I)$, und sendet eine entsprechende Aufdeckung. Dabei wird die Homomorphie des Hinterlegungsverfahrens ausgenutzt. Tatsächlich zeigt diese Methode nur, daß das Geheimnis x im Intervall $[-I, 2I)$ liegt, d.h. wenn das Geheimnis nicht in $[-I, 2I)$ liegt, kann A in jeder Ausführung nur mit Wahrscheinlichkeit $\frac{1}{2}$ betrügen — sofern er das Faktorisierungsproblem nicht effizient lösen und damit in der Aufdeckungsphase betrügen kann. Insgesamt beträgt die Betrugswahrscheinlichkeit unter der Annahme, daß Faktorisieren schwierig ist, daher 2^{-K}.

Wir betrachten Damgårds Release-Protokoll am Beispiel einer Rabin-Unterschrift [R79]. Gegeben sei ein RSA-Modul $N = pq$ und eine Nachricht $m \in \mathbb{Z}_N^*$. Nur der Unterschreiber kennt die Primfaktorzerlegung p, q von N. Eine Unterschrift ist eine Wurzel zu m, d.h. ein Wert $s \in \mathbb{Z}_N^*$ mit $s^2 = m \bmod N$. Da der Unterschreiber die Primfaktoren p, q kennt, kann er eine solche Wurzel $\bmod p$ und $\bmod q$ berechnen und dann mit dem Chinesischen Restsatz zusammensetzen. Andererseits kann man zeigen, daß die Berechnung von Wurzeln für zufällig gewählte $m \in \mathbb{Z}_N^*$ impliziert, daß man N faktorisieren kann. Folglich basiert die Sicherheit des Unterschriftenschemas auf der Faktorisierungsannahme. Eine Unterschrift s zu m kann öffentlich verifiziert werden, indem man die Gleichung $s^2 = m \bmod N$ überprüft. Damgårds Release-Protokoll für Rabin-Unterschriften sieht wie folgt aus: Mit der Cut-And-Choose-Methode kann man ebenfalls zeigen, daß für hinterlegte Werte x_1, x_2 die Gleichung $x_1 x_2 = c$ für eine Konstante c gilt, wobei das Produkt $x_1 x_2$ durch Reduktion mod$\mathbb{Z}_{3I}$ und Subtraktion von I als Wert in $[-I, 2I)$ dargestellt wird. Der Unterschreiber in Damgårds Protokoll hinterlegt daher die Werte $x_1 = s \bmod N$ und $x_2 = s \bmod N$ und zeigt mit der Cut-And-Choose-Methode, daß die beiden Hinterlegungen denselben Wert darstellen und im Intervall $[-N, 2N)$ liegen, und daß $x_1 x_2 = m$ gilt, wobei $x_1 x_2$ wieder aus dem Intervall $[-N, 2N)$ sei. Damit gilt insbesondere auch $x_1 x_2 = m \bmod N$ und folglich hat der Unterschreiber eine Wurzel von m modulo $\mathbb{Z}_N$ hinterlegt. Diese Hinterlegung kann der Unterschreiber dann schrittweise beginnend mit dem untersten Bit aufdecken.

E.Fujisaki und T.Okamoto [FO97] verbessern Dagårds Verfahren bezüglich Kommunikations- und Rechenaufwand, indem sie die Cut-and-Choose-Methode eliminieren. Die Sicherheit des Protokolls von Fujisaki, Okamoto basiert auf der *modifizierten RSA-Annahme*, d.h. gegeben einen zufälligen RSA-Modul $N = pq$ und ein zufälliges $y \in \mathbb{Z}_N^*$, kann man ohne Kenntniss von p, q nicht in Polynomialzeit ein x und ein $e \geq 2$ mit $y = x^e \bmod N$ finden. Die ursprüngliche RSA-Annahme besagt dagegegen, daß man für zufälligen RSA-Modul $N = pq$, zufälliges $e \in \mathbb{Z}_{\phi(N)}^*$ und zufälliges y ohne Kenntnis von p, q nicht in Polynomialzeit ein x mit $y = x^e \bmod N$ berechnen kann. Offensichtlich impliziert die modifizierte RSA-Annahme die RSA- und Faktorisierungsannahme. Die Umkehrung ist offen. Als elementares Protokoll verwenden Fujisaki und Okamoto wie Damgård ein Hinterlegungsprotokoll für Werte x_1, x_2, so daß der Hinterleger zusätzlich zeigen kann, daß $x_1 x_2 + c = 0 \bmod N$ für eine beliebige Konstante $c \in \mathbb{Z}_N$. Die Hinterlegung kann unter der modifizierten RSA-Annahme nicht auf zwei verschiedene Weisen aufgedeckt werden und garantiert umgekehrt, daß der andere Teilnehmer keine Informationen über x_1, x_2 erhält. Weiterhin erlaubt die Hinterlegung die schrittweise Herausgabe der Werte x_1, x_2, beginnend mit dem untersten Bit usw.

Während die Protokolle von Damgård und Fujisaki, Okamoto auf der speziellen zahlentheoretischen Annahme des Faktorisierungsproblems beruht, geben T.Okamoto und K.Ohta ein faires Austauschprotokoll basierend auf der allgemeineren Annahme der Existenz injektiver One-Way-Funktionen an [OO94]. Dieses Protokoll erlaubt allerdings nur den Austausch von Geheimnissen, und — wie bei Damgård und Fujisaki, Okamoto — dies nur bitweise in einer festen Reihenfolge beginnend mit dem untersten Bit, gefolgt von dem zweituntersten Bit usw.

Diese Diplomarbeit baut vor allem auf Damgårds Arbeit [D95] auf. Im Unterschied zu Damgård beschäftigen wir uns primär mit dem fairen Austausch Diskreter-Logarithmus-Unterschriften. Neben den RSA- und Rabin-Unterschriften sind dies die bekanntesten und in der Praxis am häufigsten verwendeten Unterschriftenverfahren [DSS, S91, EG85]. Die Si-

cherheit unserer Protokolle beruht vollständig auf der *Diskreten-Logarithmus-Annahme*. Diese Annahme besagt, daß man nur mit sehr kleiner Wahrscheinlichkeit für eine Primzahl p, einen Generator g einer Untergruppe G von $\mathbb{Z}_p^*$ und einen zufälligen Wert $X \in G$ in Polynomialzeit das $x \in \mathbb{Z}_{\operatorname{ord} G}$ mit $X = g^x \bmod p$ finden kann. Dadurch reduziert sich die Sicherheitsannhame des Austauschprotokolls auf die des Unterschriftenverfahrens.

Weiterhin erlaubt unser Protokoll die *adaptive* Herausgabe der Blöcke der Unterschrift, d.h. nach jedem Schritt kann man neu entscheiden, welchen Teil der Unterschrift man als nächstes an den anderen Teilnehmer sendet. Im Unterschied dazu erlauben die Protokolle von Damgård bzw. Fujisaki und Okamoto nur die bitweise Herausgabe, beginnend mit dem untersten Bit, dann dem zweitunterstem Bit usw. Die adaptive Herausgabe schwächt die Anforderung an die uniforme Ergänzungskomplexität des Geheimnisses ab, da A beispielsweise zunächst die "leichten" Teile der Unterschrift herausgeben kann.

Die Komplexität unseres Verfahrens ist vergleichbar mit der von Damgårds Protokoll. Während die Rundenanzahl bis auf eine additiven Konstante optimal ist, ist die Kommunikations- und Rechenkomplexität wie bei Damgård relativ groß. Dieser Aufwand wird in Damgårds Protokoll vor allem durch die Cut-And-Choose-Methode verursacht. Bei unserem Protokoll ergibt sie sich ebenfalls durch den Nachweis, daß jeder Teil der Unterschrift in einem festen Bereich liegt — obwohl wir nicht die Cut-And-Choose-Methode verwenden (können). Dagegen erreichen Fujisaki und Okamoto unter der stärkeren, modifizierten RSA-Annahme ein effizienteres Protokoll.

Kapitel 3

Grundlagen

3.1 Zufallsvariablen

Sei A ein probabilistischer Algorithmus bzw. eine probabilistische Turingmaschine.[1] Wir schreiben $y \leftarrow A(x)$, wenn der Variablen y die Ausgabe einer Ausführung von A auf Eingabe x zugewiesen wird. Analog sei $y \leftarrow Y$ für eine Zufallsvariable Y definiert. Ist A deterministisch, schreiben wir im allgemeinen $y = A(x)$. Wenn wir einen Wert y uniform aus einer endlichen Menge M wählen, schreiben wir $y \in_R M$. Mit

$$\mathrm{Prob}[\, y_1 \leftarrow A_1(x_1), \ldots, y_n \leftarrow A_n(x_n) \;:\; P(y_1, \ldots, y_n)]$$

bezeichnen wir die Wahrscheinlichkeit, daß das Prädikat P von $(y_1, \ldots, y_n)$ erfüllt wird, wobei $y_1, \ldots, y_n$ durch die (geordnete) Ausführung der probabilistischen Algorithmen (oder Zufallsvariablen) $A_1, \ldots, A_n$ für Eingabe $x_1, \ldots, x_n$ erzeugt werden. Dabei schließen wir den Fall ein, daß x_i die leere Eingabe ist bzw. daß y_i Teil der Eingabe x_j für $i < j$ ist. Die Wahrscheinlichkeit wird über die Münzwürfe der Algorithmen bzw. die zufällige Wahl gemäß der Zufallsvariablen gebildet.

Für einen probabilistischen Algorithmus A mit Eingabe x sei

$$[A(x)] = \{ z \in \{0,1\}^* \mid \mathrm{Prob}[\, A(x) = z] > 0 \} .$$

die Menge der mit positiver Wahrscheinlichkeit von $A(x)$ ausgegebenen Werte. Analog sei $[Y]$ für eine Zufallsvariable Y definiert. Wir können einen probabilistischen Algorithmus A für eine fixierte Münzwurffolge ω_A als deterministische Funktion $A(\cdot, \omega_A)$ auffassen, d.h. wenn die Münzwurffolge ω_A gemäß der Zufallsvariablen Ω_A erzeugt wird, gilt

$$\mathrm{Prob}[\, \omega_A \leftarrow \Omega_A,\, y = A(x, \omega_A) \;:\; P(x, y)] = \mathrm{Prob}[\, y \leftarrow A(x) \;:\; P(x, y)]$$

für alle x, y und alle Prädikate P.

Für eine Menge $I \subseteq \{0,1\}^*$ heißt eine Folge $X = \{X_i\}_{i \in I}$ von Zufallsvariablen ein *Ensemble* für die *Indexmenge* I. Entspricht $I = \{1^n \mid n \in \mathbb{N}\}$ den natürlichen Zahlen, schreiben wir $\{X_n\}_{n \in \mathbb{N}}$ statt $\{X_i\}_{i \in I}$.

[1] Wir verwenden die Begriffe "Algorithmus" und "Turingmaschine" synonym.

Um die kryptographischen Sicherheitsbegriffe in den folgenden Kapiteln zu formalisieren, benötigen wir die Definition der statistischen und der Polynomialzeit-Ununterscheidbarkeit. Als elementar stellt sich dabei der Begriff einer vernachlässigbaren Funktion heraus. Informell ist eine Funktion $\epsilon : \mathbb{N} \to \mathbb{R}$ vernachlässigbar, wenn sie schneller gegen 0 konvergiert als jeder polynomielle Bruchteil. Beispielsweise ist $\lambda n.2^{-n}$ vernachlässigbar, da $2^{-n} < 1/p(n)$ für alle Polynome p und alle hinreichend großen $n \in \mathbb{N}$. In der folgenden Definition erweitern wir den Begriff einer vernachlässigbaren Funktionen auch auf Indexmengen.

Definition 3.1.1 (Vernachlässigbare Funktion)
Eine Funktion $\epsilon : \mathbb{N} \to \mathbb{R}$ heißt vernachlässigbar, wenn es für alle Polynome $p : \mathbb{N} \to \mathbb{R}$ ein n_0 gibt, so daß $\epsilon(n) < 1/p(n)$ für alle n mit $n \geq n_0$. Für eine Indexmenge I heißt $\epsilon : I \to \mathbb{R}$ vernachlässigbar, wenn $\epsilon_{\max}(n) := \max\{\epsilon(x) \mid x \in I \cap \{0,1\}^n\}$ vernachlässigbar ist.[2]

Für $I = \{1^n \mid n \in \mathbb{N}\}$ stimmen beide Definitionen überein. Informell heißen zwei Ensembles von Zufallsvariablen statistisch ununterscheidbar, wenn die Zufallsvariablen annähernd identisch verteilt sind.

Definition 3.1.2 (Statistisch Ununterscheidbar)
Zwei Ensembles $X = \{X_i\}_{i \in I}$ und $Y = \{Y_i\}_{i \in I}$ von Zufallsvariablen heißen statistisch ununterscheidbar, wenn die statistische Differenz

$$\Delta_{X,Y}(i) = \tfrac{1}{2} \cdot \sum_{z \in [X_i] \cup [Y_i]} |\mathrm{Prob}[X_i = z] - \mathrm{Prob}[Y_i = z]|$$

vernachlässigbar ist.

Der Faktor $1/2$ normiert die statistische Differenz $\Delta_{X,Y}(i)$ auf eine reelle Zahl zwischen 0 (X_i und Y_i sind identisch verteilt) und 1 ($[X_i]$ und $[Y_i]$ sind disjunkt). Man kann den Begriff der statistischen Ununterscheidbarkeit auf Ensembles $\{X_i\}_{i \in I}$ und $\{Y_j\}_{j \in J}$ mit Indexmengen I und J erweitern, wenn es eine surjektive Projektion π von J auf I gibt. Man definiert dann $X' = \{X'_j\}_{j \in J}$ durch $X'_j \equiv X_{\pi(j)}$ und setzt $\Delta_{X,Y}(j) = \Delta_{X',Y}(j)$ für $j \in J$.

Für kryptographische Anwendungen genügt es im allgemeinen, daß zwei Zufallsvariablen für probabilistische Polynomialzeitalgorithmen nicht zu unterscheiden sind. Wir formalisieren dies durch sogenannte *Unterscheider*. Ein solcher probabilistischer Polynomialzeitalgorithmus D soll für einen vorgelegten Wert $z_i \leftarrow X_i$ bzw. $z_i \leftarrow Y_i$ entscheiden, ob z_i gemäß X_i oder Y_i gezogen wurde. Der *Vorteil* von D gibt an, wie gut D die beiden Verteilungen unterscheiden kann.

Definition 3.1.3 (Polynomialzeit-Ununterscheidbar)
Zwei Ensembles $X = \{X_i\}_{i \in I}$ und $Y = \{Y_i\}_{i \in I}$ von Zufallsvariablen heißen polynomialzeit-ununterscheidbar, wenn für alle probabilistischen Polynomialzeitalgorithmen D der Vorteil

$$|\mathrm{Prob}[z_i \leftarrow X_i,\, d \leftarrow D(i, z_i) \;:\; d = 1] - \mathrm{Prob}[z_i \leftarrow Y_i,\, d \leftarrow D(i, z_i) \;:\; d = 1]|$$

vernachlässigbar ist.

[2] Dabei sei $\max \emptyset = 0$.

Analog zur statistischen Ununterscheidbarkeit kann man den Begriff der Polynomialzeit-Ununterscheidbarkeit auf Indexmengen I und J erweitern, bei denen eine Indexmenge auf die andere surjektiv projiziert werden kann. Wenn die Bilder von X_i und Y_i nur polynomielle Länge bezüglich der Indexlänge $|i|$ besitzen, dann ist die Laufzeit von D polynomiell in der Länge des Index beschränkt. Speziell für $I = \{1^n \mid n \in \mathbb{N}\}$ besitzt D für X_n, Y_n dann polynomielle Laufzeit in n. Offensichtlich sind Ensembles von identisch verteilten Zufallsvariablen statistisch ununterscheidbar. Man zeigt leicht, daß statistisch ununterscheidbare Ensembles auch polynomialzeit-ununterscheidbar sind [G95]. Die Umkehrung gilt in beiden Fällen nicht.

Als alternatives Berechnungsmodell kann man polynomielle Schaltkreisfamilien bzw. nicht-uniforme Polynomialzeitalgorithmen für die Unterscheider wählen. Goldreich und Mayer zeigen, daß es Ensembles gibt, die von polynomiellen Schaltkreisfamilien mit nicht vernachlässigbarem Vorteil unterscheiden werden können, während probabilistische Polynomialzeitalgorithmen nur vernachlässigbaren Vorteil erreichen [GM96]. Die Aussagen in dieser Diplomarbeit bleiben gültig, wenn man den Begriff der Polynomialzeit-Ununterscheidbarkeit durch polynomielle Schaltkreisfamilien formalisiert. In diesem Fall definiere man die zugrundeliegende Annahmen (z.B. die Diskrete-Logarithmus-Annahme im nächsten Abschnitt) ebenfalls im nicht-uniformen Modell.

3.2 Repräsentationsproblem

In diesem Abschnitt führen wir die Sicherheitsannahme unseres Austauschprotokolls ein. Im folgenden seien p, q Primzahlen, so daß $q \mid (p-1)$ und $q^2 \nmid (p-1)$. Sei $\mathbb{G}_q$ die eindeutig bestimmte Untergruppe von $\mathbb{Z}_p^*$ mit Ordnung q. Generatoren von $\mathbb{G}_q$ bezeichnen wir im allgemeinen mit g, g_i, G, G_i, h und H. Sofern nicht anders angegeben, erfolgen alle Berechnungen in $\mathbb{Z}_p^*$.

Definition 3.2.1 (Repräsentation)
Seien $g_1, \ldots, g_n$ Generatoren von $\mathbb{G}_q$. Eine Repräsentation von $Z \in \mathbb{G}_q$ bezüglich $(g_1, \ldots, g_n)$ ist ein Tupel $(z_1, \ldots, z_n) \in \mathbb{Z}_q^n$ mit $Z = \prod_{i=1}^n g_i^{z_i} \bmod p$.

Für $n = 1$ bezeichnen wir die eindeutig bestimmte Repräsentation eines Wertes Z bezüglich g_1 als *diskreten Logarithmus* $\log_{g_1} Z$ von Z bezüglich g_1. Das folgende Lemma zeigt, daß jedes $Z \in \mathbb{G}_q$ genau q^{n-1} Repräsentationen bezüglich eines Generatorentupels $(g_1, \ldots, g_n)$ besitzt. Dazu wähle man im Lemma $z_1, \ldots, z_{n-1}$ beliebig, sowie $\overline{z}_1, \ldots, \overline{z}_{n-1} = 0$ und $\overline{z}_n = \log_{g_n} Z$.

Lemma 3.2.2
Seien $g_1, \ldots, g_n$ Generatoren von $\mathbb{G}_q$. Für alle $z_1, \ldots, z_{n-1} \in \mathbb{Z}_q$ und $\overline{z}_1, \ldots, \overline{z}_n \in \mathbb{Z}_q$ existiert ein eindeutig bestimmtes $z_n \in \mathbb{Z}_q$ mit

$$\prod_{i=1}^{n} g_i^{z_i} = \prod_{i=1}^{n} g_i^{\overline{z}_i}.$$

Beweis. Da g_1 die Gruppe $\mathbb{G}_q$ erzeugt, existieren $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_q^*$ mit $g_i = g_1^{\alpha_i} \bmod p$ für alle $i = 1, \ldots, n$. Daher kann man die Gleichung wie folgt umschreiben:

$$g_1^{\sum \alpha_i z_i} = g_1^{\sum \alpha_i \overline{z}_i} \bmod p.$$

Die äquivalente Gleichung

$$\sum_{i=1}^{n} \alpha_i z_i = \sum_{i=1}^{n} \alpha_i \overline{z}_i \bmod q$$

in $\mathbb{Z}_q$ besitzt wegen $\alpha_n \neq 0$ für gegebene $z_1, \ldots, z_{n-1}, \overline{z}_1, \ldots, \overline{z}_n$ eine eindeutig bestimmte Lösung $z_n \in \mathbb{Z}_q$. ■

Die folgende Annahme besagt, daß es schwierig ist, in Polynomialzeit eine Repräsentation eines zufälligen Wertes zu finden. Dabei nehmen wir an, daß ein probabilistischer "Instanzengenerator" I auf Eingabe 1^k Primzahlen p, q mit Bitlänge $|q| = k$ und $|p| = \text{poly}(k)$ sowie $q \mid (p-1)$ und $q^2 \not| (p-1)$ erzeugt und $n = n(k) = \text{poly}(k)$ zufällige Generatoren $g_1, \ldots, g_n \in_R \mathbb{G}_q - \{1\}$ wählt. Zur Erzeugung solcher Primzahlpaare und Generatoren verweisen wir auf die Arbeiten von Ueli Maurer [M92, M95]. Wenn zusätzlich die Primfaktorzerlegung $w_1^{e_1}, \ldots, w_m^{e_m}$ (inklusive der Primfaktoren $w_1, \ldots, w_m$) von w mit $p = qw+1$ öffentlich bekannt gemacht wird, kann man effizient überprüfen, daß ein Wert g ein Generator von $\mathbb{G}_q$ ist. Dazu verifiziere man, daß $g^{w_i^{e_i}} \neq 1 \bmod p$ für alle $i = 1.\ldots, m$. Ist ferner $g \neq 1$ und $g^q = 1 \bmod p$, so ist g ein Generator von $\mathbb{G}_q$. Die Annahme wurde zunächst nur für den Fall $n = 1$ eines Generators formuliert. Die angegebene Definition verallgemeinert die Annahme auf den Fall $n > 1$.

Definition 3.2.3 (Verallgemeinerte Diskrete-Logarithmus-Annahme)
Sei $n : \mathbb{N} \to \mathbb{N}$ ein Polynom. Die verallgemeinerte Diskrete-Logarithmus-Annahme für n gilt, wenn für jeden probabilistischen Polynomialzeitalgorithmus A die Wahrscheinlichkeit

$$\text{Prob}\Big[(p, q, g_1, \ldots, g_{n(k)}) \leftarrow I(1^k),\ Z \in_R \mathbb{G}_q,\\ (x_1, \ldots, x_{n(k)}) \leftarrow A(p, q, g_1, \ldots, g_{n(k)}, Z)\ :\ Z = \textstyle\prod_{i=1}^{n(k)} g_i^{x_i} \bmod p\Big]$$

vernachlässigbar in 1^k ist.

Wir schreiben im folgenden kurz VDL$_n$-Annahme für die verallgemeinerte Diskrete-Logarithmus-Annahme. Im Fall $n \equiv 1$ sprechen wir einfach von der Diskreten-Logarithmus-Annahme (DL-Annahme). Die folgende Repräsentationsproblem-Annahme besagt, daß es schwierig ist, in Polynomialzeit einen Wert $Z \in \mathbb{G}_q$ mit zwei verschiedene Repräsentationen zu finden.

Definition 3.2.4 (Repräsentationsproblem-Annahme)
Sei $n : \mathbb{N} \to \mathbb{N}$ ein Polynom. Die Repräsentationsproblem-Annahme für n gilt, wenn für jeden probabilistischen Polynomialzeitalgorithmus A die Wahrscheinlichkeit

$$\text{Prob}\Big[(p, q, g_1, \ldots, g_{n(k)}) \leftarrow I(1^k),\\ \big((x_1, \ldots, x_{n(k)}), (x_1^*, \ldots, x_{n(k)}^*)\big) \leftarrow A(p, q, g_1, \ldots, g_{n(k)})\ :\\ \textstyle\prod_{i=1}^{n(k)} g_i^{x_i^*} = \prod_{i=1}^{n(k)} g_i^{x_i} \bmod p\ \wedge\ (x_1, \ldots, x_{n(k)}) \neq (x_1^*, \ldots, x_{n(k)}^*)\Big]$$

vernachlässigbar in 1^k ist.

Bei der Repräsentationsproblem-Annahme können wir uns o.B.d.A. auf den Fall $n(k) \geq 2$ beschränken, da für $n(k) = 1$ der diskrete Logarithmus eindeutig bestimmt ist. Wir nennen zwei Annahmen $\mathcal{A}_1$ und $\mathcal{A}_2$ *äquivalent*, wenn $\mathcal{A}_1$ genau dann gilt, wenn $\mathcal{A}_2$ gilt. Wir skizzieren kurz, daß DL-Annahme, VDL_n-Annahme und Rep_n-Annahme äquivalent sind. Die im Beweis verwendete Technik ist elementar für kryptographische Sicherheitsbeweise und wird daher anhand einfacher Teile des Beweises dargestellt. Um zu zeigen, daß eine Annahme $\mathcal{A}_1$ eine andere Annahme $\mathcal{A}_2$ impliziert, nimmt man an, daß es einen probabilistischen Polynomialzeitalgorithmus A_2 gibt, der $\mathcal{A}_2$ widerlegt. Aus A_2 erhält man dann — im Widerspruch zur Voraussetzung — einen probabilistischen Polynomialzeitalgorithmus A_1, der $\mathcal{A}_1$ widerlegt.

Satz 3.2.5
Seien $m, n : \mathbb{N} \to \mathbb{N}$ Polynome. Dann sind DL-, Rep_m- *und* VDL_n-*Annahme äquivalent.*

Beweis (Skizze). Wir zeigen die Behauptungen durch Widerspruch. Zur Abkürzung sei im folgenden stets $n = n(k)$ und $m = m(k)$.

1. Sei A_{DL} ein probabilistischer Polynomialzeitalorithmus, der den diskreten Logarithmus mit nicht vernachlässigbarer Wahrscheinlichkeit berechnet, d.h. für unendlich viele $k \in \mathbb{N}$ sei die Erfolgswahrscheinlichkeit von A_{DL} mindestens $1/p(k)$ für ein Polynom p. Wir betrachten im folgenden nur noch solche $k \in \mathbb{N}$. Aus A_{DL} konstruieren wir einen Algorithmus A_{VDL_n}, der eine Repräsentation eines zufälligen Wertes Z mit nicht vernachlässigbarer Wahrscheinlichkeit findet. A_{VDL_n} erhält als Eingabe $p, q, g_1, \ldots, g_n$ und Z für Sicherheitparameter k und simuliert A_{DL} für Eingabe (p, q, g_1, Z). Da diese Werte identisch verteilt sind zu einer "echten" Eingabe für A_{DL} erhalten wir mit Wahrscheinlichkeit mindestens $1/p(k)$ einen Wert $x_1 = \log_{g_1} Z$. Folglich ist $(x_1, 0, \ldots, 0)$ eine Repräsentation für Z bezüglich $(g_1, \ldots, g_n)$. Die Laufzeit von A_{VDL_n} ist offensichtlich polynomiell in der von A_{DL}.

2. Sei A_{VDL_n} mit nicht vernachlässigbarer Erfolgswahrscheinlichkeit und polynomieller Laufzeit gegeben. Wir konstruieren daraus einen Polynomialzeitalgorithmus A_{Rep_m}, das Repräsentationsproblem mit nicht vernachlässigbarer Wahrscheinlichkeit löst. Sei $m(k) \geq n(k)$. Auf Eingabe $p, q, g_1, \ldots, g_m$ wählt A_{Rep_m} Werte $x_1, \ldots, x_n \in_R \mathbb{Z}_q$, bildet $Z = \prod_{i=1}^{n} g_i^{x_i}$ und simuliert A_{VDL_n} für $p, q, g_1, \ldots, g_n, Z$. Mit Wahrscheinlichkeit mindestens $1/p(k)$ gibt A_{VDL_n} eine Repräsentation $(x_1^*, \ldots, x_n^*)$ für Z aus. In diesem Fall ist $(x_1^*, \ldots, x_n^*) \neq (x_1, \ldots, x_n)$ mit Wahrscheinlichkeit mindestens $1 - q^{-n+1}$, da die Ausgabe von A_{VDL_n} unabhängig von der Repräsentation $(x_1, \ldots, x_n)$ der q^{n-1} möglichen Repräsentationen von Z ist. Daher ist die Erfolgswahrscheinlichkeit von A_{Rep_m} mindestens $(1 - q^{-n(k)+1})/p(k) \geq 1/2p(k)$, wobei wir o.B.d.A. $n(k) \geq 2$ vorausgesetzt haben.

   Sei $m(k) < n(k)$. A_{Rep_m} erzeugt Werte $y_i \in_R \mathbb{Z}_q^*$ und setzt $g_i = g_m^{y_i}$ für $i = m + 1, \ldots, n$. Dann wählt er $x_1, \ldots, x_n$ und berechnet $Z = \prod g_i^{x_i} \bmod p$. Durch Simulation von A_{VDL_n} für $p, q, g_1, \ldots, g_n, Z$ erhält er eine Repräsentation $(x_1^*, \ldots, x_n^*)$ von Z bezüglich $(g_1, \ldots, g_n)$. Mit Wahrscheinlichkeit $1 - q^{-m+1} \geq \frac{1}{2}$ ist $(x_1^*, \ldots, x_{m-1}^*) \neq (x_1, \ldots, x_{m-1})$, so daß wir in diesem Fall verschiedene Repräsentationen $(x_1, \ldots, x_m)$ und $(x_1^*, \ldots, x_{m-1}^*, x_m^* + \sum y_i x_{m+i}^*)$ von Z bezüglich $(g_1, \ldots, g_m)$ erhalten.

Die Umwandlung eines A_{Rep_m}-Algorithmus' in einen A_{DL}-Algorithmus erfolgt wie in [BGG94] angegeben. ■

3.3 Interaktive Protokolle

In diesem Abschnitt führen wir den Begriff der interaktive Berechnungen zweier Algorithmen ein. Dazu betrachten wir zunächst das formale Modell und erläutern die Definitionen interaktiver Beweissysteme anhand des Beispiels des Repräsentationsproblems. Diese sogenannten Proofs of Knowledge für das Repräsentationsproblem bilden den elementaren Baustein unseres Release-Protokolls in Kapitel 4. Interaktive Protokolle dienen ferner als Grundlage für die Definition fairer Austauschprotokolle im nächsten Abschnitt.

Ein Paar *interaktiver Turingmaschinen* besteht aus zwei Turingmaschinen, die außer den individuellen Eingabe-, Münzwurf-, Arbeits- und Ausgabebändern über ein gemeinsames Kommunikationsband und ein gemeinsames Eingabeband verfügen.[3] Zu Beginn einer gemeinsamen Berechnung eines Paares interaktiver Maschinen (A, B) erhalten A und B als gemeinsame Eingabe x. Ferner werden die Münzwurfbänder unabhängig mit ω_A bzw. ω_B initialisiert und x_A bzw. x_B auf die individuellen Eingabebänder geschrieben. Dabei können x_A und x_B voneinander und insbesondere von x abhängen. Über das gemeinsame Kommunikationsband tauschen A und B Nachrichten aus, wobei wir o.B.d.A. annehmen, daß A die erste Nachricht sendet. Wir können die interaktive Berechnung wie folgt darstellen: Wir betrachten A und B für feste Münzwurffolgen als Funktionen in der Eingabe, den Münzwurffolgen und den bisher ausgetauschten Nachrichten. Induktiv definieren wir dann die *Kommunikation* zwischen $A(x, x_A, \omega_A)$ und $B(x, x_B, \omega_B)$ durch $\alpha_1 = A(x, x_A, \omega_A)$ und

$$\alpha_{i+1} = A\big(x, x_A, \omega_A, (\alpha_1, \beta_1, \dots, \alpha_i, \beta_i)\big) \text{ und } \beta_{i+1} = B\big(x, x_B, \omega_B, (\alpha_1, \beta_1, \dots, \alpha_i, \beta_i, \alpha_{i+1})\big)$$

für $i \in \mathbb{N}$. Wir schreiben $\mathcal{P}(A, B)$ für die Spezifikation der Funktionen A und B und nennen $\mathcal{P}$ das *Protokoll*. Eine *Runde* des Protokolls besteht aus einer gesendeten Nachricht. Wir erhalten folgende Zufallsvariablen: Mit

$$\{\mathrm{output}_B \langle A(x, x_A) \leftrightarrow_{\mathcal{P}} B(x, x_B)\rangle\}_{x, x_A, x_B}$$

bezeichnen wir das Ensemble, das die Ausgabe von Maschine B bei Ausführung des Protokolls $\mathcal{P}(A, B)$ mit gemeinsamer Eingabe x und individueller Eingaben x_A und x_B für A bzw. B beschreibt. Die Wahrscheinlichkeit wird über die Münzwürfe von A und B gebildet. Das Ensemble

$$\{\mathrm{view}_B \langle A(x, x_A) \leftrightarrow_{\mathcal{P}} B(x, x_B)\rangle\}_{x, x_A, x_B}$$

beschreibt die Kommunikation eines Protokollablaufs zwischen A und B (konkateniert mit dem Münzwurfband von B). Insbesondere ist $\{\mathrm{output}_B \langle A(x, x_A) \leftrightarrow_{\mathcal{P}} B(x, x_B)\rangle\}_{x, x_A, x_B}$ eindeutig durch $\{\mathrm{view}_B \langle A(x, x_A) \leftrightarrow_{\mathcal{P}} B(x, x_B)\rangle\}_{x, x_A, x_B}$ bestimmt. Wenn B nur ein Bit ausgibt, schreiben wir $\mathrm{Acc}_B \langle A(x, x_A, \omega_A) \leftrightarrow_{\mathcal{P}} B(x, x_B, \omega_B)\rangle$ genau dann, wenn die Ausgabe $\mathrm{output}_B \langle A(x, x_A, \omega_A) \leftrightarrow_{\mathcal{P}} B(x, x_B, \omega_B)\rangle = 1$ ist. Entsprechend sei das Ensemble

$$\{\mathrm{Acc}_B \langle A(x, x_A) \leftrightarrow_{\mathcal{P}} B(x, x_B)\rangle\}_{x, x_A, x_B}$$

[3] Wir verzichten auf eine formale Definition und verweisen auf die Arbeiten von Goldwasser, Micali und Rackoff [GMR89] und Goldreich und Ostrovsky [GO96]. Eine Ausführliche Darstellung findet man in [G95].

definiert. Im allgemeinen ist B ein probabilistischer Polynomialzeitalgorithmus, so daß wir in diesem Fall o.B.d.A. annehmen, daß x_B und jede Nachricht von A an B polynomielle Länge in $|x|$ hat.

Das Protokoll $\mathcal{P}(A, B)$ gibt an, wie sich "ehrliche" Teilnehmer A und B verhalten. Wir verwenden im weiteren auch den Fall, daß ein "unehrlicher" Teilnehmer statt A oder B am Protokoll mit dem anderen ehrlichen Teilnehmer kommuniziert. In diesem Fall schreiben wir im allgemeinen A^* bzw. B^* für den unehrlichen Teilnehmer.[4] Dabei können sich A^* bzw. B^* ganz oder teilweise wie A bzw. B verhalten. Analog seien die oben angegebenen Zufallsvariablen für A^* mit Eingabe x_{A^*} bzw. B^* mit Eingabe x_{B^*} definiert.

Sei A^* eine interaktive Turingmaschine im Protokoll $\mathcal{P}(A, B)$. Wir sagen, daß eine Orakelturingmaschine[5] E *Orakelzugriff auf* A^* hat, wenn zu Beginn der Ausführung für E nichtsichtbar eine hinreichend lange Münzwurffolge ω_{A^*} für A^* erzeugt wird, und E als Orakel die Funktion $A^*(x, x_{A^*}, \omega_{A^*})$ erhält. Für eine Kommunikation $(\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$ erhält E als Antwort des Orakels $\alpha_{i+1} = A^*\big(x, x_{A^*}, \omega_{A^*}, (\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)\big)$. Wir schreiben $E^{A^*(x, x_{A^*})}$ für die entsprechende Zufallsvariable (über die Münzwurfe von E und A^*) bzw. einfach E^* sofern das Orakel $A^*(x, x_A)$ eindeutig aus dem Kontext hervorgeht. Analog definieren wir Orakelmaschinen für B^*. Beachte, daß die Münzwurfe für A^* zu Beginn zufällig gewählt werden, dann allerdings fest sind. Insbesondere kann E^* verschiedene Orakelanfragen $(\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i) \neq (\alpha'_1, \beta'_1, \ldots, \alpha'_i, \beta'_i)$ für die gleiche Münzwurffolge ω_{A^*} stellen. Dies entspricht der Möglichkeit des *Zurücksetzens* (Reset) einer Ausführung bei fester Münzwurffolge des Orakels. Wir erweitern die Definition einer solchen Orakelmaschine um eine *vollständige Zurücksetzung*, bei der nicht-sichtbar für E^* eine neue Münzwurffolge für das Orakel gewählt wird.

Sei $Z \in \mathbb{G}_q$ und $(z_1, \ldots, z_n)$ eine Repräsentation für Z bezüglich $(g_1, \ldots, g_n)$. Mit einem sogenannten *Proof of Knowledge* [GMR85, BG92] kann man dem anderen Teilnehmer beweisen, daß man eine Repräsentation für Z kennt (Vollständigkeit). Dieser Beweis garantiert umgekehrt, daß — wenn der anderer Teilnehmer mit nicht vernachlässigbarer Wahrscheinlichkeit akzeptiert — der Beweiser tatsächlich eine Repräsentation kennt oder zumindest effizient berechnen kann (Gültigkeit).

Wir formalisieren Proofs of Knowledge allgemeiner für $\mathcal{NP}$*-Relationen.* Sei $R \subseteq \{0,1\}^* \times \{0,1\}^*$ eine binäre Relation. Dann ist R eine $\mathcal{NP}$-Relation, wenn R in Polynomialzeit berechenbar ist und ein Polynom p existiert, so daß $(x, y) \in R$ impliziert $|y| \leq p(|x|)$. Sei $R(x) = \{y \in \{0,1\}^* \mid (x, y) \in R\}$ die Menge der *Zeugen* für x. Dann definieren wir die Sprache L_R durch $L_R = \{x \in \{0,1\}^* \mid R(x) \neq \emptyset\}$. Offenbar ist eine Sprache L genau dann in $\mathcal{NP}$, wenn es eine $\mathcal{NP}$-Relation R mit $L = L_R$ gibt.

Definition 3.3.1 (Proof of Knowledge)
Ein interaktives Protokoll $\mathcal{P}(P, V)$ für die $\mathcal{NP}$-Relation R ist ein Proof of Knowledge mit Fehler $\kappa : L_R \to \mathbb{R}$, wenn gilt:

- *Vollständigkeit: Für alle $x \in L_R$ und alle $w \in R(x)$ gilt:*

$$\mathrm{Prob}\left[\,\mathrm{Acc}_V\,\langle P(x, w) \leftrightarrow_{\mathcal{P}} V(x)\rangle\right] = 1.$$

[4] Alternativ wird in der Literatur die Notation $\widetilde{A}$ bzw. $\widetilde{B}$ für unehrliche Teilnehmer verwendet [FS86]. Wir folgen der Notation in [G95].

[5] Eine ausführliche Darstellung von Orakelturingmaschinen findet man in [BDG95].

- *Gültigkeit: Es gibt ein Orakelmaschine E, so daß für jeden Beweiser P^* und alle $x \in L_R$, $x_{P^*}, x_V \in \{0,1\}^*$ gilt: Wenn*

$$p(x, x_{P^*}, x_V) = \text{Prob}[\,\text{Acc}_V \langle P^*(x, x_{P^*}) \leftrightarrow_{\mathcal{P}} V(x, x_V)\rangle] > \kappa(x),$$

dann gibt $E^{P^*(x,x_{P^*})}$ auf Eingabe x, x_V in erwarteter Schrittzahl

$$\frac{\text{poly}\,(|x|)}{p(x, x_{P^*}, x_V) - \kappa(x)}$$

einen Zeugen $w \in R(x)$ aus. Wir nennen E den Knowledge Extractor.

Wir nennen P bzw. P^* einen Beweiser oder Prover und V einen Verifier oder Prüfer. Die Vollständigkeit garantiert, daß ein ehrlicher Verifier bei Interaktion mit einem ehrlichen Prover, der einen Zeugen kennt, stets akzeptiert. Im allgemeinen ist die Fehlerfunktion $\kappa(x)$ vernachlässigbar, so daß die Gültigkeit besagt, daß ein Beweiser, der den Prüfer mit nicht vernachlässigbarer Wahrscheinlichkeit überzeugt, für unendlich viele Eingaben $x \in L_R$ mit Orakelzugriff auf P^* einen Zeugen in Polynomialzeit berechnen kann. Wenn P^* zusätzlich polynomielle Laufzeit besitzt, erhalten wir einen probabilistischen Polynomalzeitalgorithmus, der für Eingabe x, x_{P^*}, x_V für unendlich viele $x \in L_R$ einen Zeugen berechnet. Dabei haben wir zur Vereinfachung angenommen, daß κ nur von x und nicht von x_{P^*}, x_V abhängt.

Für eine ausführliche Darstellung von Proofs of Knowledge verweisen wir auf die Arbeit von Bellare und Goldreich [BG92]. Dort findet man auch eine äquivalente Definition der Gültigkeit. Bei dieser Definition hat der Knowledge Extractor erwartete polynomielle Laufzeit, findet dafür aber nicht immer einen Zeugen, sondern nur mit Wahrscheinlichkeit $p(x, x_{P^*}, x_V) - \kappa(x)$:

> Es gibt ein Orakelmaschine E mit erwarteter polynomieller Laufzeit, so daß für jeden Beweiser P^* und alle hinreichend langen Eingaben $x \in L_R$ gilt:
>
> $$\text{Prob}\left[\, E^{P^*(x,x_{P^*})}(x, x_V) \in R(x)\right] \geq p(x, x_{P^*}, x_V) - \kappa(x)$$
>
> wobei $p(x, x_{P^*}, x_V) = \text{Prob}[\,\text{Acc}_V \langle P^*(x, x_{P^*}) \leftrightarrow_{\mathcal{P}} V(x, x_V)\rangle]$.

Proofs of Knowledge sind trivial zu realisieren, indem der Beweiser einfach den Zeugen sendet. In vielen kryptographischen Anwendungen ist es dagegen notwendig, daß der Prüfer "so wenig wie möglich" aus einer Protokollausführung lernt. Wir fordern daher, daß alles, was der Verifier lernt, bereits ohne den Beweiser — und damit ohne Kenntnis des Zeugen — berechnet werden kann. Dafür führen wir einen sogenannten Simulator ein, der die gleichen Eingaben wie der Prüfer erhält, und dessen Ausgabe und die Kommunikation des Protokolls identisch verteilt bzw. statistisch oder polynomialzeit-ununterscheidbar sind. Dies gilt sogar, wenn der Prüfer unehrlich ist.

Definition 3.3.2 (Zero-Knowledge Proof of Knowledge)
Sei $\mathcal{P}(P,V)$ ein Proof of Knowledge für die $\mathcal{NP}$-Relation R. Dann heißt $\mathcal{P}(P,V)$ perfekt/ statistisch/ polynomialzeit-zero-knowledge, wenn es für jeden probabilistischen Polynomialzeitalgorithmus V^* einen Algorithmus S^* mit erwarteter polynomieller Laufzeit gibt, so daß die Ensembles

- $\{S^*(x, x_{V^*})\}_{x \in L_R, x_{V^*} \in \{0,1\}^*}$
- $\{\text{view}_{V^*} \langle P(x,w) \leftrightarrow_{\mathcal{P}} V^*(x, x_{V^*})\rangle\}_{x \in L_R, w \in R(x), x_{V^*} \in \{0,1\}^*}$

identisch verteilt/statistisch ununterscheidbar/polynomialzeit-ununterscheidbar sind. Wir nennen S^* einen Zero-Knowledge-Simulator oder kurz Simulator.

Aus der Definition der Begriffe der statistischen und der Polynomialzeit-Ununterscheidbarkeit folgt unmittelbar, daß jeder perfekte zero-knowledge Proof of Knowledge auch statistisch zero-knowledge ist, und jedes statistische zero-knowledge Protokoll auch polynomialzeit-zero-knowledge ist.

Beachte, daß der Simualtor S^* nur im Erwartungswert polynomielle Laufzeit haben muß. Dies ist die üblicherweise in der Literatur verwendete Definition [GMR85]. Goldreich [G95] fordert dagegen, daß der Simulator strikte polynomielle Laufzeit besitzt. Dafür darf der Simulator mit Wahrscheinlichkeit höchstens 1/2 eine ungültige Ausgabe $\perp$ geben. Die Verteilung der Zufallsvariablen $S^*_\perp(x, x_{V^*})$ definiert man dann als die bedingte Wahrscheinlichkeit von $S^*(x, x_{V^*})$, gegeben, daß $S^*(x, x_{V^*}) \neq \perp$. Man zeigt leicht, daß diese Formulierung Definition 3.3.2 impliziert. Ob beide Definitionen äquivalent sind, ist ein offenes Problem. Alle in dieser Diplomarbeit angegebenen zero-knowledge Proofs of Knowledge erfüllen beide Definitionen.

Im allgemeinen arbeitet S^* wie der Knowledge Extractor eines Proofs of Knowledge als Orakelmaschine S^{V^*} mit Orakelzugriff auf V^*. Anderer Zero-Knowledge-Simulatoren sind bis heute nicht bekannt. Wir werden daher im folgenden stets Simulatoren angeben, die als Orakelmaschinen mit Zugriff auf den Verifier arbeiten. Daraus folgt insbesondere, daß S^* nur die Kommunikation simulieren muß, da die Münzwürfe des Verifiers V^* korrekt gewählt werden.

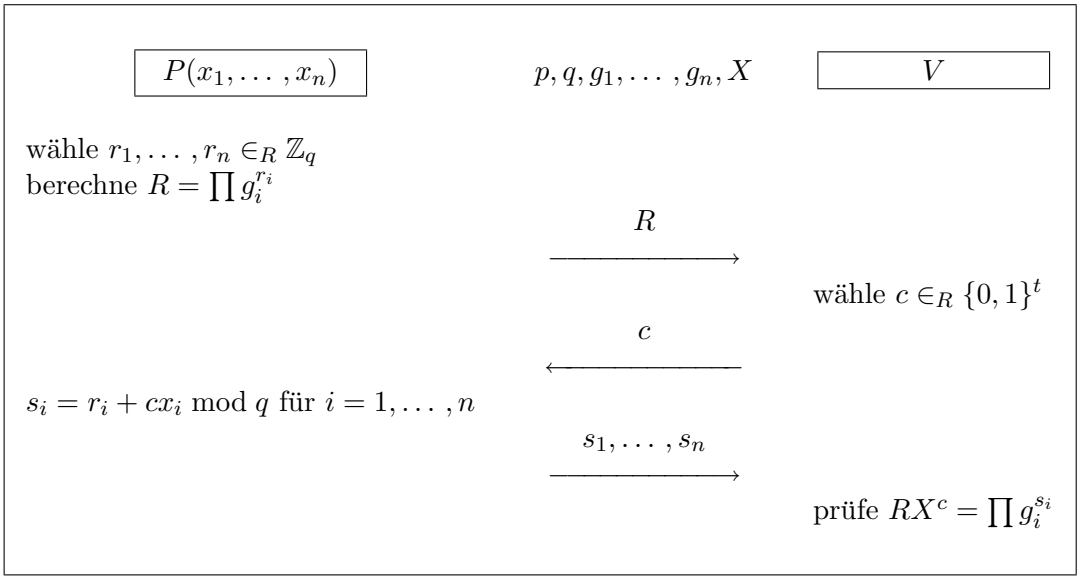

Abbildung 3.1: Proof of Knowledge für $X = \prod g_i^{x_i}$

Als Beispiel betrachten wir einen perfekt zero-knowledge Proof of Knowledge für das Repräsentationsproblem (Abbildung 3.1). Wir zeigen zunächst, daß das Protokoll ein Proof of Knowledge darstellt, wobei der im Protokoll angegebene Sicherheitsparameter t beliebig aus $\{1, \ldots, k-1\}$ für $k = |q|$ sei. Die gemeinsame Eingabe ist $p, q, g_1, \ldots, g_n$ und $X = \prod g_i^{x_i} \bmod p$. Der Beweiser P erhält als zusätzliche Eingabe eine Repräsentation $(x_1, \ldots, x_n)$ von X bezüglich $(g_1, \ldots, g_n)$.

Satz 3.3.3
Das in Abbildung 3.1 angegebene Protokoll ist für $t < k$ ein Proof of Knowledge mit Fehler 2^{-t}.

Beweis. Die Vollständigkeit ist offensichtlich. Betrachten wir die Gültigkeit. Sei

$$p = p(x, x_{P^*}, x_V) = \text{Prob}\left[\,\text{Acc}_V \left\langle P^*(x, x_{P^*}) \leftrightarrow_{\mathcal{P}} V(x, x_V)\right\rangle\right].$$

Die Orakelmaschine E versucht, zwei akzeptierende Protokollabläufe zu finden, bei denen in der ersten Runde jeweils der gleiche Wert R gesendet wird, aber die Nachrichten c, c' des Verifiers verschieden sind. In diesem Fall kann E eine Repräsentation von X berechnen.

Behauptung 1: Sei $p^R = p^R(x, x_{P^*}, x_V) = \text{Prob}\left[\,\text{Acc}_V \left\langle P^*(x, x_{P^*}) \leftrightarrow_{\mathcal{P}} V(x, x_V)\right\rangle \mid \omega_{P^*}\right]$ die bedingte Wahrscheinlichkeit, daß V akzeptiert, gegeben die feste Münzwurffolge ω_{P^*}. Dann gibt es einen Algorithmus E', der in erwarteter polynomieller Laufzeit mit Wahrscheinlichkeit $p^R - 2^{-t}$ eine Repräsentation von X bezüglich $(g_1, \ldots, g_n)$ ausgibt.

Beweis. Beachte, daß mit ω_{P^*} auch $R = P^*(x, x_{P^*}, \omega_{P^*})$ fixiert wird. Wir geben zunächst einen Algorithmus E'' an, der erwartete Laufzeit $\text{poly}(k)/(p^R - 2^{-t})$ hat und stets eine Repräsentation findet. E'' wählt $c \in_R \{0,1\}^t$ zufällig und simuliert den Rest des Protokolls, d.h. fragt das Orakel $P^*(x, x_{P^*}, \omega_{P^*})$ an der Stelle (R, c). Im Erwartungswert benötigen wir $1/p^R$ Versuche, um als Antwort Werte $s_1, \ldots, s_n$ zu erhalten, die die Verifikationsbedingung erfüllen. Wir wiederholen diese Vorgehensweise und finden einen akzeptierenden Ablauf mit $c' \neq c$ in erwarteten $1/(p^R - 2^{-t})$ Wiederholungen. Insgesamt ist die erwartete Laufzeit daher höchstens $\text{poly}(k)/(p^R - 2^{-t})$. Seien $s_1, \ldots, s_n$ und $s'_1, \ldots, s'_n$ die Antworten von P^* im dritten Schritt. Da $c \neq c'$, können wir das Inverse $(c - c')^{-1}$ von $c - c'$ in $\mathbb{Z}_q^*$ berechnen. Aus der Verifikationsbedingung

$$R = X^{-c} \cdot \prod_{i=1}^{n} g_i^{s_i} = X^{-c'} \cdot \prod_{i=1}^{n} g_i^{s'_i}$$

folgt, daß $(x_1^*, \ldots, x_n^*)$ mit $x_i^* = (s_i - s'_i)(c - c')^{-1} \bmod q$ eine Repräsentation von X bezüglich $(g_1, \ldots, g_n)$ ist:

$$\prod_{i=1}^{n} g_i^{s_i - s'_i} = X^{c - c'}$$

Wir wandeln E'' in einen Algorithmus E' um, der erwartete polynomielle Laufzeit besitzt und nur mit Wahrscheinlichkeit $p^R - 2^{-t}$ eine Repräsentation findet. E' wählt einen zufälligen Wert $c \in_R \{0,1\}^t$ und fragt $P^*(x, x_{P^*}, \omega_{P^*})$ an der Stelle (R, c). Falls die Antwort von P^*

die Korrektheitsbedingung nicht erfüllt, stoppt E'. Sonst stoppt E' mit Wahrscheinlichkeit 2^{-t}. Falls E' noch nicht angehalten hat, simuliert er Algorithmus E''. Folglich findet E' eine Repräsentation mit Wahrscheinlichkeit $p^R - 2^{-t}$ (genau dann, wenn E'' aufgerufen wird). Die erwartete Laufzeit beträgt $\text{poly}(k) + (p^R - 2^{-t}) \cdot \text{poly}(k) / (p^R - 2^{-t}) = \text{poly}(k)$. □

Aus E' können wir einen Knowledge Extractor konstruieren, der eine Repräsentation mit erwarteter Laufzeit $\text{poly}(k) / (p - 2^{-t})$ findet. Nachdem eine Münzwurffolge ω_{P^*} für P^* gewählt wurde, simulieren wir E'. Mit Wahrscheinlichkeit $p - 2^{-t}$ findet E' in erwarteter polynomieller Laufzeit eine Repräsentation. Andernfalls setzen wir P^* *vollständig* zurück und wiederholen diese Vorgehensweise. Im Erwartungswert machen wir folglich nur $1/(p - 2^{-t})$ Wiederholungen. ■

Offenbar gibt es einen Beweiser P^*, für den V mit Wahrscheinlichkeit 2^{-t} akzeptiert, unabhängig davon, ob P^* eine Repräsentation kennt oder nicht. Dazu wählt P^* zu Beginn Werte $r_1, \ldots, r_n \in_R \mathbb{Z}_q$ und $\tilde{c} \in_R \{0,1\}^t$ und sendet

$$\tilde{R} = X^{-\tilde{c}} \cdot \prod_{i=1}^{n} g_i^{r_i}$$

an V. Dieser antwortet mit $c \in_R \{0,1\}^t$. Mit Wahrscheinlichkeit 2^{-t} gilt $\tilde{c} = c$. In diesem Fall sendet P^* die Werte $r_1, \ldots, r_n$. Es gilt:

$$\tilde{R} X^c = \tilde{R} X^{\tilde{c}} = \prod_{i=1}^{n} g_i^{r_i}$$

Folglich akzeptiert V. Insbesondere ist die Verteilung der Kommunikation identisch zur Verteilung bei einer Ausführung mit P.

Wir verwenden diese Idee, um zu zeigen, daß der Proof of Knowledge für $t = O(\log |q|)$ perfekt zero-knowledge ist. Allerdings ist dann auch die Erfolgswahrscheinlichkeit des oben angegebenen Angreifers P^* nicht mehr vernachlässigbar. Wir zeigen weiter unten, wie man den Fehler verkleinert, ohne daß die Zero-Knowledge-Eigenschaft verloren geht.

Satz 3.3.4
Für $t = O(\log k)$ ist der Proof of Knowledge in Abbildung 3.1 perfekt zero-knowledge.

Beweis. Wir geben einen Simulator S^* für V^* an. Zu Beginn wird ein Münzwurfband für V^* fixiert. S^* arbeitet genau wie der oben angegebene Beweiser P^*, d.h. er versucht c zu erraten. Mit Wahrscheinlichkeit $2^{-t} = 1/\text{poly}(k)$ ist S^* erfolgreich und kann eine Ausgabe erzeugen, die identisch zu einer "echten" Ausführung des Protokolls ist. Andernfalls wiederholt S^* das Verfahren mit unabhängig gewählten Werten $r_1, \ldots, r_n, \tilde{c}$. Da die Ratewahrscheinlichkeit $1/\text{poly}(k)$ beträgt, muß S^* im Erwartungswert nur $\text{poly}(k)$ viele Wiederholungen ausführen. ■

Durch die Beschränkung an t erreichen wir erwartete polynomielle Laufzeit des Simulators. Man kann den Fehler des Protokolls auf 2^{-tm} senken, indem man $m = m(k) = \text{poly}(k)$ unabhängige Ausführungen (bei gleicher Eingabe x) sequentiell wiederholt, und der Verifier

genau dann akzeptiert, wenn er in allen diesen Wiederholungen akzeptiert. Der Simulator arbeitet bei jeder dieser Ausführungen wie für den Fall $t = O(\log k)$. Ist er bei einer dieser Wiederholungen nicht erfolgreich, braucht er lediglich zum Beginn dieser Ausführung zurückzusetzen. Daher benötigt er im Erwartungswert nur $m(k) \cdot \text{poly}(k) = \text{poly}(k)$ viele Wiederholungen. Man sieht leicht, daß das Protokoll ein Proof of Knowledge mit Fehler 2^{-tm} ist, da es genügt, wie in Beweis zu Satz 3.3.3 zwei Folgen $(c_1, \ldots, c_m) \neq (c'_1, \ldots, c'_m)$ zu finden.

Bei der sequentiellen Wiederholung steigt die Anzahl der Runden um den Faktor m. Unter der Annahme, daß sogenannte Familien von Clawfree Funktionen existieren, kann man diese Ausführungen auf fünf Runden parallelisieren [GK96]. Solche Familien von Clawfree Funktionen existieren beispielsweise unter der Diskreten-Logarithmus-Annahme bzw. unter der allgemeineren Annahme, daß Black-Box-Reduktionen mit bestimmten Eigenschaften existieren [F98b]. In diesem Fall ist das Protokoll allerdings nur noch polynomialzeit-zero-knowledge bzw. statistisch zero-knowledge, sofern der Prover auf polynomielle Laufzeit beschränkt wird.

Der Proof of Knowledge aus Abbildung 3.1 hat die weitere Eigenschaft, daß *witness-indistinguishable* [FS90] ist, d.h. daß die Verteilung der Kommunikation nicht von dem spezifischen Zeugen des ehrlichen Beweisers abhängt. Offenbar ist jeder perfekte zero-knowledge Proof of Knowledge auch witness-indistinguishable, da bei zero-knowledge Protokollen die Kommunikation von einem Simulator ohne Kenntnis eines Zeugen erzeugt werden kann. Um den Fehler bei zero-knowledge Proofs of Knowledge klein zu machen, benötigen wir sequentielle oder parallele Wiederholungen. Wir werden sehen, daß der Proof of Knowledge für das Repräsentationsproblem dagegen für alle $t < k$ witness-indistinguishable ist. Folglich erhält man einen kleiner Fehler ohne Wiederholung des Protokolls, und damit ohne zusätzlichen Rechen- und Kommunikationsaufwand. Genügt in einer Anwendung der schwächere Begriff des witness-indistinguishable Proof of Knowledge, so ist dieser einem zero-knowledge Proof of Knowledge aus Effizienzgründen daher vorzuziehen.

Definition 3.3.5 (Witness-Indistinguishable Proof of Knowledge)
Ein Proof of Knowledge $\mathcal{P}(P, V)$ für die $\mathcal{NP}$-Relation R ist witness-indistinguishable, wenn für alle probabilistischen Polynomialzeitalgorithmen V^* die Ensembles

- $\{\text{view}_{V^*} \langle P(w_x^1) \leftrightarrow_{\mathcal{P}} V^*(x_{V^*}) \rangle\}_{x \in L_R, x_{V^*} \in \{0,1\}^*}$
- $\{\text{view}_{V^*} \langle P(w_x^2) \leftrightarrow_{\mathcal{P}} V^*(x_{V^*}) \rangle\}_{x \in L_R, x_{V^*} \in \{0,1\}^*}$

für alle $\{w_x^1 \in R(x) \mid x \in L_R\}$ und $\{w_x^2 \in R(x) \mid x \in L_R\}$ identisch verteilt sind.

Es gilt:

Satz 3.3.6 (Okamoto [O92])
Der in Abbildung 3.1 angegebene Proof of Knowledge mit Fehler 2^{-t} ist für alle $t < k$ witness-indistinguishable.

Wir schreiben im folgenden $\text{WIPOK}_{(g_1,\ldots,g_n)}(X)$ oder kurz $\text{POK}_{(g_1,\ldots,g_n)}(X)$ für einen Proof of Knowledge einer Repräsentation von X bezüglich $(g_1, \ldots, g_n)$.

Wir betrachten als Beispiel für eine Anwendung eines witness-indistinguishable Proof of Knowledge ein *Hinterlegungsprotokoll* zwischen A und B. Teilnehmer A besitze einen Wert x. In der Hinterlegungsphase des Protokolls sendet A an B einen (mit Zufallsbits ω erzeugten) Wert $C = C(\omega, x)$, so daß B keine Informationen über x erhält.[6] In der Aufdeckphase sendet A einen Wert $D(C, \omega, x)$ zusammen mit x, so daß B effizient überprüfen kann, daß C korrekt gebildet wurde. Ferner gelte unter einer kryptographischen Annahme, daß A in der Aufdeckphase keine verschiedenen Werte x, x' korrekt aufdecken kann.

Wir verzichten auf eine formale Definition, da wir Hinterlegungsprotokolle nur implizit als Spezialfall des Repräsentationsproblems verwenden. Wir betrachten ein solches Hinterlegungsprotokoll für einen Wert $x \in \mathbb{Z}_q$ nach Pedersen [P91]. Gegeben seien öffentliche Parameter p, q und $g, h \in_R \mathbb{G}_q \setminus \{1\}$. In der Hinterlegungsphase wählt A zufällig $w \in_R \mathbb{Z}_q$, sendet $C = g^x h^w$ und führt zusammen mit B einen witness-indistinguishable Proof of Knowledge für C durch. Dabei erhält B keine Informationen über x, da der Wert C uniform in $\mathbb{G}_q$ verteilt ist und andererseits für jedes $x' \in \mathbb{Z}_q$ genau ein $w' \in \mathbb{Z}_q$ mit $C = g^{x'} h^{w'}$ existiert, und damit der witness-indistinguishable Proof of Knowledge keine Information über x preisgibt. Umgekehrt kann A unter der Diskreten-Logarithmus-Annahme nicht zwei verschiedene Werte x, x' finden, so daß $C = g^x h^w = g^{x'} h^{w'}$, da dann $\log_g h = (x - x')(w' - w)^{-1} \bmod q$.

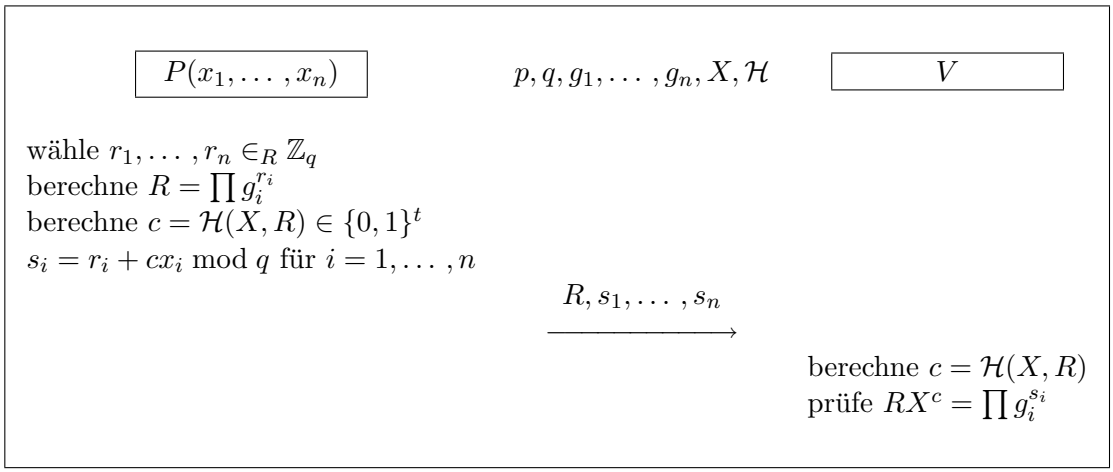

Abbildung 3.2: Proof of Knowledge für $X = \prod g_i^{x_i}$ mit idealer Hashfunktion

Die interaktiven Proofs of Knowledge kann man unter der Annahme der Existenz sogenannter *idealer* Hashfunktionen $\mathcal{H} : \{0,1\}^* \to \{0,1\}^t$ in nicht-interaktive Protokolle umwandeln [FS86, BR93]. Eine solche ideale (öffentliche) Hashfunktion $\mathcal{H}$ kann man als Orakel beschreiben, daß für jeden Wert x einen Zufallswert $\mathcal{H}(x) \in_R \{0,1\}^t$ ausgibt und für gleiche Orakelanfragen jeweils den gleichen Wert reproduziert. Die nicht-interaktive Variante eines Proofs of Knowledge ist in Abbildung 3.2 angegeben. Dabei ersetzen wir den Wert $c \in_R \{0,1\}^t$ des Verifiers durch den Hashwert $\mathcal{H}(X, R) \in_R \{0,1\}^t$ der öffentlichen Hashfunktion $\mathcal{H}$. Das Protokoll ist ein Proof of Knowledge, sofern man die Wahrscheinlichkeit auch über die zufällige Wahl der idealen Hashfunktion bildet. Ferner bleibt die Eigenschaft erhalten, für alle

[6] Dabei haben wir zur Vereinfachung angenommen, daß diese Phase nicht-interaktiv abläuft. Die Verallgemeinerung auf interaktive Hinterlegungsphasen folgt unmittelbar.

$t < k$ witness-indistinguishable zu sein. Für $t = O(\log k)$ ist der Proof of Knowledge zero-knowledge. Die sequentielle Wiederholung entspricht dann der Auswertung der Hashfunktion $\mathcal{H}$ an den Stellen (X, R_i, i) für $i = 1, \ldots, m$. In diesem Fall werden alle m Runden parallel nicht-interaktiv ausgeführt, so daß wir keine Modifikation des Protokolls zur Senkung der Rundenanzahl benötigen.

Die Verwendung solcher idealen Hashfunktionen setzt eine sehr starke Annahme voraus. Andererseits vereinfacht diese Annahme den Entwurf von kryptographischen Protokollen und Verfahren wesentlich [BR93]. In der Praxis verwendet man geeignete Hashfunktionen, die durch Modifikation aus den bekannten Hashfunktionen wie SHA-1 [SHA] hervorgehen.

3.4 Fairer Austausch von Geheimnissen

Ein Protokoll $\mathcal{A}(A, B)$ für den fairen Austausch von Geheimnissen besteht aus zwei interaktiven Teilnehmern A und B, wobei A das Geheimnis $x = (x_1, \ldots, x_n) \in \mathcal{X}$ und B das Geheimnis $y = (y_1, \ldots, y_n) \in \mathcal{Y}$ besitzt. Dabei sind $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^*$ und $x_i, y_i \in \{0,1\}^b$ für alle $i = 1, \ldots, n$. Wir nennen b die *Blockgröße* und x_i bzw. y_i einen *Block* oder *Teil* des Geheimnisses. Dabei haben wir zur Vereinfachung angenommen, daß sowohl x als auch y aus genau n Blöcken zu b bestehen. Für den allgemeineren Fall kann man die folgenden Protokolle und Sätze entsprechend anpassen. Wir nehmen an, daß A den Bitstring $\mathcal{K}_{A,\mathcal{Y}}(y)$ als à-priori-Wissen über B's Geheimnis kennt, und umgekehrt B à-priori-Informationen $\mathcal{K}_{B,\mathcal{X}}(x)$ über x erhält. Diese à-priori-Informationen ergeben sich beispielsweise aus der Ausführung des Austauschprotokolls als Unterprotokolls oder aus der öffentlich bekannten Verteilung der Geheimnisse.

Als gemeinsame Eingabe erhalten A und B die öffentlichen Systemparameter, die beispielsweise den Sicherheitsparameter k, Primzahlen, Generatoren, Blöckgröße $b = b(k) = O(\log k)$, Anzahl $n = n(k) = \mathrm{poly}(k)$ der Teile, öffentliche Schlüssel, die zu unterschreibende Nachricht usw. enthalten. Diese Systemparameter setzen sich aus den beiden öffentlichen Parametern pub_A und pub_B der Teilnehmer zusammen. Zu diesen öffentlichen Parametern gibt es individuelle, geheime Parameter priv_A und priv_B für A und B, die z.B. jeweils den geheimen Schlüssel des Teilnehmers enthalten. Wir nehmen an, daß diese Parameter pub_A und priv_A für A sowie pub_B und priv_B für B durch "Instanzengeneratoren" $I_A(1^k)$ und $I_B(1^k)$ erzeugt werden. Zur Abkürzung schreiben wir $\mathsf{pub}_{A,B}$ für $(\mathsf{pub}_A, \mathsf{pub}_B)$. Ferner gebe es probabilistische Polynomialzeitalgorithmen $\mathcal{S}_A$, $\mathcal{S}_B$, die auf Eingabe pub_A und priv_A bzw. pub_B und priv_B die Geheimnisse x und y mit den zugehörigen Informationen $\mathcal{K}_{B,\mathcal{X}}(x)$ und $\mathcal{K}_{A,\mathcal{Y}}(y)$ erzeugen. Ein solcher "Geheimnisgenerator" ist im Fall des Austauschs digitaler Unterschriften beispielsweise der Algorithmus zur Erzeugung einer Unterschrift.

Um zu überprüfen, ob ein erhaltener Teil x_i ein korrekter Teil des Geheimnisses ist, muß es öffentliche Prädikate $\mathcal{V}_A$ und $\mathcal{V}_B$ für pub_A bzw. pub_B geben. Im Fall des fairen Austausches digitaler Unterschriften ist dieses Prädikat genau dann wahr, wenn die erhaltenen Teile zu einer korrekten Unterschrift (für eine gegebene Nachricht und dem öffentlichen Schlüssel des Unterschreibers) ergänzt werden können. Wir definieren dieses Prädikat später formal. Wir nennen ein Paar $(\mathcal{V}_X, \mathcal{S}_X)$ ein *Geheimnis*. Beachte, daß wir die Ausgabe von $\mathcal{S}_X$ ebenfalls als Geheimnis bezeichnen.

Informell soll ein faires Austauschprotokoll garantieren, daß zu jedem Zeitpunkt der Ausführung A soviel über y gelernt hat, wie umgekehrt B über x. Als Grundlage von Austauschprotokollen wählen wir *sichere Release-Protokolle*. Bei einem Release-Protokoll zwischen Sender A und Empfänger B gibt Teilnehmer A mit Geheimnis $x = (x_1, \ldots, x_n)$ in jeder der n Runden Teilnehmer B einen neuen Teil x_i seines Geheimnisses, so daß folgende Eigenschaften gelten:

1. Vollständigkeit: Wenn A und B dem Protokoll folgen, dann erhält B am Ende der n Runden das Geheimnis x.
2. Korrektheit: Wenn A in einer Runde einen (gemäß $\mathcal{V}_A$) falschen Wert $\tilde{x}_i$ an B weitergibt, wird dies sofort von B entdeckt.
3. Geheimhaltung: Zu jedem Zeitpunkt der Ausführung hat B nicht mehr Informationen über x erhalten, als sich aus den bisherigen ausdrücklich preisgegebenen Teil und $\mathcal{K}_{B,\mathcal{X}}(x)$ ergibt.

Aus einem sicheren Release-Protokoll zwischen Sender A und Empfänger B und einem sicheren Release-Protokoll zwischen Sender B und Empfänger A erhalten wir ein sicheres Austauschprotokoll zwischen A und B, indem jeweils abwechselnd eine Runde des entsprechenden Release-Protokolls ausgeführt wird. Als zusätzliche Annahme benötigt man die uniforme Ergänzungskomplexität der Geheimnisse. Ferner müssen die beiden Release-Protokolle symmetrisch sein, d.h. die Verteilung der Kommunikation muß in beiden Release-Protokollen identisch sein. Dies ist insbesondere erfüllt, wenn das gleiche Unterschriftenschema und das gleiche Release-Protokoll verwendet wird.

Bevor wir auf die Definition der uniformen Ergänzungskomplexität und der Symmetrie eingehen, definieren wir zunächst sichere Release-Protokolle formal. Während sich die beiden ersten Eigenschaften unmittelbar aus der informellen Beschreibung ergeben, ist die dritte Anforderung nicht offensichtlich zu formalisieren. Wir wählen den Simulator-Ansatz von Micali und Rogaway [MR91] und Beaver [B91] für sichere Mehrparteienprotokolle (siehe auch [CFGN96]). Informell besagt dieser Ansatz daß es für jeden probabilistischen Polynomialzeitalgorithmus B^* einen Simulator $S^* = S^{B^*}$ gibt, der die gleichen Eingaben wie B^* erhält, und in erwarteter polynomieller Laufzeit eine Ausgabe erzeugt, die statistisch ununterscheidbar von der Sicht von B^* bei Kommunikation mit dem ehrlichen A ist. Wie bei zero-knowledge Proofs of Knowledge ist die Idee dabei, daß folglich alles, was B^* aus der Kommunikation mit $A(x)$ lernt, bereits ohne die Kenntnis von x durch S^* berechnet werden kann.

Wir spezifieren die Arbeitsweise des Simulators. S^* erhält die gleichen Eingaben wie B^* und simuliert zunächst B^* bis dieser eine Anfrage open(j) stellt, um den j-ten Block von A's Geheimnis x zu erhalten. Sei release(open(j)) $= (x_j, j)$. Nach Erhalt von x_j setzt S^* die Simulation von B^* fort. S^* stoppt, wenn B^* anhält, und gibt die simulierte Kommunikation aus.[7] Wir lassen im folgenden auch zu, daß A den Block bestimmt, den er an B gibt. In diesem Fall gibt B lediglich einen open($\cdot$)-Befehl an A weiter, der dann ein j wählt und den Block x_j und die Position j an B sendet. Wir nennen S^* einen *Release-Protokoll-Simulator für* B^*.

[7] Damgård [D95] betrachtet nur Protokolle, bei denen die Blöcke in fest vorgegebener Reihenfolge *nicht-adaptiv* herausgegeben werden. In diesem Fall kann man dem Release-Protokoll-Simulator die ersten i Blöcke als zusätzliche Eingabe geben, ohne eine "On-line-Simulation" zu verwenden.

Die Korrektheitsbedingung eines Release-Protokolls garantiert, daß B sofort bemerkt, wenn A bzw. A^* einen inkorrekten Wert sendet. Zur Formalisierung führen wir folgende Notation ein: Für $(\mathsf{pub}_A, \mathsf{priv}_A) \in [I_A(1^k)]$ gelte $x \in \mathcal{V}_A(\mathsf{pub}_A)$ genau dann, wenn es ein $\mathcal{K}_{B,\mathcal{X}}(x)$ gibt, so daß $(x, \mathcal{K}_{B,\mathcal{X}}(x)) \in [\mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)]$. Ferner gelte

$$\big(\mathrm{release}(\mathrm{open}(j_1)), \ldots, \mathrm{release}(\mathrm{open}(j_i))\big) \in \mathcal{V}_A(i, \mathsf{pub}_A),$$

genau dann, wenn $(\mathrm{release}(\mathrm{open}(j_1)), \ldots, \mathrm{release}(\mathrm{open}(j_i)))$ zu einem korrektem Geheimnis bzw. einer korrekten Unterschrift ergänzt werden können, d.h. wenn es ein $x' \in \mathcal{V}_A(\mathsf{pub}_A)$ gibt, so daß $x'_{j_\ell} = \mathrm{release}(\mathrm{open}(j_\ell))$ für alle $\ell = 1, \ldots, i$. Für gegebene $(\mathsf{pub}_A, \mathsf{priv}_A) \in [I_A(1^k)]$, $(x, \mathcal{K}_{B,\mathcal{X}}(x)) \in [\mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)]$ bezeichne

$$\mathrm{psecret}_B^i \langle A(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle$$

die Zufallsvariable (über die Münzwürfe von A und B), die die auf die ersten i Befehle open$(\cdot)$ erhaltenen Blöcke bei Ausführung des Release-Protokolls $\mathcal{R}(A, B)$ mit den gegebenen Parametern beschreibt. Falls das Protokoll vorzeitig stoppt, sei die Ausgabe $\perp$ mit $\perp \notin \mathcal{V}_A(i, \mathsf{pub}_A)$ für alle i, pub_A. Entsprechend bezeichne

$$\mathrm{secret}_B \langle A(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle$$

alle n erhaltenen Teile bzw. $\perp$. Die Zufallsvariable

$$\mathrm{pass}_B^i \langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle$$

sei genau dann wahr, wenn B den $(i+1)$-ten open$(\cdot)$-Befehl an A^* weitergibt. Wir sagen, daß B die i-te *Release-Runde beendet*, wenn er den $(i+1)$-ten Befehl open$(\cdot)$ an A^* sendet. Mit dieser Nachricht *beginnt* die $(i+1)$-te Release-Runde. Zur Vereinfachung der Notation nehmen wir an, daß B ein spezielles Ende-Symbol sendet, wenn er das n-te Teil akzeptiert, und daß damit die $(n+1)$-te Release-Runde beginnt. Somit ist $\mathrm{pass}_B^i \langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle$ genau dann wahr, wenn B die $(i+1)$-te Release-Runde beginnt. Die Korrektheitsbedingung läßt sich damit charakterisieren, daß B nur mit vernachlässigbarer Wahrscheinlichkeit die $(i+1)$-te Release-Runde beginnt, wenn die ersten i Teile *nicht* zu einem korrekten Geheimnis ergänzt werden können. Dabei nehmen wir an, daß die Parameter pub_A von einer öffentlichen Stelle authentifiziert werden, so daß das Protokoll stets mit korrekten öffentlichen Parametern ausgeführt wird. Ferner erhält A^* die geheimen Parameter priv_A von A. Mit

$$\mathrm{pview}_B^i \langle A(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle$$

bezeichnen wir die Kommunikation zwischen A und B bis Release-Runde i einschließlich.

Definition 3.4.1 (Sicheres Release-Protokoll)
Ein Release-Protokoll $\mathcal{R}(A, B)$ für Geheimnis $(\mathcal{V}_A, \mathcal{S}_A)$ ist sicher, wenn gilt:

- *Vollständigkeit:*

$$\mathrm{Prob}\big[\omega_A \leftarrow \Omega_A,\, \omega_B \leftarrow \Omega_B,\, (\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),$$
$$(x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A) \;:$$
$$\mathrm{secret}_B \langle A(x, \mathsf{pub}_A, \mathsf{priv}_A, \omega_A) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A, \omega_B)\rangle \in \mathcal{V}_A(\mathsf{pub}_A)] = 1$$

- *Korrektheit: Für alle* $i : \mathbb{N} \to \mathbb{N}$ *mit* $i(k) \in \{1, \ldots, n(k)\}$ *ist*

$$\begin{aligned}\text{Prob}\big[&\omega_{A^*} \leftarrow \Omega_{A^*},\ \omega_B \leftarrow \Omega_B, (\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\\ &(x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)\ :\\ &\text{pass}_B^{i(k)}\langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A, \omega_{A^*}) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A, \omega_B)\rangle \wedge\\ &\text{psecret}_B^{i(k)}\langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A, \omega_{A^*}) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A, \omega_B)\rangle \notin \mathcal{V}_A(i(k), \mathsf{pub}_A)\big]\end{aligned}$$

  vernachlässigbar in 1^k.

- *Geheimhaltung: Für jeden Algorithmus* B^* *gibt es einen Release-Protokoll-Simulator* $S^* = S^{B^*}$ *für* B^* *mit erwarteter polynomieller Laufzeit, so daß für alle* $(\mathsf{pub}_A, \mathsf{priv}_A) \in [I_A(1^k)]$, $(x, \mathcal{K}_{B,\mathcal{X}}(x)) \in [\mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)]$ *und* $i : \mathbb{N} \to \mathbb{N}$ *mit* $i(k) \in \{1, \ldots, n(k)\}$ *die Ensembles*

  - $\{\text{pview}_{B^*}^{i(k)}\langle A(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}} B^*(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle\}_{\mathsf{pub}_A, \mathsf{priv}_A, x, \mathcal{K}_{B,\mathcal{X}}(x), i}$
  - $\{S^*(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\}_{\mathsf{pub}_A, \mathsf{priv}_A, x, \mathcal{K}_{B,\mathcal{X}}(x), i}$

  statistisch ununterscheidbar sind.

Wie bei zero-knowledge Proofs of Knowledge kann man den Release-Protokoll-Simulator mit strikter polynomieller Laufzeit und $\perp$-Ausgabe definieren. Unsere Protokolle bleiben unter dieser Definition sicher.

Aus sicheren Release-Protokollen erhält man ein sicheres Austauschprotokoll, indem man abwechselnd jeweils eine Runde des sicheren Release-Protokolls ausführt. Als zusätzliche Annahme benötigt man die *uniforme Ergänzungskomplexität* des Geheimnisses. Informell besagt die uniforme Ergänzungskomplexität: Wenn man für einen nicht vernachlässigbaren Anteil der Geheimnisse aus einem Teil $(x_{j_1}, \ldots, x_{j_i})$ das gesamte Geheimnis $(x_1, \ldots, x_n)$ vollständig rekonstruieren kann, dann kann man dies fast immer für alle Geheimnisse. Folglich ist die Berechnung des vollständigen Geheimnisses $(x_1, \ldots, x_n)$ aus $(x_{j_1}, \ldots, x_{j_i})$ für alle Geheimnisse ungefähr gleich schwierig. Wir geben die Definition nur für den Fall der Herausgabe in fester Reihenfolge:

Definition 3.4.2 (Uniforme Ergänzungskomplexität eines Geheimnisses)
Ein Geheimnis $(\mathcal{V}_A, \mathcal{S}_A)$ *hat uniforme Ergänzungskomplexität, wenn gilt: Wenn es einen probabilistischen Polynomialzeitalgorithmus* C *gibt, so daß für ein* $i : \mathbb{N} \to \mathbb{N}$ *mit* $i(k) \in \{1, \ldots, n(k)\}$ *und eine Funktion* j *mit* $j(k) = \{j_1(k)), \ldots, j_{i(k)}(k)\} \subseteq \{1, \ldots, n(k)\}$ *für alle* k *aus einer unendlichen Menge* $K \subseteq \mathbb{N}$ *gilt*

$$\begin{aligned}\text{Prob}\big[&(\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\ (x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A),\\ &x' \leftarrow C(\mathsf{pub}_A, \mathcal{K}_{B,\mathcal{X}}(x), x_{j_1(k)}, \ldots, x_{j_{i(k)}(k)})\ :\\ &x'_{j_1(k)} = x_{j_1(k)}, \ldots, x'_{j_{i(k)}(k)} = x_{j_{i(k)}(k)} \wedge x' \in \mathcal{V}_A(\mathsf{pub}_A)\big] \geq \frac{1}{p(k)}\end{aligned}$$

für ein Polynom p, dann gibt es einen probabilistischen Polynomialzeitalgorithmus D, so daß

$$\begin{aligned}\text{Prob}\big[&(\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\ (x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A),\\ &x' \leftarrow D(\mathsf{pub}_A, \mathcal{K}_{B,\mathcal{X}}(x), x_{j_1(k)}, \ldots, x_{j_{i(k)}(k)}) \ :\\ &x'_{j_1(k)} = x_{j_1(k)}, \ldots, x'_{j_{i(k)}(k)} = x_{j_{i(k)}(k)} \ \wedge\ x' \in \mathcal{V}_A(\mathsf{pub}_A)\big] \geq 1 - \frac{1}{q(k)}\end{aligned}$$

für alle Polynome q und alle hinreichend großen $k \in K$.

Die Forderung, daß die uniforme Ergänzungskomplexität für *alle* Teile des Geheimnisses gilt, kann im Fall adaptiver Herausgaben abgeschwächt werden, indem man die Wahrscheinlichkeit über die zufällige Wahl der ersten $i(k)$ herausgegebene Teile gemäß Release-Protokoll bildet.

Austauschprotokolle lassen sich allgmemeiner aus zwei *verschiedenen* Release-Protokollen konstruieren, z.B. wenn die Teilnehmer verschiedene Unterschriftenschemata verwenden. In diesem Fall können wir die Sicherheit nachweisen, wenn die übertragenen Parameter und Daten in beiden Fällen identisch verteilt sind:

Definition 3.4.3 (Symmetrische Release-Protokolle)
Seien $\mathcal{R}_A(A,B)$ und $\mathcal{R}_B(B,A)$ Release-Protokolle für Geheimnisse $(\mathcal{V}_A, \mathcal{S}_A)$ bzw. $(\mathcal{V}_B, \mathcal{S}_B)$. Dann heißen $\mathcal{R}_A(A,B)$ und $\mathcal{R}_B(B,A)$ symmetrisch, wenn $I_A(1^k)$ und $I_B(1^k)$ bzw. $\mathcal{S}_A$ und $\mathcal{S}_B$ identisch verteilt sind, und für alle $(\mathsf{pub}_A, \mathsf{priv}_A) \in [I_A(1^k)]$, $(x, \mathcal{K}_{B,\mathcal{X}}(x)) \in [\mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)]$ und $(\mathsf{pub}_B, \mathsf{priv}_B) \in [I_B(1^k)]$, $(y, \mathcal{K}_{A,\mathcal{Y}}(y)) \in [\mathcal{S}_B(\mathsf{pub}_B, \mathsf{priv}_B)]$ die Ensembles

- $\{\text{view}_B \langle A(x, \mathsf{pub}_A, \mathsf{priv}_A) \leftrightarrow_{\mathcal{R}_A} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A)\rangle\}_{\mathsf{pub}_A, \mathsf{priv}_A, x, \mathcal{K}_{B,\mathcal{X}}(x)}$
- $\{\text{view}_A \langle B(y, \mathsf{pub}_B, \mathsf{priv}_B) \leftrightarrow_{\mathcal{R}_B} A(\mathcal{K}_{A,\mathcal{Y}}(y), \mathsf{pub}_B)\rangle\}_{\mathsf{pub}_B, \mathsf{priv}_B, y, \mathcal{K}_{A,\mathcal{Y}}(y)}$

identisch verteilt sind.

Seien $\mathcal{R}_A(A,B)$ und $\mathcal{R}_B(B,A)$ sichere, symmetrische Release-Protokolle für $(\mathcal{V}_A, \mathcal{S}_A)$ bzw. für $(\mathcal{V}_B, \mathcal{S}_B)$. Wenn A und B jeweils abwechselnd eine Release-Runde des sicheren Release-Protokolls $\mathcal{R}_A$ bzw. $\mathcal{R}_B$ ausführen, erhalten wir ein *von $\mathcal{R}_A, \mathcal{R}_B$ induziertes Austauschprotokoll.*

Definition 3.4.4 (Faires induziertes Austauschprotokoll)
Seien $\mathcal{R}_A(A,B)$ und $\mathcal{R}_B(B,A)$ sichere, symmetrische Release-Protokolle für die Geheimnisse $(\mathcal{V}_A, \mathcal{S}_A)$ und $(\mathcal{V}_B, \mathcal{S}_B)$. Dann heißt das von $\mathcal{R}_A(A,B)$, $\mathcal{R}_B(B,A)$ induzierte Austauschprotokoll $\mathcal{A}(A,B)$ fair für $(\mathcal{V}_A, \mathcal{S}_A)$ und $(\mathcal{V}_B, \mathcal{S}_B)$, wenn es für jeden probabilistischen Polynomialzeitalgorithmus B^{**} und jede Funktion $i : \mathbb{N} \to \{1, \ldots, n(k)\}$ mit

$$\begin{aligned}\text{Prob}\big[&(\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\ (x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A),\\ &(\mathsf{pub}_B, \mathsf{priv}_B) \leftarrow I_B(1^k),\ (y, \mathcal{K}_{A,\mathcal{Y}}(y)) \leftarrow \mathcal{S}_B(\mathsf{pub}_B, \mathsf{priv}_B),\\ &v \leftarrow \text{pview}^{i(k)}_{B^*} \left\langle A(x, \mathcal{K}_{A,\mathcal{Y}}(y), \mathsf{priv}_A, \mathsf{pub}_{A,B}) \leftrightarrow_{\mathcal{A}} B^*(y, \mathcal{K}_{B,\mathcal{X}}(x), \mathsf{priv}_B, \mathsf{pub}_{A,B})\right\rangle,\\ &x^{**} \leftarrow B^{**}(\mathsf{priv}_B, \mathsf{pub}_{A,B}, \mathcal{K}_{B,\mathcal{X}}(x), v) \ :\ x^{**} \in \mathcal{V}_A(\mathsf{pub}_A)\big] \geq \frac{1}{p(k)}\end{aligned}$$

für ein Polynom p und k in einer unendlichen Menge $K \subseteq \mathbb{N}$ einen probabilistischen Polynomialzeitalgorithmus A_0 gibt, so daß

$$\begin{aligned}\text{Prob}\big[&(\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\ (x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A),\\ &(\mathsf{pub}_B, \mathsf{priv}_B) \leftarrow I_B(1^k),\ (y, \mathcal{K}_{A,\mathcal{Y}}(y)) \leftarrow \mathcal{S}_B(\mathsf{pub}_B, \mathsf{priv}_B),\\ &v \leftarrow \text{pview}_{B^*}^{i(k)} \left\langle A(x, \mathcal{K}_{A,\mathcal{Y}}(y), \mathsf{priv}_A, \mathsf{pub}_{A,B}) \leftrightarrow_{\mathcal{A}} B^*(y, \mathcal{K}_{B,\mathcal{X}}(x), \mathsf{priv}_B, \mathsf{pub}_{A,B})\right\rangle,\\ &y_0 \leftarrow A_0(\mathsf{pub}_{A,B}, \mathcal{K}_{A,\mathcal{Y}}(y), v)\ :\ y_0 \in \mathcal{V}_B(\mathsf{pub}_B)\big] \geq 1 - \frac{1}{q(k)}\end{aligned}$$

für alle Polynome q und alle hinreichend großen $k \in K$. Eine entsprechende Eigenschaft gelte zusätzlich für B's Geheimnis.

Wir verweisen auf die Arbeit von Damgård [D95] für den formalen Beweis, daß die Sicherheit der symmetrischen Release-Protokolle zusammen mit der uniformen Ergänzungskomplexität die Fairness des induzierten Austauschprotokolls impliziert. Wir skizzieren die Beweisidee nur für den Fall, daß das Geheimnis in fester Reihenfolge herausgegeben wird. Anschließend zeigen wir, daß der Beweis auch im Fall der adaptiven Herausgabe gültig bleibt.

Satz 3.4.5 (Damgård [D95])
Seien $\mathcal{R}_A(A,B)$ und $\mathcal{R}_B(B,A)$ sichere, symmetrische Release-Protokolle für die Geheimnisse $(\mathcal{V}_A, \mathcal{S}_A)$ und $(\mathcal{V}_B, \mathcal{S}_B)$ mit uniformer Ergänzungskomplexität. Dann ist das von $\mathcal{R}_A, \mathcal{R}_B$ induzierte Austauschprotokoll $\mathcal{A}(A,B)$ für $(\mathcal{V}_A, \mathcal{S}_A)$ und $(\mathcal{V}_B, \mathcal{S}_B)$ fair.

Beweis (Idee). Angenommen, es gibt einen Polynomialzeitalgorithmus B^{**}, der bei einer Protokollausführung aus i Teilen des Geheimnisses x mit Wahrscheinlichkeit mindestens $1/\operatorname{poly}(k)$ bereits das gesamte Geheimnis x berechnen kann. Wir konstruieren daraus wie in Definition 3.4.4 gefordert einen Polynomialzeitalgorithmus A_0, der aus einer Kommunikation zwischen A und B^* und den öffentlichen Parametern das gesamte Geheimnis y von B^* mit Wahrscheinlichkeit $1 - 1/\operatorname{poly}(k)$ berechnen kann.

Aus B^{**} entwickeln wir zunächst einen Algorithmus A_0^*, der als Eingabe i Blöcke eines Geheimnisses und die zugehörigen öffentlichen Parameter erhält, und daraus fast immer ein vollständiges Geheimnis extrahiert. Dazu erzeugen wir gemäß Vorschrift $I_B(1^k)$ öffentliche Parameter pub_B und private Parameter priv_B, sowie ein Geheimnis $(y, \mathcal{K}_{A,\mathcal{Y}}(y)) \leftarrow \mathcal{S}_B(\mathsf{pub}_B, \mathsf{priv}_B)$. Seien pub_X, $\mathcal{K}_{X,\mathcal{Z}}(x)$ und $s_{j_1}, \ldots, s_{j_i}$ die Eingabe von A_0^*. Da das zugrundeliegende Release-Protokoll $\mathcal{R}_A$ sicher ist, können wir einen Protokollablauf zwischen X in der Rolle von A (mit Eingabe pub_X, priv_X usw.) und B^{**} (mit Eingabe $\mathsf{pub}_B, \mathsf{priv}_B, y, \mathcal{K}_{X,\mathcal{Z}}(x)$) aufgrund der Informationen pub_X und den i Teilen $s_{j_1}, \ldots, s_{j_i}$ simulieren. Dabei haben wir ausgenutzt, daß die Release-Protokolle symmetrisch sind, so daß wir die Rollen von A und B vertauschen können. B^{**} extrahiert mit Wahrscheinlichkeit $1/\operatorname{poly}(k)$ ein vollständiges Geheimnis s zu pub_X. Da der Release-Protokoll-Simulator das Protokoll fast perfekt simulieren kann, erhalten wir einen Algorithmus, der mit Wahrscheinlichkeit $1/2\operatorname{poly}(k)$ ein vollständiges Geheimnis extrahiert. Wegen der uniformen Ergänzungskomplexität können wir daher o.B.d.A. annehmen, daß es einen Algorithmus A_0^* gibt, der aus der Eingabe pub_X und i Teilen ein vollständiges Geheimnis mit Wahrscheinlichkeit $1 - 1/\operatorname{poly}(k)$ berechnet.

Aus A_0^* konstruieren wir A_0. Algorithmus A_0 erhält als Eingabe eine Kommunikation zwischen A und B^{**} und die öffentlichen Parameter. Die Korrektheit des Release-Protokolls $\mathcal{R}_A$ garantiert, daß B^{**} mit nicht vernachlässigbarer Wahrscheinlichkeit mindestens $i-1$ Teile von y in der Kommunikation an A gesendet hat. In diesem Fall versuchen wir alle maximal $2^{b(k)} = \text{poly}(k)$ Möglichkeiten, die die nächste open-Anfrage und die mögliche Antwort liefern können, und simulieren A_0^* auf den ersten $i-1$ Teilen von B^*s Geheimnis y und der jeweiligen Möglichkeit, sowie pub_B und $\mathcal{K}_{A,\mathcal{Y}}(y)$. Da A_0^* fast immer korrekt antwortet, erhalten wir mit hoher Wahrscheinlichkeit den Wert y. ■

Man kann den Beweis leicht auf den Fall der adaptiven Herausgabe erweitern. In diesem Fall versucht A_0 neben allen 2^b möglichen Werten zusätzlich alle maximal n Positionen des open-Befehls mit Aufwand $n2^b = \text{poly}(k)$.

Kapitel 4

Protokoll zum fairen Austausch von digitalen Unterschriften

In diesem Abschnitt stellen wir unsere Release-Protokolle vor. Dabei gehen wir ausführlich auf Schnorr-Unterschriften ein und skizzieren kurz, wie man dieses Protokoll für andere Unterschriftentypen anpaßt. Insbesondere erlauben unsere Protokolle den fairen Austausch verschiedener Unterschriftentypen, z.B. Schnorr- und DSS-Unterschriften, da diese Protokolle symmetrisch sind.

Wir verzichten auf eine formale Definition von Unterschriftenverfahren und Sicherheitsbegriffen, da wir Unterschriftenschemata nur als "Black-Box" betrachten. Bis auf das in Kapitel 4.3 angegebene Signaturschema basieren alle Verfahren auf der Diskreten-Logarithmus-Annahme: Gilt diese Annahme nicht, können die Unterschriftenschemata leicht gebrochen werden. Da die Diskrete-Logarithmus-Annahme andererseits die Sicherheit unseres Release-Protokolls impliziert, gehen wir nicht weiter auf die Sicherheit des jeweiligen Unterschriftenverfahrens ein. Für eine Formalisierung von Signaturverfahren verweisen wir auf die grundlegende Arbeit von Goldwasser, Micali und Rivest [GMR88]. Sicherheitsaspekte des jeweiligen Verfahrens findet man in der zugörigen Publikation.

4.1 Fairer Austausch von Schnorr-Unterschriften

Wir wiederholen kurz das Schnorr-Unterschriftenschema, das unmittelbar aus dem Schnorr-Identifikationsprotokoll hervorgeht [S91]. Wir nennen die so erhaltenen Unterschriften "lange" Schnorr-Unterschriften, da man im Unterschriftenschema die Länge der Signaturen durch One-Way-Hashfunktionen verkürzen kann ("kurze" Schnorr-Unterschriften). Eine "lange" Schnorr-Unterschrift ist ein Proof of Knowledge für das Repräsentationsproblem für $n = 1$, bei dem der Wert $c \in \{0,1\}^t$ durch den Hashwert $\mathsf{H}(R, X, m)$ einer öffentlichen One-Way-Hashfuntion $\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$ ersetzt wird. Um eine Nachricht m zu unterschreiben, wählt A zufällig $r \in \mathbb{Z}_q$ und bildet $R = g^r \bmod p$ sowie $s = r + cx \bmod q$ für $c = \mathsf{H}(R, X, m)$. Eine Unterschrift für m ist (R, s). Die Unterschrift ist öffentlich überprüfbar, indem der Prüfer $c = \mathsf{H}(R, X, m)$ berechnet und verifziert, daß $RX^c = g^s \bmod p$. Bei "kurzen" Schnorr-Unterschriften wird der Wert R durch den kürzeren Hashwert $c = \mathsf{H}(R, X, m)$

ersetzt, d.h. die Unterschrift zu m ist (c, s). In diesem Fall lautet die Korrektheitsbedingung $c = \mathsf{H}(X^{-c}g^s, X, m)$.

Für unser Austauschprotokoll der "langen" Schnorr-Unterschriften unterteilt der Unterschreiber den Wert s in n Teile $s_1, \ldots, s_n$ zu je b Bits (wobei wir zur Vereinfachung annehmen, daß b den Wert $|q|$ teilt). Das erste herausgegebene Teil der Unterschrift (R, s) ist stets der zufällig gewählte Wert $R = g^r \bmod p$, da wir zum Überprüfen der Korrektheit der Teile $\tilde{s}_\ell$ den Hashwert $c = \mathsf{H}(R, X, m)$ kennen müssen. Da R eine spezielle Rolle zufällt, sagen wir, daß R in der nullten Release-Runde preisgegeben wird. Das Prädikat $\mathcal{V}_A(i, \mathsf{pub}_A)$ ist somit genau dann wahr, wenn die in den ersten i Release-Runden erhaltenen Blöcke $\tilde{s}_{j_1}, \ldots, \tilde{s}_{j_i}$ aus je b Bits bestehen und durch $n - i$ weitere Werte $\tilde{s}_\ell \in \{0,1\}^b$ zu $\tilde{s}$ ergänzt werden können, so daß zusammen mit dem in der nullten Runde gesendeten Wert R die Gleichung $g^{\tilde{s}} = RX^c \bmod p$ gilt. Die Annahme über die uniforme Ergänzungskomplexität der Unterschriftenschemas lautet in diesem Fall: Für alle zufälligen p, q, g, r, R, x, X, c ist die Berechnung von $s = \log_g RX^c$ aus i Teilen $s_{j_1}, \ldots, s_{j_i}$ für alle s gleich schwierig.

"Kurze" Schnorr-Unterschriften werden wie "lange" Signaturen ausgetauscht. Statt (R, s) gibt der Empfänger (c, s) aus. Da "lange" Schnorr-Unterschriften aus dem zugehörigen Identifikationsprotokoll hervorgehen, erhalten wir aus unserem Release-Protokoll zusätzlich ein "faires Identifikationsprotokoll", bei dem sich zwei Teilnehmer auf faire Weise gegenseitig identifizieren.

4.1.1 Überblick

Wir geben zunächst einen Überblick über unser Release-Protokoll (Abbildung 4.1). Wir unterscheiden zwischen drei Phasen: dem *Preprocessing*, das unabhängig von der speziellen Unterschrift ist und nur einmal für mehrere Unterschriften ausgeführt werden muß, der *Off-line-Phase*, die von der speziellen Unterschrift abhängt und bei Verwendung einer idealen Hashfunktion nicht-interaktiv abläuft, sowie der *On-line-Phase*, bei der interaktiv die Unterschrift preisgegeben wird.

Im Preprocessing erzeugen wir Generatoren $g_1, \ldots, g_b, h$, so daß A bzw. A^* nach Ausführung die diskreten Logarithmen *nicht* kennt, während wir einen Simulator angeben können, der durch Zurücksetzen des Protokolls die diskreten Logarithmen ermitteln kann. Dies ist notwendig, da der Release-Protokoll-Simulator in der Off-line-Phase Hinterlegungen für Werte finden muß, die er zu diesem Zeitpunkt noch nicht kennt. Diese Werte lernt der Simulator erst im Verlauf der On-line-Simulation. Zur Vereinfachung der Notation schreiben wir im folgenden auch g_{b+1} statt h. Ferner zeigt der Unterschreiber A in der Off-line-Phase, daß die hinterlegten Teile eine korrekte Unterschrift bilden. Dazu beweist A durch einen witness-indistinguishable Proof of Knowledge, daß jeder Teil aus b Bits besteht, und in zero-knowledge, daß sich die Teile korrekt zu $\log_g RX^c$ aufaddieren. Da der Beweis zero-knowledge ist, kann der Release-Protokoll-Simulator diesen Beweis ohne die Kenntnis des Geheimnisses simulieren. Wir nehmen an, daß diese Beweise nicht-interaktiv durch eine ideale Hashfunktion implementiert werden, wobei die Ausgabelänge t der Hashfunktion linear im Sicherheitsparameter k sei.

In der On-line-Phase gibt A jeweils einen Teil des Geheimnisses preis. Dabei lassen wir B den Teil wählen, den er als nächstes sehen möchte. Alternativ können wir A den Teil selbst

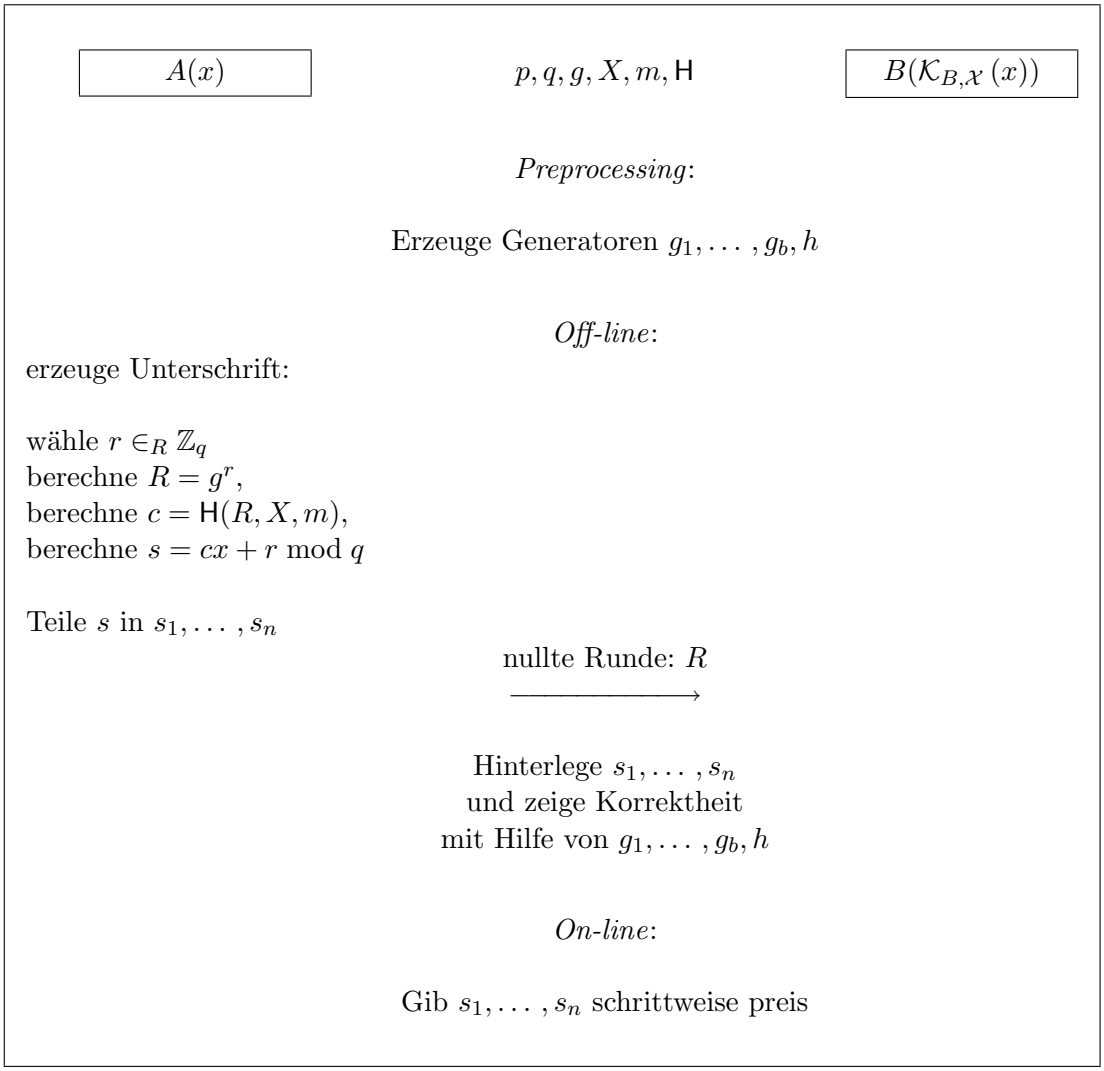

Abbildung 4.1: Release-Protokoll für Schnorr-Unterschriften (Überblick)

wählen lassen. In Abschnitt 4.1.8 diskutieren wir Probleme, die bei beiden Ansätzen auftreten können, und zeigen einen Lösungsansatz. Der Vorteil dieser Wahlmöglichkeit ist die in Abschnitt 3.4 angesprochenen Abschwächung an die Anforderung der uniformen Schwierigkeit des Geheimnisses.

4.1.2 Preprocessing

Wir zeigen, wie A und B Generatoren $g_1, \ldots, g_{b+1}$ aus einem Generator g erzeugen können, so daß selbst ein unehrlicher Teilnehmer A^* nicht die diskreten Logarithmen von g_i bezüglich g_j für $i \neq j$ kennt, während ein Simulator durch Wiederholung des Protokolls diese Logarithmen leicht berechnen kann. Ferner ist die Ausgabe des Simulators statistisch ununterscheidbar von einer “echten” Kommunikation. Die zugrundeliegende Idee wurde von Brassard, Crépeau und

Yung [BCY91] verwendet, um perfekte zero-knowledge Protokolle anzugeben, bei denen der Prover nur polynomielle Laufzeit besitzt.

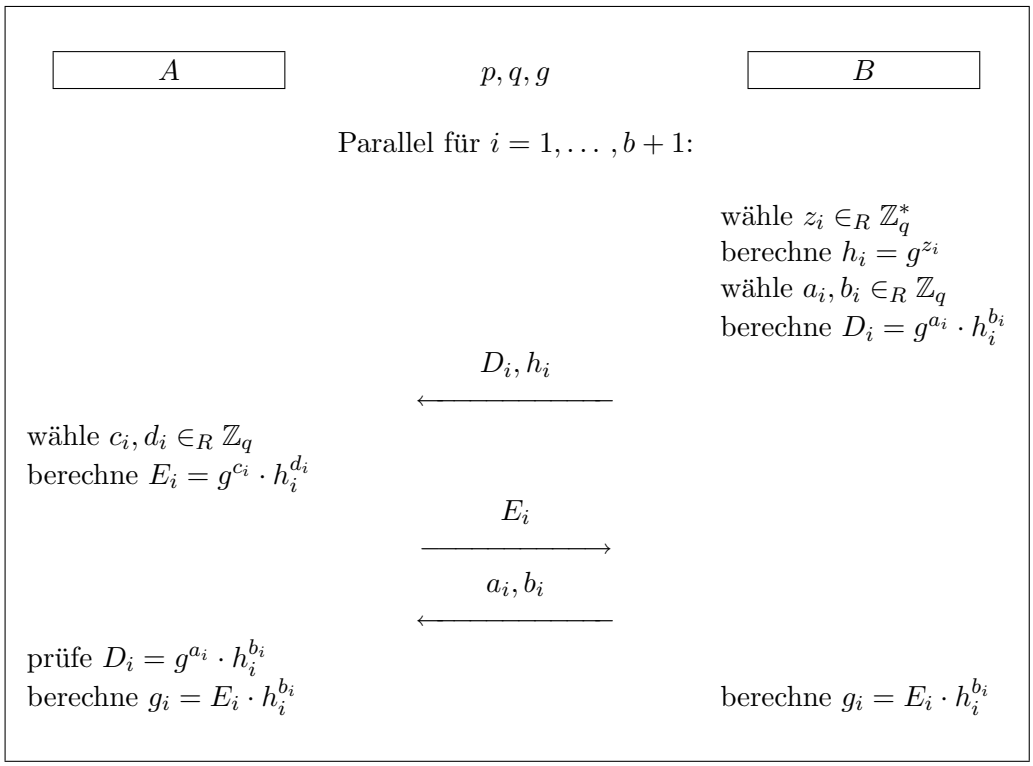

Abbildung 4.2: Preprocessing: Erzeuge $g_1, \ldots, g_{b+1}$

Wir geben einen Simulator S^* an, der für jeden Algorithmus B^* eine Kommunikation erzeugt, die statistisch ununterscheidbar von einer "echten" Kommunikation ist, aber so daß S^* die diskreten Logarithmen $\log_{g_i} g_j$ für alle i, j berechnen kann. Tatsächlich gibt der Simulator nur $\delta_i = \log_g g_i$ aus. Daraus lassen sich

$$\log_{g_i} g_j = \delta_j \delta_i^{-1}$$

leicht berechnen, wobei δ_i^{-1} das Inverse zu δ_i in $\mathbb{Z}_q^*$ sei.[1]

Lemma 4.1.1
Für jeden Algorithmus B^* in Protokoll 4.2 gibt es eine Orakelmaschine $S^* = S^{B^*}$ mit erwarteter polynomieller Laufzeit, so daß die Ensembles

- $\{S^*((p,q,g), x_{B^*})\}_{(p,q,g),x_{B^*}}$
- $\{\text{view}_{B^*} \langle A(p,q,g) \leftrightarrow B^*(p,q,g,x_{B^*})\rangle\}_{(p,q,g),x_{B^*}}$

[1] Der Fall $\delta_i = 0$ tritt nur mit vernachlässigbarer Wahrscheinlichkeit $1/q \approx 2^{-k}$ ein, so daß wir im folgenden davon ausgehen, daß $\delta_i \neq 0$ für alle $i = 1, \ldots, b+1$.

statistisch ununterscheidbar sind, und so daß der Simulator zusätzlich $\log_g g_1, \ldots, \log_g g_{b+1}$ *ausgibt.*

Beweis. Der Simulator arbeitet in zwei Phasen. In der ersten Phase versucht S^* einen erfolgreichen Protokollablauf zu finden. Falls er diese Phase abschließt, wiederholt S^* in der zweiten Phase das Protokoll und verwendet die Informationen aus der ersten Phase, um g_i so zu erzeugen, daß er $\log_g g_i$ berechnen kann.

In der ersten Phase führt S^* das Protokoll einmal mit festen Münzwürfen ω_{B^*} für B^* aus. Falls B^* im letzten Schritt keine korrekte Aufdeckung sendet, stoppt der Simulator und gibt die Kommunikation aus. Dies ist identisch mit A's Verhalten in diesem Fall. Andernfalls lernt S^* Werte a_i, b_i mit $D_i = g^{a_i} \cdot h_i^{b_i}$ für alle $i = 1, \ldots, b+1$ und setzt das Protokoll bis nach der ersten Runde zurück. Wir nehmen zunächst an, daß B^* auch bei der Wiederholung stets korrekte Aufdeckungen sendet. Dann wählt S^* in der zweiten Runde $E_i = g^{c'_i} \cdot h_i^{-b_i}$ für zufälliges $c'_i \in_R \mathbb{Z}_q$. Beachte, daß dieser Wert identisch verteilt zu einem "echten" Wert ist. Falls B^* in der neuen dritten Runde wieder a_i, b_i sendet, gilt $g_i = g^{c'_i}$ und damit $\log_g g_i = c'_i$. Angenommen, B^* sendet Werte $(a'_i, b'_i) \neq (a_i, b_i)$ mit

$$D_i = g^{a_i} \cdot h_i^{b_i} = g^{a'_i} \cdot h_i^{b'_i}.$$

Dann gilt

$$g^{a_i - a'_i} = h_i^{b'_i - b_i} \bmod p$$

Folglich ist

$$g_i = g^{c'_i} \cdot h_i^{b'_i - b_i} = g^{c'_i + a_i - a'_i} \bmod p$$

und damit kann S^* den diskreten Logarithmus $\log_g g_i = c'_i + a_i - a'_i \bmod q$ berechnen.

Wir betrachten den allgemeinen Fall, daß B^* nur mit Wahrscheinlichkeit $\rho = \rho(k)$ (über die zufällige Wahl des Wertes E_i) korrekte Aufdeckungen sendet. Der Beweis folgt der Vorgehensweise in [GK96]. Wir fügen zunächst zwischen den beiden Phasen eine Phase $1b$ ein, in der eine Schätzung für ρ ermittelt wird. Dazu wiederholen wir das Protokoll (mit gleicher Münzwurffolge ω_{B^*}) bis wir eine feste, polynomielle Anzahl von akzeptierenden Ausführungen erreicht haben. Dies gelingt in erwarteter Laufzeit $\mathrm{poly}(k)/\rho(k)$. Ferner erhalten wir nach der Chernoff-Schranke mit Wahrscheinlichkeit $1 - 2^{-\mathrm{poly}(k)}$ eine Schätzung $\overline{\rho}$ von ρ bis auf einen konstanten Faktor. Wir führen Phase 2 höchstens $(\mathrm{poly}(k)/\overline{\rho})$-mal aus. Falls wir dabei keinen akzeptierenden Ablauf finden, geben wir eine undefinierte Ausgabe $\perp$. Die erwartete Laufzeit des Simulators beträgt somit $\mathrm{poly}(k) + \rho(k) \cdot \mathrm{poly}(k)/\rho(k) = \mathrm{poly}(k)$.

Wir zeigen, daß die Ausgabe statistisch nah an einer "echten" Kommunikation ist. Es genügt zu zeigen, daß der Simulator nach Verlassen von Phase 1 nicht zu oft $\perp$ ausgibt. Wir betrachten nur den Fall, daß die Schätzung $\overline{\rho}$ bis auf einen konstanten Faktor gut ist. Dies geschieht mit Wahrscheinlichkeit $1 - 2^{-\mathrm{poly}(k)}$. In diesem Fall gibt S^* nur mit Wahrscheinlichkeit

$$\rho(k) \cdot (1 - \rho(k))^{\mathrm{poly}(k)/\rho(k)} \leq e^{-\mathrm{poly}(k)}$$

die Ausgabe $\perp$. Folglich sind beide Kommunikationen statistisch ununterscheidbar. ■

Wir haben ausgenutzt, daß A nicht die Werte c_i, d_i an B sendet, so daß der Simulator in Schritt 2 nach dem Reset $C_i = g^{c'_i} \cdot h_i^{-b_i}$ wählen und somit statt eines zufälligen Wertes d'_i den Wert $-b_i$ verwenden konnte. In unserem Release-Protokoll benötigen wir dagegen, daß ein Korrektheitssimulator, der mit A^* kommuniziert, eine Repräsentation für *gegebene* Generatoren $G_1, \ldots, G_{b+1}, g$ findet, wenn A^* versucht zu betrügen. Wir modifizieren daher das Protokoll so, daß ein Simulator nach Ende des Preprocessing die erzeugten Generatoren kennt. Dazu wählt der Simulator $h_i = G_i^{z_i}$ und A zeigt in Schritt 2 durch Proofs of Knowledge, daß er Repräsentationen für C_i bezüglich g, h_i kennt. Der Knowledge Extractor kann dann diese Repräsentationen extrahieren und damit zwei verschiedene Repräsentationen eines Wertes Z bezüglich $(g_1, \ldots, g_{b+1}, g)$ in verschiedene Repräsentationen von Z bezüglich $(G_1, \ldots, G_{b+1}, g)$ überführen.

Das modifizierte Protokoll ist in Abbildung 4.3 angegeben. Beachte, daß $h_i = G_i$ ebenfalls uniform in $G_q \backslash \{1\}$ verteilt ist. Es genügt, daß der Proof of Knowledge witness-indistinguishable ist, da dann der Wert C_i uniform verteilt ist und somit die Werte c_i, d_i informationstheoretisch geschützt sind. Dies gilt auch für den Simulator S^* aus Lemma 4.1.1 und den in der zweiten Phase gewählten Wert $C_i = g^{c'_i} \cdot h_i^{-b_i}$.

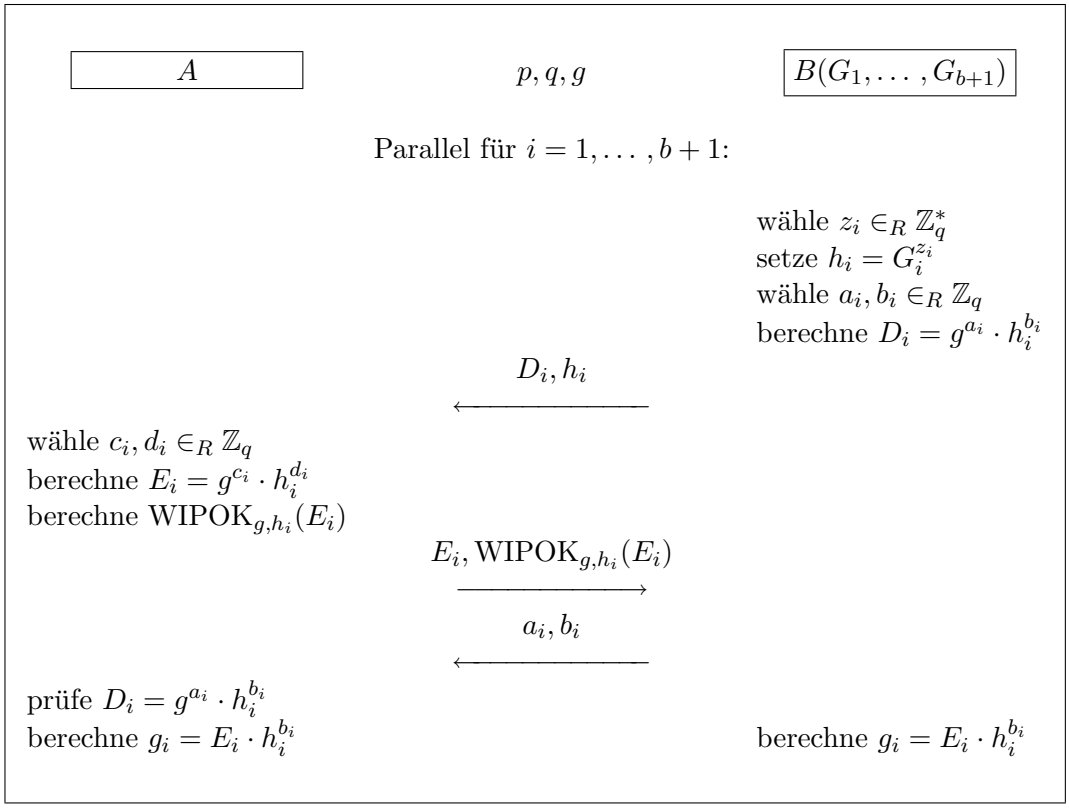

Abbildung 4.3: Modifiziertes Preprocessing: Erzeuge $g_1, \ldots, g_{b+1}$ aus $G_1, \ldots, G_{b+1}$

4.1.3 Off-line-Phase

In der Off-line-Phase berechnet A eine Unterschrift zur Nachricht m und teilt s mit $g^s = RX^c$ in Blöcke $s_1, \ldots, s_n$ zu je b Bits, so daß

$$s = \sum_{i=1}^{n} s_i \cdot \alpha_i \qquad \text{mit } \alpha_i = 2^{(i-1)b}$$

Dann berechnet A Hinterlegungen für $s_1, \ldots, s_n$, wobei wir jedes Bit s_{ij} des i-ten Blockes separat hinterlegen. Da A dann noch zeigen muß, daß sich die Bits korrekt zu $\log_g RX^c$ aufaddieren, verwenden wir aus Effizienzgründen folgende Darstellung: Wir gruppieren jeweils das j-te Bit s_{ij} von s_i zu einem Wert

$$\overline{s}_j = \sum_{i=1}^{n} s_{ij} \cdot \overline{\alpha}_{ij} \qquad \text{mit } \overline{\alpha}_{ij} = 2^{(i-1)b+(j-1)}$$

für $j = 1, \ldots, b$. Die Umgruppierung ist in Abbildung 4.4 für $n = 4$ und $b = 3$ dargestellt. Mit dieser Darstellung gilt:

$$s = \sum_{j=1}^{b} \overline{s}_j$$

A hinterlegt die einzelnen Bits wie in Abschnitt 3.3 angegeben mit Pedersens Protokoll, indem er $v_{ij} \in_R \mathbb{Z}_q$ zufällig wählt und

$$C_{ij} = g_j^{s_{ij} \cdot \overline{\alpha}_{ij}} \cdot h^{v_{ij}}$$

berechnet. Für $v = \sum_{i,j} v_{ij} \bmod q$ gilt damit:

$$C = \prod_{i,j} C_{ij} = h^v \cdot \prod_{j=1}^{b} g_j^{\overline{s}_j}$$

Der Release-Protkoll-Simulator für die Geheimhaltung muß identisch verteilte Hinterlegungen berechnen, ohne die korrekten zu hinterlegenden Werte zu diesem Zeitpunkt zu kennen. Diese korrekten Werte erhält er adaptiv im Laufe des Protokolls. Andererseits kennt der Simulator durch das Preprocessing die diskreten Logarithmen $\gamma_j = \log_{g_j} h$ von h bezüglich g_j und kann daher triviale Werte hinterlegen und die Hinterlegungen später nach Belieben öffnen. Dazu wählt der Simulator in der Off-line-Phase $v^*_{11}, \ldots, v^*_{nb} \in_R \mathbb{Z}_q$ und hinterlegt Werte $s^*_{11}, \ldots, s^*_{nb} = 0$, indem er $C^*_{ij} = h^{v^*_{ij}}$ berechnet. Wenn der Simulator später einen korrekten Wert s_{ij} erhält, berechnet er $v_{ij} = v^*_{ij} - \gamma_j^{-1} \cdot \overline{\alpha}_{ij} \cdot s_{ij} \bmod q$ und öffnet C^*_{ij} durch (s_{ij}, v_{ij}):

$$C^*_{ij} = h^{v^*_{ij}} = h^{v_{ij} + \gamma_j^{-1} \cdot \overline{\alpha}_{ij} \cdot s_{ij}} = h^{v_{ij}} g_j^{\overline{\alpha}_{ij} \cdot s_{ij}} \bmod p$$

A muß zusätzlich zeigen, daß die hinterlegten Werte s_{ij} korrekt sind. Dazu beweist A, daß er einerseits Bits hinterlegt hat,[2] und andererseits, daß diese Bits zusammen den dis-

[2] Die Cut-and-Choose-Methode kann nicht vewendet werden, da sie nur zeigt, daß die Hinterlegung in $[-I, 2I)$ liegt. Die Sicherheit unseres Protokolls beruht u.a. auf der Tatsache, daß die Blöcke exakt aus dem Intervall $[0, 2^b)$ sind.

kreten Logarithmus s von RX^c ergeben. Im ersten Teil führen wir parallel $|q|$ viele witness-indistinguishable Proofs of Knowledge aus, während wir für den Nachweis der korrekten Zusammensetzung zwei zero-knowledge Proofs of Knowledge benötigen. Dabei nehmen wir an, daß alle Beweise nicht-interaktiv ausgeführt werden. Am Ende der Off-line-Phase sendet A die entsprechenden Werte an B, der die Korrektheit der erhaltenen Werte überprüft.

Abbildung 4.4: Umgruppieren der Bits

In Abbildung 4.5 ist ein witness-indistinguishable Proof of Knowledge angegeben, mit dem man zeigen kann, daß man eine Repräsentation (b, w) von C bezüglich zweier Generatoren G und H kennt, für die $b \in \{0, 1\}$ gilt.

Lemma 4.1.2
Das in Abbildung 4.5 angegebene Protokoll ist ein witness-indistinguishable Proof of Knowledge mit Fehler 2^{-t}. Weiterhin existiert ein Knowledge Extractor, so daß für den extrahierten Zeugen (b^*, v^*) gilt: $b^* \in \{0, 1\}$.

Beweis. Die Vollständigkeit ist offensichtlich. Der Nachweis, daß das Protokoll witness-indistinguishable ist, folgt wie in [O92]. Wir zeigen, daß das Protokoll ein Proof of Knowledge ist. Der Knowledge Extractor E^* arbeitet wie im Beweis zu Satz 3.3.3, d.h. extrahiert zwei akzeptierende Ausführungen für die gleichen Werte D, D_0, D_1, aber mit $c \neq c'$. Analog seien d_0, d_1, w_0, w_1 und d'_0, d'_1, w'_0, w'_1 die gesendeten Werte in Schritt 3. Aus der Korrektheitsbedingung folgt:

$$D^{d'_0-d_0} = G^{d'_0-d_0} \cdot H^{w'_0-w_0} \quad \text{und} \quad D^{d'_1-d_1} = H^{w'_1-w_1}$$

Für $c \neq c'$ gilt $d_a \neq d'_a$ für ein $a \in \{0, 1\}$ und damit existiert das Inverse zu $d'_a - d_a$ in $\mathbb{Z}_q^*$. In diesem Fall erhalten wir eine Repräsentation

$$\left(1, (w'_0 - w_0)(d'_0 - d_0)^{-1}\right) \qquad \text{oder} \qquad \left(0, (w'_1 - w_1)(d'_1 - d_1)^{-1}\right)$$

für D bezüglich (G, H). ■

Man kann den Proof of Knowledge leicht auf beliebige Werte $b \in \{b_0, b_1\}$ für feste b_0, b_1 erweitern. In diesem Fall ersetze man in den Berechnungen den Wert G^{-1} durch G^{-b_1} und füge analog zum Fall $b = 0$ den Wert G^{-b_0} ein.

A führt den angebenen Proof of Knowledge parallel für alle Hinterlegungen C_{ij} aus, wobei in allen Ausführungen der gleiche Wert $c \in \{0, 1\}^t$ verwendet wird, d.h. im nicht-interaktiven Modell der Hashwert über alle in der ersten Runde berechneten Werte gebildet wird. Gemäß unserer Darstellung $\overline{s}_1, \ldots, \overline{s}_b$ verwenden wir als Generatoren für C_{ij} die Werte $g_j^{\overline{\alpha}_{ij}}$ und h. Da

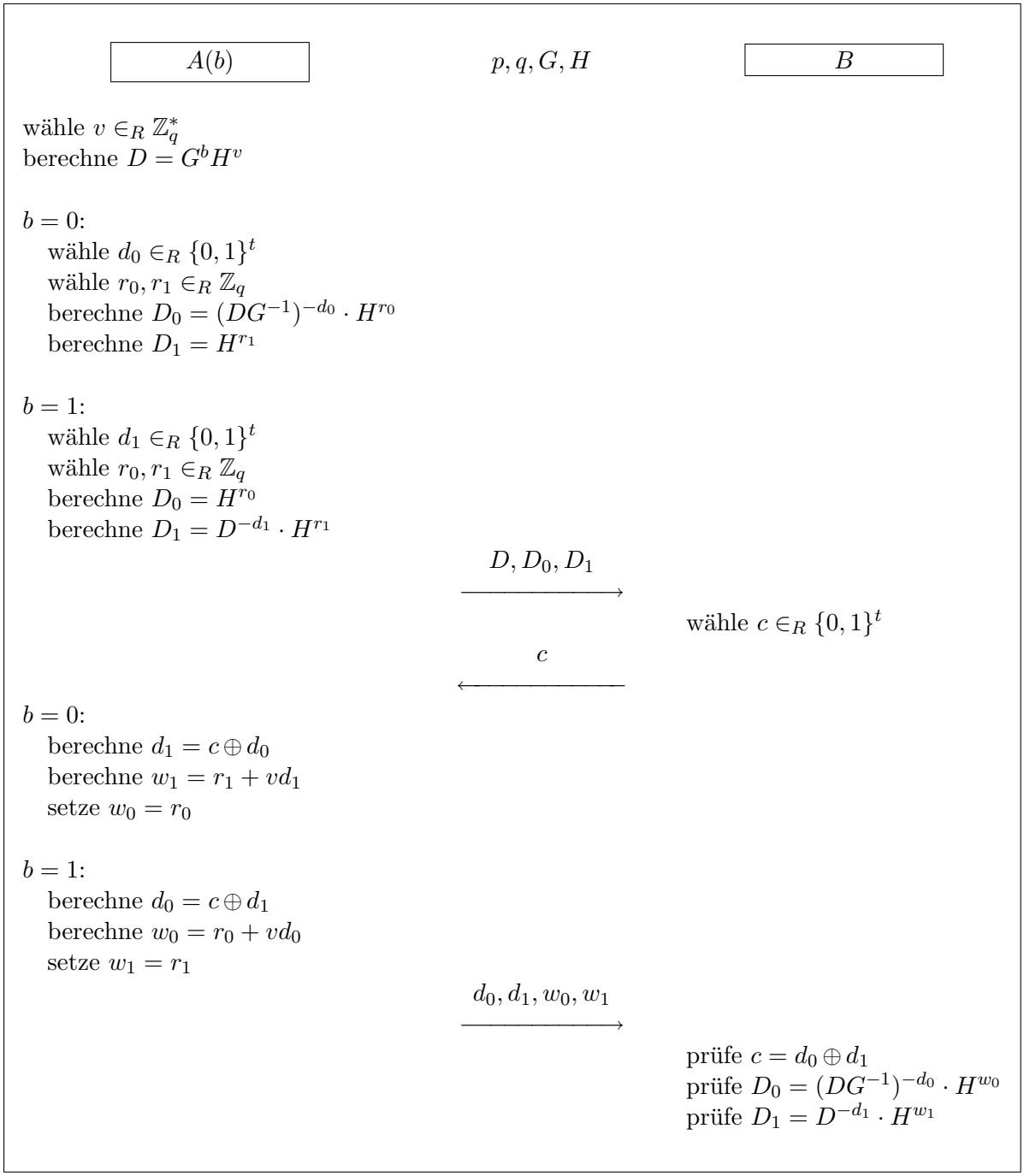

Abbildung 4.5: Proof of Knowledge für Repräsentation $(b, v) \in \{0,1\} \times \mathbb{Z}_q$

diese Proofs of Knowledge witness-indistinguishable sind, kann auch der Release-Protokoll-Simulator diese Beweise für seine "Pseudo"-Hinterlegungen ausführen, so daß die Kommunikation identisch zu einer "echten" Kommunikation ist. Seien im folgenden $s_1^*, \ldots, s_n^* \in \{0,1\}^b$ und $v^* \in \mathbb{Z}_q$ die extrahierten Zeugen des Knowledge Extractors in obigen Proofs of Knowledge für die Hinterlegungen. Entsprechend seien $\overline{s}_1^*, \ldots, \overline{s}_b^*$ definiert.

Zusätzlich zum Beweis, daß er Bits hinterlegt hat, zeigt A durch Brands Protokoll [B97], daß sich die hinterlegten Bits zu einer gültigen Unterschrift aufaddieren. Dazu beweist A mit einem zero-knowledge Proof of Knowledge, daß er eine Repräsentation von Cg^{-s} bezüglich $(g_1 g^{-1}, \ldots, g_b g^{-1}, h)$ kennt. Wegen $s = \sum \overline{s}_j$ gilt für die korrekten Werte $\overline{s}_1, \ldots, \overline{s}_b, v$ die Gleichung

$$Cg^{-s} = g^{-s} \cdot \prod_{i,j} C_{ij} = g^{-\sum_{i=1}^{b} \overline{s}_i} \cdot h^v \cdot \prod_{i=1}^{b} g_i^{\overline{s}_i}$$

Beachte, daß B nur den Wert $C = \prod_{i,j} C_{ij}$, aber nicht g^{-s} kennt. Andererseits erfüllt ein korrekter Wert s die Gleichung $g^s = RX^c \bmod p$ für $c = \mathsf{H}(X, R, m)$. Daher genügt zur Überprüfung der gesendeten Werte im Proof of Knowledge die Kenntnis von m, R, X bzw. von c, R, X. In diesem Fall ersetzt B im Korrektheitstest den Wert Cg^{-s} durch $C(RX^c)^{-1}$. Wir betrachten den aus dem Proof of Knowledge vom Knowledge Extractor extrahierten Zeugen $(\overline{s}_1^{**}, \ldots, \overline{s}_b^{**}, v^{**})$. Dieser Zeuge erfüllt

$$C(RX^c)^{-1} = h^{v^{**}} \cdot g^{-\sum_{i=1}^{b} \overline{s}_i^{**}} \cdot \prod_{i=1}^{b} g_i^{\overline{s}_i^{**}}$$

Wenn $\sum_{i=1}^{b} \overline{s}_i^{**} \neq s = \log_g(RX^c)$ ist, dann liegen zwei verschiedene Repräsentationen von Cg^{-s} bezüglich $(g_1, \ldots, g_b, h, g)$ vor. Da der Knowledge Extractor E^* den Wert s nicht explizit kennt, sondern nur $g^s = RX^c$, fügen wir noch einen zero-knowledge Proof of Knowledge für eine Repräsentation von RX^c bezüglich g hinzu. In diesem Fall kann E^* einen Zeugen s^* extrahieren — insbesondere gilt $s^* = s$, da der diskrete Logarithmus von RX^c eindeutig bestimmt ist — und damit zwei verschiedene Repräsentationen für Cg^{-s} bezüglich $(g_1, \ldots, g_b, h, g)$ ausgeben.

4.1.4 On-line-Phase

In der On-line-Phase gibt A schrittweise die Blöcke $s_1, \ldots, s_n$ preis. Dazu deckt er die Hinterlegungen C_{ij} für $i = j = 1, \ldots, b$ auf. B überprüft lediglich, daß diese Aufdeckungen korrekt sind.

4.1.5 Korrektheit

Wir zeigen, daß unser Protokoll korrekt ist. Angenommen, es gibt einen Algorithmus A^*, so daß

(4.1)

$$\begin{aligned}
\text{Prob}\big[&\omega_{A^*} \leftarrow \Omega_{A^*},\ \omega_B \leftarrow \Omega_B,\ (\mathsf{pub}_A, \mathsf{priv}_A) \leftarrow I_A(1^k),\\
&(x, \mathcal{K}_{B,\mathcal{X}}(x)) \leftarrow \mathcal{S}_A(\mathsf{pub}_A, \mathsf{priv}_A)\ :\\
&\text{pass}_B^{i(k)} \langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A, \omega_{A^*}) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A, \omega_B)\rangle \ \wedge\\
&\text{psecret}_B^{i(k)} \langle A^*(x, \mathsf{pub}_A, \mathsf{priv}_A, \omega_{A^*}) \leftrightarrow_{\mathcal{R}} B(\mathcal{K}_{B,\mathcal{X}}(x), \mathsf{pub}_A, \omega_B)\rangle \notin \mathcal{V}_A(i(k), \mathsf{pub}_A)\big]
\end{aligned}$$

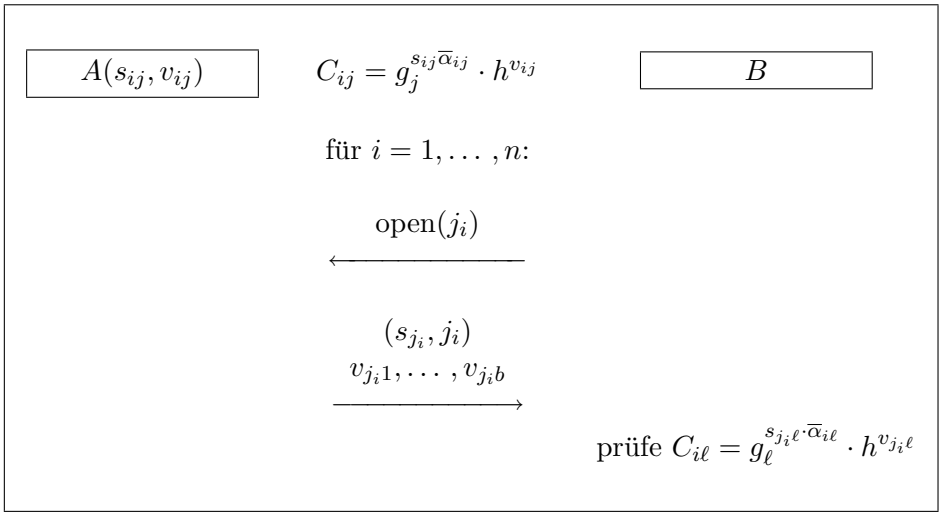

Abbildung 4.6: On-line-Phase

für ein $i : \mathbb{N} \to \{0, \ldots, n(k)\}$, ein Polynom $p(k)$ und unendlich viele $k \in \mathbb{N}$. Wir leiten daraus einen Widerspruch zur Diskreten-Logarithmus-Annahme ab. Dazu geben wir einen probabilistischen Polynomialzeitalgorithmus $\mathcal{A}^*$ an, der mit nicht vernachlässigbarer Wahrscheinlichkeit ein Element $Z \in \mathbb{G}_q$ und zwei verschiedene Repräsentationen für Z bezüglich zufällig gewählter Generatoren $(G_1, \ldots, G_{b+1}, g)$ für $b = b(k) \in \text{poly}(k)$ findet. Dafür genügt es, verschiedene Repräsentationen für die im Preprocessing erzeugten Generatoren $(g_1, \ldots, g_{b+1}, g)$ zu finden.

Wir fixieren ein k, so daß die in Ungleichung (4.1) angegebene Wahrscheinlichkeit mindestens $1/p(k)$ beträgt. Ferner seien $i = i(k)$, $n = n(k)$ und $b = b(k)$ fest. Algorithmus $\mathcal{A}^*$ verwendet A^* als Orakel, d.h. für A^* wird nicht-sichtbar für $\mathcal{A}^*$ zu Beginn eine Folge von Münzwürfen gewählt. Für $i = 0$, d.h. nachdem $R = g^r$ gesendet wurde, ist nichts zu zeigen, da es ein s gibt, daß $g^s = RX^c \bmod p$ erfüllt. Sei $i > 0$ und $\tilde{s}_{j_1}, \ldots, \tilde{s}_{j_i}$ die Werte, die A^* an B gesendet hat. $\mathcal{A}^*$ simuliert den Knowledge Extractor E^{A^*}, um Zeugen $s_1^*, v_1^* \ldots, s_n^*, v_n^*$ aus den Proofs of Knowledge für die Hinterlegungen C_{ij} zu extrahieren — insbesondere ist $s_1^*, \ldots, s_n^* \in \{0,1\}^b$. Das gleiche wiederholt er für die Proofs of Knowledge für g^s und die lineare Relation $s = \sum \overline{s}_j$. Seien s^* und $s_1^{**}, \ldots, s_n^{**}, v^{**}$ diese Zeugen. Dabei nehmen wir o.B.d.A. an, daß der Knowledge Extractor erwartete polynomielle Laufzeit besitzt und die Zeugen wegen $t = \Omega(k)$ mit Wahrscheinlichkeit $1/p(k) - 2^{-t} \geq 1/2p(k)$ für hinreichend große k extrahiert. Wir zeigen daher zunächst, daß $\mathcal{A}^*$ in erwarteter polynomieller Laufzeit mit nicht vernachlässigbarer Wahrscheinlichkeit das Repräsentationsproblem löst.

Die Beweisidee sieht wie folgt aus: Wir zeigen, daß die extrahierten Zeugen s_ℓ^* und s_ℓ^{**} übereinstimmen, sofern man nicht das Repräsentationsproblem löst. Unter der gleichen Annahme folgt, daß die geöffneten Werte $\tilde{s}_\ell$ mit s_ℓ^* identisch sind. Andererseits ergeben die Werte s_ℓ^{**} eine gültige Unterschrift. Folglich kann A^* nur dann falsche Werte hinterlegen bzw. aufdecken, wenn er das Repräsentationsproblem löst.

Wir betrachten den Beweis im Detail. Die folgende Behauptung ergibt sich unmittelbar:

BEHAUPTUNG 1: Falls $\tilde{s}_\ell \neq s^*_\ell$ für ein $\ell \in \{j_1, \ldots, j_i\}$, dann hat $\mathcal{A}^*$ zwei verschiedene Repräsentationen für $C_\ell = \prod_h C_{\ell h}$ bezüglich $(g_1, \ldots, g_b, h)$ gefunden.

Wir nehmen daher im weiteren an, daß $\tilde{s}_\ell = s^*_\ell$ für alle $\ell \in \{j_1, \ldots, j_i\}$. Analog folgt:

BEHAUPTUNG 2: Wenn $s^*_\ell \neq s^{**}_\ell$ für ein $\ell \in \{1, \ldots, n\}$, dann hat $\mathcal{A}^*$ ein Element mit zwei unterschiedlichen Repräsentationen bezüglich $(g_1, \ldots, g_b, h, g)$ gefunden.

Für $\ell = j_1, \ldots, j_i$ muß $s^*_\ell = \tilde{s}_\ell$ gelten, da wir sonst zwei verschiedene Repräsentationen für C_ℓ bezüglich $(g_1, \ldots, g_b, h)$ gefunden haben. Unter der Voraussetzung, daß keine verschiedenen Repräsentationen gefunden werden, erhalten wir:

$$s = s^* = \sum_{\ell=1}^{n} \alpha_\ell s^*_\ell = \sum_{\ell=1}^{n} \alpha_\ell s^{**}_\ell = \sum_{\ell=1}^{i} \alpha_{j_\ell} \tilde{s}_{j_\ell} \; + \; \sum_{\ell=i+1}^{n} \alpha_{j_\ell} s^{**}_{j_\ell}$$

wobei $s^*_\ell = s^{**}_\ell \in \{0,1\}^b$ für $\ell = i+1, \ldots, n$. Folglich können in diesem Fall die Werte $R, \tilde{s}_{j_1}, \ldots, \tilde{s}_{j_i}$ zu einer korrekten Unterschrift ergänzt werden.

Sei $p_0(k)$ ein Polynom, das die erwartete Laufzeit des Knowledge Extractors beschränkt. Dann ist die Wahrscheinlichkeit, daß der Knowledge Extractor mehr als $4p(k)p_0(k)$ Schritte macht, höchstens $1/4p(k)$. Folglich können wir die Simulation des Knowledge Extractors nach $4p_0(k)p(k)$ Schritten abbrechen, um mit Wahrscheinlichkeit mindestens $1/2p(k) - 1/4p(k) = 1/4p(k)$ einen Wert mit verschiedenen Repräsentationen zu finden.

4.1.6 Geheimhaltung

Die Geheimhaltung folgt unmittelbar. Im Preprocessing kann ein Simulator S^* das Protokoll so simulieren, daß er die diskreten Logarithmen $\log_{g_i} g_j$ berechnen und damit "Pseudo"-Hinterlegungen senden kann. Diese simulierte Kommunikation ist statistisch ununterscheidbar von einer echten Kommunikation. Die Korrektheitsbeweise für die Hinterlegungen sind witness-indistinguishable und daher identisch zu Proofs of Knowledge für "echte" Hinterlegungen. Da die Proofs of Knowledge für s bzw. die lineare Relation statistisch zero-knowledge sind, gibt es einen Simulator, der eine Kommunikation erzeugt, die statistisch ununterscheidbar von einer "echten" Kommunikation ist. Nach Erhalt eines Blocks der Unterschrift kann S^* die Aufdeckungen korrekt öffnen und zur weiteren Simulation an B^* weitergeben.

4.1.7 Sicherheit und Effizienz

Insgesamt erhalten wir:

Theorem 4.1.3
Unter der Diskreten-Logarithmus-Annahme ist das in Abbildung 4.1 angegebene Release-Protokoll statistisch sicher.

Wir betrachten die Komplexität unseres Protokolls:

- *Rundenanzahl*: Wenn die Proofs of Knowledge mit einer idealen Hashfunktion nicht-interaktiv ausgeführt werden, hat unser Protokoll $n+4$ Runden (drei im Preprocessing, eine in der Off-line-Phase und n in der On-line-Phase). Im interaktiven Fall kommen vier Runden in der Off-line-Phase für die zero-knowledge Beweise hinzu.

- *Kommunikation*: Wir betrachten die Kommunikation bei Verwendung einer idealen Hashfunktion, da die Kommunikation bei interaktiven Proofs of Knowledge fast identisch ist. Im Preprocessing senden A und B jeweils $2(b+1)(|p|+|q|)$ Bits. In der Off-line-Phase sendet A insgesamt $|q| \cdot (3|p| + 4|q|)$ Bits für die Proofs of Knowledge der Bithinterlegungen, und $2|p| + (b+2)|q|$ Bits für die zero-knowledge Beweise. In der On-line-Phase tauschen beide Teilnehmer $(|q|+1) \cdot |q| + 2\log n$ Bits aus. Der gesamte Kommunikationsaufwand beträgt somit höchstens $3|q| \cdot |p| + 5|q|^2 + 6b|p| + 10b|q| \approx 3|q| \cdot |p| + 5|q|^2$.

- *Exponentiationen*: Wir betrachten nur die Anzahl der Exponentiationen, da sie im wesentlichen die Laufzeit bestimmen. Eine Exponentiation einer zufälligen Zahl in einer Untergruppe der Ordnung $|q|$ kann man im Erwartungswert mit $\frac{3}{2}|q|$ Multiplikationen ausführen. Schnellere Verfahren erhält man für einen festen Generator durch berechnen und speichern von Potenzen des Generators in Tabellen [LL94]. Für einen 160-Bit-Exponenten und 512-Bit-Modul p kann man beispielsweise eine Exponentiation mit 30 Multiplikationen (im Worst-Case) und Speicherbedarf 3 KB durchführen.

  Im Preprocessing führt A insgesamt $6(b+1)$ Exponentiationen durch, gegenüber $5(b+1)$ Exponentiationen von B. In der Off-line-Phase muß A genau $(5|q|+b+2)$-mal exponentieren. Am Ende der Off-line-Phase, nach Erhalt der Werte, führt B zur Überprüfung $4|q|+b+2$ Exponentiation aus. Die $2|q|$ Exponentiationen in der On-line-Phase muß nur B durchführen. Insgesamt machen A und B somit jeweils $5|q|+7b+8$ bzw. $6|q|+6b+7$ Exponentiationen.

Wir vergleichen den Aufwand mit Damgårds Protokoll [D95]. Für RSA-Unterschriften mit Exponent $e \geq 2$ — damit schließen wir Rabin-Unterschriften für $e = 2$ ein — benötigen A und B in Damgårds Release-Protokoll jeweils ca. $4K \log e$ Exponentiationen, wobei K der Sicherheitsparamter des Cut-And-Choose-Protokolls sei (z.B. $K = 80$). Für $e = 3$ und $K = |q|/2$ benötigt unser Protokoll folglich etwa doppelt so viele Exponentiationen pro Teilnehmer. Andererseits werden unsere Exponentiationen in einer Gruppe der Ordnung p durchgeführt, während Damgårds Protokoll in $\mathbb{Z}_N^*$ ausgeführt wird. Da nach dem heutigen Stand der Technik RSA-Moduln mit etwa 2048 Bits verwendet werden, während Primzahlen p für das Diskrete-Logarithmus-Problem Bitlänge ca. 1024 haben, ist der Aufwand bei beiden Protokollen fast identisch. Für Diskrete-Logarithmus-Unterschriften benötigt Damgårds Protokoll ca. $4K$ Exponentiationen in $\mathbb{Z}_N^*$, so daß auch hier der Aufwand annähernd gleich ist.

Die Anzahl der Runden beträgt bei Damgård $n+2$ im nicht-interaktiven Fall bzw. $n+9$ bei interaktiven zero-knowledge Proofs of Knowledge. Die Kommunikationskomplexität ist bei RSA-Unterschriften ungefähr $4|N| \cdot K \log e + n|N|$ bzw. bei Diskreten-Logarithmus-Unterschriften ca. $8|N| \cdot K + n|K|$ und damit fast identisch zum Aufwand unserer Protokolle.

Das Release-Protokoll von Fujisaki, Okamoto benötigt im Vergleich zu Damgårds Protokoll für RSA-Unterschriften mit $e \geq 2$ nur ca. $60 \log e$ Exponentiationen und $8|N| \log e$ Bits Kommunikation. Insbesondere ist das Protokoll damit effizienter als unser Protokoll, benötigt

allerdings eine zusätzliche kryptographische Annahme. Die Rundenanzahl ist bei Fujisaki, Okamoto identisch zu der von Damgård.

4.1.8 Verdecktes Release-Protokoll

Laut Release-Protokoll wählt der Empfänger jeweils den nächsten Teil des Geheimnisses. Dadurch tritt folgendes Problem auf: Betrachte zwei betrügerische Teilnehmer B_1^* und B_2^*, die parallel mit A kommunizieren. Dann kann B_1^* zunächst eine Hälfte von A's Geheimnis anfordern, während sich B_2^* parallel die andere Hälfte geben läßt. Danach stoppen beide und geben beispielsweise einen Kommunikationsfehler vor. In diesem Fall können B_1^* und B_2^* das gesamte Geheimnis von A zusammensetzen, obwohl sie jeweils nur die Hälfte ihrer Geheimnisse preisgeben. Ein weiteres Problem tritt auf, wenn das Geheimnis keine uniforme Ergänzungskomplexität besitzt. Wenn A weiß, welche Teile er bereits herausgegeben hat. Handelt es sich dabei um "wertlose" Teile, während A bereits "wertvolle" Teile von B's Geheimnis erhalten hat, dann kann es günstig für A sei, das Protokoll vorzeitig abzubrechen. Wir skizzieren kurz Lösungen für diese Probleme. Unsere Ansätze kann man kombinieren, so daß in jeder Runde A nicht weiß, welchen Wert er herausgegeben hat, und B nicht bestimmen kann, welchen Teil er erhält.

Wir wenden ein $\binom{n}{1}$-Oblivious-Transfer-Schema an. Dadurch kann B einen Teil des Geheimnisses erhalten ohne daß A weiß, welchen Teil B erhält. Andererseits erhält selbst ein betrügerischer B^* nur genau einen Teil. Die restlichen Teile kann B^* nur mit vernachlässigbarem Vorteil erraten. Solche Ein-Runden-Oblivious-Transfer-Protokolle exisiteren beispielsweise unter der Quadratischen Residuozitätsannahme [KO97, GIKM97] oder unter der *Diffie-Hellman-Annahme*[3] [BM89, BR96], wobei letztere die Diskrete-Logarithmus-Annahme impliziert.

Durch das Release-Protokoll mit Oblivious-Transfer tritt folgendes Problem auf: Wenn beispielsweise B's Geheimnis sehr schwierig zu erraten ist, während nur die Hälfte von A's Bits schwierig vorherzusagen sind, dann läßt sich B zunächst den schwierigen Anteil von A's Geheimnis geben und berechnet den Rest selbst. Umgekehrt gibt B nur die Hälfte seines Geheimnisses preis. Daher ist es zusätzlich wünschenswert, daß B vor Beginn einer Runde nicht festlegen kann, welchen Teil er als nächstes erhält. Wir erreichen dies, indem A nach der Hinterlegung des Geheimnisses $x_1, \ldots, x_n$ die Werte durch eine zufällige Permutation π umordnet und dann die Werte $(x_{\pi(1)}, \ldots, x_{\pi(n)})$ statt $(x_1, \ldots, x_n)$ als "neues" Geheimnis verwendet. Durch Anhängen des Wertes $\pi(i)$ an $x_{\pi(i)}$ kann B nach Erhalt (eventuell in Kombination mit einem Oblivious-Transfer-Protokoll) feststellen, welche Nummer der erhaltene Block hat.

Bei Verwendung von Oblivious-Transfer-Protokollen kennt der Simulator eventuell nicht die als nächstes aufzudeckende Position und kann sich diese nicht vom echten Teilnehmer geben lassen. In diesem Fall können wir keine Aussage über die Sicherheit machen — oder wir modifizieren das Modell und gehen davon aus, daß der Simulator stets die richtige Postion vom echten Teilnehmer erhält. Verwendet man ein Oblivious-Transfer-Protokoll in Verbindung mit der oben beschriebenen Permutationstechnik, so kann der Simulator eine zufällige Position

[3] Gegeben p, q, g und $g^a, g^b \bmod p$ für zufällige $a, b, \in_R \mathbb{Z}_q$ kann man in probabilistischer Polynomialzeit nicht $g^{ab} \bmod p$ berechnen [DH76].

wählen und an B weitergeben. Zusätzlich muß B dann mit jedem open-Befehl durch einen Zero-Knowledge-Beweis zeigen, daß er die entsprechende Position noch nicht abgefragt hat. Sonst könnte B testen, ob er für die gleiche Anfrage den gleichen Wert erhält. In diesem Fall wäre die Simulation inkorrekt, da der Simulator jedesmal eine neue zufällige Position wählt.

4.2 Fairer Austausch anderer Diskreter-Log-Unterschriften

Im folgenden sei — sofern nicht anders angegeben — stets $X = g^x \bmod p$ der öffentliche Schlüssel zum geheimen Schlüssel $g^x \bmod p$, wobei p, q, g wie beim Schnorr-Unterschriftenschema definiert seien.

4.2.1 Release von Okamoto-Unterschriften

Okamoto [O92] erweitert die Schnorr-Identifikation auf den Fall $n = 2$, so daß der zugehörige Proof of Knowledge witness-indistinguishable ist — obwohl mehrere Zeugen existieren. Der geheime Schlüssel besteht in diesem Fall aus zwei Komponenten $(x_1, x_2) \in \mathbb{Z}_q^2$. Der öffentliche Schlüssel ist $X = g^{x_1} g_*^{x_2} \bmod p$. Eine Okamoto-Unterschrift ist damit ein Proof of Knowledge für das Repräsentationsproblem für $n = 2$ bei Verwendung von One-Way-Hashfunktion wie im Fall der Schnorr-Unterschriften. Das Release-Protokoll entspricht dem des vorigen Abschnitts. Wir haben dann eine Repräsentation $s^{(1)}$ und $s^{(2)}$ mit

$$RX^c = g^{s^{(1)}} g_*^{s^{(2)}}$$

Die Blöcke dieser Repräsentation hinterlegt man wie im Fall der Schnorr-Unterschriften. Diesmal verwenden A und B allerdings $2b$ Generatoren $g_1, \ldots, g_b$ und $g_{*,1}, \ldots, g_{*,b}$. Der Generator h kann in beiden Fällen verwendet werden. Die lineare Relation $\sum_i \alpha_i s_i^{(a)} = s^{(a)}$ für $a = 1, 2$ kann parallel durch Brands Protokoll [B97] gezeigt werden.

4.2.2 Release von ElGamal-Unterschriften

Eine ElGamal-Unterschrift [EG85] wird wie folgt erzeugt: Der Unterschreiber berechnet $R = g^k \bmod p$ für zufälliges $k \in \mathbb{Z}_q^*$ und löst die Gleichung $m = xr + ks \bmod p - 1$ nach s. Die Unterschrift ist (R, s). Um das Paar (R, s) zu überprüfen, verifziere man, daß $X^R R^s = g^m \bmod p$. Da g ein Generator von $\mathbb{G}_q$ ist, sind auch X und R Generatoren. Insbesondere hängt der Wert R^s nur von $s \bmod q$ ab. Wir können deshalb analog zu Schnorr-Unterschriften den Wert R ind er nullten Release-Runde senden. Im Proof of Knowledge der linearen Relation verwenden wir dann R statt g und ersetzen in der Korrektheitsbedingung R^s durch $X^{-R} g^m \bmod p$. Beachte, daß der Unterschreiber A den im Release-Protokoll verwendeten, "neuen" Generator R bestimmt. Daß der Wert tatsächlich ein Generator ist, kann man trivialerweise überprüfen. Ferner kann A unter der Diskreten-Logarithmus-Annahme nicht die diskreten Logarithmen $\log_R g_i$ bzw. $\log_{g_i} R$ berechnen. Sonst könnte er bereits die diskreten Logarithmen $\log_{g_i} g$ bzw. $\log_g g_i$ durch

$$\log_{g_1} g_i = \log_{g^k} g_i = 1/k \cdot \log_g g_i \quad \text{und} \quad \log_{g_i} g_1 = \log_{g_i} g^k = k \cdot \log_{g_i} g$$

bestimmen. Pointcheval und Stern [PS96a] geben eine ElGamal-Variante an, die beweisbar sicher bei Verwendung einer idealen Hashfunktion statt einer One-Way-Hashfunktion ist. Man kann leicht zeigen, daß unsere Protokolle für dieses Unterschriftenschema angepaßt werden können.

4.2.3 Release von DSS-Unterschriften

Wir betrachten eine Variation des DSS-Unterschriftenschemas. Zunächst beschreiben wir die original DSS-Unterschriften [DSS]. Sei $R = g^k \bmod p$ und $r = R \bmod q$ für $k \in_R \mathbb{Z}_q^*$. Ferner sei $s = k^{-1}(m+rx) \bmod q$. Die Unterschrift besteht aus dem Paar (r, s). Um eine Unterschrift zu überprüfen, verifiziere man, daß

$$(g^{ms^{-1}} X^{rs^{-1}} \bmod p) \bmod q = r$$

wobei s^{-1} das Inverse von s in $\mathbb{Z}_q^*$ sei.

Unser Release-Protokoll können wir nicht direkt anwenden, da die Verifikationsbedingung nur mod$q$ und nicht mod$p$ gilt. Gennaro, Jarecki, Krawczyk and Rabin [GJKR96] zeigen, daß die mod$q$-Reduktion nur ausgeführt wird, um die Unterschriftenlänge klein zu halten. Wir können daher annehmen, daß die Unterschrift (R, s) und die Korrektheitsbedingung

$$g^m X^R = R^s \bmod p$$

lautet. Wie im Fall der ElGamal-Unterschriften können wir für den Proof of Knowledge der linearen Relation den Wert R statt g verwenden. Das Protokoll kann durch eine Modifikation ebenfalls für die blinden DSS-Unterschriften von Camenisch, Piveteau and Stadler [CPS94] verwendet werden.

4.3 Unterschriften basierend auf Faktorisieren und RSA

Wir diskutieren kurz, daß man unsere Austauschprotokolle für Okamotos Variante [O92] der Guillou-Quisquater-Unterschriften [GQ88] anwenden kann. Unser Prokoll ist etwa so effizient wie Damgårds Verfahren und erlaubt die adaptive Herausgabe der Blöcke. Für andere Unterschriftenschemata wie (Feige-)Fiat-Shamir [FS86, FFS88] bzw. Ong-Schnorr [OS90, S96] ist unser Protokoll nicht geeignet, da diese Unterschriften- bzw. Identifikationsschemata nicht auf dem Repräsentationsproblem basieren.

Wir erweitern zunächst das Repräsentationsproblem auf die Gruppe $\mathbb{Z}_N^*$ für einen RSA-Modul $N = pq$. Eine Repräsentation von $Z \in \mathbb{Z}_N^*$ bezüglich $(g_1, \ldots, g_n, v)$ mit $g_1, \ldots, g_n \in \mathbb{Z}_N^*$ und $v \in \mathbb{Z}_N$ prim ist ein Tupel $(z_1, \ldots, z_n, x)$ mit

$$Z = x^v \cdot \prod_{i=1}^{n} g_i^{z_i} \bmod N$$

In diesem Fall ist die Repräsentationsproblem-Annahme, d.h. gegeben einen zufälligen k-Bit-RSA-Modul $N = pq$ und zufällige $g_1, \ldots, g_n, v \in_Z Z_N^*$ kann man nur mit vernachlässigbarer

Wahrscheinlichkeit in Polynomialzeit ein Z mit verschiedenen Repräsentationen finden, äquivalent zur RSA-Annahme [O92, B97]. Ein Proof of Knowledge für diesen Fall ist in Abbildung 4.7 angegeben. Die Eigenschaften im Fall des Diskreten-Logarithmus-Protokolls übetragen sich unmittelbar. Wir betrachten zur Erläuterung nur die Vollständigkeit:

$$\begin{aligned} RZ^c &= y^v \cdot \prod g_i^{r_i} \cdot x^{cv} \cdot \prod g_i^{cz_i} \bmod N \\ &= x^{cv} \cdot y^c \cdot \prod g_i^{r_i+cz_i} \bmod N \\ &= x^{cv} \cdot y^c \cdot \prod g_i^{s_i+v\lfloor (r_i+cz_i)/v \rfloor} \bmod N \\ &= \left(x^c \cdot y \cdot g_i^{\lfloor (r_i+cz_i)/v \rfloor} \right)^v \cdot \prod g_i^{s_i} \bmod N \\ &= w^v \cdot \prod g_i^{s_i} \bmod N \end{aligned}$$

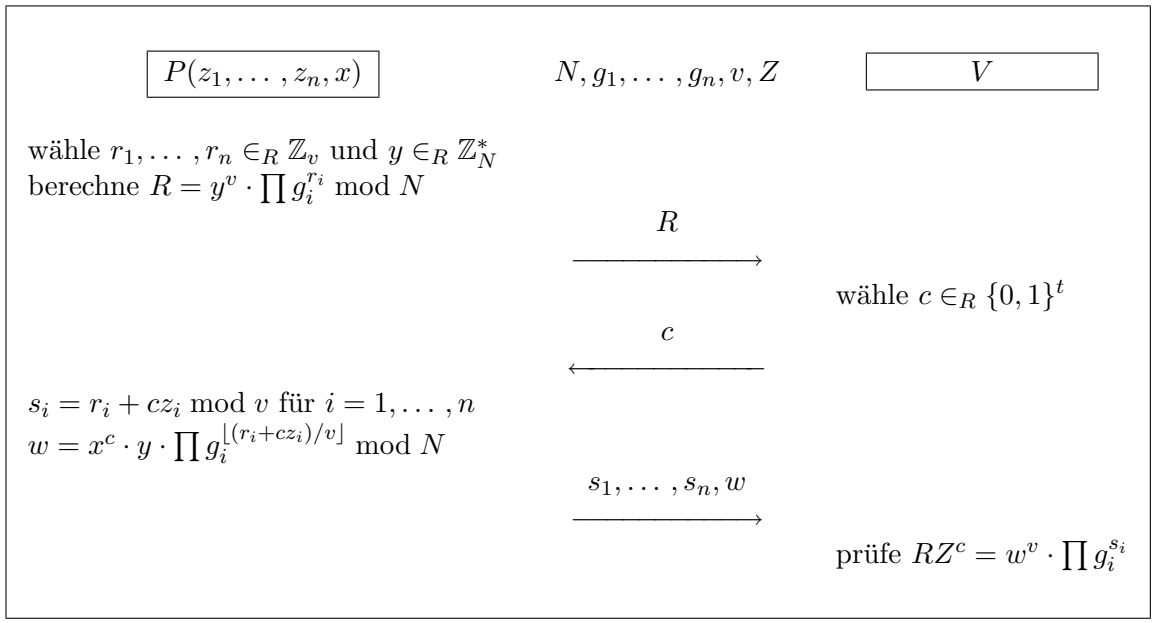

Abbildung 4.7: Proof of Knowledge für $Z = x^v \cdot \prod g_i^{z_i} \bmod N$

Guillou-Quisquater-Unterschriften [GQ88] stimmen mit dem Proof of Knowledge für $n = 0$ überein, wobei $Z = x^v \bmod N$ der öffentliche Schlüssel und x der geheime Schlüssel ist. Eine "lange" Unterschrift zu $m \in \mathbb{Z}_N^*$ ist ein Paar (R, w), wobei $R = y^v \bmod N$ für ein zufälliges $y \in \mathbb{Z}_N^*$ und $w = x^c \cdot y \bmod N$ mit $c = \mathsf{H}(R, Z, m)$. Die Korrektheitsbedingung lautet $RZ^c = w^v \bmod N$. Durch eine One-Way-Hashfunktion erhalten wir analog zu Schnorr-Unterschriften "kurze" Guillou-Quisquater-Unterschriften.

Okamoto [O92] erweitert die Unterschriften auf den Fall $n = 1$, so daß der zugehörige Proof of Knowledge witness-indistinguishable, obwohl mehrere Zeugen existieren. Für diesen Fall erhalten wir ein sicheres Release-Protokoll, wenn der Unterschreiber in der nullten Runde die Werte R und w sendet. Es genügt daher zu zeigen, daß die Unterprotokolle unseres

Verfahrens zum Austausch Diskreter-Logarithmus-Unterschriften auf diesen Fall übetragen werden können.

Wir teilen $s \in \mathbb{Z}_v$ mit $g^s = w^{-v} \cdot RZ^c$ in Blöcke $s_1, \ldots, s_n$ zu je b Bits. Entsprechend seien $\overline{s}_1, \ldots, \overline{s}_n$ definiert. Die Hinterlegung für ein Bit $\overline{s}_{ij}$ erfolgt durch Berechnung von

$$C_{ij} = g_j^{\overline{\alpha}_{ij} \cdot s_{ij}} \cdot u_{ij}^v \bmod N$$

für ein zufälliges $u_{ij} \in_R \mathbb{Z}_N^*$. Diese Hinterlegung hat die gleichen Eigenschaften wie die entsprechende Diskrete-Logarithmus-Hinterlegung, d.h. sie schützt den Wert s_{ij} informationstheoretisch. Insbesondere kann man durch einen witness-indistinguishable Proof of Knowledge nachweisen, daß s_{ij} nur ein Bit darstellt. Brands [B97] zeigt, daß man den Proof of Knowledge einer linearen Relation (mod$v$ statt mod$q$) auch für denn RSA-Fall durchführen kann. Daher kann man die zero-knowledge Proofs of Knowledge für die lineare Relation und s übetragen. Wir zeigen weiter unten, daß das Preprocessing übertragen werden kann, so daß der Release-Protokoll-Simulator "Pseudo"-Hinterlegungen wählen kann. Folglich erhalten wir ein sicheres Release-Protokoll für die Okamoto-Guillou-Quisquater-Unterschriften basierend auf der RSA-Annahme. Während sich Kommunikations- und Rechenaufwandaufwand etwa verdoppeln (da $|N| \approx 2 \cdot |p|$), bleibt die Rundenanzahl identisch.

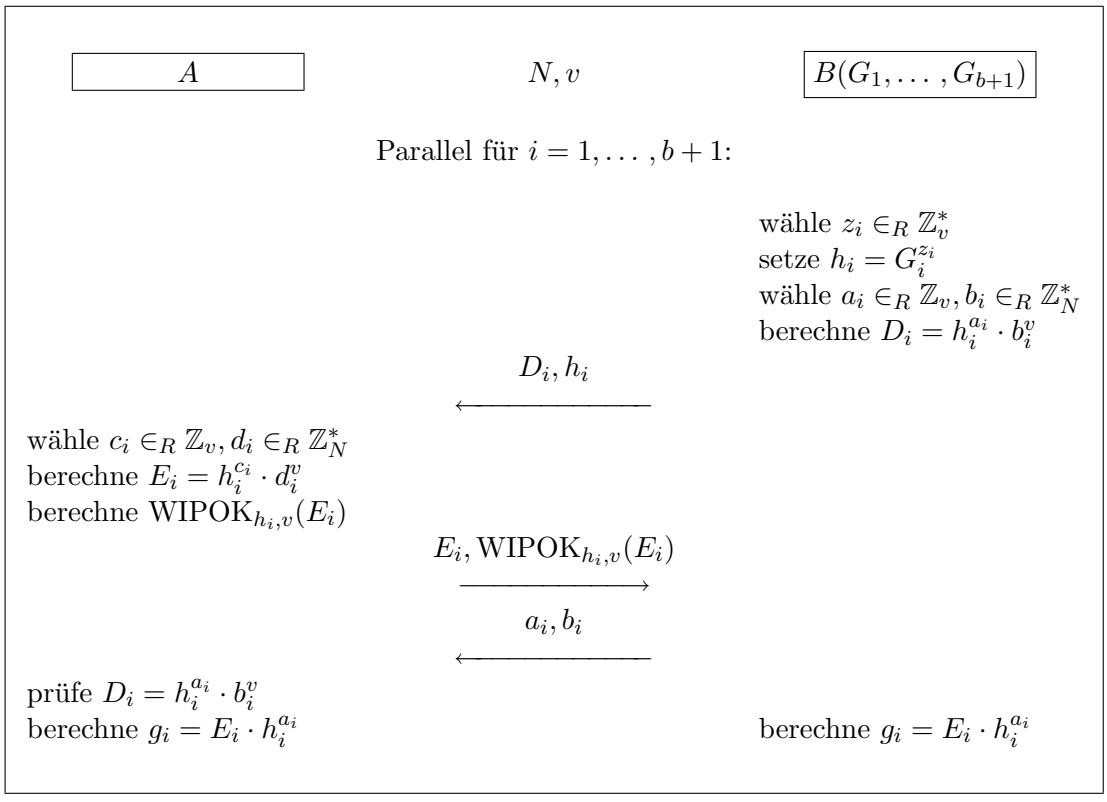

Abbildung 4.8: Preprocessing im RSA-Fall: Erzeuge $g_1, \ldots, g_{b+1}$ aus $G_1, \ldots, G_{b+1}$

Wir betrachten das Preprocessing. Im Diskreten-Logarithmus-Fall kann der Simulator "Pseudo"-Hinterlegungen bezüglich g_i, h erzeugen, wenn er $\log_{g_i} h$ kennt. Im RSA-Fall muß

der Simulator dagegen eine v-te Wurzel $g_i^{1/v}$ von g_i kennen, da er dann eine Hinterlegung

$$C = g_i^{z_i} y^v \bmod N$$

von $(z_i, y) \in \mathbb{Z}_v \times \mathbb{Z}_N^*$ in eine Hinterlegung für (z_i', y') mit $y' = y g_i^{(z_i - z_i')/v}$ umwandeln kann:

$$C = g_i^{z_i} y^v = g_i^{z_i'} \cdot \left(g_i^{(z_i - z_i')/v} y\right)^v = g_i^{z_i'} \cdot y'^v \bmod N$$

Das Preprocessing ist in Abbildung 4.8 angegeben. Die zugrundeliegende Idee und der Sicherheitsbeweis entsprechen denen des Diskreten-Logarithmus-Protokolls. Bei der Simulation wählt der Simulator in der zweiten Phase den Wert $E_i = h_i^{-a_i} d_i'^v$, wobei b_i der in der ersten Phase erhaltene Wert sei. Wenn B in der zweiten Phase dann Werte $(a_i', b_i') \neq (a_i, b_i)$ sendet, die beide den Wert D_i ergeben, dann ist

$$h_i^{a_i - a_i'} = (b_i^{-1} b_i')^v \bmod N$$

und damit kann der Simulator eine v-te Wurzel $(b_i^{-1} b_i')^{1/(a_i - a_i')}$ von h_i und damit von g_i berechnen.

Index

W

Z

Literaturverzeichnis

[ASW97] N.Asokan, V.Shoup, M.Waidner: Optimistic Fair Exchange of Digitial Signatures, *IBM Technical Report RZ 2973, 11/17/1997*, 1997.

[BDG95] J.Balcázar, J.Díaz, J.Gabarró: Structural Complexity I, *Second Edition, Springer Verlag*, 1995.

[B91] D.Beaver: Foundations of Secure Interactive Computing, *Crypto '91, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag*, 1991.

[BG92] M.Bellare, O.Goldreich: On Defining Proofs of Knowledge, *Crypto '92, Lecture Notes in Computer Science, Vol. 740, Springer-Verlag, pp. 390–420*, 1992.

[BGG94] M.Bellare, O.Goldreich, S.Goldwasser: Incremental Cryptography: The Case of Hashing and Signing, *Crypto '94, Lecture Notes in Computer Science, Vol. 839, Springer-Verlag, pp. 216–233*, 1994.

[BJY97] M.Bellare, M.Jakobsson, M.Yung: Round-Optimal Zero-Knowledge Arguments Based on any One-Way Function, *Eurocrypt '97, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp. 280–305*, 1997.

[BM89] M.Bellare, S.Micali: Non-Interactive Oblivious Transfer and Applications, *Crypto '89, Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, pp. 547–557*, 1989.

[BR96] M.Bellare, R.Rivest: Translucent Cryptography — An Alternative to Key Escrow, and its Implementation via Fractional Oblivious Transfer, *ACM Conference on Computer and Communication Security*, 1996.

[BR93] M.Bellare, P.Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *ACM Conference on Computer and Communication Security, pp. 62–73*, 1993.

[BGMR90] M.Ben-Or, O.Goldreich, S.Micali, R.Rivest: A Fair Protocol for Signing Contracts, *IEEE Transaction on Information Theory, Vol. 36, pp. 40–46*, 1990.

[B83] M.Blum: How to Exchange (Secret) Keys, *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing, pp. 440–447*, 1983.

[B93] S.Brands: An Efficient Off-line Electronic Cash System Based on the Representation Problem, *Technical Report CS-R9323, Centrum voor Wiskunde en Informatica*, 1993.

[B97] S.Brands: Rapid Demonstration of Linear Relations Connected by Boolean Operators, *Eurocrypt '97, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp. 318–333*, 1997.

[BCR86a] G.Brassard, C.Crépeau, J.-M.Robert: All-or-Nothing Disclosure of Secrets, *Crypto '86, Lecture Notes in Computer Science, Springer-Verlag*, 1987.

[BCR86b] G.Brassard, C.Crépeau, J.-M.Robert: Information Theoretic Reductions Among Disclosure Problems, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pp. 168–173*, 1986.

[BCS96] G.Brassard, C.Crépeau, M.Santha: Oblivious Transfers and Intersecting Codes, *IEEE Transaction on Information Theory*, 1996.

[BCY91] G.Brassard, C.Crépeau, M.Yung: Constant Round Perfect Zero-Knowledge Computationally Convincing Arguments, *Theoretical Computer Science, Vol. 84, No. 1*, 1991.

[BCDG87] E.F.Brickell, D.Chaum, I.Damgård, J.van de Graaf: Gradual and Verifiable Release of a Secret, *Crypto '87, Lecture Notes in Computer Science, Springer-Verlag, pp. 156–166*, 1987.

[BM92] E.F.Brickell, K.S.McCurley: An Interactive Identification Scheme Based on Discrete Logarithms and Factoring, *Journal of Cryptology, Vol. 5, pp. 29–39*, 1992.

[CPS94] J.Camenisch, J.-M.Piveteau, M.Stadler: Blind Signatures Based on the Discrete Logarithm Problem, *Eurocrypt '94, Lecture Notes in Computer Science, Vol. 950, Springer-Verlag, pp. 428–432*, 1994.

[CFGN96] R.Canetti, U.Feige, O.Goldreich, M.Naor: Adaptively Secure Multi-Party Computation, *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, pp. 639–648*, 1996.

[C89] R.Cleve: Controlled Gradual Disclosure Schemes for Random Bits and Their Applications, *Crypto '89, Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, pp. 573–588*, 1989.

[D93] I.Damgård: Practical and Provably Secure Release of a Secret and Exchange of Signature, *Eurocrypt '93, Lecture Notes in Computer Science, Vol. 765, Springer-Verlag, pp. 200–214*, 1993.

[D95] I.Damgård: Practical and Provably Secure Release of a Secret and Exchange of Signature, *Journal of Cryptology, Vol. 8, pp. 201–222*, 1995.

[DH76] W.Diffie, M.Hellman: New Directions in Cryptography, *IEEE Transaction on Information Theory, Vol. 22, pp. 644–654*, 1976.

[EG85] T.ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transaction on Information Theory, Vol. 31, pp. 469–472*, 1985.

[EGL85] S.Even, O.Goldreich, A.Lempel: A Randomized Protocol for Signing Contracts, *Communications of ACM, Vol. 28, No. 6, pp. 637–647*, 1985.

[FFS88] U.Feige, A.Fiat, A.Shamir: Zero Knowledge Proofs of Identity, *Journal of Cryptology, Vol. 1, pp. 77–94*, 1988.

[FS89] U.Feige, A.Shamir: Zero-Knowledge Proofs of Knowledge in Two Rounds, *Crypto '89, Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, pp. 526–544*, 1989.

[FS90] U.Feige, A.Shamir: Witness Indistinguishable and Witness Hiding Protocols, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing, pp. 416–426*, 1990.

[FS86] A.Fiat, A.Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Schemes, *Crypto '86, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, pp. 186–194*, 1986.

[F98a] M.Fischlin: Fair Exchange of Discrete-Log-Signatures with Flexible Block-By-Block Release, *Manuskript*, 1998.

[F98b] M.Fischlin: Sufficient Conditions for Constructing One-Way Functions from Black-Box-Reductions, *Manuskript*, 1998.

[FR97] M.Franklin, M.Reiter: Fair Exchange with a Semi-Trusted Third Party, *ACM Conference on Computer and Communication Security*, 1997.

[FO97] E.Fujisaki, T.Okamoto: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations, *Crypto '97, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1294, pp. 16–30*, 1997.

[GJKR96] R.Gennaro, S.Jarecki, H.Krawczyk, T.Rabin: Robust Threshold DSS Signatures, *Eurocrypt '96, Lecture Notes in Computer Science, Vol. 1070, Springer-Verlag, pp. 354–371*, 1996.

[GIKM97] Y.Gertner, Y.Ishai, E.Kushilevitz, T.Malkin: Protecting Data Privacy in Private Information Retrieval Schemes, *Manuskript*, 1997.

[G95] O.Goldreich: Foundations of Cryptography (Fragments of a Book), *available at ECCC,* `http://www.eccc.uni-trier.de/eccc/`, 1995.

[GK96] O.Goldreich, A.Kahan: How to Construct Constant-Round Zero-Knowledge Proof Systems for NP, *Journal of Cryptology, Vol. 9, No. 3, pp. 167–189*, 1996.

[GL89] O.Goldreich, L.Levin: A Hard-Core Predicate for any One-Way Function, *Proceedings of the 21st Annual Symposium on the Theory of Computing, pp. 25–32*, 1989.

[GM96] O.Goldreich, B.Mayer: Computational Indistinguishability — Algorithms vs. Circuits, *available at Oded Goldreich's homepage,* `http://theory.lcs.mit.edu/~oded/papers.html`, 1996.

[GMW86] O.Goldreich, S.Micali, A.Wigderson: Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design, *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, pp. 174–187*, 1986.

[GMW91] O.Goldreich, S.Micali, A.Wigderson: Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems, *Journal of ACM, Vol. 38, pp. 691–729*, 1991.

[GO96] O.Goldreich, R.Ostrovsky: Software Protection and Simulation on Oblivious RAM, *Journal of ACM, Vol. 43(3), pp. 431–473*, 1996.

[GM84] S.Goldwasser, S.Micali: Probabilistic Encryption, *Journal of Computer and System Sciences, Vol. 28(2), pp. 270–299*, 1984.

[GMR85] S.Goldwassser, S.Micali, C.Rackoff: The Knowledge Complexity of Interactive Proof Systems, *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, pp. 291–304*, 1985.

[GMR89] S.Goldwassser, S.Micali, C.Rackoff: The Knowledge Complexity of Interactive Proof Systems, *SIAM Journal on Computation, Vol. 18, pp. 186–208*, 1989.

[GMR88] S.Goldwassser, S.Micali, R.Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks, *SIAM Journal on Computation, Vol. 17(2), pp. 281–308*, 1988.

[GQ88] L.C.Guillou, J.-J.Quisquater: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory, *Eurocrypt '88, Lecture Notes in Computer Science, Vol. 330, Springer-Verlag, pp. 123–128*, 1988.

[HL91] L.Harn, H.-Y.Lin: An Oblivious Transfer Protocol for the Exchange of Secrets, *Asiacrypt '91, Lecture Notes in Computer Science, Vol. 739, Springer-Verlag, pp. 312–320*, 1991.

[KO97] E.Kushilevitz, R.Ostrovsky: Replication is Not Needed: Single Database, Computationally-Private Information Retrieval, *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, pp. 364–373*, 1997.

[LL94] C.H.Lim, P.J.Lee: More Flexible Exponentiation with Precomputation, *Crypto '94, Lecture Notes in Computer Science, Vol. 839, Springer-Verlag, pp. 95–107*, 1994.

[LMR83] M.Luby, S.Micali, C.Rackoff: How to Simultanously Exchange a Secret Bit by Flipping a Symmetrically Biased Coin, *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, 1983.

[M92] U.Maurer: Some Number-Theoretic Conjectures and Their Relation to the Generation of Cryptographic Primes, *Cryptography and Coding II, Oxford University Press, pp. 173–191*, 1992.

[M95] U.Maurer: Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters, *Journal of Cryptology, Vol. 8, pp. 123–155*, 1995.

[MR91] S.Micali, P.Rogaway: Secure Computation, *Crypto '91, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag*, 1991.

[DSS] National Institute of Standards and Technology: Digital Signature Standard (DSS), *Federal Register Vol. 56(169)*, 1991.

[SHA] National Institute of Standards and Technology: Secure Hash Standard (SHS), *Federal Information Processing Standard, Publication 180*, 1993.

[R79] M.Rabin: Digitalized Signatures and Public-Key Functions as Intractable as Factorization, *MIT-LCS-TR 212, MIT Laboratory for Computer Science*, 1979.

[RSA78] R.Rivest, A.Shamir, L.Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of ACM, Vol. 21(2), pp. 120–126*, 1978.

[O92] T.Okamoto: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes, *Crypto '92, Lecture Notes in Computer Science, Springer-Verlag, Vol. 740, pp. 31–53*, 1992.

[OO94] T.Okamoto, K.Ohta: How to Simultaneously Exchange Secrets by General Assumptions, *ACM Conference on Computer and Communication Security, pp. 184–192*, 1994.

[OS90] H.Ong, C.P.Schnorr: Fast Signature Generation with a Fiat-Shamir Identification Scheme, *Eurocrypt '90, Lecture Notes in Computer Science, Vol. 473, Springer-Verlag, pp. 432–440*, 1990.

[P91] T.P.Pedersen: Non-Interactive and Information-Theoretical Secure Verifiable Secret Sharing, *Crypto '91, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag, pp. 129–140*, 1991.

[PS96a] D.Pointcheval, J.Stern: Security Proofs for Signature Schemes, *Eurocrypt '96, Lecture Notes in Computer Science, Vol. 1070, Springer-Verlag, pp. 387–398*, 1996.

[PS96b] D.Pointcheval, J.Stern: Provably Secure Blind Signature Schemes, *Asiacrypt '96, Lecture Notes in Computer Science, Springer-Verlag*, 1996.

[PS98] D.Pointcheval, J.Stern: Security Arguments for Digital Signatures and Blind Signatures, *Sumbitted to Journal of Cryptology*, 1998.

[S91] C.P.Schnorr: Efficient Signature Generation by Smart Cards, *Journal of Cryptology, Vol. 4, pp. 161–174*, 1991.

[S96] C.P.Schnorr: Security of 2^t-Root Identification and Signatures, *Crypto '96, Lecture Notes in Computer Science, Vol. 1109, Springer-Verlag*, 1996.

[T84] T.Tedrick: Fair Exchange of Secrets, *Crypto '84, Lecture Notes in Computer Science, Vol. 196, Springer-Verlag*, 1985.

[VV83] U.Vazirani, V.Vazirani: Trapdoor Pseudorandom Number Generators with Applications to Protocol Design, *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science, pp. 23–30*, 1983.

[Y86] A.Yao: How to Generate and Exchange Secrets, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986.