Zur Analyse und Struktur von Sicherheitsbegriffen

Zur Erlangung des akademischen Grades eines Doktors
der Naturwissenschaften von der Fakultät für Informatik
der Universität Karlsruhe (Technische Hochschule)

genehmigte

Dissertation

von

Dennis Hofheinz

aus Siegen

Tag der mündlichen Prüfung:	9. Mai 2005
Erster Gutachter:	Prof. Dr. Thomas Beth
Zweiter Gutachter:	Prof. Dr. Dieter Gollmann

Inhaltsverzeichnis

Meinen Eltern

nrtrtaxwlzszcvhqahhmmfjel
xkoxqbdhwlzf
hcvqgnogvdblvkqtrahbzcda
hmdumxcapphvvpwdypto
faclgawxwtteanwvt
dgmzwqgthzddah
pzyvhkfnurtapcyvzr
tcprttuvubxwvva

Dank

An erster Stelle möchte ich mich bei Herrn Professor Dr. Thomas Beth bedanken, ohne dessen Unterstützung die vorliegende Arbeit nicht möglich gewesen wäre. Er hat mein Interesse für kryptographische Mechanismen geweckt, und ihm verdanke ich eine algebraische Sicht auf die Informatik.

Weiter möchte ich mich bei Herrn Professor Dr. Dieter Gollmann für die Übernahme des Korreferats und für viele hilfreiche und konstruktive Verbesserungsvorschläge bedanken.

Den Mitarbeitern des Instituts für Algorithmen und Kognitive Systeme danke ich für eine überaus angenehme Arbeitsatmosphäre. Insbesondere möchte ich Herrn Dr. Jörn Müller-Quade und Herrn Dr. Rainer Steinwandt für eine Vielzahl interessanter und fruchtbarer fachlicher und nicht-fachlicher Diskussionen danken.

Auch danke ich Herrn Dipl.-Inform. Jens Bohli, Herrn Dr. Willi Geiselmann, Herrn Dr. Markus Grassl, Herrn Dipl.-Inform. Stefan Röhrich und Herrn Dipl.-Inform. Dominique Unruh für fachliche, organisatorische und TEXnische Hilfestellung.

Für interessante fachliche Kooperation danke ich Frau Dr. María Isabel González Vasco, Herrn Dr. Michael Backes und Herrn Dr. Ran Canetti.

1 Einleitung

1.1 Überblick

In dieser Arbeit soll am Beispiel der Public-Key-Kryptographie zunächst eine Motivation für simulationsbasierte Sicherheitsbegriffe entwickelt werden. Daß hier eine rein informelle Betrachtung von Sicherheit für Public-Key-Kryptosysteme problematisch ist, wird anhand verschiedener Kryptoanalysen von Public-Key-Kryptosystemen gezeigt. Anschließend wird eine Verbindung zwischen simulationsbasierten und klassischen Sicherheitsbegriffen für die kryptographischen Aufgaben

- Public-Key-Kryptographie,
- digitale Signaturen,
- Schlüsselaustausch und
- Bit Commitment

hergestellt. Dabei können jeweils klassische Begriffe identifiziert werden, die hinreichend für oder sogar äquivalent zu simulierbarer Sicherheit sind und damit eine sichere modulare Verwendbarkeit der untersuchten Protokolle gewährleisten.[1] Hierbei wird immer wieder an Beispielen in der Literatur auf die Notwendigkeit von genauen Beweisen hingewiesen.

Abschließend wird die Bedeutung der Netzwerkmodellierung bei Mehrparteienprotokollen verdeutlicht: Es ist bekannt, daß die Primitive „Oblivious Transfer“ hinreichend für die simulierbar sichere Realisierung allgemeiner Mehrparteienberechnungen ist – bei unterstelltem asynchronen Netzwerk. Es kann gezeigt werden, daß dies bei einem *verläßlichen* Netzwerk nicht mehr der Fall ist.

1.2 Public-Key-Kryptographie

1.2.1 Einführung

Ein damals in der öffentlichen Forschung gänzlich neuer Ansatz, der ohne einen bei Secret-Key-Verfahren nötigen gemeinsamen *geheimen* Schlüssel auskommt, wurde 1976 von Diffie und Hellman in ihrer Arbeit [54] vorgestellt. Jeder Teilnehmer

[1] Man beachte, daß eine sichere modulare Verwendbarkeit kryptographischer Verfahren im allgemeinen nicht gegeben ist.

in einem *Public-Key*-Kryptosystem generiert zunächst ein *Paar* von Schlüsseln: einen öffentlichen (eben den *Public* Key), der allen anderen Teilnehmen zugänglich gemacht wird, und einen geheimen Schlüssel. Mit dem öffentlichen Schlüssel eines Teilnehmers werden an ihn adressierte Nachrichten chiffriert, sollten aber *nicht* dechiffriert werden können. Die Dechiffrierung sollte idealerweise nur unter Kenntnis des zugehörigen geheimen Schlüssels möglich sein.

Ein Public-Key-Kryptosystem kann durch drei Algorithmen KeyGen, Encrypt und Decrypt mit offensichtlicher Semantik modelliert werden. Um nun kryptographische Sicherheit sinnvoll zu fassen, reicht es *nicht*, zu fordern, daß Encrypt „schwer" invertierbar sein sollte – etwa in dem Sinne, daß man von Encrypt analoge Eigenschaften zu denen einer Einwegfunktion fordert. Man betrachte hierzu etwa ein System P', das aus einem wie beschrieben „sicheren" System P gewonnen wird, indem der Algorithmus Encrypt nur zur Chiffrierung der ersten $\lfloor |m|/2 \rfloor$ Bits einer Nachricht m verwendet wird; der restliche Teil von m wird dem Chiffrat im Klartext hinzugefügt. Die Änderungen für Decrypt ergeben sich kanonisch, und das resultierende System P' ist damit ein Public-Key-Kryptosystem *und* immer noch sicher im vorgeschlagenen Sinn. Dennoch ist P' intuitiv ganz offenbar unsicher.

Auch sollte sich Encrypt nicht deterministisch verhalten. Andernfalls liefert beispielsweise bei kleinem Nachrichtenraum (der etwa lediglich die beiden Nachrichten „`annehmen`" und „`ablehnen`" enthalte) eine vollständige Suche über die – bei deterministischem Encrypt eindeutigen – Chiffrate der möglichen Klartexte eine komplette Invertierung von Encrypt.

Man ist versucht, dieses Problem mit einem geeigneten „padding" der Klartexte anzugehen, d. h. die Klartexte einfach mit einem zufällig erzeugten Bitstring gewisser Länge aufzufüllen und so die Chiffrate gewissermaßen „künstlich" zu randomisieren. Eine solche Methode wurde beim Entwurf des Verschlüsselungsstandards PKCS #1 (siehe [118]) verwendet. Version 1.5 dieses Standards wurde von Bleichenbacher in [23] mittels eines Chosen-Ciphertext-Angriffs[2] erfolgreich angegriffen – in neueren Versionen des PKCS #1 kommt deshalb eine Variante des bzgl. einer idealisierten Hashfunktion als sicher bewiesenen OAEP-Verfahrens [19] von Bellare und Rogaway zum Einsatz.

1.2.2 Klassische Sicherheitsdefinitionen

Ein entscheidender Fortschritt bezüglich einer stimmigen Sicherheitsdefinition für Public-Key-Kryptosysteme gelang Goldwasser und Micali in ihrer grundlegenden Arbeit [75]. Diese Arbeit lieferte nicht nur eine formale und dennoch intuitive Formulierung von Sicherheit („*semantische Sicherheit*"), es konnte auch ein handliches Kriterium für Sicherheit in diesem Sinne angegeben und hiermit eine einfache – wenngleich sehr aufwendige – Konstruktion als der neuen Definition genügend gezeigt werden. Informell heißt ein System „semantisch sicher", wenn jede Funktion

[2] Bei einem Chosen-Ciphertext-Angriff steht dem Angreifer ein Entschlüsselungsorakel zur Verfügung, vgl. Naor und Yung [110], Rackoff und Simon [121], Shoup [126].

auf Klartexten ohne Kenntnis des jeweiligen Chiffrats so gut approximiert werden kann wie unter Kenntnis dieses Chiffrats. Zwar legt ein Chiffrat eindeutig einen Klartext fest, doch „hilft“ das bei keiner effizient durchführbaren Berechnung!

Obgleich semantische Sicherheit alle genannten ad-hoc-Sicherheitskriterien impliziert, wurden in [75] prinzipiell keine Chosen-Ciphertext-Angriffe betrachtet. In der Tat wurde etwa für das ElGamal-Kryptosystem aus [58] semantische Sicherheit – unter Annahme der Härte des Decisional-Diffie-Hellman-Problems in der verwendeten Familie von Gruppen – von Tsiounis und Yung in [135] gezeigt. Und doch ist die Eignung des ElGamal-Systems für gewisse Szenarien begrenzt; es sei im folgenden ein Beispiel von Dolev u. a. aus [55] wiedergegeben.

Angenommen, eine Auktion findet statt, bei der jeder Bieter dem Auktionator sein Gebot (etwa in Euro) kundtut, indem er es ihm – chiffriert mit dem öffentlichen Schlüssel des Auktionators – zuruft. Der als ehrlich angenommene Auktionator entschlüsselt dann alle diese Chiffrate und verkündet das entsprechende Ergebnis. Intuitiv sollte in diesem Szenario ein „sicheres“ Public-Key-Kryptosystem insbesondere gewährleisten, daß kein Bieter sein Gebot in Abhängigkeit der Gebote der anderen Bieter abgeben kann. Obwohl semantisch sicher, kann das ElGamal-System dies nicht erbringen: Erlauscht Bieter B das Gebot $c_A \leftarrow \mathsf{Encrypt}(m_A)$ eines Bieters A, so kann er beispielsweise im Falle $10|m_A$ mit eigenem Gebot $c_B := (11/10) \cdot c_A$ das Gebot von A um 10% überbieten.

In *diesem* Auktionsszenario tritt der beobachtete Mißstand nicht auf bei Verwendung eines *non-malleable*[3] Public-Key-Kryptosystems. Dieser Sicherheitsbegriff wurde in [55] eingeführt und fordert intuitiv, daß aus einem gegebenen Chiffrat kein weiteres erzeugt werden kann, so daß die zugehörigen Klartexte in einer nicht-trivialen Relation stehen. Als weiteres Resultat wurde in [55] gezeigt, daß non-malleability schon semantische Sicherheit impliziert.

Prinzipiell existieren noch einschränkendere Sicherheitsbegriffe für Public-Key-Kryptosysteme als die gerade vorgestellten, etwa der in [19] zunächst nur im „Random-Oracle-Modell“ (d. h. unter Zuhilfenahme idealisierter Hashfunktionen) formulierte Begriff der „plaintext awareness“. Es sind andererseits alle erwähnten Begriffe – teils im Random-Oracle-Modell – „erreichbar“. Stark abhängig davon, ob der Beweis im Random-Oracle-Modell geführt wird oder nicht, und natürlich welcher Sicherheitsbegriff gefordert wird, variiert allerdings der Aufwand für bekannte „sichere“ Konstruktionen beträchtlich: Zu einer Nachricht m benötigt etwa im Falle des semantisch sicheren ElGamal-Systems ein Chiffrat $2|m|$ Bits; fordert man semantische Sicherheit auch gegen Chosen-Ciphertext-Angriffe, benötigt ein Chiffrat etwa im diesbezüglich sicheren System [49] von Cramer und Shoup $4|m|$ Bits. Das unter anderem im SSL-Verfahren [129] praktisch eingesetzte und in Fujisaki u. a. [63] im Random-Oracle-Modell als semantisch sicher gegen Chosen-Ciphertext-Angriffe bewiesene „RSAES-OAEP“-Verfahren (vgl. [119]) hingegen benötigt je nach Parameterwahl etwa $1{,}5|m|$ Bits.

[3] Übersetzen könnte man dies etwa mit der Forderung nach einer *Nicht-Verformbarkeit* der Chiffrate.

1.2.3 Fragestellung

Es stellen sich mehrere Fragen:

- Sind die bei einem Sicherheitsbeweis eines Systems getroffenen Annahmen realistisch? Anders gefragt, sind die genutzten mathematischen Probleme „hart“ genug, um für kryptographische Zwecke von Nutzen zu sein? Im Hinblick auf die Bedrohung der zwei häufig für kryptographische Zwecke genutzten Probleme „diskreter Logarithmus“ und „Faktorisieren“ durch Algorithmen auf Quantencomputern (vgl. Shor [125]) ist hier insbesondere eine Untersuchung neuer mathematischer Primitive von Interesse.

- *Welcher* Sicherheitsbegriff ist *wann* hinreichend für kryptographische Sicherheit eines größeren Protokolls, welches Public-Key-Kryptographie verwendet? Hier ist auch die Frage nach der *Komponierbarkeit* von Public-Key-Kryptosystemen relevant: Was kann über die Sicherheit von – für sich genommen – „idealen“ Public-Key-Kryptosystemen bei der Verwendung in größeren Protokollen ausgesagt werden?

In den folgenden beiden Abschnitten soll auf diese beiden Fragen eingegangen werden.

1.2.4 Untersuchung neuer Primitive

Um die erste gestellte Frage zu behandeln, sollen in Kapitel 2 zunächst einige neuartige und experimentelle Wege der Public-Key-Kryptographie untersucht werden. Dabei sollen praktisch auftretende Probleme aufgezeigt werden – etwa im Falle des anzusprechenden Zopfgruppensystems ist die konkrete Wahl der verwendeten Parameter eine vieldiskutierte Frage. Aber auch grundsätzliche Probleme wie im Falle eines auf Grigorchuk-Gruppen basierenden Systems werden aufgezeigt.

Von Fellows und Koblitz in [61] vorgeschlagen, sollte *Polly Cracker* aller Skepsis von Barkee u. a. [15] zum Trotze als Beleg dafür dienen, daß das Problem des Lösens eines multivariaten Gleichungssystems als Grundlage für ein Public-Key-Kryptosystem dienen kann. Genauer konnte in den Arbeiten [61] und Koblitz [98] das Finden des geheimen Schlüssels allein aus dem öffentlichen für vorgeschlagene Varianten von Polly Cracker als äquivalent zu den Suchvarianten verschiedener NP-vollständiger Entscheidungsprobleme gezeigt werden.

Für die hiesige Betrachtung ist dabei nicht von Belang, inwieweit schwierige *Instanzen* von NP-Problemen kanonisch generiert werden können; vielmehr konnte Hofheinz in [81] einen Weg entwickeln, den Verschlüsselungsalgorithmus durch Ausnutzen struktureller Eigenschaften von Polly Cracker „punktweise“ zu invertieren. Die praktische Anwendbarkeit dieser Methode wurde von Hofheinz und Steinwandt in [88] demonstriert, indem eine bis dahin ungebrochene Polly-Cracker-Variante aus [98] implementiert und – experimentell verifiziert – erfolgreich angegriffen wurde.

Ein Beispiel für einen kompletten strukturellen Bruch eines Systems liefert das von Garzon und Zalcstein in [65] vorgestellte Public-Key-Kryptosystem. Das System nutzt eine durch eine ternäre Folge χ bestimmte Grigorchuk-Gruppe G_χ und die Schwierigkeit des Wortproblems in G_χ bei Unkenntnis von χ. G_χ kann dabei als Permutationsgruppe der Pfade durch den unendlichen Binärbaum aufgefaßt werden und ist nicht endlich präsentierbar.

Allerdings konnte von González Vasco, Hofheinz, Martínez und Steinwandt in [78] aufgezeigt werden, daß die vorgeschlagene Nutzung der Schwierigkeit des Wortproblems für ein Public-Key-Kryptosystem unzureichend ist: Ein logarithmisch kleiner Teil des geheimen Schlüssels ermöglicht schon Entschlüsselung beliebiger Chiffrate. Letztlich kann so mittels einer vollständigen Suche über einen hinreichend großen (aber dennoch logarithmisch kleinen) Teilschlüssel ein äquivalenter geheimer Schlüssel gefunden werden. Dies ist insofern bemerkenswert, als daß das System damit allein unter Kenntnis des öffentlichen Schlüssels vollständig strukturell gebrochen wird.

Umgekehrt kann auch versucht werden, allein das zugrundeliegende Problem eines Public-Key-Kryptosystems anzugreifen: Hier seien als Beispiel die Systeme aus Ko u. a. [97] und Anshel u. a. [3], Anshel u. a. [2], deren Sicherheit auf der Schwierigkeit des Konjugationsproblems in Zopfgruppen fußt, genannt. Das Konjugationsproblem in Zopfgruppen kann – für die in [97, 3, 2] vorgeschlagene Wahl der Instanzen – mittels eines in Hofheinz und Steinwandt [89] entwickelten Algorithmus gelöst werden. Die Güte dieses heuristischen Algorithmus konnte zudem in [89] experimentell verifiziert werden.

Insbesondere konnte die in [97, 3, 2] vorgeschlagene Wahl der Instanzen des Konjugationsproblems erstmals als nicht hinreichend gezeigt werden. Erst die späteren Arbeiten Dehornoy [53], Sibert [128] schlagen einen Weg vor, den Angriff aus [89] zu umgehen. Beachtenswert hierbei ist, daß das System [97] im Random-Oracle-Modell von Ko u. a. in [96] als semantisch sicher gezeigt werden konnte; jedoch wurde dabei die Härte des Decisional-Diffie-Hellman-Problems in Zopfgruppen vorausgesetzt. Diese Annahme ist, wie [89] zeigt, bei vorgeschlagener Instanzenwahl nicht gerechtfertigt.[4]

1.2.5 Anwendungsspezifische Sicherheit

In Kapitel 3 soll anschließend die zweite aufgeworfene Frage besprochen werden, also die Frage nach einem anwendungsspezifisch *geeigneten* Sicherheitsbegriff für Public-Key-Kryptosysteme. Wie eingangs angesprochen existiert eine Vielfalt von unterschiedlichen und teilweise unvergleichbaren Sicherheitsbegriffen für Public-Key-Kryptosysteme. Selbst wenn man bereit ist, sich auf eine gewisse Anwendung festzulegen, ist nicht offensichtlich, welcher Sicherheitsbegriff anwendungsbezogene Sicherheit eines Systems garantiert. Für viele praktische Belange liegt es nahe, sich

[4] Tatsächlich konnte kürzlich sogar für das (härtere) Search-Diffie-Hellman-Problem in Zopfgruppen ein polynomialer Algorithmus angegeben werden, vgl. Cheon und Jun [45].

auf die Betrachtung des Szenarios „sichere Nachrichtenübertragung“ zurückzuziehen. Genauer ist eine Realisierung folgender Protokollaufgabe gefragt:

1. Eine dedizierte Partei (der „Empfänger“) schickt eine Initialisierungsnachricht (ihren öffentlichen Schlüssel) an alle anderen Protokollteilnehmer.

2. Danach können alle anderen Parteien durch Senden jeweils nur einer Nachricht beliebige Texte an den Empfänger schicken. Ein Angreifer soll dabei *nichts* außer der Größe der übertragenen Nachricht lernen.[5]

Diese Protokollaufgabe ist auf die Verwendung eines Public-Key-Kryptosystems zugeschnitten; jedoch ist noch zu entscheiden, wann ein gegebenes System als „sichere Realisierung“ dieser Protokollaufgabe anzusehen ist.

Hierzu soll auf die Arbeit [27] von Canetti zurückgegriffen werden, die ein formales Modell zur Betrachtung von Mehrparteienprotokollen festlegt. Bei diesem Ansatz sind alle Parteien und der Angreifer als interaktive Turingmaschinen (ITMs) modelliert; gleichsam wird eine *Protokollumgebung*, die Protokolleingaben festlegt und -ausgaben liest, als ITM modelliert. Ein konkretes („*reales*“) Protokoll kann in diesem Kontext von den Parteien ausgeführt und Attacken durch den Angreifer ausgesetzt werden; beispielsweise hat der Angreifer Zugriff auf die Kommunikation zwischen den Parteien.

Das Modell [27] erlaubt aber auch die formale Modellierung der zu erfüllenden Protokollaufgabe. Genauer kann die Protokollaufgabe (z. B. „sichere Nachrichtenübertragung“) als „*ideale Funktionalität*“ genannte ITM beschrieben werden, die *alle* Eingaben der Protokollumgebung liest, und danach – anstelle der eigentlichen „realen“ Parteien – Protokollausgaben generieren kann. In diesem „*idealen*“ Modell existiert zwar ein Angreifer, dessen Angriffsmöglichkeiten aber allein aufgrund der Modellierung der idealen Funktionalität als einzelne und unkorrumpierbare ITM stark eingeschränkt sind.

Ein Protokoll π wird nun als sichere Realisierung einer Protokollaufgabe (d. h. einer idealen Funktionalität) $\mathcal{F}$ betrachtet, wenn für jeden Angreifer $\mathcal{A}$ auf das reale Protokoll ein Angreifer $\mathcal{S}$ auf die ideale Funktionalität existiert, so daß für keine Protokollumgebung feststellbar ist, ob sie mit π und $\mathcal{A}$ oder mit $\mathcal{F}$ und $\mathcal{S}$ läuft. Durch diesen Zusammenhang ist gewährleistet, daß jeder Angriff auf π durch einen (nach Konstruktion „harmlosen“) Angriff auf $\mathcal{F}$ *simulierbar* ist. Man spricht deshalb von *Simulierbarkeit*.

Die Frage der *Komposition*, d. h. der Benutzung von in diesem Sinne sicheren Protokollen in größeren Protokollen, kann durch ein Kompositionstheorem aus [27] beantwortet werden: Benutzt ein Protokoll ρ eine ideale Funktionalität $\mathcal{F}$, so kann $\mathcal{F}$ durch Invokationen eines Protokolls π ersetzt werden, sofern π die Funktionalität $\mathcal{F}$ realisiert. Bei dieser Ersetzung geht in folgendem Sinne keine Sicherheit verloren: Realisiert ρ eine Funktionalität $\mathcal{G}$, dann realisiert auch das durch Ersetzung von $\mathcal{F}$ durch π entstandene Protokoll $\mathcal{G}$.

[5] Es muß davon ausgegangen werden, daß die Größe einer übertragenen Nachricht nicht verschleiert werden kann, siehe hierzu etwa Goldreich [69, Appendix A].

Im Falle des konkreten Beispiels „sichere Nachrichtenübertragung“ von oben konnte nun in Hofheinz, Müller-Quade und Steinwandt [85] gezeigt werden, daß genau die IND-CCA-sicheren Public-Key-Kryptosysteme (vgl. etwa Bellare u. a. [17]) beliebige Verwendung in größeren Protokollen erlauben: Für ein Public-Key-Kryptosystem ist IND-CCA-Sicherheit *äquivalent* dazu, eine intuitive und komponierbare Idealisierung eines Public-Key-Kryptosystems zu realisieren, welche ihrerseits „sichere Nachrichtenübertragung“ realisiert.

1.2.6 Public-Key-Signatursysteme

Es bietet sich an, auch nach Kriterien für die Komponierbarkeit von *digitalen Signatursystemen* zu fragen; diese Frage soll in Abschnitt 3.3 vertieft werden. Hierfür soll unter einem digitalen Signatursystem gemäß Goldwasser u. a. [77] ein Tupel (KeyGen, Sign, Verify) von Algorithmen verstanden werden, wobei KeyGen wie bei einem Public-Key-Kryptosystem zur Erzeugung von Schlüsselpaaren dient. Mit Sign können unter Verwendung des *geheimen* Schlüssels beliebige Nachrichten signiert werden, deren Signaturen dann mittels Verify und unter Verwendung des passenden *öffentlichen* Schlüssels überprüft werden können. Natürlich sollte es *ohne* Kenntnis des passenden geheimen Schlüssels schwierig sein, Nachrichten (bzgl. Verify „gültig“) zu signieren.

Es wurde in [40] von Canetti und Rabin eine Idealisierung $\mathcal{F}_{\text{SIG}}$ von Public-Key-Signaturen gefunden. Diese Idealisierung wurde dort als genau durch solche Signatursysteme realisiert bewiesen, die im Sinne von „*existentially unforgeable under adaptive chosen-message attacks*“ (EF-CMA) sicher sind. Allerdings war der Beweis dieser Aussage fehlerhaft; schlimmer noch, die angegebene Idealisierung $\mathcal{F}_{\text{SIG}}$ von digitalen Signaturen konnte von Backes und Hofheinz in [7] als von *keinem* Protokoll realisierbar gezeigt werden. In [7] wurde deshalb eine korrigierte Funktionalität $\mathcal{F}_{\text{SIG}}^{(i)}$ angegeben und die Äquivalenz zum Sicherheitsbegriff EF-CMA bewiesen. Demnach ist mit EF-CMA auch ein komponierbarer Sicherheitsbegriff für digitale Signatursysteme identifiziert. Zudem konnte in [7] ein allgemeines Problem von simulierbar sicheren Signatursystemen aufgezeigt werden, welches im Zusammenhang mit einem korrumpierten (d. h. einem sich nicht an das Protokoll haltenden) Signierer auftritt.[6]

1.3 Weitere kryptographische Anwendungen

1.3.1 Schlüsselaustausch

In Kapitel 4 soll auch für andere Protokollaufgaben der Zusammenhang zwischen simulierbarer Sicherheit (und damit verbunden *intuitiven* und *Komponierbarkeit*

[6] Hier sollte nicht unerwähnt bleiben, daß Canetti [30, 31, 32] seinerseits eine Abwandlung der Funktionalität $\mathcal{F}_{\text{SIG}}^{(i)}$ aus [7] vorschlägt; ein Kommentar hierzu kann in Abschnitt 3.3 gefunden werden.

garantierenden Idealisierungen) und klassischen Sicherheitsdefinitionen untersucht werden.

Im Zusammenhang mit symmetrischen Kryptosystemen, welche ein vorhandenes gemeinsames Geheimnis erfordern, ist die Zweiparteien-Protokollaufgabe „Schlüsselaustausch“ betrachtenswert. Informell soll hier ein einzelner, gemäß einer vorgegebenen Verteilung zufällig gewählter Wert an beide Protokollteilnehmer verteilt werden. Ein Angreifer soll hierbei – abgesehen von Längeninformation – keine Information über den verteilten Schlüssel erhalten; es sei denn natürlich, eine der Parteien ist korrumpiert, hält sich also nicht an das Protokoll und kooperiert mit dem Angreifer.

Im korrumpierten Fall kann es, je nach Protokoll und korrumpierter Partei, unvermeidlich sein, daß der Angreifer (bzw. die korrumpierte Partei) zudem die Wahl des Schlüssels beeinflußt. Etwa im Falle eines Diffie-Hellman-Schlüsselaustauschs kann der Empfänger des initialen Wertes g^x durch geschickte Wahl von y erzwingen, daß etwa die letzten 8 Bits des endgültigen Sitzungsschlüssels g^{xy} alle 0 sind.

In [36] wurde deshalb für die Betrachtung im schon vorgestellten Simulierbarkeitsmodell [27] von Canetti und Krawczyk eine Idealisierung $\mathcal{F}_{\text{KE}}$ von „Schlüsselaustausch“ definiert, bei der im Falle einer korrumpierten Partei der Schlüssel vollständig vom (idealen) Angreifer festgelegt werden darf. Es wurden dann eine Variante des Diffie-Hellman-Protokolls und ein Public-Key-gestütztes Schlüsseltransportprotokoll als simulierbar sicher im Sinne von $\mathcal{F}_{\text{KE}}$ realisierend gezeigt. Jedoch war auch dieser Beweis fehlerhaft; in [84] konnte von Hofheinz, Müller-Quade und Steinwandt nachgewiesen werden, daß $\mathcal{F}_{\text{KE}}$ tatsächlich durch *kein* Protokoll realisierbar ist. Im Zuge dessen konnte die Idealisierung $\mathcal{F}_{\text{KE}}$ korrigiert und die ursprüngliche Behauptung für die korrigierte Version bewiesen werden.

Zudem konnte in [84] eine stärkere und neuartige Idealisierung $\mathcal{F}_{\text{KE}}^{+}$ angegeben werden, bei der nur im Falle eines korrumpierten *Empfängers* (d. h. des Empfängers der ersten Protokollnachricht) der Angreifer den Sitzungsschlüssel frei wählen darf; im Falle eines korrumpierten *Senders* (der ersten Protokollnachricht) wird der Sitzungsschlüssel jedoch ideal und ohne Einflußmöglichkeit des Angreifers gewählt. Natürlich ist eine solche Idealisierung nur sinnvoll, wenn auch Protokolle angegeben werden können, die diese Idealisierung erfüllen. Dies geschah in Form eines Sicherheitsbeweises für Varianten der schon angesprochenen Protokolle. Außerdem konnte ein einfaches Konstruktionsverfahren für in diesem Sinne „Sender-robuste“ Schlüsselaustauschprotokolle entwickelt werden.

1.3.2 Bit Commitment

Für Mehrparteienprotokolle zur *sicheren Funktionsauswertung* (etwa allgemeine Konstruktionen wie Goldreich u. a. [73]) wird als Grundbaustein vielfach die Primitive „Bit Commitment“ verwendet. Dabei handelt es sich um einen Baustein, der informell folgende Protokollaufgabe erfüllt:

1. In der ersten Phase legt sich der *Committer* auf ein Bit b fest (ohne daß eine

andere Partei das Bit b erfährt).

2. In der zweiten Phase deckt der Committer b auf.

Entscheidend hierbei ist, daß in der zweiten Phase *nur* das b aus der ersten Phase aufgedeckt werden kann; man sagt, ein Commitment sei *bindend*. Andererseits kennt *vor* der zweiten Phase nur der Committer das Bit b – das Commitment ist also auch *verbergend*. Deshalb wird in größeren Protokollen Bit Commitment oft benutzt, um b von zwischen der ersten und zweiten Commitment-Phase durchgeführten Berechnungen zu „entkoppeln“.

Nun zeigen Canetti und Fischlin in [34] mit einem einfachen Argument, daß es kein simulierbar sicheres Protokoll für Bit Commitment geben kann. Diese Situation ist höchst unbefriedigend, da sehr wohl Protokolle für Bit Commitment existieren, welche sowohl bindend als auch verbergend sind. Als Abhilfe konnte in [34] unter der *Zusatzannahme* eines öffentlich verfügbaren und ideal gewählten *common reference strings* (CRS) ein simulierbar sicheres Protokoll für Bit Commitment angegeben werden. Jedoch ist dieses Protokoll sehr komplex und verliert jede intuitive Sicherheit (insbesondere die Eigenschaften „verbergend“ und „bindend“), sobald der CRS vom Angreifer beeinflußt werden kann.

Als realistischere Zusatzannahme wird deshalb in [83] von Hofheinz und Müller-Quade das schon erwähnte Random-Oracle-Modell motiviert. Hierbei wird eine idealisierte Hashfunktion (das „Random Oracle“) angenommen, die nicht teilweise ausgewertet werden kann und bei jedem Protokolldurchlauf neu als zufällige Funktion gewählt wird. Diese etablierte Annahme wurde schon für Sicherheitsbeweise von Public-Key-Kryptosystemen verwendet, siehe etwa Bellare und Rogaway [18, 19].

Weiter wurde in [83] eine allgemeine Konstruktion entwickelt, mit der klassisch sichere (d. h. bindende und verbergende) Commitment-Protokolle umgewandelt werden können in (im Random-Oracle-Modell) simulierbar sichere Commitment-Protokolle. Als Besonderheit sind diese umgewandelten Protokolle immer noch klassisch sicher, wenn man das Random-Oracle-Modell verläßt und das Random Oracle durch eine kryptographische Hashfunktion ersetzt. Durch diese – im allgemeinen nicht gegebene – „doppelte Absicherung“ konnten also Protokolle erarbeitet werden, die sowohl klassisch sicher als auch (im Random-Oracle-Modell) simulierbar sicher sind.

1.4 Modellierungen von Netzwerken

Es scheint sehr wünschenswert, in dem konkreten Modell [27] erzielte Ergebnisse auch auf andere Modelle (etwa das Mehrparteienmodell Pfitzmann und Waidner [117], Backes u. a. [11]) zu übertragen. Eine konsequente Weiterführung dieses Gedankens ist der Ansatz, von Modelldetails zu abstrahieren, um Ergebnisse in einer naheliegenden, gewissermaßen „modellunabhängigen“ Form darstellen zu können. Man könnte sich damit insbesondere erhoffen, Beweise frei von „lästigen“

Details wie der expliziten Berücksichtigung des Kontrollflusses und der Netzwerkmodellierung formulieren zu können, um den „intuitiven“ Beweisgedanken klarer hervortreten zu lassen.

Es kann allerdings in Hofheinz und Müller-Quade [82] (diese Arbeit wird in Kapitel 5 vorgestellt) ein starkes Indiz dafür gegeben werden, daß diese Hoffnung nicht berechtigt ist. Genauer zeigt [82], daß die Primitive „Oblivious Transfer“ *nicht vollständig* für allgemeine, simulierbar sicher zu realisierende Protokollaufgaben ist, sobald das zugrundegelegte Netzwerk *verläßlich* ist. Andererseits wurde in Canetti u. a. [39] bewiesen, daß Oblivious Transfer sehr wohl vollständig ist, wenn das Netzwerk als *asynchron* modelliert ist. (D. h. jede Protokollaufgabe, welche gewissen technischen Randbedingungen genügt, kann unter Zuhilfenahme von Oblivious-Transfer-Bausteinen asynchron realisiert werden.) Die Arbeit [82] zeigt also, daß sogar Vollständigkeitsaussagen für Primitiven vom Typ des verwendeten Netzwerks abhängen können.[7]

[7] Weitere Zusammenhänge zwischen verschiedenen Netzwerkmodellierungen wurden in Backes, Hofheinz, Müller-Quade und Unruh [8] in dem sehr vielseitigen Modell [117, 11] in ausführlicher Weise untersucht.

2 Untersuchung neuer Primitive

Wie schon besprochen sollen jetzt einige neuartige und experimentelle Wege der Public-Key-Kryptographie untersucht werden. Genauer soll das Public-Key-Kryptosystem *Polly Cracker*, ein *zopfgruppenbasiertes* Verfahren, sowie ein Public-Key-Kryptosystem basierend auf dem Wortproblem in *Grigorchuk-Gruppen* betrachtet werden. Die hier gegebenen Betrachtungen sollen keinen Überblick darstellen, sondern eine Detailanalyse ausgewählter Kandidaten für neue Primitive wiedergeben.

Andere diskutierte gruppenbasierte Systeme – wobei hier keinesfalls ein Anspruch auf Vollständigkeit erhoben werden soll – sind etwa das MST- (vgl. Magliveras u. a. [106], Bohli u. a. [25]) oder das MOR-Public-Key-Kryptosystem (vgl. Paeng u. a. [112, 111], Tobias [134]). Auch sei als eine weitere Klasse von polynombasierten Systemen noch die Familie der HFE-Systeme (vgl. Patarin [113], Patarin u. a. [114]) erwähnt, für die allerdings die Wahl einiger Parameter ebenfalls noch nicht restlos geklärt zu sein scheint, siehe dazu Kipnis und Shamir [95], Courtois [46], Faugère und Joux [60].

Um die zu besprechenden Verfahren konkret vorzustellen, wird hier zunächst der bislang nur informell besprochene Begriff eines *Public-Key-Kryptosystems* fixiert:

Definition 2.1. *Ein* Public-Key-Kryptosystem P *besteht aus drei PPT-Algorithmen*[1] KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$. *Für beliebige* $k \in \mathbb{N}$ *und* $pk, sk, m, c \in \{0,1\}^*$ *werden folgende* syntaktische *Bedingungen gestellt:*

- $\mathsf{KeyGen}(1^k) \in \{0,1\}^* \times \{0,1\}^*$,
- $\mathsf{Encrypt}(1^k, pk, m) \in \{0,1\}^*$,
- $\mathsf{Decrypt}(1^k, sk, c) \in \{0,1\}^* \cup \{\perp\}$.

Des weiteren wird verlangt, daß jeder mittels $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ *generierte öffentliche Schlüssel* pk *eindeutig eine* Klartextmenge $\mathcal{M}_{pk} \subseteq \{0,1\}^*$ *spezifiziert. Schließlich wird* Korrektheit *in dem Sinne gefordert, daß die Wahrscheinlichkeit, ein Schlüsselpaar* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ *zu generieren, so daß eine Nachricht* $m \in \mathcal{M}_{pk}$ *existiert, für die* $\mathsf{Decrypt}(1^k, sk, \mathsf{Encrypt}(1^k, pk, m)) \neq m$ möglich *ist, dabei als Funktion in* k *vernachlässigbar sein muß.*

Man beachte hier die unäre Codierung eines expliziten *Sicherheitsparameters* k; ein solcher wird auch für die Definition anderer kryptographischer Bausteine zur Anwendung kommen. Insbesondere dient er – oft im Zusammenhang mit dem gerade benutzten Begriff vernachlässigbarer Funktionen – als technisches Hilfsmittel

[1] vgl. Definition A.6 in Anhang A

für asymptotische Sicherheitsaussagen. Für die in diesem Kapitel noch anzusprechenden konkreten Kryptosysteme ist allerdings nicht immer ersichtlich, in welcher Weise ein Sicherheitsparameter etwa in Schlüsselgenerierung oder Verschlüsselung einfließt. Schon die ursprünglichen Spezifikationen der Systeme beschreiben keine diesbezügliche Skalierbarkeit. Dennoch sollen aus Konsistenzgründen auch diese Systeme als Public-Key-Kryptosysteme im Sinne von Definition 2.1 betrachtet werden, wobei natürlich offen gelassen werden muß, wie eine Skalierung in k konkret auszusehen hat.

Definition 2.1 erlaubt – analog zu der Definition aus Cramer und Shoup [50] – zwar seltene „schlechte“ Schlüsselpaare, jedoch wird für „gute“ Schlüsselpaare gefordert, daß Entschlüsselung eines Chiffrates *immer* den zugehörigen Klartext liefert. Hier sind Abschwächungen denkbar; etwa könnte man seltene und klartextunabhängige Fehlentschlüsselungen erlauben. Dies würde die später präsentierten Ergebnisse nicht beeinflussen. Allerdings genügen die im folgenden konkret betrachteten Kryptosysteme, sowie – mit Ausnahme des NTRU-Systems (siehe Hoffstein u. a. [80]) – alle etablierten Kryptosysteme schon der Formulierung aus Definition 2.1; insofern scheint eine Abschwächung nicht nötig.

2.1 Polly Cracker

Vorgeschlagen in Fellows und Koblitz [61], sollte *Polly Cracker* aller Skepsis von Barkee u. a. [15] zum Trotze als Beleg dafür dienen, daß das Problem des Lösens eines multivariaten Gleichungssystems als Grundlage für ein Public-Key-Kryptosystem dienen kann. Genauer konnte in den Arbeiten [61] und Koblitz [98] das Finden des geheimen Schlüssels allein aus dem öffentlichen Schlüssel für vorgeschlagene Varianten von Polly Cracker als äquivalent zu den Suchvarianten verschiedener NP-vollständiger Entscheidungsprobleme gezeigt werden; in einer Konstruktion beispielsweise impliziert solches Finden des geheimen Schlüssels das Finden einer Drei-Färbung eines durch den öffentlichen Schlüssel gegebenen Graphen.

Doch obwohl das Entscheidungsproblem etwa der Drei-Färbbarkeit NP-vollständig ist, garantiert das noch nicht die Schwierigkeit zufällig erzeugter *Instanzen*. Wünschenswert wäre hier noch mehr: eine Schwierigkeit der zugrundegelegten Instanz in „fast allen“ Fällen. Eine Diskussion dieser Problematik und Vorschläge zur Generierung „harter“ Probleminstanzen finden sich in Levy-dit-Vehel und Peret [103]. Im weiteren Verlauf soll sich aber auf die Sicherheit *einzelner Chiffrate* konzentriert werden. Es geht also nicht darum, den geheimen Schlüssel offenzulegen (und damit die Suchvariante eines NP-vollständigen Problems zu lösen), sondern es soll der Prozeß der Verschlüsselung in einem gegebenen Fall invertiert werden.

Nachdem hiermit das Ziel der weiteren Betrachtungen abgesteckt ist, folgt eine Beschreibung von Polly Cracker. Es sei angemerkt, daß ein Überblick über das System und die vorgestellten Angriffe auch in Hofheinz [81] zu finden ist. Dort nicht enthalten ist vor allem die Beschreibung eines *konkret* durchgeführten und in Hofheinz und Steinwandt [88] veröffentlichten Angriffs auf eine in [98] vor-

geschlagene *Instanz* von Polly Cracker.

2.1.1 Das System

Eine noch nicht vollständig spezifizierte Beschreibung des Systems vorausschickend wird hier, [98] folgend, Polly Cracker zunächst in einer allgemeinen Form beschrieben:

- $\mathsf{KeyGen}(1^k)$ wählt einen endlichen Körper $\mathbb{F}$ und eine endliche Familie $T = (t_i)_i$ von Variablen. Sodann wird ein Punkt $y \in \mathbb{F}^{|T|}$ und eine endliche Menge $Q = \{q_j\}$ von Polynomen $q_j \in \mathbb{F}[T]$ mit $q_j(y) = 0$ für alle j ermittelt. Ausgabe ist schließlich (pk, sk) mit $pk = (\mathbb{F}, T, Q)$ (assoziierte Klartextmenge ist $\mathcal{M}_{pk} = \mathbb{F}$) und $sk = (\mathbb{F}, T, y)$.

- $\mathsf{Encrypt}(1^k, pk, m)$ wählt – falls $m \in \mathbb{F}$ und pk von der Form $(\mathbb{F}, T, Q)$ mit $Q \subset \mathbb{F}[T]$ ist – Polynome $\{h_j\}_{q_j \in Q} \subset \mathbb{F}[T]$ und gibt das Chiffrat $c = m + \sum h_j q_j \in \mathbb{F}[T]$ aus. Der (uninteressante) Fall ungültiger Eingaben kann beispielsweise durch die leere Ausgabe behandelt werden.

- $\mathsf{Decrypt}(1^k, sk, c)$ gibt, wenn $c \in \mathbb{F}[T]$ und sk von der Form $(\mathbb{F}, T, y)$ mit $y \in \mathbb{F}^{|T|}$ ist, den Klartext $m = c(y) \in \mathbb{F}$ aus. Bei ungültigen Eingaben wird „$\perp$“ ausgegeben.

Für ein wie beschrieben erzeugtes Chiffrat $c \in \mathbb{F}[T]$ zu einem beliebigen Klartext $m \in \mathbb{F}$ folgt $c(y) = m + \sum h_j(y) \cdot q_j(y) = m + \sum h_j(y) \cdot 0 = m$. Das betrachtete Schema ist damit ein Public-Key-Kryptosystem gemäß Definition 2.1, sobald nur die informell spezifizierten Algorithmen KeyGen, Encrypt und Decrypt implementierbar durch PPT-Algorithmen sind. Erkennt man weiter, daß durch Auswerten von $c \in \mathbb{F}[T]$ am Punkt $y \in \mathbb{F}^{|T|}$ letztlich modulo des maximalen Ideals $\langle \{t_i - y_i\}_{t_i \in T} \rangle$ reduziert wurde, liegt eine Verallgemeinerung auf der Hand.

Statt des Punktes y könnte auch eine Gröbnerbasis G mit $\langle Q \rangle \subseteq \langle G \rangle \subset \mathbb{F}^{|T|}$ als geheimer Schlüssel dienen. Dechiffrierung fände dann durch Reduktion von c modulo G statt, und die Klartextmenge $\mathcal{M}_{pk}$ könnte als Repräsentantensystem von $\mathbb{F}[T]/\langle G \rangle$ – sofern G nur ein nicht-maximales echtes Ideal aufspannt – größer als $\mathbb{F}$ gewählt werden. Dieses Vorgehen ist in [98, Kapitel 5, Abschnitt 3.3] angesprochen, jedoch bislang nicht eingehender untersucht. Abgesehen vom alsbald zu besprechenden Chosen-Ciphertext-Angriff von Steinwandt und Geiselmann aus [131] hat eine solche Verallgemeinerung auch keinen Einfluß auf die vorgestellten Angriffe, und sei hier deshalb nur angedeutet.

Um vor der Präsentation der Angriffe auf Polly Cracker ein Beispiel für ein konkreter spezifiziertes Polly-Cracker-System zu sehen, betrachte man etwa folgende, [98, Kapitel 5, Abschnitt 3.3] entnommene Wahl von Q: Sei ein Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E gegeben; hierzu eine Drei-Färbung von G in Form einer Abbildung $\varphi : V \to \{1, 2, 3\}$, die ihrerseits $\varphi(u) \neq \varphi(v)$ für alle

$uv \in E$ genügt. Sei nun $\mathbb{F} = \mathbb{F}_2$, $T = \{t_{v,i} \mid v \in V,\ 1 \leq i \leq 3\}$ und $Q = Q_1 \cup Q_2 \cup Q_3$, wobei

$$\begin{aligned} Q_1 &= \{t_{v,1} + t_{v,2} + t_{v,3} + 1 \mid v \in V\}, \\ Q_2 &= \{t_{v,i} t_{v,j} \mid v \in V,\ 1 \leq i < j \leq 3\}, \\ Q_3 &= \{t_{u,i} t_{v,i} \mid uv \in E,\ 1 \leq i \leq 3\}. \end{aligned}$$

Jede Nullstelle $y \in \mathbb{F}_2^{|T|}$ *aller* $q \in Q$ kann als Drei-Färbung von G aufgefaßt werden: Ein Knoten $v \in V$ hat genau dann die Farbe i, wenn $t_{v,i}(y) = 1$ ist. Da y Nullstelle aller $q \in Q_1 \cup Q_2$ ist, liegt hiermit eindeutig und widerspruchsfrei eine Färbung von G fest; $q(y) = 0$ für alle $q \in Q_3$ garantiert weiter, daß je zwei benachbarte Knoten von G von unterschiedlicher Farbe sind. Aber auch umgekehrt gibt jede Drei-Färbung von G – als Belegung der $t_{v,i}$ aufgefaßt – Anlaß zu einer Nullstelle aller $q \in Q$; insgesamt ist damit im zugehörigen Polly-Cracker-System das Finden eines (nicht notwendig eindeutigen!) geheimen Schlüssels allein unter Kenntnis des öffentlichen Schlüssels äquivalent zum Finden einer Drei-Färbung von G.

So verlockend das klingt, kann eine solche Äquivalenz selbst bei angenommener Härte der Instanz des zugrundeliegenden Suchproblems die Sicherheit *einzelner* Chiffrate nicht immer gewährleisten; eine Diskussion folgt. Zuvor jedoch soll ein verblüffend einfaches und gleichsam wirkungsvolles Vorgehen beschrieben werden, das zur Offenlegung selbst eines in diesem Sinne (also *nur* unter Kenntnis des öffentlichen Schlüssels) „schwer“ zu findenden geheimen Schlüssels dient.

2.1.2 Ein Chosen-Ciphertext-Angriff

Als eindrucksvoller Beleg für die Kraft eines Chosen-Ciphertext-Angriffs kann im Falle von Polly Cracker der Angriff aus [131] dienen. Es sei hier daran erinnert, daß bei einem Chosen-Ciphertext-Angriff dem Angreifer die Möglichkeit offensteht, Chiffrate seiner Wahl entschlüsseln zu lassen – je nach Angriffsziel allerdings mit leichten Einschränkungen, was die Wahl dieser Chiffrate angeht.[2] Sei also die Instanz eines Polly-Cracker-Systems in Form eines öffentlichen Schlüssels $(\mathbb{F}, T, Q)$ gegeben. Generiert ein Angreifer nun ein Chiffrat $m = t$ für $t \in T$ (etwa mit $c = t + \sum h_j q_j$), so liefert dieses – obwohl *nicht* „legitim“ generiert – beim Entschlüsseln durch das Decryption-Orakel den Wert $t(y) \in \mathbb{F}$. Iterieren über alle $t \in T$ legt damit für den Angreifer ganz $y \in \mathbb{F}^{|T|}$ offen; ein kompletter Bruch des Systems also, durch einen überaus „sparsamen“ Angriff noch dazu.

Die Frage liegt nahe, wie dieser Angriff vermieden werden kann. Eine Erkennung wie beschrieben „illegitim“ generierter Chiffrate *ohne* Änderung des Encrypt-Algorithmus scheint nicht offensichtlich: Gegeben ein $c \in \mathbb{F}[T]$ muß hierbei entschieden werden, ob $c_Q \in \mathbb{F}$, wobei c_Q die Reduktion von c modulo einer *Gröbnerbasis* von Q meint. Ein einfacherer Weg zur Vermeidung von Chosen-Ciphertext-Angriffen

[2] Die formale Definition eines Chosen-Ciphertext-Angriff steht hier noch immer aus; sie wird erst später von Bedeutung sein.

ist eine geschickte „Authentifizierung“ der Klartexte. Beispielsweise wurden in den Arbeiten Naor und Yung [110], Rackoff und Simon [121] nichtinteraktive Zero-Knowledge-Beweise verwendet, um zusätzlich zum Chiffrat einen Beweis der Kenntnis des Klartextes zu generieren. Wesentlich effizientere Mechanismen sind im Random-Oracle-Modell bekannt, im Falle von Polly Cracker bietet sich etwa die allgemeine Konstruktion aus Fujisaki und Okamoto [62] an. Andererseits verlangen alle diese Verfahren ein schon – je nach Konstruktion in unterschiedlichem Maße – sicheres Ausgangssystem. Deshalb soll sogleich die Sicherheit von Polly Cracker gegen schwächere Angriffe untersucht werden.

2.1.3 Ein Angriff mittels Linearer Algebra

Die nachfolgend untersuchten Angriffe auf Polly Cracker sind Ciphertext-Only-Angriffe; gegeben sind also nur ein öffentlicher Schlüssel $(\mathbb{F}, T, Q)$ und ein Chiffrat $c \in \mathbb{F}[T]$. Ziel eines Angriffs soll in diesem Falle sein, an Information über den zu c gehörigen Klartext zu gelangen. Die folgende Notation ist zur Beschreibung dieser Angriffe nützlich: Für ein Polynom $f \in \mathbb{F}[T]$ bezeichne $\mathbf{M}(f)$ die Menge der in f enthaltenen *Monome*, also die Menge der T-Potenzprodukte, die in f mit nichtverschwindendem Koeffizienten vorkommen. Hingegen bezeichne $\mathbf{T}(f)$ die Menge der *Terme* von f, also die Menge der in f enthaltenen Monome *mit* ihren jeweiligen Koeffizienten. Beispielsweise für $f = 3t_1t_2 - 5$ ist damit $\mathbf{M}(f) = \{t_1t_2, 1\}$ und $\mathbf{T}(f) = \{3t_1t_2, -5\}$. Für eine Menge $F \subseteq \mathbb{F}[T]$ von Polynomen sei weiter $\mathbf{M}(F) = \bigcup_{f\in F} \mathbf{M}(f)$ und $\mathbf{T}(F) = \bigcup_{f\in F} \mathbf{T}(f)$.

Zur Einführung und für die spätere Einordnung des Angriffs aus [81] sei nun ein bekannter Angriff auf Polly Cracker vorgestellt. Erwähnt in [15] und ausführlich behandelt in [98, Kapitel 5, Abschnitt 6], wird bei diesem Angriff zunächst $c = m + \sum h_j q_j$ mit noch unbekannten $m \in \mathbb{F}$ und $h_j \in \mathbb{F}[T]$ angesetzt. Angenommen, es sei eine Menge S von Monomen mit $S \supseteq \mathbf{M}(\{h_j\})$ bekannt. Dann können die h_j ihrerseits als Summen dieser Monome, jeweils versehen mit unbekannten Koeffizienten, angesetzt werden, und es ergibt sich die Verfeinerung

$$c = m + \sum_{q_j \in Q} \left(\sum_{s \in S} c_{j,s} s \right) q_j$$

mit unbekannten Koeffizienten $c_{j,s} \in \mathbb{F}$. Ausmultiplizieren, Aufsummieren der Koeffizienten und ein Koeffizientenvergleich mit den Koeffizienten im (bekannten) Chiffrat c liefert schließlich lineare Gleichungen über $\mathbb{F}$ für die $c_{j,s}$ *und* m. *Jede* Lösung dieses linearen Gleichungssystems bestimmt in eindeutiger Weise m, da die q_j eine gemeinsame Nullstelle haben und somit $1 \notin \langle Q \rangle$ ist. Umgekehrt existiert nach Voraussetzung eine Lösung dieses Gleichungssystems, und somit ist der Angriff immer erfolgreich, sobald nur die Menge S – und damit verbunden das entstandene Gleichungssystem – eine „handhabbare“ Größe hat.

Zu einer Menge S „handhabbarer“ Größe kann man zum Beispiel gelangen, wenn man annimmt, daß zu jedem $q_j \in Q$ und jedem Monom $s \in \mathbf{M}(h_j)$ ein Monom

$s_c \in \mathbf{M}(c)$ existiert mit $s_c \in \mathbf{M}(s \cdot q_j)$ – mit anderen Worten, wenn *nicht alle* „Konsequenzen“ der Teilmultiplikation $s \cdot q_j$ in c annulliert wurden. Dann ist

$$s \in \{s_c/s_q \mid s_c \in \mathbf{M}(c),\, s_q \in \mathbf{M}(Q),\, s_q | s_c\}$$

und damit die Konstruktion einer „kleinen“ Menge S klar. (In [98, Kapitel 5, Abschnitt 6] ist nachzulesen, daß dieser Vorschlag zur Konstruktion von S einer persönlichen Korrespondenz mit H. W. Lenstra entstammt.) Die gerade beschriebene Technik läßt sich durch mehr oder weniger offensichtliche Vorsichtsmaßnahmen bei der Wahl der h_i beim Verschlüsseln „in die Irre“ führen. Dazu ist es lediglich erforderlich, wenigstens ein Monom s in ein h_j einzubauen, dessen „Konsequenzen“ $\mathbf{M}(s \cdot q_j)$ in c nicht auftreten. Ein Weg, dieses zu bewerkstelligen, wurde in [98, Kapitel 5, Abschnitt 7] vorgeschlagen, ein weiterer in [103]; der erste dieser Vorschläge soll sogleich im Lichte einer weiteren Attacke betrachtet werden.

2.1.4 Eine „differentielle“ Attacke

In [81] vorgeschlagen und in [88] praktisch demonstriert, stellt auch dieser Angriff einen Ciphertext-Only-Angriff dar. Dabei wird von einigen heuristischen Annahmen ausgegangen, die im folgenden noch erläutert werden sollen. Auch hier bietet sich eine handliche Notation an: Für ein Polynom $f \in \mathbb{F}[T]$ ist die Menge der Termquotienten $\mathbf{D}(f) \subseteq \mathbb{F}[T \cup T^{-1}]$ (mit $T^{-1} = \{t^{-1}\}_{t \in T}$) gegeben durch $\mathbf{D}(f) = \{t_1/t_2 \mid t_1, t_2 \in \mathbf{T}(f)\}$. Man beachte, daß hier *nicht* $t_2 | t_1$ gefordert wird und damit auch Terme mit negativen t-Exponenten in $\mathbf{D}(f)$ auftreten können. In gewisser Weise gibt $\mathbf{D}(f)$ die „Struktur“ von f wieder. Bemerkenswert dabei ist das Verhalten dieser Struktur unter Addition und Multiplikation der zugehörigen Polynome. Direkt aus der Definition von $\mathbf{D}(\cdot)$ folgt $\mathbf{D}(t \cdot f) = \mathbf{D}(f)$ für beliebige Terme $t \neq 0$ und Polynome $f \in \mathbb{F}[T]$. Gleichsam zeigt sich $\mathbf{D}(f + g) \supseteq \mathbf{D}(f) \cup \mathbf{D}(g)$ für $f, g \in \mathbb{F}[T]$ mit $\mathbf{M}(f) \cup \mathbf{M}(g) = \emptyset$; hier lassen sich auch in naheliegender Weise schwächere Aussagen unter schwächeren Bedingungen formulieren.

Sei also – zurückkehrend zu Polly Cracker – wie weiter oben ein öffentlicher Schlüssel $(\mathbb{F}, T, Q)$ und ein Chiffrat $c \in \mathbb{F}[T]$ gegeben. Man betrachte nun die Mengen „charakteristischer Quotienten“ $D_j = \mathbf{D}(q_j) \setminus \bigcup_{\ell \neq j} \mathbf{D}(q_\ell)$ der jeweils *nur* in $\mathbf{D}(q_j)$ auftretenden Quotienten. Angenommen, alle diese Mengen D_j sind nichtleer, dann kann von einer *eindeutigen* Struktur D_j von q_j gesprochen werden. Die gemachten Bemerkungen zur Abbildung $\mathbf{D}(\cdot)$ lassen nun die – nicht zwingende! – Vermutung zu, daß auch $D_j \cap \mathbf{D}(c)$ nichtleer für wenigstens ein j ist. Intuitiv bedeutet dies, daß sich Teile der eindeutigen Struktur eines q_j in c wiederfinden.

Etwa aus dem größten dieser Mengenschnitte kann dann eine Schätzung für einen Term aus dem zugehörigen h_j abgeleitet werden: Ist $t_1/t_2 = u_1/u_2 \in D_j \cap \mathbf{D}(c)$ mit $t_1, t_2 \in \mathbf{T}(q_j)$ und $u_1, u_2 \in \mathbf{T}(c)$, so daß $v = u_1/t_1 = u_2/t_2$ ein Term ist, liegt der Verdacht $v \in \mathbf{T}(h_j)$ nahe. Das Chiffrat c kann dann durch Subtraktion von $v \cdot q_j$ „vereinfacht“ und die Methode iteriert werden. Hiermit ist auch eine Überprüfung der Schätzung v möglich: Eine *Erhöhung* der Anzahl der Terme des Chiffrats

durch Subtraktion von $v \cdot q_j$ deutet auf eine falsche Schätzung hin. Insgesamt kann mit dieser Technik – unter heuristischen Annahmen – der Prozeß der Chiffrierung schrittweise rückgängig gemacht werden, bis sich schließlich (ein eindeutiges) $c = m \in \mathbb{F}$ findet. Variationen sind möglich: Beispielsweise könnte man anstelle von Termquotienten auch Quotienten von Monomen betrachten; auch könnte im Verlauf der Attacke überprüft werden, ob zwischenzeitlich ein Angriff mittels Linearer Algebra möglich wurde.

Entscheidend ist jedoch die Frage der praktischen Einsetzbarkeit: Sind die genannten heuristischen Annahmen bei vorgeschlagenen – gegebenenfalls im Hinblick auf Angriffe mittels Linearer Algebra entworfenen – Systemen gegeben? Eine in dieser Hinsicht vom Standpunkt eines Kryptoanalytikers positive Teilantwort konnte in [88] gegeben werden. Ausgangspunkt ist hier eine in [98, Kapitel 5, Abschnitt 7] angeregte Instanz von Polly Cracker, im Falle derer das Finden des geheimen Schlüssels allein aus dem öffentlichen ein *Graph Perfect Code* genanntes kombinatorisches Suchproblem löst. Ohne auf Details einzugehen sei noch die dabei verwendete Methode zur Verschlüsselung angesprochen: In einem ersten Schritt wird ein vorläufiges Chiffrat $c_1 \in \mathbb{F}[T]$ konstruiert, das in mehreren Iterationen zu einem endgültigen Chiffrat c_d umstrukturiert wird, wobei durch die letzten dieser Iterationen gewährleistet werden soll, daß die Voraussetzungen eines Angriffs mittels Linearer Algebra nicht erfüllt sind.

Implementierung der beschriebenen „differentiellen“ Attacke zeigte jedoch, daß die konkret vorgeschlagenen Parameter immer noch die Enthüllung des Klartextes m erlauben. Je nach Parameterwahl konnte bei Verwendung einer Implementierung in C und eines mit 1,33 GHz getakteten PCs die Attacke in optimierter Form den Klartext nach wenigen Sekunden liefern; bei Parametern, die zu größeren Chiffraten führten, dauerte die Attacke bis zu einigen Stunden. Tatsächlich war es unvermeidbar, die vorgeschlagenen Systemparameter zu reduzieren, um eine handhabbare Chiffratgröße (weniger als einige hundert Megabyte) zu gewährleisten. Bemerkenswerterweise schlug der Angriff fast immer genau dann fehl, wenn $m = 1 \in \mathbb{F}_2$ chiffriert wurde. Eine Erklärung hierfür steht noch aus.

2.1.5 Weitere Entwicklungen

Zusammenfassend kann gesagt werden, daß augenblicklich kein konkretes Polly-Cracker-System vorgeschlagen zu sein scheint, auf das nicht einer der vorgestellten Angriffe anwendbar ist. Beispielsweise konnte in der Arbeit [103], welche zunächst ein Polly-Cracker-System vorschlägt, anschließend gezeigt werden, daß eine Variante des hier vorgestellten differentiellen Angriffs anwendbar auf das vorgeschlagene System ist.

Nicht unerwähnt bleiben sollten in diesem Zusammenhang auch die Public-Key-Kryptosysteme von Grant u. a. [79] und Banks u. a. [12], welche als Variante eines Polly-Cracker-Systems aufgefaßt werden können. Allerdings konnten bezüglich der vorgeschlagenen Parameter konkrete Angriffe angegeben werden, vgl. Bao u. a. [14], Bao u. a. [13].

Schließlich wurde in Ly [105] eine tiefgreifende strukturelle Änderung des ursprünglichen Polly-Cracker-Systems vorgeschlagen (die dann folgerichtig auch den Namen „Polly Two“ trägt), für welche eine mögliche Anwendbarkeit des hier vorgestellten Angriffs noch nicht geklärt ist.

2.2 Ein Grigorchuk-Gruppen-basiertes System

Als weiteres Primitiv soll nun ein auf der Härte des Wortproblems in Grigorchuk-Gruppen basierendes Kryptosystem vorgestellt werden. Eine anschließende Analyse des Systems führt dabei zu einem vollständigen strukturellen Bruch. Bemerkenswert dabei ist vor allem, daß der Angriff allein aus dem öffentlichen Schlüssel einen passenden geheimen Schlüssel ableitet, der zwar alle mit dem öffentlichen Schlüssel generierten Chiffrate entschlüsselt, aber dennoch (im allgemeinen) *nicht* der „legitime“ geheime Schlüssel ist. In der Tat legt der öffentliche Schlüssel den geheimen nicht eindeutig fest, und diese Unmöglichkeit der Bestimmung des „legitimen“ geheimen Schlüssels allein aus dem öffentlichen wurde von Garzon und Zalcstein in [65] sogar als Argument für die Sicherheit des dort beschriebenen Systems gewertet. Doch vor einer Behandlung des Systems und eines Angriffs darauf sollen zunächst einige im Verlauf der weiteren Diskussion wichtige Eigenschaften von Grigorchuk-Gruppen vorgestellt werden.

2.2.1 Grigorchuk-Gruppen

Sei eine unendliche ternäre Folge $\chi = \{\chi_i\}_{i\in\mathbb{N}}$ mit $\chi_i \in \{0,1,2\}$ für alle i gegeben. Dann wird die mit χ assoziierte (unendliche) *Grigorchuk-Gruppe* G_χ generiert von $\{a, b_\chi, c_\chi, d_\chi\}$, wobei nur a unabhängig von χ ist. Es bezeichne hier und im folgenden $D = \{a, b_\chi, c_\chi, d_\chi\}$ das Alphabet, welches durch diese Generatoren definiert wird. Weiter sei nur angedeutet, daß die Struktur von G_χ durch ihre Interpretation als Permutationsgruppe der Pfade durch den unendlichen Binärbaum gegeben ist; Details sind für diese Diskussion nicht von Bedeutung.

Folgende Aussage (vgl. [65]) zeigt die „Unhandlichkeit“ von G_χ bei geeigneter Wahl von χ: Treten wenigstens zwei Zahlen aus $\{0,1,2\}$ in χ unendlich oft auf, so ist G_χ *nicht* endlich präsentierbar und besitzt subexponentielles Wachstum. Andererseits erschließt sich durch das folgende Ergebnis (ebenfalls aus [65]) eine effiziente Lösung des Wortproblems in G_χ *unter Kenntnis von* χ: Es existiert ein deterministischer, polynomial laufzeitbeschränkter Algorithmus, der bei Eingabe eines beliebigen Wortes $\mathbf{w} \in D^*$ und der $\lceil \log_2 |\mathbf{w}| \rceil$ ersten Glieder von χ entscheidet, ob das durch $\mathbf{w}$ gegebene Gruppenelement $w \in G_\chi$ das neutrale Element in G_χ ist.

2.2.2 Das System

Ausgehend von den gerade wiedergegebenen Resultaten wurde in [65] ein Public-Key-Kryptosystem vorgeschlagen, dessen Sicherheit dem Wortproblem in ge-

eignet gewählten Grigorchuk-Gruppen zugrundegelegt werden sollte. Dieses System sei hier in gewohnter – allerdings in Ermangelung konkreter Parameter abstrakter – Form beschrieben.

- $\mathsf{KeyGen}(1^k)$ wählt zunächst eine ternäre Folge $\chi = \{\chi_i\}_{i\in\mathbb{N}}$ mit $\chi_i \in \{0,1,2\}$ für alle i, so daß insbesondere wenigstens zwei dieser möglichen drei Werte für die χ_i unendlich oft in χ auftreten. Teil des geheimen Schlüssels sk ist dann mit M_{G_χ} eine Turing-Maschine (konstruiert gemäß [65]), die das Wortproblem in G_χ löst, also bei Eingabe eines Wortes $\mathbf{w} \in D^*$ entscheidet, ob das zugehörige Element $w \in G_\chi$ gleich der Identität ist.

  Anschließend wird eine endliche Menge R von Worten $\mathbf{r}$ über D bestimmt, die – als Elemente $r \in G_\chi$ aufgefaßt – gleich der Identität sind; außerdem zwei Worte $\mathbf{w}_0, \mathbf{w}_1 \in D^*$, so daß die zugehörigen $w_0, w_1 \in G_\chi$ *ungleiche* Elemente in G_χ darstellen. Ausgabe von KeyGen ist schließlich (pk, sk) mit $pk = (R, \mathbf{w}_0, \mathbf{w}_1)$ und, wie besprochen, $sk = (M_{G_\chi}, \mathbf{w}_0, \mathbf{w}_1)$. Die Klartextmenge $\mathcal{M}_{pk}$ ist, unabhängig von k, immer $\{0,1\}$.

- $\mathsf{Encrypt}(1^k, pk, m)$ startet – sofern $m \in \{0,1\}$ und pk von der beschriebenen Form $(R, \mathbf{w}_0, \mathbf{w}_1)$ ist – mit einem vorläufigen Chiffrat $\mathbf{c} = \mathbf{w}_m$ und ändert dieses Wort über D sukzessive ab durch Einfügen und Löschen von Worten aus R. (Leider sind hierfür in [65] keine konkreteren Richtlinien vorgeschlagen.) Ausgabe ist dann das solchermaßen erhaltene endgültige Wort $\mathbf{c} \in D^*$ oder, im Falle ungültiger Eingaben, das leere Wort.

- $\mathsf{Decrypt}(1^k, sk, \mathbf{c})$ gibt $m = 0$ aus, falls $\mathbf{c} \in D^*$ und sk die Beschreibung einer Turing-Maschine M_{G_χ} ist, so daß M_{G_χ} das Wort $\mathbf{c}^{-1}\mathbf{w}_0$ als Identität in G_χ findet. Analog wird $m = 1$ ausgegeben, falls M_{G_χ} das Wort $\mathbf{c}^{-1}\mathbf{w}_1$ als Identität in G_χ identifiziert. In allen anderen Fällen, insbesondere bei ungültigen Eingaben, wird „$\perp$“ ausgegeben.

Sobald diese Algorithmenbeschreibungen auch durch PPT-Algorithmen implementierbar sind, ist aufgrund der Assoziativität der Gruppe G_χ klar, daß dieses System ein Public-Key-Kryptosystem nach Definition 2.1 ist.

2.2.3 Ein Angriff

In diesem Abschnitt soll der Angriff aus González Vasco, Hofheinz, Martínez und Steinwandt [78] auf das soeben beschriebene System vorgestellt werden. Sei dazu ein öffentlicher Schlüssel $(R, \mathbf{w}_0, \mathbf{w}_1)$ gegeben. Sei weiter $N = \max\{|\mathbf{r}| \mid \mathbf{r} \in R \cup \{\mathbf{w}_0^{-1}\mathbf{w}_1\}\}$. Dann ist aufgrund der vorangestellten Bemerkungen über Grigorchuk-Gruppen ein Test auf Identität in G_χ für alle Worte in R möglich unter Kenntnis *lediglich* der ersten $\lceil \log_2 N \rceil$ Glieder von χ. Gleichwohl ermöglichen selbige Voraussetzungen einen Test auf $w_0 \neq w_1$ mit durch die Worte $\mathbf{w}_0, \mathbf{w}_1 \in D^*$ gegebenen Elementen $w_0, w_1 \in G_\chi$.

Anders ausgedrückt: Die „hinteren“ Glieder von χ sind für einen öffentlichen Schlüssel gar nicht von Belang, in diesem Sinne ist also der öffentliche Schlüssel *unabhängig* von diesen Gliedern. Folglich ist ein Dechiffrieren beliebiger Klartexte auch mittels einer Folge χ' möglich, die nur in den ersten $\lceil \log_2 N \rceil$ Gliedern mit χ übereinstimmt – der gegebene öffentliche Schlüssel könnte ja auch bei vorangehender Wahl der Gruppe $G_{\chi'}$ (statt G_χ) erzeugt worden sein.

Über alle $3^{\lceil \log_2 N \rceil} \approx N^{\log_2 3}$ möglichen Belegungen der ersten $\lceil \log_2 N \rceil$ Glieder eines solchen χ' kann aber effizient eine vollständige Suche durchgeführt werden; für eine „richtige“ Belegung muß nur gelten, daß hiermit $\mathbf{w}_0 \neq \mathbf{w}_1$ und alle $\mathbf{r} \in R$ als Identität in $G_{\chi'}$ erkannt werden. (Obwohl die weiteren Glieder einer in ihren Anfangsgliedern geratenen Folge χ' dafür nicht benötigt werden, kann man sie sich beliebig, aber fest denken.)

Insgesamt kann also mit vergleichsweise geringem Aufwand ein zum „tatsächlichen“ geheimen Schlüssel äquivalenter Schlüssel gewonnen werden, und zwar ausschließlich unter Nutzung eines vorgegebenen öffentlichen Schlüssels. Es ist klar, daß das behandelte System so keine „vernünftige“ Sicherheitsdefinition erfüllen kann, geschweige denn einen der in der Einleitung umrissenen asymptotischen Sicherheitsbegriffe.

2.3 Zopfgruppenbasierte Systeme

Schließlich soll noch eine Klasse von Systemen angesprochen werden, der gerade in den letzten Jahren eine größere Aufmerksamkeit zugekommen ist. Es handelt sich dabei um Systeme, die auf der Schwierigkeit des Konjugationsproblems in Zopfgruppen basieren. Zopfgruppen selbst sind mathematisch wohlbekannt und gut untersucht (siehe etwa Artin [4], Garside [64], Birman [20]); deshalb steht in dieser Arbeit die Untersuchung der kryptographischen Eignung insbesondere des Konjugationsproblems in Zopfgruppen im Vordergrund. Doch zunächst sollen einige im folgenden benutzte Eigenschaften von Zopfgruppen vorgestellt werden.

2.3.1 Zopfgruppen

Für $n \in \mathbb{N}$ ist die *Zopfgruppe* B_n die von den Elementen $\sigma_1, \ldots, \sigma_{n-1}$ erzeugte Gruppe, in der die Relationen

$$\begin{array}{rcll} \sigma_i\sigma_j\sigma_i & = & \sigma_j\sigma_i\sigma_j & \text{für } |i-j| = 1, \\ \sigma_i\sigma_j & = & \sigma_j\sigma_i & \text{für } |i-j| > 1 \end{array} \tag{2.1}$$

gelten. Dabei werden die σ_i *Artin-Generatoren* und Elemente aus B_n *Zöpfe* genannt. Ein Zopf heißt *positiv*, wenn er als Produkt von Artin-Generatoren geschrieben werden kann; formal wird auch die Identität $\varepsilon \in B_n$ als positiv betrachtet. Es kann gezeigt werden (siehe etwa [64]), daß die B_n-Untermenge aller positiven Worte bezüglich der B_n-Gruppenverknüpfung einen Monoid B_n^+ bildet; weiter folgt direkt aus (2.1), daß die *Länge* $|w|$ eines positiven Wortes $w \in B_n^+$ als Länge einer

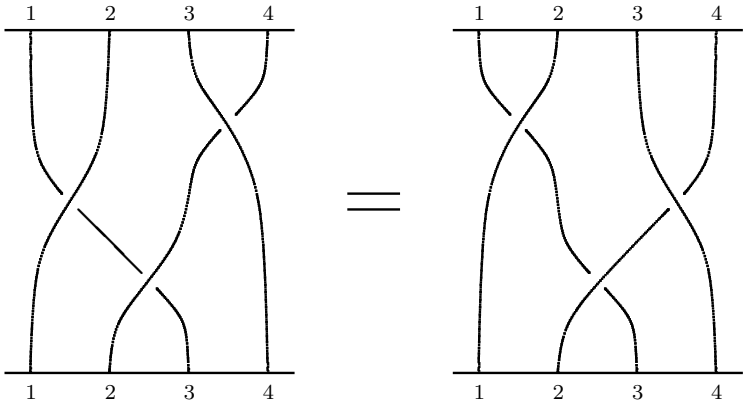


Abbildung 2.1: Zwei Darstellungen des Zopfes $\sigma_3^{-1}\sigma_1\sigma_2 = \sigma_1\sigma_3^{-1}\sigma_2 \in B_4$

Zerlegung in Artin-Generatoren wohldefiniert ist. Weiter existiert ein eindeutig bestimmter Homomorphismus $\pi : B_n \to S_n$ in die symmetrische Gruppe S_n, der σ_i auf die Transposition $(i, i+1)$ abbildet.

Insbesondere dieser Sachverhalt, aber auch schon die Relationen (2.1) legen folgende Anschauung nahe: Ein Zopf $w \in B_n$ kann interpretiert werden als Verdrillung von n Strängen, exemplarisch dargestellt in Abbildung 2.1. Die Gruppenverknüpfung in der Gruppe B_n kann dann als die Operation „Hintereinanderausführung von Verdrillungen“ interpretiert werden. Die Relationen (2.1) induzieren dabei eine Äquivalenz von genau solchen Verdrillungen, die bei befestigten Strangenden allein durch „Umsortieren“ der Überkreuzungen von Strängen ineinander überführt werden können.

Der spezielle, *fundamental braid* genannte Zopf $\Delta \in B_n$ ist induktiv durch $\Delta_1 := \sigma_1$, $\Delta_i := \sigma_1 \cdots \sigma_i \Delta_{i-1}$ für $1 < i < n$ und schließlich $\Delta := \Delta_{n-1}$ definiert. Die Schreibweise $v \leq w$ für Zöpfe $v, w \in B_n$ drückt aus, daß positive Zöpfe $\alpha, \beta \in B_n^+$ existieren, für die $w = \alpha v \beta$ ist. Mit dieser Schreibweise wird jeder Zopf $w \in B_n^+$ mit $\varepsilon \leq w \leq \Delta$ auch *kanonischer Faktor* genannt. Da die oben besprochene Abbildung π, eingeschränkt auf die Menge aller kanonischen Faktoren, bijektiv ist, können kanonische Faktoren also mit Permutationen auf n Stellen identifiziert werden.

Eine Zerlegung $\gamma = \alpha\beta$ von $\gamma \in B_n^+$ in einen kanonischen Faktor α und positives β heißt *linksgewichtet*, wenn für jede andere solche Zerlegung $\gamma = \alpha'\beta'$ schon $|\alpha'| \leq |\alpha|$ ist, d. h. wenn die Länge von α maximal ist. Jeder Zopf $w \in B_n$ kann in einer eindeutigen, Δ-*Normalform* genannten Weise als Wort in Artin-Generatoren bzw. deren Inversen geschrieben werden. Genauer gibt es für w eindeutig bestimmte $r \in \mathbb{Z}$ und $W_1, \ldots, W_s \in B_n^+$ mit $w = \Delta^r W_1 \cdots W_s$, so daß die W_i von ε verschiedene kanonische Faktoren sind, und weiter alle Zerlegungen $W_i W_{i+1}$ linksgewichtet sind. Man nennt $\inf w := r$ das *Infimum* und $\sup w := r + s$ das *Supremum* von w. Die Δ-Normalform eines Wortes kann effizient berechnet und rechnerintern dargestellt werden – vgl. Cha u. a. [44] für optimierte Verfahren hierzu.

Zwei Zöpfe $v, w \in B_n$ heißen *konjugiert*, wenn ein $x \in B_n$ mit $w = x^{-1}vx$ existiert. Das Konjugationsproblem in Zopfgruppen bezeichnet die Aufgabe, für konjugierte $v, w \in B_n$ ein solches $x \in B_n$ mit $w = x^{-1}vx$ zu finden.[3] Ein Zugang zur Lösung des Konjugationsproblems führt über die Berechnung der (endlichen) *super summit sets* $\mathcal{S}(v)$ bzw. $\mathcal{S}(w)$ – siehe hierzu etwa Elrifai und Morton [59]. Diese Mengen bestehen dabei aus allen zu v bzw. w konjugierten Zöpfen mit minimalem Supremum und maximalem Infimum. Leider ist die Mächtigkeit solcher Mengen nicht geklärt, und eine explizite Berechnung von $\mathcal{S}(z)$ scheint für hinreichend lange Wörter $z \in B_n$ nicht praktikabel, vgl. Ko u. a. [97].

2.3.2 Ein Kryptosystem

Exemplarisch soll hier skizziert werden, wie aus der Härte des Konjugationsproblems ein Public-Key-Kryptosystem abgeleitet werden kann. Hierzu wird das System aus [97] vorgestellt, wobei $\{H_k\}_{k\in\mathbb{N}}$ eine Familie von Hashfunktionen bezeichne.[4] Es soll dabei auch möglich sein, Zöpfe eindeutig als Bitstrings zu interpretieren und insbesondere als Argumente eines H_k einzusetzen. (Die Eindeutigkeitsforderung kann beispielsweise durch Berechnung der Δ-Normalform erfüllt werden.) Weiter sei noch folgende Notation vereinbart: Für $1 < \ell, r < n$ bezeichne LB_ℓ bzw. RB_r die von $\sigma_1, \ldots, \sigma_{\ell-1}$ bzw. $\sigma_{n-r+1}, \ldots, \sigma_{n-1}$ generierte Untergruppe von B_n. Man beachte, daß damit ein Element $v \in LB_\ell$ mit beliebigen Elementen $w \in RB_r$ kommutiert, sofern nur $\ell + k \leq n$ ist.

- $\mathsf{KeyGen}(1^k)$ wählt zunächst zwei Zahlen $\ell, r \in \mathbb{N}$ und setzt $n := \ell + r$. Anschließend werden Zöpfe $a \in LB_\ell$ und $x \in B_n$ gewählt und $y := a^{-1}xa$ berechnet. Damit ist $pk = (n, \ell, r, x, y)$, $sk = (n, a)$ und die Ausgabe von KeyGen wie gewohnt von der Form (pk, sk). Die assoziierte Klartextmenge $\mathcal{M}_{pk}$ ist dabei $\{0,1\}^k$.

- $\mathsf{Encrypt}(1^k, pk, m)$ wählt – für $m \in \{0,1\}^k$ und wie beschriebenes $pk = (n, \ell, r, x, y)$ – einen Zopf $b \in RB_r$ und berechnet $d := b^{-1}xb$ und $e := H_k(b^{-1}yb) \oplus m$. Ausgabe ist dann das Chiffrat $c := (d, e)$ oder, im Falle ungültiger Eingaben, die leere Ausgabe.

- $\mathsf{Decrypt}(1^k, sk, c)$ gibt $m \in \{0,1\}^k \cup \{\perp\}$ aus, wobei $m := H_k(a^{-1}da) \oplus e$, falls sk von der Form $(n, a) \in \mathbb{N} \times B_n$ und c von der Form $(d, e) \in B_n \times \{0,1\}^k$ ist. Andernfalls wird $m := \perp$ gesetzt.

Das beschriebene System entschlüsselt immer korrekt, da (in der oben benutzten Notation) $z := a^{-1}da = a^{-1}b^{-1}xba = b^{-1}a^{-1}xab = b^{-1}yb$ ist. Andererseits genügt

[3] Zwar existiert auch eine Entscheidungsvariante des Konjugationsproblems, jedoch ist für die weitere Diskussion nur die gegebene Formulierung als Suchproblem von Belang.

[4] In [97] wird eine „ideale“ Hashfunktion gefordert, so daß man sich hier letztlich ein Random Oracle vorstellen kann. Nimmt man aber „nur“ eine Familie von kollisionsresistenten Hashfunktionen gemäß Definition A.9 an, kann ein in jedem intuitiven Sinne unsicheres System entstehen.

es, für gegebenes $y = a^{-1}xa$ und $d = b^{-1}xb$ dieses Element z zu finden, um das mit d assoziierte Chiffrat zu entschlüsseln. Findet man sogar ein $a' \in LB_\ell$ mit $y = a'^{-1}xa'$ (oder ein $b' \in RB_r$ mit $d = b'^{-1}xb'$), so kann man hiermit insbesondere $z = a'^{-1}da'$ (bzw. $z = a'^{-1}ya'$) berechnen. In diesem Sinne beruht die Sicherheit des gerade skizzierten Systems auf der Schwierigkeit des Konjugationsproblems in der Gruppe B_n.

Es sei hier jedoch auf einen feinen Unterschied hingewiesen: Eine Lösung des Konjugationsproblems liefert in der obigen Situation nur einen „konjugierenden" Zopf $a' \in B_n$ mit $y = a'^{-1}xa'$; um zu dechiffrieren wird jedoch ein Zopf a' benötigt, welcher mit b kommutiert, was beispielsweise für $a' \in LB_\ell$ der Fall ist. Dieses verallgemeinerte Konjugationsproblem wird in [97] „Generalized Conjugacy Search Problem" genannt.

Deshalb sei bemerkt, daß der nachfolgend beschriebene heuristische Algorithmus – den experimentellen Ergebnissen nach zu urteilen – für die in [97] vorgeschlagenen Parameter auch dieses verallgemeinerte Konjugationsproblem erfolgreich löst. Bezüglich der genauen Wahl der beteiligten Zöpfe x, a, b werden in [97] keine näheren Angaben gemacht. Jedoch regen die dortigen Sicherheitsabschätzungen dazu an, Zöpfe x, a, b mit jeweils etwa 5–12 kanonischen Faktoren aus einer Zopfgruppe B_n bzw. LB_ℓ, RB_r mit $50 \leq n \leq 90$ und $\ell \approx r$ zu betrachten. Die konkrete Wahl der kanonischen Faktoren erfolgt dabei, in Ermangelung vorgeschlagener Alternativen, zufällig gemäß [44].

Der Vollständigkeit halber seien hier noch die gleichsam auf Varianten des Konjugationsproblems fußenden Schlüsselaustauschprotokolle aus Anshel u. a. [3], [97], Anshel u. a. [2] erwähnt. Der untenstehende heuristische Algorithmus wurde auch für Instanzen des Konjugationsproblems erfolgreich getestet, welche sich aus den für diese Systeme vorgeschlagenen Parametern ergeben. Die genauen Ergebnisse finden sich im nächsten Abschnitt.

2.3.3 Ein heuristischer Algorithmus

An dieser Stelle soll der heuristische Algorithmus aus Hofheinz und Steinwandt [89] für das besprochene Konjugationsproblem in Zopfgruppen angegeben werden. Es sei nochmals darauf hingewiesen, daß eine Lösung von Instanzen des Konjugationsproblems einen Weg darstellt, ein Kryptosystem zu brechen, ohne sich eine Schwäche in der Struktur des Systems zunutze zu machen – es wird einfach die zugrundegelegte Annahme (im vorliegenden Fall also die Annahme der Härte des Konjugationsproblems) angegriffen. Hierfür ist keine allgemeine Lösung des Konjugationsproblems vonnöten; es reicht vielmehr, für kryptographische Zwecke vorgeschlagene Instanzen zu betrachten und in einigen Fällen Erfolg zu haben.

Zuvor sollen jedoch noch zwei Bezeichnungen vereinbart werden: Ein Zopf $\gamma \in B_n^+$ wird *Wortende* eines Zopfes $\alpha \in B_n^+$ genannt, wenn eine Zerlegung $\alpha = \beta\gamma$ mit $\beta \in B_n^+$ existiert. Weiter bezeichne τ den inneren B_n-Automorphismus „Konjugation mit Δ", d. h. $\tau(w) = \Delta^{-1}w\Delta$ für $w \in B_n$. Das folgende Ergebnis wird für den zu besprechenden Algorithmus von Bedeutung sein; der erste Teil hiervon wurde

schon für den Angriff aus Lee und Lee [102] genutzt.

Lemma 2.2. *Seien $v, w \in B_n$ mit $w = x^{-1}vx$ für ein $x \in B_n^+$. Es bezeichne dabei $\Delta^r W_1 \cdots W_s$ die Δ-Normalform von w. Dann gilt:*

- *Ist* $\inf w < \inf v$*, so ist* $\Delta\tau^r(W_1^{-1})$ *ein Wortende von* x*.*
- *Ist* $\sup w > \sup v$*, so ist* W_s *ein Wortende von* x*.*

Beweis. Ein Beweis der ersten Aussage findet sich im Beweis von [59, Lemma 4.3]. Damit folgt die zweite Behauptung, da – in der Terminologie von [59] – „reverse cycling“ eines Zopfes $w \in B_n$ durch „cycling“ von w^{-1} erreicht werden kann. □

Zunächst ist zu beachten, daß die Forderung $x \in B_n^+$ für die hiesigen Zwecke keine Einschränkung darstellt: Da Δ^2 das Zentrum von B_n erzeugt (vgl. [64]), folgt aus $w = x'^{-1}vx'$ für ein $x' = \Delta^r X_1 \cdots X_s$ automatisch $w = x^{-1}vx$ für $x = \Delta^{r \bmod 2} X_1 \cdots X_s \in B_n^+$. Damit kann Lemma 2.2 wie folgt genutzt werden: Durch Konjugation von w mit $(\Delta\tau^r(W_1^{-1}))^{-1}$ bzw. W_s^{-1} kann ein gegebenes Konjugationsproblem (unter den genannten Voraussetzungen) auf ein Konjugationsproblem mit kürzerem konjugierenden Zopf x reduziert werden. Diese Technik wurde ursprünglich in [59] benutzt, um das super summit set $\mathcal{S}(w)$ eines Zopfes w zu berechnen.

Der erste Schritt des heuristischen Algorithmus aus [89] besteht nun darin, durch wiederholte Anwendung von Lemma 2.2 zu einem reduzierten Konjugationsproblem $w' = x'^{-1}vx'$ mit $\inf w' = \inf v$ und $\sup w' = \sup v$ zu gelangen. (Hier sei bemerkt, daß diese Bedingung nach endlich vielen Konjugationen mit $(\Delta\tau^r(W_1^{-1}))^{-1}$ bzw. W_s^{-1} eintritt, vgl. Birman u. a. [21] für eine genaue diesbezügliche Abschätzung.) Für alles Weitere entscheidend ist nun die folgende Beobachtung: Die bei initialer Parameterwahl gemäß [97, 44] erzeugten Instanzen des Konjugationsproblems führen oft zu reduzierten Instanzen $w' = x'^{-1}vx'$, bei denen der konjugierende Zopf x' schon *kanonischer Faktor* ist. Ein kanonischer Faktor x' ist andererseits schon eindeutig durch sein Bild $\pi(x')$ unter π, also durch die von ihm induzierte Permutation bestimmt.

Betrachtet man also das Konjugationsproblem $w' = x'^{-1}vx'$ in der *symmetrischen* Gruppe S_n, untersucht man also $\pi(w') = \tau^{-1}\pi(v)\tau$ für $\tau \in S_n$, so ist aufgrund der Homomorphieeigenschaft von π gewiß $\tau = \pi(x')$ eine gültige Lösung dieser Gleichung. Eine zweite Beobachtung ist nun, daß die durch Parameterwahl gemäß [97, 44] gewonnenen Instanzen dieses S_n-Konjugationsproblems oft nur wenige (in einem noch genauer zu fassenden Sinne) „einfache“ Lösungen x besitzen. Es bietet sich folgendes Vorgehen an: Bei gegebenem reduziertem Konjugationsproblem $w' = x'^{-1}vx'$ löse man diese Gleichung in der symmetrischen Gruppe für $\tau = \pi(x')$ und transportiere die Lösungen als kanonische Faktoren x' zurück in die Zopfgruppe B_n. Dargestellt in Pseudocode ergibt sich insgesamt der Algorithmus aus Abbildung 2.2.

In diesem Algorithmus wird $\inf x = 0$ für den konjugierenden Zopf $x \in B_n$ mit $w = x^{-1}vx$ angenommen. Dies geschieht ohne Beschränkung der Allgemeinheit, da

Algorithmus

- Eingabe: $v, w \in B_n$ mit $w = x^{-1}vx$ für ein $x \in B_n^+$ mit $\inf x = 0$.
- Ausgabe: entweder $\alpha \in B_n^+$ mit $w = \alpha^{-1}v\alpha$ oder '`failed`'.

1. Initialisiere α als das leere Wort ε.
2. Berechne die Δ-Normalform von v und w,
   so daß $w = \Delta^r W_1 \cdots W_s$.
3. While $\inf w < \inf v$ do
   Setze $\gamma := \Delta\tau^r(W_1^{-1})$, $\alpha := \gamma\alpha$, $w := \gamma w\gamma^{-1}$,
   Berechne die Δ-Normalform von w wie in 2.
4. While $\sup w > \sup v$ do
   Setze $\gamma := W_s$, $\alpha := \gamma\alpha$, $w := \gamma w\gamma^{-1}$,
   Berechne die ΔNormalform von w wie in 2.
5. Setze $\mu :=$ `GuessPermutation`(v, w), $\alpha := \mu\alpha$, $w := \mu w\mu^{-1}$.
6. If $v = w$ then
   Return α,
   else
   Return '`failed`'.

Abbildung 2.2: Ein heuristischer Algorithmus zur Lösung des Konjugationsproblems in der Zopfgruppe B_n

wie geschildert $0 \leq \inf x \leq 1$ vorausgesetzt werden darf, und der Fall $\inf x = 1$ in den Fall $\inf x = 0$ mittels Konjugation mit Δ^{-1} überführt werden kann. Dabei ist nicht von Belang, daß $\inf x$ nicht bekannt ist; da nur $\inf x = 0$ oder $\inf x = 1$ in Frage kommt, kann der Algorithmus für beide Werte ausgeführt werden, wobei im Fall $\inf x = 1$ der Zopf w zunächst mit Δ^{-1} konjugiert wird.

Es ist soweit noch nicht besprochen worden, wie eine Lösung $\tau \in S_n$ der Gleichung $\pi(w') = \tau^{-1}\pi(v)\tau$ in der symmetrischen Gruppe S_n gefunden werden kann. Im Algorithmus wird diese Aufgabe in eine Unterroutine „`GuessPermutation`“ ausgelagert. Es ist klar, daß Konjugation mit einer Permutation $\tau \in S_n$ einen Zyklus $(v_1, \ldots, v_i)$ der Permutation $\pi(v)$ auf den Zyklus $(\tau(v_1), \ldots, \tau(v_i))$ abbildet. Insbesondere wird, wenn $\pi(v)$ etwa j Zyklen der Länge k hat, jede Lösung τ Anlaß zu weiteren $j! \cdot k^j - 1$ Lösungen geben.

Nun zeigen allerdings praktische Experimente, daß – in obiger Notation – x' meist nicht nur kanonischer Faktor ist, sondern die induzierte Permutation $\pi(x')$ noch dazu eine einfache, „lokale“ Form hat. Es empfiehlt sich deshalb für die Routine „`GuessPermutation`“, eine möglichst einfache Permutation τ aus einer möglicherweise großen Lösungsmenge auszuwählen. Experimentell zeigt sich dabei die Formulierung des Algorithmus aus Abbildung 2.3 erfolgreich.

Die Idee dieser Formulierung von „`GuessPermutation`“ ist denkbar einfach: In Schritt 3 werden Anfänge von Zyklen von $\pi(v)$ mit Anfängen von Zyklen von

`GuessPermutation`

- Eingabe: $v, w \in B_n$ mit $w = x^{-1}vx$ für ein $x \in B_n^+$ mit $\inf x = 0$.
- Ausgabe: ein kanonischer Faktor $\mu \in B_n^+$.

1. Initialisiere $\tau \in S_n$ als Identität.
2. Setze $(\chi_1, \ldots, \chi_n) := (\texttt{false}, \ldots, \texttt{false})$.
3. For i from n downto 1 do
   Setze $r := i$, $s := i$,
   While $\chi_r = \texttt{false}$ do
   Setze $\chi_r := \texttt{true}$,
   Setze $r := \pi(v)(r)$, $s := \pi(w)(s)$,
   If $r \neq s$ then
   Setze $\tau(r) := s$.
4. Setze $(\xi_1, \ldots, \xi_n) := (\texttt{false}, \ldots, \texttt{false})$.
5. For i from n downto 1 do
   If $\xi_i = \texttt{false}$ then
   Setze $\xi_i := \texttt{true}$, $r := i$,
   While $\xi_{\tau(r)} = \texttt{false}$ and $\tau(r) \neq r$ do
   Setze $r := \tau(r)$, $\xi_r := \texttt{true}$.
   Setze $\tau(r) := i$.
6. Return $\pi^{-1}(\tau)$.

Abbildung 2.3: Die Unterroutine `GuessPermutation`

$\pi(w)$ in Verbindung gebracht; anschließend werden die gesamten jeweiligen Zyklen identifiziert. Dabei wird ausgenutzt, daß die „Enden“ der jeweiligen Zyklen identisch sind, was natürlich nur bei einfachen konjugierenden Permutationen $\pi(x)$ der Fall ist. Hierbei eventuell auftretende Unvollständigkeiten in der Darstellung von $\pi(x)$ werden in Schritt 5 bereinigt. Die für die Bestimmung des Rückgabewertes in Schritt 6 benutzte Schreibweise $\pi^{-1}(\tau)$ ist so zu interpretieren: $\pi^{-1}(\tau)$ liefert den eindeutig bestimmten kanonischen Faktor $\alpha' \in B_n^+$ mit $\pi(\alpha') = \tau$.

2.3.4 Experimentelle Ergebnisse

Um den vorangehenden Behauptungen Kraft zu verleihen, wurde der Algorithmus aus Abbildung 2.2 nebst der Routine „`GuessPermutation`“ in C++ implementiert. Hierbei wurde die `CBraid`-Bibliothek aus Cha [43] verwendet. Die Versuche fanden auf einem gängigen, mit 1,8 GHz getakteten PC statt. Erfreulicherweise bearbeitet der Algorithmus dabei einzelne Konjugationsprobleme sehr schnell, so daß lange Versuchsreihen durchgeführt werden konnten. Die aus einer Parameterwahl gemäß [97, 44] resultierenden Instanzen waren dabei der Form $w = x^{-1}vx$ für Zöpfe $x \in LB_{n/2} \subseteq B_n$ und $v \in B_n$, die jeweils aus p kanonischen Faktoren bestanden.

Dabei variiert n von 45 bis 90, und p von 3 bis 12; ferner wurde $\ell = \lfloor \frac{n}{2} \rfloor$ und $r = \lceil \frac{n}{2} \rceil$ gewählt. Für diese Wahl der Instanzen konnten die folgenden Ergebnisse beobachtet werden:

n	p	Versuche	Erfolgsquote
45	3	1000	78,1%
50	5	1000	79,1%
70	7	1000	79,0%
90	12	1000	80,0%

Hierbei wurde eine Ausgabe des Algorithmus genau dann als „Erfolg“ gewertet, wenn ein $\alpha \in B_n^+$ mit $w = x^{-1}vx$ gefunden werden konnte, d. h. wenn das Konjugationsproblem erfolgreich gelöst werden konnte. Es ist klar, daß angesichts dieser Ergebnisse das vorhin vorgestellte System aus [97] – für die dort vorgeschlagenen Parameter – nicht mehr als sicher angesehen werden darf.

Zwar existierten zum Zeitpunkt der Publikation [89] des besprochenen Algorithmus und der damit erzielten Ergebnisse schon Angriffe sehr unterschiedlicher Natur auf zopfgruppenbasierte Systeme; genannt seien hier insbesondere die Angriffe aus [102], Hughes [92]. Jedoch waren diese Angriffe allesamt theoretischer Natur; die Komplexität dieser Angriffe konnte theoretisch nicht vollständig geklärt werden, und praktische Implementierungen funktionierten nur für sehr kleine Parameter (etwa für Zopfgruppenindizes $n < 10$). Damit war [89] die erste Arbeit, die vorgeschlagene zopfgruppenbasierte Systeme *praktisch* und damit nachvollziehbar erfolgreich angreifen konnte.

Tatsächlich konnten in [89] die Ergebnisse auch auf das in [97] vorgeschlagene Schlüsselaustauschverfahren ausgedehnt werden; dies erforderte keine weiteren Bemühungen, da dieses System auf demselben algebraischen Problem beruht. Auch konnte der gegebene Angriff auf das Schlüsselaustauschverfahren aus [3, 2] übertragen werden, das seine kryptographische Stärke aus der Schwierigkeit des *simultanen* Konjugationsproblems in Zopfgruppen zieht. Dieses Problem besteht in der Aufgabe, bei gegebenen Paaren $(v_i, w_i = x^{-1}v_ix)$ ein konjugierendes x mit $w_i = x^{-1}v_ix$ für alle i zu finden. Die Erfolgsquote für Instanzen dieses simultanen Konjugationsproblems, die aus den in [3, 2] vorgeschlagenen Parametern resultieren, war generell sogar noch höher. Hierzu war natürlich eine in [89] geschilderte Anpassung des Algorithmus aus Abbildung 2.2 vonnöten.

Abschließend muß also festgestellt werden, daß die Systeme aus [3, 97, 2] für die dort vorgeschlagene Parameterwahl als nicht sicher angesehen werden sollten. Es konnte ein Algorithmus für das zugrundeliegende algebraische Problem – das Konjugationsproblem in Zopfgruppen – angegeben werden. Zwar löst dieser das Problem nicht im mathematischen Sinne, denn es kann keine Lösung garantiert werden, und insbesondere kam es bei den Experimenten vor, daß der Algorithmus keine Lösung lieferte. Jedoch konnte im Falle von für kryptographische Zwecke vorgeschlagenen Instanzen eine ausreichende Erfolgswahrscheinlichkeit des Algorithmus experimentell nachgewiesen werden.

2.3.5 Neuere Entwicklungen

Seit Publikation der Arbeit [89] gab es einige neuere Entwicklungen. Die Arbeit Cheon und Jun [45] entwickelt einen polynomialen Algorithmus zur Lösung des Search-Diffie-Hellman-Problems in Zopfgruppen. Dies ist insofern hochinteressant, als dieses Problem eng verwandt ist mit dem Problem, auf dem die Systeme aus [97] basieren. In der Tat bricht der Algorithmus aus [45] schon die in [97] vorgeschlagenen Systeme. Weiter konnte auch die in [44] angedeutete Variante des Systems aus [97], welche nicht mehr auf dem Diffie-Hellman-Problem in Zopfgruppen fußt, in der Arbeit Lee und Park [101] erfolgreich angegriffen werden.

Andererseits konnte in Dehornoy [53], Sibert [128] ein eleganter Weg der Instanzenwahl für das Konjugationsproblem in Zopfgruppen angegeben werden, welcher durch eine geschickte Vorverarbeitung der Zöpfe v und w versucht, den in diesem Kapitel behandelten Angriff auszuschließen. Tatsächlich bestätigen Experimente den diesbezüglichen Erfolg dieser Vorbehandlung, soweit es den hier beschriebenen Angriff betrifft. Bislang steht allerdings eine theoretische Analyse dieser Instanzenwahl aus. Überdies wirft die in Gebhardt [66] beschriebene allgemeine Methode, das Konjugationsproblem in Zopfgruppen anzugehen, ernste Fragen bezüglich einer sicheren Instanzenwahl auf.

3 Eine intuitive Modellierung von Public-Key-Systemen

Nachdem im vorangehenden an einigen Beispielen zugrundeliegende Bausteine von Public-Key-Kryptosystemen untersucht wurden, soll jetzt der schon anfangs aufgeworfenen Frage nach einer sinnvollen *Sicherheitsdefinition* eines solchen Systems nachgegangen werden. Es ist klar, daß die im letzten Kapitel angesprochenen Systeme angesichts der dortigen Betrachtungen in keinem intuitiven Sinn als „sicher“ betrachtet werden können. Offen bleibt aber immer noch die Frage nach einem hinreichenden Kriterium für Sicherheit – wobei soweit noch nicht einmal geklärt wurde, was mit Sicherheit überhaupt gemeint ist.

In der Einleitung wurde dargestellt, daß sich, beginnend mit der Arbeit Goldwasser und Micali [75], eine Reihe von sehr unterschiedlich motivierten Definitionen herausgebildet hat. Ohne auf Details einzugehen, sei in Abbildung 3.1 allein zur Darstellung der Unübersichtlichkeit der Sachlage ein Diagramm aus Bellare u. a. [17] wiedergegeben, welches Implikationen zwischen verschiedenen formalen Sicherheitsdefinitionen für Public-Key-Kryptosysteme veranschaulicht.

Ein Pfeil zwischen zwei Begriffen $A \to B$ bedeutet eine Implikation, ein durchgestrichener Pfeil die Existenz eines trennenden Beispiels (sofern überhaupt ein System den Anforderungen von A genügt); das Diagramm ist in dem Sinne vollständig, als daß durch Ausnutzen von Transitivität für je zwei Begriffe A, B entweder $A \to B$ oder $A \not\to B$ erkannt werden kann. Die Abkürzungen sind wie folgt zu lesen: Der Teil vor dem „-“ bezeichnet das Ziel des Angriffs, wobei „IND“ für „indistinguishability of ciphertexts“ steht (siehe[1] [75]), und „NM“ für „non-malleability“ (vgl. Dolev u. a. [55]). Der Teil nach dem „-“ legt den Typ eines Angriffs fest, wobei „CPA“ für „chosen-plaintext-attack“ steht, und „CCA1“ und „CCA2“ für die Chosen-Ciphertext-Angriffe aus Naor und Yung [110] bzw. Rackoff und Simon [121] stehen. „IND-CPA“ meint demnach „indistinguishability of ciphertexts under chosen-plaintext attacks“ und damit eine vergleichsweise schwache Eigenschaft eines Public-Key-Kryptosystems.

In Abbildung 3.1 sind längst nicht alle gängigen Sicherheitsdefinitionen erfaßt; prominente Abwesende sind insbesondere jegliche Formulierung semantischer Sicherheit (geprägt in [75]) und der Begriff der „plaintext awareness“ (vgl. Bellare und Rogaway [19]).

Hier sollen diese Zusammenhänge nicht weiter erforscht werden; vielmehr soll ein intuitiver Zugang zu einer Sicherheitsdefinition für Public-Key-Kryptosysteme

[1] In dieser Arbeit wurde der Begriff noch „polynomial security“ genannt.

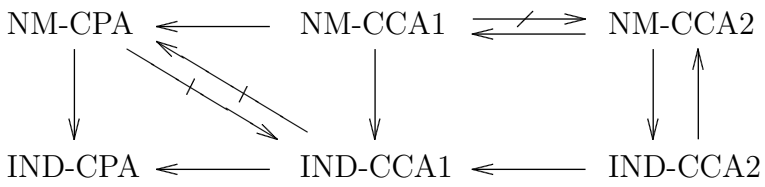


Abbildung 3.1: Implikationen zwischen verschiedenen Sicherheitsdefinitionen für Public-Key-Kryptosysteme, vgl. [17]

gewählt werden. Hierbei soll im Vordergrund stehen, daß ein Public-Key-Kryptosystem direkt zur Übermittlung von Nachrichten genutzt wird. In diesem Szenario fällt eine Formulierung von Sicherheit leichter; es drängt sich folgende Forderung auf: Ein Public-Key-Kryptosystem ist „sicher“, wenn mit ihm (bei naheliegender Benutzung) ein sicherer Kanal realisiert werden kann. Es soll also für einen Teilnehmer A möglich sein, Nachrichten sicher zu einem anderen Teilnehmer B zu schicken, indem eine Nachricht einfach mit dem öffentlichen Schlüssel von B verschlüsselt wird. Wie gesagt bedeutet „sicher“ in diesem Zusammenhang nun, daß durch dieses Vorgehen ein sicherer Kanal von A nach B realisiert wird.

Aus verschiedenen Gründen war diese Betrachtung sehr vage, und sie kann auch zum jetzigen Zeitpunkt nur vage sein, da noch gewisse technische Hilfsmittel fehlen. Es ist noch nicht geklärt, was es heißt, „einen sicheren Kanal zu realisieren“. Es ist noch nicht einmal klar, was ein sicherer Kanal überhaupt ist. Auch wurde die Frage der Authentifikation bislang nicht besprochen; sollen die Chiffrate schon über einen authentifizierten Kanal geschickt werden, oder soll eine explizite Authentifikation erfolgen?

All diese Fragen werden im folgenden nach und nach wieder aufgegriffen und geklärt werden. Für den Augenblick sollte allein die Anschauung im Vordergrund stehen, die hinter der in diesem Kapitel vorgestellten Modellierung steckt. Es soll nicht verschwiegen werden, daß damit auch eine Motivation geschaffen werden sollte, den folgenden Abschnitt gründlich zu lesen, obwohl dieser auf den ersten Blick keine Verbindung zur Problematik zu haben scheint. Jedoch ist es wichtig, zunächst alle Rahmenbedingungen abzustecken, um schließlich mit einem gefestigten Begriff von „Sicherheit“ belohnt werden zu können.

3.1 Gewählte Modellierung von Mehrparteienberechnungen

Was folgt, ist zunächst eine Fixierung des Umfelds, in dem Sicherheitsdefinitionen betrachtet werden sollen. Dazu wird auf ein sehr allgemeines Berechnungsmodell für Mehrparteienprotokolle zurückgegriffen. Dies geschieht in voller Allgemeinheit, um auch für die später noch zu besprechenden Protokollaufgaben gerüstet zu sein. Es kann damit eine einheitliche Betrachtung sehr unterschiedlicher Protokollaufgaben erfolgen; eine Beschränkung auf die Protokollaufgabe „sichere Nachrichtenüber-

mittlung“ bzw. „sicherer Kanal“ ist hier weder nötig noch vorteilhaft. Zudem kann auch die *Verwendung* beispielsweise eines Public-Key-Kryptosystems in größeren Protokollen auf diese Weise nachvollzogen werden.

Da für die hiesigen Zwecke entscheidend, soll sich hier auf Berechnungsmodelle konzentriert werden, welche sichere Komposition von Protokollen erlauben. Damit ist der Erhalt von Sicherheitseigenschaften bei Verwendung von ein oder mehreren Instanzen des betrachteten Protokolls in größeren Protokollen gemeint.

Irritierenderweise folgt diese Eigenschaft nicht automatisch: Beispielsweise verliert jedes Drei-Runden-Protokoll für einen „zero-knowledge proof of knowledge“ (für „interessante“ Sprachen und „black-box-Sicherheit“) seine Sicherheit unter paralleler Komposition, vgl. Goldreich und Krawczyk [72]. Glücklicherweise kann allerdings im Falle von *simulierbarkeitsbasierten* Modellen eine intuitive und allgemeine Formulierung von Sicherheit mit der Garantie einer solchen *Komponierbarkeit* verbunden werden.

Alternativ zu dem hier gewählten und alsbald zu besprechenden Modell aus Canetti [27] hätten sich insbesondere die Modellierungen Pfitzmann u. a. [115], Canetti [26], Pfitzmann und Waidner [117], Maurer [107], Backes u. a. [11] angeboten. Die Wahl von [27] erfolgte dabei vor allem wegen einer schon vorhandenen „Infrastruktur“ zahlreicher formulierter Idealisierungen und untersuchter Protokolle. Doch nun zum eigentlichen Modell aus [27].

3.1.1 Maschinenmodell

Die Parteien eines Protokolls, insbesondere aber auch ein explizit als eigenständige Maschine aufgefaßter Angreifer, sind modelliert als interaktive Turingmaschinen (ITMs). Wichtig dabei ist, daß eine ITM mehrmals aktiviert werden kann und somit sowohl eine Menge von *Wartezuständen* als auch eine Menge von *Haltezuständen* besitzt. Angelehnt an Goldwasser u. a. [76] wird in [27] eine *interaktive Turingmaschine (ITM)* als Erweiterung einer Turingmaschine definiert.

Eine ITM A verfügt neben dem Arbeitsband über ein nur lesbares Identitätsband, welches die (im Kontext mehrerer ITMs als eindeutig geforderterte) Identität von A enthält. Weiter besitzt A ein nur lesbares Sicherheitsparameterband, auf welchem der betrachtete Sicherheitsparameter k codiert ist. ([27] legt nicht fest, ob dies in binärer oder unärer Codierung zu geschehen hat; für alles Folgende ist dies aber nicht von Belang.) Ferner hat A ein Zufallsband, welches eine bei der initialen Aktivierung von A gewählte unendliche Folge unabhängig gleichverteilter Zufallsbits enthält.

Außerdem besitzt A ein nur lesbares Eingabeband und ein nur einmal beschreibbares Ausgabeband; ferner ein Eingangs-Kommunikationsband und ein Ausgangs-Kommunikationsband. Es wird hier eine Codierung vorausgesetzt, die es erlaubt, den Inhalt dieser Kommunikationsbänder als Folge von Nachrichten zu interpretieren, welche ihrerseits jeweils aus einem Sender- bzw. Empfängerfeld (welches die Identität der sendenden bzw. empfangenden ITM festlegt) und einem Inhaltsfeld bestehen. (Diese Codierung ist zunächst syntaktisch zu verstehen; es ist nicht

immer garantiert, daß eine Nachricht nicht mit *gefälschtem* Absender ausgeliefert wird. Details hierzu werden im nächsten Abschnitt geklärt.)

Schließlich besitzt A ein les- und beschreibbares Ein-Bit-*Aktivierungsband*, welches festlegt, ob A gerade aktiv ist. Eine Aktivierung von A wird automatisch beendet, wenn A einen Warte- oder einen Haltezustand einnimmt. Wichtig für die weiteren Betrachtungen ist, daß zwischen Halte- und Wartezuständen einer ITM A unterschieden wird. Geht A in einen Wartezustand über, so kann A zu einem späteren Zeitpunkt, etwa im Falle neuer Nachrichten oder Eingaben, erneut aktiviert werden. Befindet sich A allerdings in einem Haltezustand, so läßt jede weitere Aktivierung A invariant.

Was Komplexitätsbetrachtungen angeht, interessiert hier nur die Unterscheidung „effizienter“ und „ineffizienter“ ITMs. Es bietet sich dafür der Begriff der polynomialen Laufzeit an: Eine ITM A heißt *(strikt) polynomial beschränkt* genau dann, wenn ein Polynom $p_A \in \mathbb{Z}[k]$ existiert, so daß A für alle Sicherheitsparameter k unabhängig von der Belegung des Zufallsbands und unabhängig von eingehenden Eingaben und Nachrichten jede Aktivierung nach höchstens $p_A(k)$ Schritten beendet und nach höchstens $p_A(k)$ Aktivierungen in einen Haltezustand übergeht.

Um ständige Textwiederholungen zu vermeiden, seien – wie in [27] – bis auf weiteres alle betrachteten ITMs als polynomial beschränkt angenommen. Man beachte hier, daß damit die Komplexität einer ITM als polynomial *allein im Sicherheitsparameter k* gefordert wird; damit sind etwa ITMs, die lediglich ihre Kommunikation bzw. Eingabe lesen, *nicht* polynomial beschränkt. In der Tat führen ITMs, deren Komplexität polynomial in der Gesamtlänge ihrer eingehenden Kommunikation und ihrer Eingaben ist, zu Problemen bei der Modellierung mehrerer, miteinander interagierender Maschinen. (Man denke hier an zwei Maschinen, die sich gegenseitig jeweils mit der verdoppelten eigenen Eingabe aktivieren; die Kommunikationskomplexität dieses „Protokolls“ steigt exponentiell in der Anzahl der Aktivierungen.) Natürlich schließt diese Regelung aus, daß Maschinen betrachtet werden können, welche beliebig lange Eingaben akzeptieren; dies kann insbesondere im Falle einer simplen Maschine zur Nachrichtenweiterleitung als wenig intuitiv empfunden werden. Eine genauere Besprechung, in der eine teilweise Behebung dieses Mißstandes besprochen werden soll, findet in Abschnitt 3.2.2 statt.

3.1.2 Das reale Modell

Das Zusammenspiel mehrerer ITMs kann am Beispiel des „realen Modells“ beschrieben werden. Es handelt sich hierbei um den Versuch, einen in der Realität auftretenden Protokollablauf zu modellieren. Der Name ist nur bedingt gerechtfertigt, da das reale Modell immer noch von gewissen Implementierungsdetails abstrahiert; etwa können Seitenkanalangriffe (siehe etwa Kocher [99], Kocher u. a. [100]) nicht in direkter Weise betrachtet werden.

Sei also ein *n-Parteien-Protokoll* π in Form von n ITMs $P_1, \ldots, P_n$ gegeben. Sei weiter $\mathcal{A}$ ein *Angreifer*, d. h. eine weitere ITM, deren Aktionsmöglichkeiten noch zu beschreiben sind. Sei schließlich $\mathcal{Z}$ eine in [27] *Umgebung* genannte ITM;

intuitiv wird mit $\mathcal{Z}$ eine Protokollumgebung für π modelliert, also letztlich ein größeres, π als Subprotokoll benutzendes Protokoll. Diese Protokollumgebung wird hier nicht fixiert, sondern man wird später verschiedene Protokollumgebungen $\mathcal{Z}$ betrachten und fordern, daß das betrachtete Protokoll in *jeder* solchen Umgebung „sicher“ bleibt (wobei noch zu klären ist, was damit genau gemeint ist, vgl. auch die Diskussion in Abschnitt 3.1.4).

Dann kann ein Protokollablauf durch folgenden Algorithmus definiert werden, wobei initial alle ITMs deaktiviert seien:

1. $\mathcal{Z}$ wird aktiviert. $\mathcal{Z}$ hat Zugriff nicht nur auf die eigenen Bänder, sondern darf auch die Ausgabebänder aller Parteien und von $\mathcal{A}$ lesen und die jeweiligen Eingabebänder (d. h. die Eingabebänder der P_i und $\mathcal{A}$) beschreiben.[2] Nimmt $\mathcal{Z}$ einen Haltezustand ein, so endet der Algorithmus, und der Inhalt der ersten Zelle des Ausgabebandes von $\mathcal{Z}$ wird kanonisch als Bit interpretiert und durch den Algorithmus ausgegeben. Nimmt $\mathcal{Z}$ einen Wartezustand ein, so wird angenommen, daß $\mathcal{Z}$ genau ein Eingabeband einer anderen ITM beschrieben hat.[3] Diese ITM wird als nächstes aktiviert (siehe Schritte 2 und 3).

2. Wird $\mathcal{A}$ aktiviert, so darf $\mathcal{A}$ insbesondere

   - die Ausgangs-Kommunikationsbänder der Parteien P_i sehen (in dem Sinne, daß nach Wechsel von $\mathcal{A}$ in einen speziellen Zustand die Belegung der Bänder auf das Eingangs-Kommunikationsband von $\mathcal{A}$ geschrieben wird),
   - eine (beliebige) Nachricht (s, c) mit Absender s und Inhalt c an eine Partei P_i ausliefern, was bedeutet, daß (s, c) auf das Eingangs-Kommunikationsband von P_i geschrieben wird, und
   - eine Partei P_i korrumpieren, was damit verbunden ist, daß der aktuelle und alle vorigen Zustände von P_i auf das Eingabeband von $\mathcal{A}$ geschrieben werden, $\mathcal{Z}$ eine Nachricht über die Korruption von P_i erhält und P_i automatisch einen Haltezustand einnimmt.

   Es wird angenommen, daß $\mathcal{A}$ in einer Aktivierung höchstens eine Nachricht an eine Partei geschrieben hat.[4] Hat $\mathcal{A}$ eine Nachricht an eine Partei P_i ausgeliefert, so wird als nächstes P_i aktiviert (siehe nächster Punkt). Wenn nicht, wird als nächstes wieder $\mathcal{Z}$ aktiviert, wobei $\mathcal{Z}$ dann wie schon beschrieben alle Ausgaben lesen kann, die $\mathcal{A}$ möglicherweise generiert hat.

[2] [27] spezifiziert nicht, wie dies konkret geschieht; hier reicht es anzunehmen, daß $\mathcal{Z}$ solche Information durch spezielle Zustandsübergänge „erfragen“ und übermitteln darf.

[3] Auch hier legt [27] keine Details fest, wie diese Annahme zu rechtfertigen ist; es reicht hier aus, sich auf Umgebungen $\mathcal{Z}$ zu beschränken, welche in diesem Sinne „gutartig“ sind.

[4] In Ermangelung einer besseren Beschreibung in [27] soll sich hier also auch auf in diesem Sinne „gutartige“ Angreifer $\mathcal{A}$ beschränkt werden.

3. Wird schließlich eine Partei P_i aktiviert, so wird nach Beendigung der Aktivierung von P_i wieder $\mathcal{Z}$ aktiviert.

Mehrere Dinge bedürfen hier einer Diskussion. Zunächst eine Bemerkung zum grundsätzlichen Protokollablauf: Die gegebene Beschreibung legt fest, daß zu jedem Zeitpunkt immer nur genau eine ITM aktiv ist; dies entspricht natürlich nicht einer „realen" Protokollsituation. Die gegebene *asynchrone* Modellierung erleichtert jedoch zum einen die praktische Handhabung des Modells (etwa, was Beweistechniken angeht), und zum anderen lassen sich in feingranularer Weise Verfeinerungen eines solchen asynchronen Scheduling-Modells angeben, welche als unter Umständen realistischer betrachtete Modellierungen abbilden. Hier sei insbesondere auf die Arbeiten Hofheinz und Müller-Quade [82] und Backes, Hofheinz, Müller-Quade und Unruh [8], sowie auf Kapitel 5 dieser Arbeit verwiesen.

Das beschriebene Scheduling verläuft in einer „nachrichtengetriebenen" Art und Weise: Wird eine Nachricht an eine ITM ausgeliefert oder bekommt eine ITM neue Eingaben, so wird diese ITM als nächstes aktiviert. Hat nach Beendigung einer Aktivierung einer ITM keine andere ITM eine Nachricht oder eine Eingabe bekommen, so wird die Umgebung $\mathcal{Z}$ als nächstes aktiviert. (Im Sinne der Modellierung von [117, 11] ist $\mathcal{Z}$ demnach „Master Scheduler".)

Es sollte hier deutlich hervorgehoben werden, daß eine *direkte* Kommunikation der Parteien untereinander nicht möglich ist. Jegliche Nachricht zwischen Protokollteilnehmern wird vom Angreifer $\mathcal{A}$ ausgeliefert. Dabei ist $\mathcal{A}$ nicht nur von jeder Auslieferungspflicht entbunden, sondern es wird noch nicht einmal die Authentizität der ausgelieferten Nachrichten gewährleistet. Für eine Betrachtung von diesbezüglichen Alternativen, etwa der Maßgabe einer *fairen* Auslieferung nach (im Sicherheitsparameter k) polynomial vielen Schritten, sei erneut auf [82] und [8], und auf Kapitel 5 dieser Arbeit verwiesen.

Jedoch schon in [27] wurde die Möglichkeit einer *authentifizierten* Nachrichtenübermittlung zwischen den Parteien berücksichtigt. Bei dieser Variation des Modells darf der Angreifer $\mathcal{A}$ nur dann eine Nachricht (P_i, c) (d. h. eine Nachricht mit Absender P_i) an eine Partei P_j ausliefern, wenn genau diese Nachricht zuvor tatsächlich von P_i gesandt wurde, also auf dem Ausgangs-Kommunikationsband von P_i stand. Im Falle einer korrumpierten Partei P_i wird diese Forderung natürlich fallengelassen, und $\mathcal{A}$ darf Nachrichten im Namen von P_i versenden. Wenn im folgenden authentifizierte Kanäle vorausgesetzt werden, ist von dieser Variation des Modells die Rede.

Angesprochen werden sollte auch, daß eine Kommunikation zwischen Protokollumgebung $\mathcal{Z}$ und Angreifer $\mathcal{A}$ ausdrücklich erlaubt ist; jedoch ist es dem Angreifer $\mathcal{A}$ nicht möglich, $\mathcal{Z}$ zu korrumpieren.

Die Besprechung eines weiteren, im Augenblick noch nicht hinreichend erklärten Details soll noch aufgeschoben werden: die Frage nach der Notwendigkeit, die Protokollumgebung $\mathcal{Z}$ über Korruptionen zu informieren. Es scheint zunächst unrealistisch und geradezu widersinnig, ein benutzendes Protokoll (welches ja durch $\mathcal{Z}$ modelliert wird) über Korruptionen zu informieren. Eine schlüssige Begründung

hierfür kann jedoch erst nach Einführung der Sicherheitsdefinition für π gegeben werden.

3.1.3 Das ideale Modell

Für die Beurteilung der Sicherheit eines konkreten Protokolls π wird bei einem simulationsbasierten Ansatz wie etwa [27] der Vergleich mit einer *Idealisierung* der jeweiligen Protokollaufgabe gesucht. Mit anderen Worten, Sicherheit eines Protokolls wird relativ zu einer idealisierten Version dieses Protokolls betrachtet. In dieser idealisierten Version wird dabei das gewünschte Ein-/Ausgabeverhalten für die jeweilige Protokollaufgabe in einer übersichtlichen Art und Weise spezifiziert. Dies geschieht nämlich durch eine einzige, *ideale Funktionalität* genannte ITM, welche alle für Protokollteilnehmer bestimmten Eingaben liest und die individuellen Ausgaben festlegt.

Ein solchermaßen idealisierter Protokollablauf kann analog zum Ablauf im soeben beschriebenen realen Modell durch einen sogleich zu beschreibenden Algorithmus modelliert werden. Vorauszuschicken ist, daß Protokollabläufe in diesem idealen Modell und im realen Modell eine wichtige Gemeinsamkeit verbindet: die Existenz eines größeren Protokolls, welches durch eine Protokollumgebung $\mathcal{Z}$ modelliert wird. Dies ist sinnvoll, um mittels $\mathcal{Z}$ Vergleiche zwischen einer realen Implementierung π eines Protokolls und seiner Idealisierung ziehen zu können.

Um für $\mathcal{Z}$ im idealen Modell zumindest eine Schnittstelle zu bieten, welcher der im realen Modell entspricht, wird die ideale Funktionalität um sogenannte *Dummy-Parteien* ergänzt, welche formal die Verbindung zwischen $\mathcal{Z}$ und der jeweiligen idealen Funktionalität $\mathcal{F}$ darstellen. Zur Unterscheidung von Parteien $P_1, \ldots, P_n$ im realen Modell werden die Dummy-Parteien $\tilde{P}_1, \ldots, \tilde{P}_n$ genannt. Praktisch stellen diese Dummy-Parteien allerdings nichts weiter als Relais dar, welche alle Eingaben von $\mathcal{Z}$ an $\mathcal{F}$ und alle Ausgaben von $\mathcal{F}$ an $\mathcal{Z}$ weiterleiten.

Schließlich wird auch ein Angreifer $\mathcal{S}$ im idealen Modell mitmodelliert, welcher oft als *Simulator* bezeichnet wird. Diese Bezeichnung ist auf die noch folgende Sicherheitsdefinition zurückzuführen; für den Moment sei nur angedeutet, daß dem Simulator die Aufgabe zufällt, Angriffe des jeweiligen realen Angreifers $\mathcal{A}$ in einer für $\mathcal{Z}$ ununterscheidbaren Weise „vorzugaukeln“. Hierfür stehen $\mathcal{S}$ jedoch aufgrund der Idealitätseigenschaft des Modells nur begrenzte Mittel zur Verfügung: Wie schon angesprochen, existiert keine (direkte oder indirekte) Kommunikation der Parteien untereinander – es existieren ja keine „echten“ Parteien, sondern nur Dummy-Parteien, welche im wesentlichen „Relais-Stationen“ sind. Der Simulator darf auch nur diese Parteien korrumpieren, was ihn zu nicht mehr befähigt, als die Protokolleingaben von korrumpierten Parteien zu ersetzen.

In gewisser Weise reflektiert der Simulator also Angriffe oder Schwächen eines Protokolls, welche prinzipiell nicht vermeidbar sind – ist eine Partei korrumpiert, kann sie immer ihre „legitimen“ Eingaben ignorieren und sich gewissermaßen vorstellen, sie sei mit anderen Eingaben gestartet worden.

Genauer ergibt sich folgender Algorithmus für die Protokollausführung im idealen

Modell, wobei wie schon gesagt die Teilnehmer an dieser Protokollausführung $\mathcal{Z}$, Dummy-Parteien $\tilde{P}_1, \ldots, \tilde{P}_n$, die ideale Funktionalität $\mathcal{F}$ und ein Angreifer $\mathcal{S}$ sind:

1. $\mathcal{Z}$ wird aktiviert. Während dieser Aktivierung hat $\mathcal{Z}$ dieselben Möglichkeiten wie im realen Modell. Nimmt $\mathcal{Z}$ einen Haltezustand ein, so terminiert der Algorithmus wie im realen Modell mit der Ausgabe von $\mathcal{Z}$.

2. Wird eine Dummy-Partei $\tilde{P}_i$ mit einer neuen Eingabe von $\mathcal{Z}$ aktiviert, so schreibt $\tilde{P}_i$ diese Nachricht direkt auf das Eingangs-Kommunikationsband von $\mathcal{F}$ und nimmt einen Wartezustand ein. Anschließend wird $\mathcal{F}$ aktiviert.

3. Nach der Aktivierung der idealen Funktionalität $\mathcal{F}$ wird $\mathcal{S}$ aktiviert, sofern $\mathcal{F}$ eine Nachricht an $\mathcal{S}$ geschrieben hat.[5] Ansonsten wird die ITM aktiviert, welche zuletzt vor $\mathcal{F}$ aktiviert war. Zu beachten ist: Hat $\mathcal{F}$ eine Nachricht an eine Dummy-Partei $\tilde{P}_i$ geschrieben, so wird diese *nicht* automatisch ausgeliefert.

4. Wird $\mathcal{S}$ aktiviert, so darf $\mathcal{S}$ insbesondere

   - die Ziele aller Nachrichten von $\mathcal{F}$ an Dummy-Parteien erfragen, *nicht* jedoch deren Inhalt;
   - eine Nachricht von $\mathcal{F}$ an eine Dummy-Partei *ausliefern*; auch hier bleibt dabei der Inhalt dieser Nachricht für $\mathcal{S}$ verborgen, außer, die betreffende Dummy-Partei ist korrumpiert;
   - eine Dummy-Partei $\tilde{P}_i$ korrumpieren, was dazu führt, daß der aktuelle und alle vorigen Zustände (darin enthalten also insbesondere alle Ein- und Ausgaben von $\tilde{P}_i$) auf das Eingabeband von $\mathcal{S}$ geschrieben werden, $\mathcal{Z}$ eine Nachricht über die Korruption von $\tilde{P}_i$ erhält, und $\tilde{P}_i$ automatisch einen Haltezustand einnimmt.

   Es wird angenommen, daß $\mathcal{S}$ in einer Aktivierung entweder höchstens eine Nachricht an eine Dummy-Partei ausgeliefert hat, oder genau eine Nachricht an $\mathcal{F}$ oder an $\mathcal{Z}$ geschrieben hat. Hat $\mathcal{S}$ also eine Nachricht an eine Dummy-Partei ausgeliefert, so wird diese Dummy-Partei als nächstes aktiviert; im zweiten Fall wird $\mathcal{F}$ bzw. $\mathcal{Z}$ mit der eingegangenen Nachricht aktiviert. Wenn keine Nachricht an eine Dummy-Partei ausgeliefert wurde und auch keine Nachricht an $\mathcal{F}$ oder $\mathcal{Z}$ geschrieben wurde, dann wird als nächstes $\mathcal{Z}$ aktiviert.

5. Wird eine Partei $\tilde{P}_i$ mit einer eingegangenen Ausgabe von $\mathcal{F}$ aktiviert, so kopiert $\tilde{P}_i$ diese Ausgabe auf ihr Ausgabeband und beendet die Aktivierung. Wird $\tilde{P}_i$ ohne neue Eingaben bzw. eingegangene Ausgaben von $\mathcal{F}$ aktiviert,

[5] Bezüglich der genauen Auslieferungsdetails schweigt sich [27] hier aus; hier sei angenommen, daß eine Auslieferung solcher Nachrichten automatisch geschieht.

beendet $\tilde{P}_i$ die Aktivierung sofort.[6] In jedem Fall wird anschließend $\mathcal{Z}$ aktiviert.

Die gegebene Beschreibung ist, wie auch schon die Beschreibung des realen Modells, nicht formal. Das hängt damit zusammen, daß schon die in [27] gegebene Beschreibung nicht formal ist, und überdies mit der Zeit einigen Änderungen unterzogen wurde (siehe z. B. die Modellbeschreibung in Canetti u. a. [39] im Vergleich zu [27]). Ein für weitere Ergebnisse irrelevantes Detail ist etwa, daß die gegebene (und aus [27] reproduzierte) Formulierung der Dummy-Parteien nicht zu polynomial beschränkten ITMs führt.

Die Frage der polynomialen Beschränkung von ITMs im allgemeinen wiegt jedoch wesentlich schwerer und ist Gegenstand der Untersuchungen in Hofheinz, Müller-Quade und Steinwandt [85]. Auf diese Frage soll in Abschnitt 3.2.2 noch genauer eingegangen werden, sobald die Problematik klarer dargestellt werden kann. Für den Augenblick kann aber bemerkt werden, daß im *speziellen* Fall der Dummy-Parteien ohne Schaden von unbeschränkten ITMs ausgegangen werden kann.

Auf einen wichtigen Zusammenhang soll hier – auch angesichts einer noch in Abschnitt 4.1.4 zu besprechenden Modelländerung in [39] – hingewiesen werden. In der gerade gegebenen Form des idealen Modells werden Eingaben von den Dummy-Parteien *sofort* zur idealen Funktionalität weitergeleitet; Ausgaben hingegen können beliebig lange von $\mathcal{S}$ verzögert werden. (Dies ist auch schon die im wesentlichen einzige Einflußmöglichkeit des Simulators auf den idealen Protokollverlauf.) Hintergrund dabei ist, daß für eine prinzipielle Vergleichbarkeit von realem und idealem Modell eine Möglichkeit für – im realen unvermeidliche – Verzögerungen von Protokollausgaben geschaffen werden soll.

3.1.4 Die Sicherheitsdefinition

Anschaulich wurde schon geschildert, wann ein Protokoll als sicher erachtet werden soll: Nämlich genau dann, wenn es „mindestens so sicher“ ist wie eine anwendungsspezifische Idealisierung des Protokolls bzw. der jeweiligen Protokollaufgabe. Es steht noch aus, den Ausdruck „mindestens so sicher“ genauer zu fassen. Für eine Simulierbarkeitsdefinition von Sicherheit geschieht das folgendermaßen: Ein Protokoll π wird als mindestens so sicher wie eine Idealisierung $\mathcal{F}$ erachtet, wenn jeder Angriff auf π durch einen Angriff auf $\mathcal{F}$ simuliert werden kann. Damit ist gemeint, daß jede „Schwäche“ von π auch schon in $\mathcal{F}$ enthalten ist. Die Beurteilung, wann ein realer Angriff erfolgreich durch einen Simulator simuliert werden konnte, fällt dabei der Protokollumgebung zu.

Intuitiv wird dies dadurch gerechtfertigt, daß die Protokollumgebung ein *benutzendes* Protokoll darstellt: Gibt es ein benutzendes Protokoll, welches zwischen

[6] Diese Situation wird in [27] nicht behandelt, tritt aber ein, wenn $\mathcal{F}$ nach einer Aktivierung durch Eingaben von einer Partei $\tilde{P}_i$ die Aktivierung ohne Nachricht an $\mathcal{S}$ beendet.

einem Angriff im realen und einem im idealen Modell unterscheidet, wird das reale Protokoll als unsicher im Sinne der idealen Spezifikation erachtet; denn diese Unterscheidbarkeit könnte bedeuten, daß im realen Protokoll unbeabsichtigte, potentiell schädliche Auswirkungen für den Benutzer des Protokolls auftreten, welche im idealen Protokoll nach Konstruktion ausgeschlossen wurden. Umgekehrt bedeutet (in diesem Sinne simulierbare) Sicherheit eines Protokolls, daß kein Benutzer des realen Protokolls einen irgendwie gearteten Schaden nimmt, welchen er nicht auch in der idealisierten Fassung des Protokolls nehmen könnte.

Gegen diese Anschauung kann nun folgendermaßen argumentiert werden: Nicht jeder reale Angriff, der als ungewollt eingestuft wird, hat Veränderungen für den Protokollbenutzer zur Folge. Beispielsweise könnte ein realer Angreifer $\mathcal{A}$ alle (geheimen) Protokollein- und -ausgaben erfahren, im Zuge dessen an geheime Schlüssel gelangen und alle Kommunikation zwischen den Protokollparteien entschlüsseln; jedoch könnte sich dieser Angreifer in keiner Weise verpflichtet fühlen, dies der Protokollumgebung mitzuteilen. Dann könnte dieser spezielle, mit Sicherheit nicht wünschenswerte reale Angriff möglicherweise im idealen Modell durch einen womöglich „stillen“ Simulator simuliert werden.

Dieser Argumentation kann leider nur auf formalem Grund begegnet werden: Gibt es einen solchen realen Angreifer $\mathcal{A}$, dann existiert auch ein Angreifer, welcher alle seine Angriffserfolge bereitwillig der Umgebung mitteilt. Für einen *solchen* Angreifer kann es nun keinen Simulator geben, der Ähnliches in einem idealen Modell mit einer hinreichend idealisierten Funktionalität ausrichten könnte.

Umgekehrt kann es Unterscheidungen zwischen realem und idealem Modell geben, die zu keinem „intuitiven“ Angriff korrespondieren. Beispielsweise wurden derart „pathologische“ Unterscheidungen in Hofheinz und Unruh [90] verwendet, um zwei Varianten von Simulierbarkeitsbegriffen zu trennen. Die dort als trennende Beispiele verwendeten Protokolle waren nur – bezüglich des stärkeren dort betrachteten Begriffs – „unsicher“, weil dem Simulator weniger Rechenzeit als der Umgebung zugestanden wurde (technisch ausgedrückt: die Umgebung $\mathcal{Z}$ wurde in Abhängigkeit vom Simulator $\mathcal{S}$ gewählt). Dem entspricht kein bekannter kryptographischer Angriff.

Trotzdem soll hier vor allem aus zwei Gründen die Unterscheidbarkeit von Protokollen als Sicherheitskriterium herangezogen werden: zum einen garantiert sie die sichere Verwendung eines Protokolls in größeren Protokollen (siehe hierzu Abschnitt 3.1.7). Zum anderen scheint es schwierig zu sein, alle „intuitiven“ Angriffe formal zu fassen; es ist leicht, eine Reihe von wünschenswerten Eigenschaften eines bestimmten Protokolls anzugeben, aber sind damit schon alle möglichen Angriffe berücksichtigt? Nach obiger Diskussion führt jeder intuitive Angriff zu einer Unterscheidung, sodaß mit Ununterscheidbarkeit höchstens zuviel, aber auf keinen Fall zuwenig gefordert wird.

Folgende Definition scheint damit gerechtfertigt:[7]

[7] Es handelt sich hierbei um eine handlichere, aber äquivalente Formulierung des Sicherheitskriteriums aus [27].

Definition 3.1. *Sei π ein n-Parteien-Protokoll und $\mathcal{F}$ eine ideale Funktionalität. Für einen Angreifer $\mathcal{A}$ und festen Sicherheitsparameter $k \in \mathbb{N}$ bezeichne*

$$\mathsf{Pr}[\mathcal{Z} \to 1 \mid \pi, \mathcal{A}](k)$$

die Wahrscheinlichkeit, daß eine Umgebung $\mathcal{Z}$ bei Abläufen im realen Modell mit Protokoll π, Angreifer $\mathcal{A}$ und bei Sicherheitsparameter k eine 1 als Ausgabe generiere.

Analog bezeichne

$$\mathsf{Pr}[\mathcal{Z} \to 1 \mid \mathcal{F}, \mathcal{S}](k)$$

die Wahrscheinlichkeit für eine 1-Ausgabe von $\mathcal{Z}$ bei Abläufen im idealen Modell für $\mathcal{F}$ und mit einem Angreifer $\mathcal{S}$ und Sicherheitsparameter k.

Mit diesen Bezeichnungen sei

$$\mathbf{Adv}^{\pi,\mathcal{F}}_{\mathcal{A},\mathcal{S},\mathcal{Z}}(k) := |\mathsf{Pr}[\mathcal{Z} \to 1 \mid \pi, \mathcal{A}](k) - \mathsf{Pr}[\mathcal{Z} \to 1 \mid \mathcal{F}, \mathcal{S}](k)|$$

der Unterscheidungsvorteil *von $\mathcal{Z}$ bezüglich π, $\mathcal{F}$, $\mathcal{A}$ und $\mathcal{S}$.*

Es heiße π eine sichere Realisierung *von $\mathcal{F}$ (geschrieben $\pi \geq \mathcal{F}$) genau dann, wenn für alle $\mathcal{A}$ ein $\mathcal{S}$ existiert, so daß für alle $\mathcal{Z}$ die Funktion $\mathbf{Adv}^{\pi,\mathcal{F}}_{\mathcal{A},\mathcal{S},\mathcal{Z}}(k)$ vernachlässigbar in k ist.*

In der Originalformulierung aus [27] wird die Maschine $\mathcal{Z}$ (und nur diese Maschine) als *nicht-uniform* angenommen, was bedeutet, daß mit einer ITM $\mathcal{Z}$ immer auch eine Familie $\{z_k\}_{k\in\mathbb{N}}$ von initialen Eingaben für $\mathcal{Z}$ betrachtet wird. Hierbei ist $z_k \in \{0,1\}^*$ die initiale Eingabe von $\mathcal{Z}$ bei Protokollabläufen mit Sicherheitsparameter k. Im Falle einer Quantifizierung über alle Umgebungen $\mathcal{Z}$ – wie bei Definition 3.1 – ist dann auch eine Quantifizierung über alle Familien $\{z_k\}$ gemeint. Eine solche Definition sichert also sogar Ununterscheidbarkeit von realem und idealem Modell zu, selbst wenn (ggf. nicht-berechenbare) a-priori-Informationen bekannt sind.

Eine häufig benutzte Abwandlung von Definition 3.1 ist die *Sicherheit bezüglich nicht-adaptiver Angreifer*. Hierbei wird nur über sogenannte nicht-adaptive Angreifer $\mathcal{A}$ und $\mathcal{S}$ quantifiziert, welche Korruptionen nur in der ersten Aktivierung vornehmen.[8] Da $\mathcal{Z}$ über Korruptionen informiert wird, wird der so entstandene Sicherheitsbegriff schon von Sicherheit im Sinne von Definition 3.1 impliziert. Daß diese Implikation „echt“ ist zeigt ein Protokollbeispiel aus [26].

3.1.5 Der Dummy-Angreifer

Es sei für spätere Zwecke noch darauf hingewiesen, daß ein *vollständiger* Angreifer $\tilde{\mathcal{A}}$ existiert; es reicht also aus, in Definition 3.1 einen Simulator $\mathcal{S}$ zu fordern, welcher Angriffe von $\tilde{\mathcal{A}}$ simuliert. Die Beschaffenheit von $\tilde{\mathcal{A}}$ ist auf den ersten Blick überraschend: Informell beschrieben führt $\tilde{\mathcal{A}}$ nur Befehle von $\mathcal{Z}$ aus – der Angriff wird also faktisch von $\mathcal{Z}$ ausgeführt. Genauer akzeptiert $\tilde{\mathcal{A}}$ von $\mathcal{Z}$ Befehle

[8] Obwohl sprachlich ungenau, schließen die bisher kennengelernten *adaptiven* Angreifer auch *nicht-adaptive* Angreifer ein.

- zur Korruption einer Menge von Parteien (wonach $\tilde{\mathcal{A}}$ an $\mathcal{Z}$ auch den Zustand der korrumpierten Parteien meldet),
- zum Bericht über den Zustand der Ausgangs-Kommunikationsbänder der Parteien und
- zum Senden von Nachrichten an Parteien (auch im Namen schon korrumpierter Parteien).

Dieser *Dummy-Angreifer* genannte Angreifer wird in [27] als vollständig in obigem Sinne gezeigt; dieses Ergebnis ist essentiell für den Beweis des Kompositionstheorems in [27] (siehe auch Abschnitt 3.1.7).

Effektiv wird der Dummy-Angreifer – als technisches Hilfsmittel in noch folgenden Sicherheitsbeweisen, aber auch für den Beweis des Kompositionstheorems – genutzt, um einen Angriff in die Umgebung auszulagern, also letztlich von der Umgebung durchführen zu lassen. (Dies legt schon die obige Definition von $\tilde{\mathcal{A}}$ nahe.) Der Grund hierfür – bzw. für die Vollständigkeit von $\tilde{\mathcal{A}}$ – ist in der Reihenfolge der Quantoren in der Sicherheitsdefinition 3.1 zu suchen: Hier wird als *erstes* ein realer Angreifer $\mathcal{A}$ fixiert, und *zuletzt* die Umgebung $\mathcal{Z}$. Es wird also die Existenz eines Simulators $\mathcal{S}$ gefordert, welcher zwar von $\mathcal{A}$ abhängen darf, jedoch *universell* für alle Umgebungen $\mathcal{Z}$ sein muß. Gelingt es – etwa mittels des beschriebenen $\tilde{\mathcal{A}}$ –, den realen Angriff letztlich von $\mathcal{Z}$ durchführen zu lassen, so wird der damit verbundene Simulator als universell für alle möglichen Angriffe gefordert. Damit reicht es, die Sicherheit eines Protokolls gegen Angriffe des Dummy-Angreifers zu zeigen – dies stellt keinen Verlust an Allgemeinheit dar.

Hierzu ist anzumerken, daß der so in [27] beschriebene Dummy-Angreifer $\tilde{\mathcal{A}}$ *nicht* polynomial beschränkt ist, denn $\tilde{\mathcal{A}}$ hält nie. Damit stellt $\tilde{\mathcal{A}}$ keinen gültigen Angreifer dar. Insofern ist der Beweis des Kompositionstheorems in [27] unvollständig. Diese Unsauberkeit kann jedoch behoben werden, indem man zu einer über polynomiale Laufzeiten parametrisierten Menge von Dummy-Angreifern $\{\tilde{\mathcal{A}}_p\}_{p\in\mathbb{Z}[k]}$ übergeht, wie von Hofheinz und Müller-Quade in [82] beschrieben. Diese Angreifermenge ist – mit dem Argument aus [82, 27] – vollständig, und ein leicht geänderter Beweis des Kompositionstheorems analog zu dem Beweis aus [82] ist möglich.

Es kann nun angeführt werden, daß eine polynomiale Beschränkung des a priori unbeschränkten $\tilde{\mathcal{A}}$ faktisch mit der Fixierung einer Umgebung $\mathcal{Z}$ einhergeht, da $\mathcal{Z}$ ja polynomial beschränkt ist. Geht man aber generell zu nur bezüglich jeder festen Umgebung $\mathcal{Z}$ polynomial beschränkten Angreifern über, deren Laufzeit dann jeweils durch ein $\mathcal{Z}$-abhängiges Polynom $p_{\mathcal{Z}} \in \mathbb{Z}[k]$ beschränkt ist, so treten massive technische Probleme beim Beweis des Kompositionstheorems auf. Konkret ist unklar, wie sich mehrere, in einer im Verlauf des Beweises konstruierten Umgebung $\mathcal{Z}$ parallel simulierte Kopien eines Simulators bezüglich ihrer Laufzeit verhalten – man beachte, daß gezeigt werden muß, daß $\mathcal{Z}$ selbst polynomial beschränkt ist.

Probleme der Laufzeitbeschränkung von Angreifer und Umgebung werden noch häufiger auftreten und angesprochen werden. Zusammenfassend kann zunächst nur

gesagt werden, daß die ursprünglich in [27] formulierte Forderung nach einer strikten polynomialen Beschränkung von $\mathcal{A}$ (bzw. $\mathcal{S}$) ihre Tücken hat. Dies wird besonders deutlich im noch zu besprechenden Fall der idealen Funktionalität $\mathcal{F}_{\mathrm{PKE}}$ zum Ausdruck kommen; dort soll diese Diskussion wieder aufgenommen werden.

3.1.6 Das hybride Modell

Ein Protokoll, welches seinerseits eine ideale Funktionalität nutzt, läßt sich weder im realen noch im idealen Modell formulieren. Deshalb führt [27] – in gewisser Weise als Zwischenstufe zwischen realem und idealem Modell – das $\mathcal{F}$-*hybride* Modell ein, wobei $\mathcal{F}$ eine ideale Funktionalität ist. Der Protokollablauf im $\mathcal{F}$-hybriden Modell ist mit dem im realen Modell identisch, jedoch existieren zusätzlich zu den Parteien P_i, dem Angreifer $\mathcal{H}$ (hier mit $\mathcal{H}$ notiert wegen des Namens „*Hybrid*-Modell-Angreifer“) und der Umgebung $\mathcal{Z}$ noch unendlich viele Kopien $\mathcal{F}_{sid}$ ($sid \in \{0,1\}^*$) der ITM $\mathcal{F}$.

Auf diese Kopien können sowohl $\mathcal{H}$ als auch die Parteien P_i ähnlich wie im idealen Modell zugreifen. Das bedeutet, daß jede Partei Eingaben für beliebige $\mathcal{F}_{sid}$ geben und Ausgaben von beliebigen $\mathcal{F}_{sid}$ lesen darf. Zum Zwecke der Unterscheidung sind deshalb alle solchen Ein- und Ausgaben mit der *Sitzungsnummer sid* versehen. In ähnlicher Weise wird auch die Kommunikation zwischen den $\mathcal{F}_{sid}$ und $\mathcal{H}$ mit Sitzungsnummern gekennzeichnet.

Man beachte hier, daß $\mathcal{H}$ nicht darüber informiert ist, für welche Sitzungsnummern *sid* schon Funktionalitäten $\mathcal{F}_{sid}$ angesprochen wurden. Dies wird für $\mathcal{H}$ erst klar, wenn $\mathcal{H}$ eine Nachricht von einer Funktionalität $\mathcal{F}_{sid}$ erhält (dieser Nachricht ist dann nach Konvention die Sitzungsnummer *sid* vorangestellt), oder wenn $\mathcal{H}$ selbst eine mit einer Sitzungsnummer *sid* versehene Nachricht an $\mathcal{F}_{sid}$ versendet.

Diese Sitzungsnummern führen wieder zu einer Problematik des Modells [27]: In der Beschreibung aus [27] wird zwecks einfacherer Benutzung im hybriden Modell von einer idealen Funktionalität verlangt, daß sie nur Eingaben der Form $(sid, \ldots)$ akzeptiert, wobei *sid* eine Sitzungsnummer ist. Damit können ideale Funktionalitäten ohne Veränderung auch im hybriden Modell betrachtet werden – jedoch ist klar, daß dafür eine ideale Funktionalität immer nur Anfragen mit ein und derselben Sitzungsnummer akzeptieren darf.

In diesem Fall allerdings tritt ein Problem auf, welches in [27] nicht behandelt ist: Eine Umgebung, welche zwei Parteien mit Anfragen bezüglich *verschiedener* Sitzungsnummern aktiviert, zwingt ein entsprechendes reales Protokoll dazu, festzustellen, welche dieser Aktivierungen zuerst erfolgt ist.[9] Das kann bei asynchronem Netzwerk nicht bewerkstelligt werden; deshalb ist eine generelle Einschränkung auf Umgebungen $\mathcal{Z}$ nötig, welche in einem Protokolldurchlauf nur Eingaben $(sid, \ldots)$ mit derselben Sitzungsnummer *sid* geben. (Wobei natürlich in verschiedenen Protokolldurchläufen verschiedene Sitzungsnummern benutzt werden dürfen.)

[9] Nach Spezifikation in [27] darf eine ideale Funktionalität nur Eingaben mit der zuerst verwendeten Sitzungsnummer beachten.

Wie im idealen Modell obliegt es $\mathcal{H}$, Ausgaben von den $\mathcal{F}_{sid}$ zu den Parteien auszuliefern; allerdings hat auch hier $\mathcal{H}$ keine Informationen über den *Inhalt* dieser Ausgaben. In diesem Sinne stellt also das hybride Modell eine „Mischung“ aus realem und idealem Modell dar.

Analog zu Definition 3.1 sei nun $\mathsf{Pr}\left[\mathcal{Z}_k \to 1 \mid \pi^{\mathcal{F}}, \mathcal{H}\right]$ die Wahrscheinlichkeit, daß eine Umgebung $\mathcal{Z}$ im $\mathcal{F}$-hybriden Modell mit Protokoll π (wobei wie schon angedeutet auch die Bezeichnung $\pi^{\mathcal{F}}$ benutzt werden soll, um deutlich zu machen, daß π im $\mathcal{F}$-hybriden Modell ausgeführt wird), Angreifer $\mathcal{H}$ und bei einem Sicherheitsparameter $k \in \mathbb{N}$ eine 1 ausgibt. Auf offensichtliche Weise kann dann die „realisiert“-Relation „$\geq$“ für die Fälle $\tau \geq \pi^{\mathcal{F}}$ und $\pi^{\mathcal{F}} \geq \mathcal{G}$ mit einer weiteren idealen Funktionalität $\mathcal{G}$ verallgemeinert werden, indem man die Vernachlässigbarkeit der Funktionen

$$\mathbf{Adv}^{\tau,\pi^{\mathcal{F}}}_{\mathcal{A},\mathcal{H},\mathcal{Z}}(k) := \left|\mathsf{Pr}[\mathcal{Z} \to 1 \mid \tau, \mathcal{A}](k) - \mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \pi^{\mathcal{F}}, \mathcal{H}\right](k)\right|$$

bzw.

$$\mathbf{Adv}^{\pi^{\mathcal{F}},\mathcal{G}}_{\mathcal{H},\mathcal{S},\mathcal{Z}}(k) := \left|\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \pi^{\mathcal{F}}, \mathcal{H}\right](k) - \mathsf{Pr}[\mathcal{Z} \to 1 \mid \mathcal{G}, \mathcal{S}](k)\right|$$

fordert.

Es kann nun auf naheliegende Weise modelliert werden, was es heißt, eine Idealisierung $\mathcal{F}$ einer Protokollaufgabe durch ein tatsächliches (d. h. im realen Modell formuliertes) Protokoll ρ zu *ersetzen*. Ist nämlich π im $\mathcal{F}$-hybriden Modell formuliert, so können – syntaktisch – alle Anfragen an eine Funktionalität $\mathcal{F}_{sid}$ durch Anfragen an ein intern gestartetes Subprotokoll der Form ρ ersetzt werden. Kommunikation von Parteien innerhalb eines solchen ρ-Subprotokolls wird durch die entsprechende Sitzungsnummer *sid* gekennzeichnet. Protokollausgaben werden dann wie Ausgaben einer Funktionalität $\mathcal{F}_{sid}$ behandelt. Es ist klar, daß Subprotokolle – also Instanzen von ρ – nur bei Bedarf erzeugt werden. Das auf diese Weise kanonisch erzeugte reale Protokoll sei mit π^{ρ} bezeichnet.

In [27] wird auch angedeutet, wie *mehrere* Hybrid-Funktionalitäten $\{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ in einem hybriden Modell formuliert werden können: Es muß dazu eine einzige Funktionalität $\mathcal{F}$ konstruiert werden, welche intern jeweils eine Instanz der Funktionalitäten $\mathcal{F}_1, \ldots, \mathcal{F}_m$ simuliert. Eingaben und Ausgaben werden durch geeignete Präfixe $1, \ldots, m$ den einzelnen Funktionalitäten zugeordnet.

3.1.7 Das Kompositionstheorem

Die Frage drängt sich auf, wie es um die Sicherheit eines wie beschrieben aus π und ρ „komponierten“ Protokolls steht. Genauer: Angenommen, es gelte $\rho \geq \mathcal{F}$ und $\pi^{\mathcal{F}} \geq \mathcal{G}$. Mit anderen Worten, ρ realisiere $\mathcal{F}$, und π realisiere im $\mathcal{F}$-hybriden Modell eine Funktionalität $\mathcal{G}$. Bleibt dann die Sicherheit von π erhalten, wenn die idealen Aufrufe von $\mathcal{F}$ durch ρ-Subprotokollaufrufe ersetzt werden, gilt also $\pi^{\rho} \geq \mathcal{G}$? Intuitiv sollte die Sicherheit von π erhalten bleiben, da ja ρ „mindestens so sicher wie $\mathcal{F}$“ ist.

Allerdings ist der Erhalt von Sicherheitseigenschaften unter Komposition von Protokollen für „klassische“ Sicherheitsbegriffe im allgemeinen *nicht* gegeben, wie das eingangs zitierte Beispiel aus [72] zeigt. Glücklicherweise hat jedoch der in Definition 3.1 vorgestellte Sicherheitsbegriff eine Eigenschaft, welche sichere Komposition von Protokollen garantiert. In [27] wird – für den Fall nicht-uniformer Umgebungen $\mathcal{Z}$ – das folgende *Kompositionstheorem* gezeigt:

Satz 3.2 ([27]). *Sei ρ ein Protokoll im realen Modell und π ein Protokoll im $\mathcal{F}$-hybriden Modell für eine ideale Funktionalität $\mathcal{F}$. Dann impliziert $\rho \geq \mathcal{F}$ schon $\pi^\rho \geq \pi^\mathcal{F}$. Im speziellen Fall $\pi^\mathcal{F} \geq \mathcal{G}$ für eine ideale Funktionalität $\mathcal{G}$ folgt aus $\rho \geq \mathcal{F}$ damit insbesondere $\pi^\rho \geq \mathcal{G}$.*

Der Beweis in [27] für den Fall nicht-uniformer Umgebungen nutzt die Nicht-Uniformität (d. h. die „Zusatzeingabe“ z_k der Umgebung $\mathcal{Z}$); jedoch kann in [85] aufgezeigt werden, daß ein simples „Raten“ der Zusatzeingabe $z_k \in \{0, \ldots, m(k)\}$ für ein $\mathcal{Z}$-abhängiges Polynom m schon hinreicht, um den Beweis auf den Fall uniformer Umgebungen – also Umgebungen *ohne* Zusatzeingaben z_k – zu übertragen. Der Fall uniformer Umgebungen ist insbesondere für die Verbindung zu klassischen Sicherheitsbegriffen interessant, etwa für den im nächsten Abschnitt zu besprechenden Fall eines Public-Key-Kryptosystems.

Satz 3.2 ermöglicht es, Protokolle modular aufzubauen und auf einer höheren Abstraktionsebene mit Idealisierungen von Protokollbausteinen zu arbeiten. Die tatsächliche Realisierung dieser Protokollbausteine kann dann separat behandelt werden. Hier sei jedoch darauf hingewiesen, daß Satz 3.2 nur ein reales und ein hybrides Modell in Beziehung setzt – *nicht* aber zwei hybride Modelle.

Eine stärkere, aber *weder dort, noch in [27] bewiesene* Formulierung des Kompositionstheorems findet sich in [39]: Dort wird Satz 3.2 mit *hybridem* Protokoll ρ behauptet. Diese Formulierung ist für die Praxis wesentlich nützlicher als die ursprüngliche Form von Satz 3.2, da für die Realisierung von Protokollbausteinen oft weitere Hilfsfunktionalitäten zwingend erforderlich sind.[10]

Andererseits ist nicht ganz offensichtlich, wie die in [39] behauptete Form des Theorems überhaupt zu verstehen ist: Es müssen beim Übergang von Protokoll π zu Protokoll π^ρ viele Vorkommen von $\mathcal{F}$ ersetzt werden, wobei *jedes* so eingesetzte Unterprotokoll ρ wieder (evtl. viele) Instanzen einer anderen idealen Funktionalität benötigt. Dabei muß gewährleistet sein, daß die verschiedenen Unterprotokolle *disjunkte* Instanzenmengen von idealen Hilfsfunktionalitäten ansprechen; andernfalls können die Unterprotokolle nicht mehr als voneinander isoliert angesehen werden. Wie dies zu geschehen hat, legt [39] nicht fest.

3.2 Idealisierung eines Public-Key-Kryptosystems

Wie schon besprochen soll ein Ziel dieses Kapitels sein, eine intuitive Sicherheitsdefinition für ein Public-Key-Kryptosystem darzustellen und, soweit möglich, Ver-

[10] Für ein konkretes diesbezügliches Beispiel sei auf Abschnitt 4.2 verwiesen.

bindungen zu bestehenden, insofern „klassischen“, Sicherheitsdefinitionen zu ziehen. Nach dem Gesagten kann also versucht werden, eine *Idealisierung* eines Public-Key-Kryptosystems in Form einer idealen Funktionalität anzugeben.

3.2.1 Die Idealisierung

Man würde erwarten, daß eine solche Funktionalität zunächst zwischen dem Benutzer, der entschlüsseln können soll, und anderen Benutzern, welche nur verschlüsseln können sollen, unterscheidet. In einem konkreten Public-Key-Kryptosystem wird dies dadurch geschehen, daß nur der entschlüsselnde Nutzer Kenntnis über einen geheimen Schlüssel sk hat, während alle Teilnehmer des Systems mit einem öffentlichen Schlüssel pk Nachrichten verschlüsseln können. Tatsächlich entfällt im Falle einer idealen Funktionalität, welche schon von vornherein einen entschlüsselnden Nutzer von anderen Teilnehmern unterscheidet (und ihm als einigem die Möglichkeit, Chiffrate zu entschlüsseln, gewährt), die Notwendigkeit eines geheimen Schlüssels.

Andererseits sollte ein öffentlicher Schlüssel auch in der Idealisierung beibehalten werden: Nur bei *unverfälschtem* öffentlichem Schlüssel kann die Geheimhaltung einer verschlüsselten Nachricht garantiert werden. Man stelle sich hier ein real implementiertes Public-Key-Kryptosystem vor, bei dem der vom legitim entschlüsselnden Nutzer generierte öffentliche Schlüssel bei der Übertragung an andere Teilnehmer durch den öffentlichen Schlüssel eines Angreifers ausgetauscht wird. Hiermit generierte Chiffrate können natürlich von diesem Angreifer entschlüsselt werden. Ein solcher Angriff könnte aber mit einer Idealisierung eines Public-Key-Kryptosystems *ohne* öffentlichen Schlüssel nicht nachgebildet werden. Es obliegt somit dem entschlüsselnden Nutzer, dafür zu sorgen, daß andere Teilnehmer ihre Nachrichten mit *seinem* öffentlichen Schlüssel verschlüsseln. Für mit einem anderen öffentlichen Schlüssel chiffrierte Nachrichten sollte keine Geheimhaltung garantiert werden.

Weiter würde eine solche Funktionalität als *Abstraktion* eines Public-Key-Kryptosystems die Aufgaben des Verschlüsselns und Entschlüsselns für die Benutzer übernehmen, wobei wie besprochen natürlich nur genau einem Nutzer die Möglichkeit offensteht, Nachrichten zu entschlüsseln. Die Entschlüsselung eines Chiffrates, welches durch die Verschlüsselung einer Nachricht m zustandegekommen ist, sollte natürlich auch eben diese Nachricht m als Ergebnis haben, um der Korrektheitsforderung an ein Public-Key-Kryptosystem nachzukommmen. Dies könnte etwa durch eine Liste von Klartext-Chiffrat-Zuordnungen innerhalb der idealen Funktionalität sichergestellt werden. Mit einem ähnlichen Argument wie im Falle der öffentlichen Schlüssel sollten allerdings Entschlüsselungen von Chiffraten, welche *nicht* von der betrachteten Funktionalität produziert wurden, keine solche Korrektheitsgarantie gewähren. Konkret könnte man in einem solchen Fall den Angreifer entscheiden lassen, wie ein gegebenes, „illegitim generiertes“ Chiffrat zu entschlüsseln sei.

Hier stellt sich die Frage, wie eine abstrakte ideale Funktionalität, welche Chiffrate bei Verschlüsselungsanfragen ausgibt, überhaupt das Format dieser Chiffrate

bestimmt. Um die Realisierbarkeit durch irgendeinen Public-Key-Kryptosystem zu gewährleisten, würde man fordern, daß ideale Chiffrate dieselbe „Form“ haben wie reale Chiffrate eines „echten“ Public-Key-Kryptosystems. Genauer könnte man hier den Kunstgriff anwenden, im idealen Modell Chiffrate vom *Simulator* erzeugen zu lassen. Selbstverständlich dürfte dieser dann kein Wissen über die entsprechende Nachricht gewinnen – lediglich die Veröffentlichung der *Länge* der Nachricht scheint hier aus praktischen Gründen unvermeidlich.[11] Eine ähnliche Argumentation legt nahe, auch das Format des öffentlichen Schlüssels vom Angreifer festlegen zu lassen.

Die Arbeit der Formulierung einer solchen Funktionalität wurde schon in [27] bewerkstelligt, und zur Anschauung, aber auch für weitergehende Betrachtungen, sei in Abbildung 3.2 diese Idealisierung in Form der idealen Funktionalität $\mathcal{F}_{\text{PKE}}$ reproduziert.

Die Funktionalität $\mathcal{F}_{\text{PKE}}$

Protokollteilnehmer: $\mathcal{F}_{\text{PKE}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. Erwarte in der ersten Aktivierung einen Wert (`KeyGen`, *sid*) von einer Partei P_i. Dann:
   (a) Sende (`KeyGen`, *sid*) an den Angreifer.
   (b) Empfange einen Wert pk vom Angreifer, sende dann pk an P_i.
2. Bei Erhalt eines Wertes (`Encrypt`, sid, pk', m) von einer Partei P_j:
   (a) Sende (`Encrypt`, $sid, pk', |m|$) zum Angreifer, wobei $|m|$ die Länge von m angibt. (Falls $pk' \neq pk$ ist oder pk noch nicht definiert ist, gib zusätzlich den vollständigen Wert m zum Angreifer.)
   (b) Empfange ein „Tag“ c vom Angreifer, sende dann c an P_j. Falls dabei $pk' = pk$, speichere das Paar (c, m). (Wenn c schon in einem solchen gespeicherten Paar auftaucht, halte an.)
3. Bei Erhalt eines Wertes (`Decrypt`, sid, c) von P_i (und nur P_i):
   (a) Falls ein Paar (c, m) gespeichert wurde, sende m an P_i.
   (b) Andernfalls sende (`Decrypt`, sid, c) zum Angreifer, empfange einen Wert m vom Angreifer und gib dann m an P_i.

Abbildung 3.2: Die Idealisierung eines Public-Key-Kryptosystems aus [27]

Hier sei kurz auf einige Merkmale der Funktionalität $\mathcal{F}_{\text{PKE}}$ eingegangen. Zunächst enthalten wie schon besprochen alle Anfragen an $\mathcal{F}_{\text{PKE}}$ eine Sitzungsnummer *sid*.

[11] Daß dem Angreifer Wissen um die Klartextlänge zugestanden wird, ist in gewisser Weise ein „notwendiges Übel“, um Simulation zu ermöglichen; vgl. hierzu Goldreich [69, Appendix A].

Eine Partei P_i kann bei $\mathcal{F}_{\text{PKE}}$ durch eine `KeyGen`-Anfrage einen öffentlichen Schlüssel pk erfragen.[12] Dieser kann dann benutzt werden (von P_i oder von anderen Parteien), um mittels `Encrypt`-Anfragen Nachrichten zu verschlüsseln. Eine Entschlüsselung so generierter Chiffrate ist allerdings nur P_i (mittels einer `Decrypt`-Anfrage) möglich.

Es gibt wie schon besprochen in der vorliegenden Beschreibung offenbar keinen *geheimen* Schlüssel. Bei `KeyGen`-Anfragen an $\mathcal{F}_{\text{PKE}}$ werden lediglich *öffentliche* Schlüssel pk generiert. Ein geheimer Schlüssel ist für diese Idealisierung auch gar nicht nötig: Chiffrate werden durch $\mathcal{F}_{\text{PKE}}$ ver- und entschlüsselt. Hierzu wird – aus ebenfalls schon genannten Gründen – der Angreifer nach konkreten Chiffraten c und, für die Schlüsselgenerierung, nach einem öffentlichen Schlüssel pk gefragt.

Allerdings wird in der Beschreibung auch klar, daß der Angreifer über die Klartexte, zu denen er die Chiffrate liefern soll, keine Information außer der Klartextlänge erhält. Insofern ist die beschriebene Funktionalität $\mathcal{F}_{\text{PKE}}$ „hinreichend ideal“, wobei jedoch alle „Sicherheitslücken“, wie etwa die Herausgabe der Länge $|m|$ eines Klartextes m, *explizit erwähnt* sind.

Man beachte auch, daß Entschüsselungen „im Normalfall“ über eine Liste durchgeführt werden. Damit ist gemeint, daß $\mathcal{F}_{\text{PKE}}$ Chiffrate c, welche durch eine Chiffrierung eines Klartextes m *mittels* $\mathcal{F}_{\text{PKE}}$ gewonnen wurden, wieder zu m entschlüsselt. Lediglich Chiffrate, welche nicht durch `Encrypt`-Anfragen an $\mathcal{F}_{\text{PKE}}$ generiert wurden, werden vom Angreifer entschlüsselt.

Bemerkenswerterweise werden beliebige Klartexte $m \in \{0,1\}^*$ zur Verschlüsselung akzeptiert, obwohl bei vielen gängigen Public-Key-Kryptosystemen die Klartextmenge $\mathcal{M}_{pk}$ im allgemeinen eine echte Untermenge von $\{0,1\}^*$ ist. Man beachte dabei, daß die naheliegende Änderung eines Public-Key-Kryptosystems durch blockweise Verschlüsselung zu einem System mit $\mathcal{M}_{pk} = \{0,1\}^*$ zu einem Verlust von Sicherheitseigenschaften führen kann.[13]

Eine sich aufdrängende diesbezügliche Anpassung von $\mathcal{F}_{\text{PKE}}$ ist nun, den Simulator nicht nur den öffentlichen Schlüssel pk wählen zu lassen, sondern dabei auch eine passende Klartextmenge $\mathcal{M}_{pk}$ der „erlaubten“ Klartexte. Bei gleichzeitiger derartiger Anpassung von $\mathcal{F}_{\text{PKE}}$ und des später noch benutzten klassischen Sicherheitsbegriffes ROR-CCA (vgl. Bellare u. a. [16]) bleiben die noch vorzustellenden Resultate gültig; jedoch soll hier der Einfachheit halber, und um den Begriff ROR-CCA „unangetastet“ zu lassen, die Funktionalität $\mathcal{F}_{\text{PKE}}$ diesbezüglich nicht verändert werden.

Es sei noch darauf hingewiesen, daß die in der späteren Arbeit Canetti u. a. [37] benutzte Parametrisierung von $\mathcal{F}_{\text{PKE}}$ über eine Familie $\mathcal{D} = \{D_k\}_{k\in\mathbb{N}}$ von Mengen „erlaubter Klartexte“ noch *nicht* optimal ist – man denke etwa an viele RSA-basierte Systeme, welche Nachrichten aus einer *Public-Key-abhängigen* Klar-

[12] Hier wird die Bezeichnung e aus [27] für den öffentlichen Schlüssel des Public-Key-Kryptosystems ersetzt durch die in Definition 2.1 benutzte Bezeichnung pk ersetzt.

[13] Eine etwas aufwendigere allgemeine Konstruktion, welche unter Benutzung eines symmetrischen Kryptosystems $\mathcal{M}_{pk} = \{0,1\}^*$ erbringt, *ohne* dabei eine IND-CCA-Sicherheit des Public-Key-Kryptosystems zu opfern, findet sich in Cramer und Shoup [50].

textmenge verschlüsseln.

Da die Funktionalität $\mathcal{F}_{\text{PKE}}$ letztlich eine Idealisierung von lokal auszuführenden *Algorithmen* darstellt, ist wünschenswert, daß Anfragen direkt und ohne Verzögerung beantwortet werden. In diesem Zusammenhang sind damit nur Simulatoren gewünscht, welche Ausgaben von $\mathcal{F}_{\text{PKE}}$ ohne Verzögerung an die Dummy-Parteien ausliefern. Konkret wird dies in [27] dadurch modelliert, daß im idealen (und im hybriden) Modell nur Simulatoren zugelassen werden, welche spezielle Befehle von $\mathcal{F}_{\text{PKE}}$ zum Anlaß nehmen, die jeweils generierte Ausgabe von $\mathcal{F}_{\text{PKE}}$ sofort an die entsprechende Partei auszuliefern. Man spricht hier von einer *direkten* (engl. „immediate“) Funktionalität $\mathcal{F}_{\text{PKE}}$. In der ursprünglichen und hier wiedergegebenen Beschreibung von $\mathcal{F}_{\text{PKE}}$ fehlen diese expliziten Anweisungen; aus dem umgebenden Text kann jedoch geschlossen werden, daß diese Nachrichten automatisch von $\mathcal{F}_{\text{PKE}}$ an den Simulator gesendet werden.

Die aus [27] entnommene und in Abbildung 3.2 wiedergegebene Beschreibung von $\mathcal{F}_{\text{PKE}}$ ist von informeller Natur. Schon in [85] wird beispielsweise angemerkt, daß die gegebene Beschreibung unklar läßt, was im Falle mehrerer `KeyGen`-Anfragen von unterschiedlichen Parteien passiert. Ein Ignorieren aller `KeyGen`-Anfragen bis auf die erste solche Anfrage ist ähnlich problematisch wie Eingaben mit verschiedenen Sitzungsnummern: Ein $\mathcal{F}_{\text{PKE}}$ realisierendes reales Protokoll müßte dann – trotz asynchronem Netzwerk – in der Lage sein, festzustellen, welche Partei als erste eine `KeyGen`-Eingabe bekommen hat. Dies führt zu einer Nicht-Realisierbarkeit einer solchermaßen ausgelegten Beschreibung der Funktionalität $\mathcal{F}_{\text{PKE}}$.

Eine Lösung dieser Problematik wird in [85] vorgeschlagen: Dort wird angeregt, Public-Key-Kryptosysteme durch eine Familie $\{\mathcal{F}_{\text{PKE}}^{(i)}\}_{P_i}$ zu modellieren, deren Elemente ideale Funktionalitäten sind, für welche jeweils festliegt, welche Partei P_i als einzige in der Lage sein soll, ein Schlüsselpaar zu generieren und Chiffrate zu entschlüsseln. Dieser Ansatz soll hier verfolgt werden.

Alternativ wird in der unabhängigen Arbeit [37] vorgeschlagen, einfach alle Anfragen an $\mathcal{F}_{\text{PKE}}$, welche nicht mit dem „erstgenerierten“ öffentlichen Schlüssel zusammenhängen, an den Angreifer weiterzuleiten und von ihm beantworten zu lassen. Damit geht natürlich jede Sicherheitsgarantie für diese Anfragen verloren. Dieser Ansatz ist problematisch, denn ein Angreifer kann nun eine Partei korrumpieren, durch sie eine `KeyGen`-Anfrage an $\mathcal{F}_{\text{PKE}}$ stellen und damit alle weiteren, durch „legitime“ `KeyGen`-Anfragen generierten Schlüssel praktisch von jeder Sicherheitsgarantie befreien.

3.2.2 Zwischenspiel: Laufzeitbeschränkungen

Im Zusammenhang mit der Diskussion von $\mathcal{F}_{\text{PKE}}$ (bzw. $\mathcal{F}_{\text{PKE}}^{(i)}$) sei hier noch ein besonders prekäres Thema angesprochen. Es wurde schon in den Abschnitten 3.1.1 und 3.1.5 – noch ohne konkretes Beispiel – bemerkt, daß eine Forderung nach polynomial beschränkten Angreifern (also insbesondere Simulatoren) problematisch ist. Am Beispiel von $\mathcal{F}_{\text{PKE}}^{(i)}$ kann dies nun konkretisiert werden.

Es ist nämlich $\mathcal{F}_{\text{PKE}}^{(i)}$ bei polynomial beschränkten Simulatoren generell nicht realisierbar. Das kann so eingesehen werden: Bei beliebigem realem Angreifer und Simulator kann eine Umgebung $\mathcal{Z}$ so formuliert werden, daß $\mathcal{Z}$ zunächst den Simulator $\mathcal{S}$ durch sinnlose Anfragen aktiviert, bis $\mathcal{S}$ gehalten haben muß. (Man beachte hier, daß $\mathcal{Z}$ von $\mathcal{S}$ abhängt und somit dessen Laufzeitschranke „kennt“.) Jede anschließende `KeyGen`-Anfrage hat im realen Modell aufgrund der Direktheitseigenschaft von $\mathcal{F}_{\text{PKE}}^{(i)}$ einen generierten Schlüssel zur Folge, während im idealen Modell mangels aktivierbarem Simulator keine Ausgabe ausgeliefert werden kann.

Unglücklicherweise ist diese Art von Angriff auch auf andere Funktionalitäten übertragbar; das auftretende Problem ist genereller Natur. Umgekehrt treten zudem, wie in Abschnitt 3.1.5 diskutiert, massive technische Probleme beim Beweis des Kompositionstheorems auf, sobald Angreifer nicht als polynomial beschränkt vorausgesetzt werden können. Eine Lösung dieses Dilemmas ist Gegenstand laufender Forschung. Hier soll ein Ausweg aufgezeigt werden, welcher auch in Backes [5] im verwandten Modell [117, 11] eingesetzt wurde, um eine ähnliche Problematik anzugehen.

Genauer kann einem Angreifer (bzw. Simulator) die Möglichkeit eingeräumt werden, Nachrichten von Maschinen seiner Wahl zu blockieren; damit werden alle Nachrichten dieser Maschine unterdrückt, und der Angreifer wird insbesondere nicht einmal mehr aktiviert. Das Scheduling wird dann weiterverfolgt, als ob der Angreifer aktiviert worden wäre, aber sofort einen Wartezustand eingenommen hätte. Diese Idee wurde auch in dem [27] sehr ähnlichen Modell [82] verwendet und soll später – für dieses Modell – noch genauer besprochen werden (vgl. Abschnitt 5.1).

Für das momentan zu behandelnde Modell [27] jedoch soll trotz der schon angesprochenen Probleme im Beweis des Kompositionstheorems vereinfachend angenommen werden, daß Angreifer (also insbesondere Simulatoren) lediglich polynomial beschränkt im Angesicht eines konkreten Protokolls und einer Umgebung sein müssen;[14] dies geschieht, um keine tiefgreifenden Änderungen im Modell [27] selbst einzuführen.

Man beachte, daß es hier *nicht* ausreicht, eine polynomiale Laufzeitbeschränkung in jeder Aktivierung eines Angreifers zu fordern: Mit einer solchen Konvention kann bei geeignet „ungeschickter“ Formulierung einer idealen Funktionalität $\mathcal{F}$ (welche beispielsweise die erste Anfrage des Angreifers an ihn zurücksendet) im $\mathcal{F}$-hybriden Modell eine Situation herbeigeführt werden, in der sich der Angreifer durch wahllose Anfragen an immer neue $\mathcal{F}$-Instanzen eine insgesamt unbeschränkte Laufzeit erschleichen kann.

Ein ähnliches Problem stellt sich im Zusammenhang mit $\mathcal{F}_{\text{PKE}}^{(i)}$ für die Protokollteilnehmer P_i und die ideale Funktionalität: $\mathcal{F}_{\text{PKE}}$ (und damit $\mathcal{F}_{\text{PKE}}^{(i)}$) ist gemäß der obigen Beschreibung *nicht* polynomial beschränkt. Hiermit kann auch nicht auf ein reales Protokoll mit polynomial beschränkten Parteien gehofft werden, welches

[14] Damit ist gemeint, daß ein Angreifer $\mathcal{A}$ für feste Umgebung $\mathcal{Z}$ nie länger als $p(k)$ Turing-Schritte läuft, wobei $p = p_{\mathcal{Z}} \in \mathbb{Z}[k]$ von $\mathcal{Z}$ abhängen darf – es wird dabei nicht gefordert, daß $\mathcal{A}$ jemals anhält.

$\mathcal{F}_{\text{PKE}}^{(i)}$ realisiert. Andererseits ist gerade eine polynomiale Beschränkung von Funktionalität und Parteien essentiell für die Gültigkeit des Kompositionstheorems.[15]

Eine künstlich erzwungene „polynomiale Einschränkung“ von $\mathcal{F}_{\text{PKE}}^{(i)}$ bzw. einem als Protokoll aufgefaßten Kryptosystem ist auch problematisch: Hier kann eine hinreichend mächtige Umgebung die Funktionalität $\mathcal{F}_{\text{PKE}}^{(i)}$ durch wiederholte P_1-Anfragen zum Anhalten bringen; spätere Anfragen bei einer anderen Partei P_2 werden nun genau im realen Modell beantwortet. Dem kann zwar oberflächlich durch eine künstliche Beschränkung der Dummy-Parteien ausgewichen werden; dabei können jedoch im Allgemeinfall unerwartete Effekte im $\mathcal{F}$-hybriden Modell (welches ja keine Dummy-Parteien kennt, vgl. Abschnitt 3.1.6) auftreten. Hier erhält $\mathcal{F}$ potentiell mehr und längere Nachrichten, da $\mathcal{F}$ nicht mehr durch beschränkte Dummy-Parteien „abgeschirmt“ wird. Damit kann die Funktionalität $\mathcal{F}$ selbst prinzipiell unterscheiden, ob sie in einem hybriden oder einem idealen Modell ausgeführt wird. (Man denke an eine Funktionalität $\mathcal{F}$, welche genau im hybriden Modell wichtige Geheimnisse ausplaudert.) Dies zeigt, daß bei diesem Ansatz Sicherheitseigenschaften unter Komposition verloren gehen können.

Glücklicherweise kann aber im *speziellen* Fall von $\mathcal{F}_{\text{PKE}}^{(i)}$, einem als reales Protokoll aufgefaßtem Kryptosystem *und* der später noch anzugebenden Simulatoren mittels der in Hofheinz, Müller-Quade und Unruh [87] entwickelten Techniken eingesehen werden, daß das Kompositionstheorem selbst seine Gültigkeit behält (*nicht* aber der in [27] angegebene allgemeine Beweis). Ähnliches gilt für die später noch zu betrachtenden Funktionalitäten und Protokolle (in Kombination mit den konkret anzugebenden Simulatoren), so daß im folgenden nicht weiter auf das Fehlen einer polynomialen Beschränkung bei Protokollen und Funktionalitäten eingegangen werden soll.

Addendum Nach Fertigstellung vorliegender Arbeit wurde die erweiterte Fassung [27] von [28] erneut überarbeitet und am 6. Januar 2005 unter der Adresse `http://eprint.iacr.org/2000/067/` veröffentlicht. In dieser jüngsten Überarbeitung wurde mit einem neuen Ansatz versucht, der gerade besprochenen Problematik der Laufzeitbeschränkungen Herr zu werden. Genauer wurde eine sogenannte *Kontrollfunktion* eingeführt, welche einen Protokolldurchlauf nach polynomial vielen Schritten – gewissermaßen gewaltsam – beendet. Bei einem Vergleich etwa eines realen Protokolls und einer idealen Funktionalität werden dabei reales und ideales Modell genau gleichermaßen „zugeschnitten“; es wird also bei einem Vergleich genau dieselbe Kontrollfunktion verwendet.

Dies bewirkt, daß eine exakt übereinstimmende Laufzeit in realem und idealem Modell gefordert wird – es wird also eine ganz neue Klasse von „Timing“-Angriffen zugelassen, welche die Unterscheidung von Protokollen allein aufgrund unterschiedlicher Laufzeiten zulassen. Eine Behebung derartiger „unerwünschter

[15] Eine Untersuchung von schwächeren Bedingungen, welche dennoch Komposition ermöglichen, findet in Hofheinz, Müller-Quade und Unruh [87] für das Mehrparteienmodell [117, 11] statt.

Nebenwirkungen“ des Einsatzes einer Kontrollfunktion ist Gegenstand augenblicklicher Forschung.

3.2.3 Anwendung für sichere Nachrichtenübermittlung

Die Nützlichkeit der gegebenen Idealisierung $\mathcal{F}_{\text{PKE}}^{(i)}$ kann demonstriert werden, indem behelfs ihrer eine sichere Nachrichtenübertragung implementiert wird. Dabei wird eine „natürliche“ Benutzung eines Public-Key-Kryptosystems zugrundegelegt, d. h. eine zu übertragende Nachricht wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und dann geschickt. Ver- und Entschlüsselung erfolgen unter Benutzung von $\mathcal{F}_{\text{PKE}}^{(i)}$.

Konkreter sei von vonherein eine Partei P_i bestimmt – der „Empfänger“ –, welche bei ihrer ersten Aktivierung mittels einer `KeyGen`-Anfrage an $\mathcal{F}_{\text{PKE}}^{(i)}$ einen öffentlichen Schlüssel pk generiert und diesen an alle anderen Parteien P_j $(j \neq i)$ verschickt. Damit sind natürlich nur Nachrichtenübermittlungen von Parteien P_j $(j \neq i)$ zu P_i möglich. Eine solche Übermittlung etwa von $m \in \{0,1\}^*$ von P_j zu P_i erfolgt dabei durch ein `Encrypt`-en von m bezüglich des anfangs von P_i erhaltenen Schlüssels pk. Ist noch kein pk empfangen, so findet keine Nachrichtenübermittlung statt.

Sei π_i nun dieses naheliegende Protokoll zur sicheren Nachrichtenübertragung, welches wie besprochen die idealisierte Form $\mathcal{F}_{\text{PKE}}^{(i)}$ eines Public-Key-Kryptosystems nutzt, also im $\mathcal{F}_{\text{PKE}}^{(i)}$-hybriden Modell formuliert ist. Dann zeigt ein Beweis in [27], daß π_i bei Annahme authentifizierter Kanäle[16] eine jeweilige ideale Funktionalität $\mathcal{F}_{\text{M-SMT}}^{(i)}$ realisiert.

Die Funktionalität $\mathcal{F}_{\text{M-SMT}}^{(i)}$

Protokollteilnehmer: $\mathcal{F}_{\text{M-SMT}}^{(i)}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. Bei Erhalt eines Wertes $(\texttt{Receiver}, sid)$ von P_i (und nur P_i): Sende $(\texttt{Receiver}, sid, P_i)$ an alle Parteien und den Angreifer. Ignoriere alle nachfolgenden $(\texttt{Receiver}, sid)$-Anfragen.

2. Bei Erhalt von $(\texttt{Send}, sid, m)$ von einer beliebigen Partei P_j: Sende (sid, m, P_j) an den Empfänger P_i und $(sid, |m|, P_j)$ an den Angreifer.

Abbildung 3.3: Die Idealisierung sicherer Nachrichtenübermittlung mit mehreren Sendern und festgelegtem Empfänger, aus [27] übernommene und angepaßte Fassung

Hierbei erlaubt $\mathcal{F}_{\text{M-SMT}}^{(i)}$ (wiedergegeben in Abbildung 3.3) allen Parteien, Nach-

[16] Es sei daran erinnert, daß es dem Angreifer im Falle authentifizierter Kanäle nicht erlaubt ist, Absender von Nachrichten zu fälschen.

richten sicher an P_i zu schicken – lediglich die Länge der übertragenen Nachricht wird dem Angreifer übermittelt. Da die Geheimhaltung von Nachrichten zwischen Parteien und idealen Funktionalitäten schon durch die Konstruktion des Modells selbst gewährleistet wird, kann somit $\mathcal{F}_{\text{M-SMT}}^{(i)}$ also als einfaches „Relais“ formuliert werden, welches nur alle Eingaben an P_i weiterleitet.

Man beachte dabei, daß [27] tatsächlich nur $\pi^{\mathcal{F}_{\text{PKE}}} \geq \mathcal{F}_{\text{M-SMT}}$ für ein Protokoll π, bei welchem der Empfänger *nicht* von vornherein festgelegt ist, behauptet. Da jedoch der Gebrauch von $\mathcal{F}_{\text{PKE}}$ in der ursprünglichen Form aus [27] wie schon besprochen problematisch ist, ist eine Aussage $\pi_i^{\mathcal{F}_{\text{PKE}}^{(i)}} \geq \mathcal{F}_{\text{M-SMT}}^{(i)}$ wünschenswerter. Glücklicherweise erschließt sich diese Aussage aber direkt aus dem Beweis von $\pi^{\mathcal{F}_{\text{PKE}}} \geq \mathcal{F}_{\text{M-SMT}}$ in [27].

Es können also die Funktionalitäten $\mathcal{F}_{\text{PKE}}^{(i)}$ genutzt werden, um in einem intuitiven und dennoch formalen Sinn Nachrichten zu übertragen. Damit ist eine Rechtfertigung für die Idealisierung $\mathcal{F}_{\text{PKE}}^{(i)}$ geschaffen, und es lohnt sich, zu untersuchen, *welche* Public-Key-Kryptosysteme dieser Idealisierung genügen (d. h. welche Public-Key-Kryptosysteme – als Protokolle aufgefaßt – $\mathcal{F}_{\text{PKE}}^{(i)}$ realisieren). Ist erst einmal ein Public-Key-Kryptosystem als sicher in diesem Sinne identifiziert, kann es dann mittels π_i zur sicheren Übertragung von Nachrichten – etwa in einem größeren Protokoll – genutzt werden. Das Kompositionstheorem stellt dann sicher, daß diese reale Implementierung gerade so sicher ist wie eine durch $\mathcal{F}_{\text{M-SMT}}$ idealisierte Nachrichtenübertragung.

Im folgenden soll deshalb nach einer Klasse von Public-Key-Kryptosystemen gesucht werden, welche $\mathcal{F}_{\text{PKE}}^{(i)}$ realisieren.

Abschließend sei noch darauf hingewiesen, daß [37] zeigt, daß auch eine abgeschwächte Variante $\mathcal{F}_{\text{RPKE}}$ von $\mathcal{F}_{\text{PKE}}$ schon hinreicht, um $\mathcal{F}_{\text{M-SMT}}$ zu realisieren. Da die Abschwächung $\mathcal{F}_{\text{RPKE}}$ aber weniger intuitiv als $\mathcal{F}_{\text{PKE}}$ selbst ist (und zudem das schon angesprochene Problem mehrfacher `KeyGen`-Eingaben nicht zufriedenstellend löst), soll sie hier nicht als Maßstab für „universell komponierbare Public-Key-Verschlüsselung“ weiter untersucht werden.

3.2.4 Real-or-Random-Sicherheit

Als Hilfsmittel wird hier auf eine – nicht durchweg gängige – „klassische“ Sicherheitsdefinition für Public-Key-Kryptosysteme aus [16] zurückgegriffen. Informell verlangt dieses „Real-or-Random-Sicherheit“ genannte Kriterium, daß Chiffrate zu selbstgewählten Klartexten nicht unterscheidbar sind von Chiffraten zu zufällig gewählten Klartexten. Genauer ergibt sich folgende, aus [16] entnommene und für den Fall von *Public*-Key-Kryptosystemen und in hiesiger Schreibweise formulierte Definition[17]:

[17] [16] fokussiert Sicherheitsdefinitionen für *symmetrische* Kryptosysteme; jedoch übertragen sich (wie schon in [16] angemerkt) alle Definitionen und Ergebnisse mühelos auf den asymmetrischen Fall.

Definition 3.3 ([16]). *Sei* $P = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *ein Public-Key-Kryptosystem,* $b \in \{0,1\}$ *und* $k \in \mathbb{N}$. *Für* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ *wird das* Real-or-Random-Orakel $\mathsf{RR}^b_{pk}(\cdot)$ *definiert, welches bei Eingabe* $m \in \{0,1\}^*$ *wie folgt verfährt:*

- *Für* $b = 1$ *berechnet* RR^b_{pk} *das Chiffrat* $c \leftarrow \mathsf{Encrypt}(1^k, pk, m)$ *und gibt* c *zurück.*
- *Ist* $b = 0$, *so zieht* RR^b_{pk} *gleichverteilt ein Element* $r \leftarrow \{0,1\}^{|m|}$, *chiffriert* $c \leftarrow \mathsf{Encrypt}(1^k, pk, r)$ *und gibt* c *zurück.*[18]

Sei A *ein Angreifer (d. h. ein PPT-Algorithmus) mit Zugriff – im Sinne von Notation A.8) – auf die Orakel* $\mathsf{RR}^b_{pk}(\cdot)$ *und* $\mathsf{Decrypt}(1^k, sk, \cdot)$. *Man betrachte das folgende Experiment*[19]

$$
\begin{array}{l}
\textit{Experiment } \mathbf{Exp}^{\text{ror-cca-b}}_{P,A}(k)\textit{:} \\
\quad (pk, sk) \leftarrow \mathsf{KeyGen}(1^k) \\
\quad \tilde{b} \leftarrow A^{\mathsf{RR}^b_{pk}(\cdot), \mathsf{Decrypt}(1^k, sk, \cdot)}(1^k, pk) \\
\quad \textbf{Return } \tilde{b}
\end{array}
$$

Hierbei wird verlangt, daß A *das Orakel* $\mathsf{Decrypt}(1^k, sk, \cdot)$ *nie mit Ausgaben von* $\mathsf{RR}^b_{pk}(\cdot)$ *startet.*

Damit sei der Unterscheidungsvorteil *von* A *definiert als*

$$
\mathbf{Adv}^{\text{ror-cca}}_{P,A}(k) := \mathsf{Pr}\left[\mathbf{Exp}^{\text{ror-cca-1}}_{P,A}(k) \to 1\right] - \mathsf{Pr}\left[\mathbf{Exp}^{\text{ror-cca-0}}_{P,A}(k) \to 1\right].
$$

Das Schema P *wird* ROR-CCA-sicher *genannt, wenn die Funktion* $\mathbf{Adv}^{\text{ror-cca}}_{P,A}(\cdot)$ *für jeden in vorstehendem Sinne erlaubten Angreifer* A *vernachlässigbar ist.*

Man beachte, daß diese Definition einen Chosen-Ciphertext-Angriff zugrundelegt, also dem Angreifer ein Entschlüsselungsorakel zur Verfügung stellt. Damit werden Situationen modelliert, in denen ein Angreifer beispielsweise in ein Kommunikationsnetzwerk Chiffrate injiziert und vom Empfänger entschlüsseln lassen kann. Die in Definition 3.3 formalisierte Form dieses Angriffs ist eine besonders starke – die Benutzung des Entschlüsselungsorakels wird lediglich minimal eingeschränkt, um eine nicht-erfüllbare Definition zu vermeiden. Hierzu sei noch abschließend angemerkt, daß durch Weglassen des Entschlüsselungsorakels in Definition 3.3 ein (echt schwächerer) Sicherheitsbegriff entsteht, welcher in [16] „ROR-CPA“ genannt wird und Sicherheit bei einem Chosen-Plaintext-Angriff modelliert.

Obwohl diese Definition künstlich wirkt, kann sie in [16] als äquivalent zum etablierten Sicherheitsbegriff „IND-CCA“ („indistinguishability under adaptive chosen-ciphertext attacks“, vgl. [121]) gezeigt werden. Bei Sicherheit bezüglich dieses Begriffs existiert kein Angreifer A, welcher ein Chiffrat einem von zwei zuvor von A gewählten Klartexten gleicher Länge zuordnen kann. Hierbei steht A ähnlich wie

[18] Es wird implizit $m \in \mathcal{M}_{pk} \Rightarrow \{0,1\}^{|m|} \subseteq \mathcal{M}_{pk}$ für die Klartextmenge $\mathcal{M}_{pk}$ unterstellt.

[19] Hier bezeichnet – wie in [16] – „Experiment“ einen Algorithmus, der intuitiv den Angreifer A einem „Test“ unterzieht.

im oben beschriebenen Fall von ROR-CCA ein Entschlüsselungsorakel zur Verfügung, welches lediglich nicht auf das zuzuordnende Chiffrat angewandt werden darf. (Auch hier ergibt sich ohne diese Einschränkung ein trivialer, weil nicht-erfüllbarer Sicherheitsbegriff.) Es kann dann analog zu Definition 3.3 einem Angreifer A eine Unterscheidungswahrscheinlichkeit $\mathbf{Adv}_{P,A}^{\text{ind-cca}}(k)$ zugeordnet werden. Formal ist ein Public-Key-Kryptosystem genau dann *IND-CCA-sicher*, wenn für alle Angreifer (d. h. PPT-Algorithmen) A die Funktion $\mathbf{Adv}_{P,A}^{\text{ind-cca}}(k)$ vernachlässigbar ist.

Satz 3.4 ([16]). *Ein Public-Key-Kryptosystem P ist IND-CCA-sicher genau dann, wenn P ROR-CCA-sicher ist.*

In [16] wird diese Äquivalenz durch Reduktionen zwischen ROR-CCA-Angreifern und IND-CCA-Angreifern gezeigt. Interessant hierbei ist, daß bezüglich dieser Reduktionen der Begriff „ROR-CCA“ in folgendem Sinne stärker ist als „IND-CCA“: Ein ROR-CCA-Angreifer kann umgewandelt werden in einen IND-CCA-Angreifer etwa gleicher Komplexität und mit gleicher Unterscheidungswahrscheinlichkeit, während ein IND-CCA-Angreifer „nur“ Anlaß zu einem ROR-CCA-Angreifer gleicher Komplexität, aber mit polynomial (in k) kleinerer Unterscheidungswahrscheinlichkeit gibt.[20] Dies widerspricht natürlich in keiner Weise der Äquivalenz von IND-CCA und ROR-CCA, da hier nur entscheidend ist, ob der Vorteil eines Angreifers (als Funktion in k) vernachlässigbar oder nicht-vernachlässigbar ist.

Die allgemeine Akzeptanz des auf den ersten Blick technisch und nicht-intuitiv wirkenden Begriffs „IND-CCA“ läßt sich wie folgt begründen. In der richtungsweisenden Arbeit [75] konnte mit „polynomialer Sicherheit“ ein handliches (wenngleich unintuitives) Kriterium für Public-Key-Kryptosysteme als hinreichend für den sehr intuitiven Begriff der semantischen Sicherheit gezeigt werden. „Polynomiale Sicherheit“ ist nun aber nichts anderes als der Begriff „IND-CPA“, also der Begriff „IND-CCA“ mit einem eingeschränkten Angreifer, welchem kein Entschlüsselungsorakel zur Verfügung steht. Für „semantische Sicherheit“ hingegen darf es keine Funktion f geben, für die *mit* einem Chiffrat zu einem Klartext m der Funktionswert $f(m)$ effizient besser approximiert werden kann als *ohne* Chiffrat. Anders gesagt: Ein Chiffrat hilft bei keiner effizient durchführbaren Berechnung auf dem Klartext.

Nun konnte in [55] gezeigt werden, daß leider semantische Sicherheit allein nicht hinreichend ist, um den Einsatz des betreffenden Public-Key-Kryptosystems in größeren Protokollen – etwa Auktionsprotokollen – zu gewährleisten. Es lag nun nahe (siehe [110, 121]), den technisch handhabbaren Begriff der polynomialen Sicherheit zu verschärfen, indem man dem Angreifer ein Entschlüsselungsorakel zubilligt – womit eine Formulierung des Begriffs IND-CCA geschaffen war. Unabhängig davon kann natürlich auch der Begriff der semantischen Sicherheit verschärft werden. Naheliegende Varianten werden beispielsweise in [16], Watanabe u. a. [137] betrachtet und als äquivalent zu IND-CCA gezeigt.

[20] Dies ist kein Artefakt der Beweistechnik in [16], wie [16, Proposition 5] zeigt.

Trotzdem ist die so geschaffene Situation nicht befriedigend: Wünschenswert ist ein an der *Verwendung* des Public-Key-Kryptosystems orientierter Begriff. Dieser konnte – wie im vorigen Abschnitt diskutiert – mit der Idealisierung $\mathcal{F}_{\text{PKE}}^{(i)}$ gefunden werden. Deshalb soll nun untersucht werden, in welchem Verhältnis $\mathcal{F}_{\text{PKE}}^{(i)}$ zu bestehenden Sicherheitsdefinitionen steht.

3.2.5 Äquivalenz zu komponierbarer Verschlüsselung

Ein Public-Key-Kryptosystem $P = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ soll dafür als Protokoll aufgefaßt werden können, welches potentiell $\mathcal{F}_{\text{PKE}}^{(i)}$ realisiert. Dazu wird ein Protokoll $\pi_P^{(i)}$ konstruiert, in dem jede Partei `Encrypt`-Anfragen mittels des Algorithmus $\mathsf{Encrypt}$ beantwortet. Weiter beantworte die dedizierte Partei P_i (der „Empfänger“) die erste `KeyGen`- und alle `Decrypt`-Anfragen durch Benutzen der Algorithmen KeyGen bzw. $\mathsf{Decrypt}$. Dabei wird bei `KeyGen`-Anfragen nur der erzeugte öffentliche Schlüssel *pk* zurückgegeben, der geheime Schlüssel *sk* wird lediglich für spätere `Decrypt`-Anfragen gespeichert. (Es ist klar, daß `Decrypt`-Anfragen *vor* der ersten `KeyGen`-Anfrage ignoriert werden.)

Man beachte, daß nicht festgelegt wurde, wieviele Parteien überhaupt am Protokoll $\pi_P^{(i)}$ teilnehmen. Da die nachfolgenden Ergebnisse nun nicht von der konkreten Teilnehmerzahl in $\pi_P^{(i)}$ abhängen, aber andererseits gerade Ergebnisse für beliebige Teilnehmerzahlen gewünscht sind (man denke nur an die Komposition von Protokollen), sei die Anzahl der Parteien in $\pi_P^{(i)}$ hier offen gelassen. Aus demselben Grund denke man sich einen festen Empfänger P_i fixiert: Alle folgenden Ergebnisse sind unabhängig von der Wahl von i. Ähnliche Betrachtungen gelten für die später vorgestellten Funktionalitäten für digitale Signaturen, Schlüsselaustausch und Commitment.

Auf diese Weise ist $\pi_P^{(i)}$ zumindest syntaktisch kompatibel zur Funktionalität $\mathcal{F}_{\text{PKE}}^{(i)}$. Es kann nun in [27] gezeigt werden, daß für IND-CCA-sichere Systeme P und bezüglich nicht-adaptiver Angreifer $\pi_P^{(i)}$ die Funktionalität $\mathcal{F}_{\text{PKE}}^{(i)}$ realisiert.[21] Der Beweis hierzu benutzt jedoch eine Reduktion eines $\pi_P^{(i)}$-Angreifers $\mathcal{A}$ auf einen IND-CCA-Angreifer A, bei der sich die zugehörigen Unterscheidungswahrscheinlichkeiten $\mathbf{Adv}_{P,A}^{\text{ind-cca}}$ und $\mathbf{Adv}_{\mathcal{A},\mathcal{S},\mathcal{Z}}^{\pi_P^{(i)},\mathcal{F}_{\text{PKE}}^{(i)}}$ um einen polynomialen (in k) Faktor unterscheiden.

Zudem wird in [27] – wie man gleich sieht, fälschlicherweise – behauptet, daß $\mathcal{F}_{\text{PKE}}^{(i)}$-Sicherheit eines Public-Key-Kryptosystems noch *nicht* IND-CCA-Sicherheit impliziert. Genauer gibt [27] dazu ein vermeintliches Gegenbeispiel an.

Eine Klärung der Sachlage führt nun das folgende Ergebnis aus [85] herbei:

[21] In [27] wird $\pi_P \geq \mathcal{F}_{\text{PKE}}$ gezeigt (es wird also nicht über Empfängerparteien quantifiziert), was strenggenommen falsch ist. Allerdings ergibt sich mit geringfügigen Änderungen ein (korrekter) Beweis für $\pi_P^{(i)} \geq \mathcal{F}_{\text{PKE}}^{(i)}$.

Satz 3.5. *Sei $P = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ ein Public-Key-Kryptosystem. Dann existiert eine bezüglich der Unterscheidungswahrscheinlichkeiten enge Korrespondenz zwischen nicht-adaptiven ROR-CCA-Angreifern und Umgebungen $\mathcal{Z}$ im folgenden Sinne:*

1. *Für jeden ROR-CCA-Angreifer A existiert eine Umgebung $\mathcal{Z}$, so daß für alle Simulatoren $\mathcal{S}$ und den Dummy-Angreifer $\tilde{\mathcal{A}}$ gilt:*

$$\mathbf{Adv}^{\pi_P^{(i)},\mathcal{F}_{\mathrm{PKE}}^{(i)}}_{\tilde{\mathcal{A}},\mathcal{S},\mathcal{Z}} = \frac{\left|\mathbf{Adv}^{\text{ror-cca}}_{P,A}\right|}{2}. \tag{3.1}$$

2. *Es existiert ein Simulator $\mathcal{S}_{\mathrm{ROR}}$, so daß für jede Umgebung $\mathcal{Z}$ und den Dummy-Angreifer $\tilde{\mathcal{A}}$ gilt:*

$$\mathbf{Adv}^{\pi_P^{(i)},\mathcal{F}_{\mathrm{PKE}}^{(i)}}_{\tilde{\mathcal{A}},\mathcal{S}_{\mathrm{ROR}},\mathcal{Z}} = \left|\mathbf{Adv}^{\text{ror-cca}}_{P,A}\right|. \tag{3.2}$$

Beweis. Die beiden Aussagen werden nacheinander bewiesen:

1. Sei ein ROR-CCA-Angreifer A gegeben. Es soll eine Umgebung $\mathcal{Z}$ konstruiert werden, welche $\pi_P^{(i)}$ von $\mathcal{F}_{\mathrm{PKE}}^{(i)}$ mit Unterscheidungsvorteil (3.1) trennt. Hierzu seien zwei Experimente E_1 und E_2 definiert, welche von $\mathcal{Z}$ durchgeführt werden.

   In E_1 wird Angreifer A mit den Eingaben 1^k und pk simuliert, wobei k der Sicherheitsparameter ist und pk einen von $\mathcal{Z}$ mittels `KeyGen` bei P_i erfragten öffentlichen Schlüssel bezeichnet. Anfragen von A an das RR-Orakel werden durch `Encrypt`-Anfragen von $\mathcal{Z}$ beantwortet, und Zugriffe auf das Decrypt-Orakel werden mittels `Decrypt`-Anfragen an P_i beantwortet.

   Das Experiment E_2 unterscheidet sich von E_1 nur dadurch, daß der jeweilige Klartext bei RR-Anfragen durch einen zufälligen (d. h. gleichverteilt gezogenen) Klartext gleicher Länge ersetzt wird. Sei $\mathcal{Z}$ die Umgebung, welche jeweils mit Wahrscheinlichkeit $1/2$ Experiment E_1 bzw. E_2 ausführt. Ausgabebit von $\mathcal{Z}$ ist dabei die Ausgabe von A, falls E_1 ausgeführt wurde, und die verneinte Ausgabe von A, falls E_2 ausgeführt wurde.

   Sei nun $\mathcal{S}$ ein beliebiger Simulator. Für festes k bezeichne dann ϵ_1^R (bzw. ϵ_1^I) die Wahrscheinlichkeit, daß A beim Ausführen von Experiment E_1 im realen Modell mit $\pi_P^{(i)}$ und $\tilde{\mathcal{A}}$ (bzw. im idealen Modell mit $\mathcal{F}_{\mathrm{PKE}}^{(i)}$ und $\mathcal{S}$) eine 1 ausgibt. Analog seien ϵ_2^R und ϵ_2^I die entsprechenden Wahrscheinlichkeiten für Experiment E_2.

   Da im idealen Modell die Antworten auf Encrypt-Anfragen nicht vom zu verschlüsselnden Klartext abhängen, gilt $\epsilon_1^I = \epsilon_2^I$. Weiter folgt aus Definition 3.3 und der Konstruktion der Experimente E_1 und E_2 direkt $|\mathbf{Adv}^{\text{ror-cca}}_{P,A}(k)| =$

$|\epsilon_1^R - \epsilon_2^R|$. Damit gilt für festes k:

$$\begin{aligned}\mathbf{Adv}^{\pi_P^{(i)},\mathcal{F}_{\mathrm{PKE}}^{(i)}}_{\tilde{\mathcal{A}},\mathcal{S},\mathcal{Z}}(k) &= \left|\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \pi_P^{(i)}, \tilde{\mathcal{A}}\right](k) - \mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \mathcal{F}_{\mathrm{PKE}}^{(i)}, \mathcal{S}\right](k)\right| \\ &= \left|\frac{1}{2}\left(\epsilon_1^R + \left(1 - \epsilon_2^R\right)\right) - \frac{1}{2}\left(\epsilon_1^I + \left(1 - \epsilon_2^I\right)\right)\right| \\ &= \frac{1}{2}\left|\epsilon_1^R - \epsilon_2^R + \epsilon_2^I - \epsilon_1^I\right| \;=\; \frac{|\mathbf{Adv}^{\text{ror-cca}}_{P,A}(k)|}{2}.\end{aligned}$$

2. Zunächst sei der Simulator $\mathcal{S}_{\mathrm{ROR}}$ beschrieben. Schlüsselgenerierung und Entschlüsselungen werden von $\mathcal{S}_{\mathrm{ROR}}$ mittels der Algorithmen KeyGen und $\mathsf{Decrypt}$ durchgeführt. Verschlüsselungen bezüglich des „korrekten“ Schlüssels $pk' = pk$ – bei denen $\mathcal{S}_{\mathrm{ROR}}$ ja nur die Länge $|m|$ des Klartextes m bekannt ist – werden mit dem Ergebnis einer Ausführung von $\mathsf{Encrypt}(1^k, pk, r)$ beantwortet, wobei r ein jeweils neu und gleichverteilt gezogenen Klartext r der Länge $|m|$ ist. Bei Verschlüsselungen bezüglich eines Schlüssels $pk' \neq pk$ andererseits kennt der Simulator $\mathcal{S}_{\mathrm{ROR}}$ den jeweiligen Klartext m und kann mit $\mathsf{Encrypt}(1^k, pk, m)$ antworten. Schließlich beantwortet $\mathcal{S}_{\mathrm{ROR}}$ alle Befehle, die ein Dummy-Angreifer befolgen würde, in offensichtlicher Weise, wobei bei Korruptionen von $\tilde{P}_i$ ein initial durch KeyGen erzeugter Schlüssel in den an $\mathcal{Z}$ gemeldeten Zustand von $\tilde{P}_i$ integriert wird.

   Hiermit ist klar, wie bei gegebener Umgebung $\mathcal{Z}$ ein ROR-CCA-Angreifer A konstruiert werden kann, der intern $\mathcal{Z}$ simuliert: Bei der ersten Schlüsselgenerierung mittels `KeyGen`-Anfrage an P_i wird $\mathcal{Z}$ die entsprechende Eingabe pk von A als Antwort serviert; `Encrypt`- und `Decrypt`-Anfragen von $\mathcal{Z}$ bezüglich dieses öffentlichen Schlüssels pk werden an die Orakel von A weitergeleitet. (`Encrypt`-Anfragen bezüglich anderer öffentlicher Schlüssel können direkt mittels des Algorithmus $\mathsf{Encrypt}$ beantwortet werden.)

   Um sicherzustellen, daß A sein Entschlüsselungsorakel nie bezüglich eines durch das RR-Orakel erhaltenen Chiffrats befragt, beantwortet A dabei derartige `Decrypt`-Anfragen von $\mathcal{Z}$ autonom. Dabei wird der entsprechende RR als Eingabe übermittelte Klartext zurückgeliefert.

   Ausgabe von A ist schließlich die Ausgabe des simulierten $\mathcal{Z}$. Hiermit ist nach Konstruktion klar, daß die Experimente $\mathbf{Exp}^{\text{ror-cca-0}}_{P,A}$ und $\mathbf{Exp}^{\text{ror-cca-1}}_{P,A}$ aus Definition 3.3 gerade Ausführungen von $\mathcal{Z}$ im realen Modell mit $\pi_P^{(i)}$ und $\tilde{\mathcal{A}}$ bzw. im idealen Modell mit $\mathcal{F}_{\mathrm{PKE}}^{(i)}$ und $\mathcal{S}_{\mathrm{ROR}}$ darstellen. Es folgt also (3.2).

□

Es findet hier eine Einschränkung auf nicht-adaptive Angreifer statt. Dies kann durch folgende Beobachtung gerechtfertigt werden: Wird ein Protokoll $\pi_P^{(i)}$ adaptiv attackiert, muß ein Simulator bei nachträglicher Korruption des Empfängers P_i einen geheimen Schlüssel liefern können, so daß *alle* vom Simulator *in Unkenntnis des jeweiligen Klartextes* generierten Chiffrate korrekt dechiffriert werden

können. Hierfür wird schon im Fall eines einzigen solchen Chiffrats der theoretisch anspruchsvolle und aufwendig zu realisierende Protokollbaustein „non-committing encryption“ (vgl. Canetti u. a. [33]) benötigt.

Man beachte weiter, daß die benutzten Reduktionen der Angreifer A bzw. Umgebungen $\mathcal{Z}$ Komplexitäten respektieren, d. h. es findet jeweils genau ein Aufruf von A bzw. $\mathcal{Z}$ statt. Da in diesem „feinauflösenden“ Sinne ROR-CCA-Sicherheit echt stärkere Sicherheitsgarantien gibt als IND-CCA-Sicherheit (vgl. [16]), gilt dies auch für Sicherheit bezüglich $\mathcal{F}_{\mathrm{PKE}}^{(i)}$. Nichtsdestotrotz sind nach [16] die *Begriffe* IND-CCA und ROR-CCA äquivalent, so daß sich folgendes Korollar ergibt:

Korollar 3.6. *Sei* $P = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *ein Public-Key-Kryptosystem. Dann sind folgende Aussagen äquivalent:*

(i) $\pi_P^{(i)} \geq \mathcal{F}_{\mathrm{PKE}}^{(i)}$ *bezüglich nicht-adaptiver Angreifer,*

(ii) P *ist IND-CCA-sicher,*

(iii) P *ist ROR-CCA-sicher.*

Damit erlauben also insbesondere praktisch vorgeschlagene und verwendete Systeme wie RSAES-OAEP (siehe [119]) oder das theoretisch bedeutsame System von Cramer und Shoup (vgl. [49]) aufgrund ihrer IND-CCA-Sicherheit schon automatisch beliebige Komposition (in Form der Idealisierung $\mathcal{F}_{\mathrm{PKE}}^{(i)}$) in größeren Protokollen.

Diese Ergebnisse sind bezüglich uniformer *und* bezüglich nicht-uniformer Angreifer gültig. Tatsächlich sind – bis auf Ausnahmen wie [75] – die meisten „klassischen“ Ergebnisse – wobei „klassisch“ hier als „nicht in einem simulierbarkeitsbasierten Umfeld formuliert“ zu lesen ist – im Umfeld uniformer Angreifer formuliert, bleiben aber gültig, wenn man zu einem nicht-uniformen Angreifermodell übergeht. Dies gilt insbesondere für die Ergebnisse aus [16]. Für Mehrparteienberechnungen hingegen ist es im Modell [27] wie schon angesprochen üblich, die unterscheidende Turing-Maschine $\mathcal{Z}$ nicht-uniform zu modellieren. Allerdings sind alle dortigen Betrachtungen auch bezüglich uniformer $\mathcal{Z}$ gültig. Dies gilt, wie schon besprochen und in [85] gezeigt, insbesondere für das Kompositionstheorem.

Abschließend sei noch auf die von [85] unabhängige Arbeit [37] hingewiesen, in der auch die Behauptung „IND-CCA-Sicherheit $\neq$ $\mathcal{F}_{\mathrm{PKE}}$-Sicherheit“ aus [27] als falsch erkannt und das Gegenteil bewiesen wurde. Wie schon angesprochen wurde in dieser Arbeit allerdings eine Abwandlung von $\mathcal{F}_{\mathrm{PKE}}$ benutzt, welche *nicht* über Empfängerparteien parametrisiert ist und insbesondere den am Ende von Abschnitt 3.2.1 skizzierten Angriff zuläßt.

3.2.6 Offene Fragen

Obwohl also mit IND-CCA (bzw. ROR-CCA) ein klassischer (d. h. nicht-simulierbarkeitsbasierter) Sicherheitsbegriff identifiziert wurde, welcher der intuitiven

Idealisierung $\mathcal{F}_{\text{PKE}}^{(i)}$ eines Public-Key-Kryptosystems genügt, stellt sich die Frage, ob nicht auch für andere klassische Sicherheitsbegriffe eine entsprechende Idealisierung gefunden werden kann. Beispielsweise ist nicht offensichtlich, wie man etwa den Begriff der „plaintext awareness“ aus [19] simulierbar charakterisieren könnte. (Informell sichert diese Eigenschaft zu, daß jeder PPT-Algorithmus, welcher ein Chiffrat produziert, auch schon den zugehörigen Klartext kennt.)

Dies ist insbesondere interessant, als „plaintext awareness“ in Verbindung mit IND-CPA-Sicherheit schon IND-CCA-Sicherheit impliziert, vgl. etwa [17]. Man könnte also prinzipiell auf eine auch intuitiv stärkere Sicherheitsgarantie hoffen.

3.3 Idealisierung digitaler Signaturen

In Abschnitt 3.2.3 wurde beschrieben, wie mittels komponierbarer Public-Key-Verschlüsselung (in Form der idealen Funktionalität $\mathcal{F}_{\text{PKE}}$) sichere Nachrichtenübertragung realisiert werden kann. Hierbei wurde durch Verschlüsselung die *Geheimhaltung* der verschickten Nachrichten gewährleistet. Nicht explizit behandelt wurde allerdings die Frage der *Authentizität* dieser Nachrichten – es wurden einfach authentifizierte Kanäle vorausgesetzt. Der Aspekt einer expliziten Authentifizierung von Nachrichten soll nun thematisiert werden.

Hierzu soll genauer auf Verfahren zur *digitalen Signatur* von Nachrichten zurückgegriffen werden. Eine Beschreibung einer Idealisierung digitaler Signaturen (in Form einer idealen Funktionalität $\mathcal{F}_{\text{SIG}}$) im Modell von [27] wurde schon in [27] angegeben. Dort findet sich auch ein Beweis für die Äquivalenz dieses Sicherheitsbegriffs mit dem klassischen Begriff EF-CMA bei Beschränkung auf nicht-adaptive Angriffe.[22] (Vorausgreifend sei bemerkt, daß EF-CMA-Sicherheit intuitiv fordert, daß selbst ein Angreifer, der sich Nachrichten seiner Wahl signieren lassen darf, nicht in der Lage ist, gültige Signaturen zu neuen Nachrichten zu erzeugen; siehe auch Definition 3.8.) In Canetti und Rabin [40] wurde zudem eine leicht veränderte Variante von $\mathcal{F}_{\text{SIG}}$ angegeben und ein Beweis für die Aussage formuliert, daß $\mathcal{F}_{\text{SIG}}$-Sicherheit eines Verfahrens zur digitalen Signatur bezüglich allgemeiner, also adaptiver Angriffe äquivalent zu Sicherheit im Sinne von EF-CMA ist.

Später wurde dann in Canetti [31] gezeigt, wie mit einer solchen Idealisierung $\mathcal{F}_{\text{SIG}}$ authentifizierte Kanäle – in Form einer geeigneten idealen Funktionalität $\mathcal{F}_{\text{AUTH}}$ – realisiert werden können. Da das Modell [27] nun aufgrund des Kompositionstheorems ein modulares Design von Protokollen (bzw. Protokollbausteinen) ermöglicht, ist in diesem Sinne das Problem der Authentifizierung von Nachrichten – etwa für die schon angesprochene sichere Nachrichtenübertragung – auf das Finden von geeigneten Verfahren zur digitalen Signatur reduziert.

Nun ist allerdings die Idealisierung $\mathcal{F}_{\text{SIG}}$ von digitalen Signaturen in den Fassungen [27] und [40] überhaupt nicht durch irgendein reales Protokoll realisierbar; die Beweise aus [27] und [40] sind fehlerhaft. Dies konnte in der Arbeit Backes

[22] Zur Erinnerung: Nicht-adaptive Angriffe sind Angriffe, bei denen der Angreifer Korruptionen nur in der ersten Aktivierung durchführt.

und Hofheinz [7] nachgewiesen werden.[23] Weiter konnte in [7] eine korrigierte Signaturfunktionalität angegeben werden und für diese die ursprünglich behauptete Äquivalenz zu EF-CMA-Sicherheit gezeigt werden. Die Ergebnisse aus [7] sollen nun vorgestellt und besprochen werden, um einen Eindruck zu geben, mit welchen Schwierigkeiten das Finden einer geeigneten Idealisierung von digitalen Signaturen verbunden sein kann.

3.3.1 Digitale Signaturen

Abstrakt und informell dient ein Verfahren zur digitalen Signatur dazu, zu einer digitalen Nachricht unter Kenntnis eines speziellen *Signaturschlüssels* eine *Signatur* zu generieren. Dabei soll zum einen mittels eines passenden *Verifikationsschlüssels* nachgewiesen werden können, daß ein Signiervorgang stattgefunden hat, d. h. es soll die Korrektheit einer Signatur nachgewiesen werden können. Zum anderen aber soll das Erzeugen einer gültigen Signatur *nur* unter Kenntnis des Signaturschlüssels möglich sein; es soll also nicht möglich sein, eine Signatur zu fälschen.

Ähnlich wie im Falle von Public-Key-Kryptosystemen fällt es nun leicht, syntaktische Anforderungen und Korrektheit eines Signatursystems formal zu fassen, was auch sogleich geschehen soll:

Definition 3.7. *Ein* Public-Key-Signatursystem S *besteht aus drei PPT-Algorithmen* KeyGen, Sign, Verify. *Für beliebige* $k \in \mathbb{N}$ *und* $vk, sk, m, s \in \{0,1\}^*$ *werden folgende* syntaktische *Bedingungen gestellt:*

- $\mathsf{KeyGen}(1^k) \in \{0,1\}^* \times \{0,1\}^*$,
- $\mathsf{Sign}(1^k, sk, m) \in \{0,1\}^*$,
- $\mathsf{Verify}(1^k, vk, m, s) \in \{\mathtt{valid}, \mathtt{invalid}\}$.

Es wird Korrektheit *in dem Sinne gefordert, daß die Wahrscheinlichkeit als Funktion in* k *vernachlässigbar sein muß, ein Schlüsselpaar* $(vk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ *zu generieren, so daß eine Nachricht* $m \in \{0,1\}^*$ *existiert, für die* $\mathtt{invalid} \leftarrow \mathsf{Verify}(1^k, vk, m, \mathsf{Sign}(1^k, sk, m))$ möglich *ist. Weiter wird* Eindeutigkeit *der Verifikation gefordert, in dem Sinne, daß für beliebige, aber feste* $k \in \mathbb{N}$ *und* $vk, m, s \in \{0,1\}^*$ *die Ausgabe von* $\mathsf{Verify}(1^k, vk, m, s)$ *konstant ist.*[24]

In dieser Definition entspricht sk dem *Signaturschlüssel*, mit welchem Nachrichten signiert werden können; vk bezeichnet den *Verifikationsschlüssel*, mit dessen Hilfe signierte Nachrichten auf Authentizität überprüft werden können. Man beachte aber, daß ähnlich wie für Public-Key-Kryptosysteme die gegebene Definition nur die *Korrektheit* eines Signatursystems einfängt. Damit genügt etwa ein triviales System, welches alle Signaturen als gültig akzeptiert (d. h. $\mathsf{Verify}(\cdot,\cdot,\cdot) = \mathtt{valid}$)

[23] Unabhängig davon wurde die Inkorrektheit des Beweises aus [40] auch in Canetti [29] erkannt.

[24] Effektiv wird hier also gefordert, daß der PPT-Algorithmus Verify durch einen deterministischen, polynomial laufzeitbeschränkten Algorithmus ersetzt werden kann.

schon Definition 3.7, obwohl hier natürlich Signaturen trivial fälschbar sind, und damit keine *Nicht-Abstreitbarkeit* von Signaturen gegeben ist.

Die Frage der *Sicherheit* eines Signatursystems soll sogleich behandelt werden, zuvor jedoch noch zwei technische Bemerkungen: Zunächst sind analog zu Definition 2.1 seltene „schlechte" Schlüsselpaare (vk, sk) erlaubt. Dies geschieht in erster Linie, um tatsächlich eingesetzte Signatursysteme zuzulassen. Man denke hier beispielsweise an RSA- oder ElGamal-basierte Systeme (etwa den „Digital Signature Algorithm", vgl. [56]), welche aus Effizienzgründen häufig probabilistische und prinzipiell fehleranfällige Primzahltests verwenden.

Des weiteren wird vom Verifikationsalgorithmus deterministisches Verhalten gefordert. Zwar sind die Verifikationsalgorithmen aller in diesem Kapitel erwähnten Signaturverfahren tatsächlich deterministisch, jedoch herrscht in der Literatur keine einhellige Auffassung darüber, ob ein „echt probabilistischer" Verifikationsalgorithmus zuzulassen ist. Etwa erlaubt Goldreich [71] ausdrücklich probabilistische Verifikationsalgorithmen; siehe dazu auch [71, Kapitel 6, Übung 1].

Man beachte allerdings, daß es bei einem probabilistischen Verifikationsalgorithmus prinzipiell möglich ist, aus einem legitim erzeugten Nachricht-Signatur-Paar (welches also wegen der Korrektheit des Signaturverfahrens immer als gültig verifiziert wird) eine neue Signatur für dieselbe Nachricht generiert werden kann, welche etwa mit Wahrscheinlichkeit 1/2 als gültig erkannt wird. Ein solches Verfahren kann sogar leicht aus einem in einem noch zu definierenden Sinne „sicheren" Public-Key-Signatursystem konstruiert werden, so daß das entstandene Verfahren immer noch „sicher" ist.

Die Existenz solcher „halbgültiger" Signaturen scheint jedoch für den Einsatz des Signaturverfahrens in einem größeren Protokoll alles andere als wünschenswert: Man denke hier etwa an signierte Verträge, die an mehrere Parteien gleichzeitig geschickt werden. Insofern scheint die Modellierung eines notwendig deterministischen Verifikationsalgorithmus berechtigt.[25]

Doch nun zur Sicherheit eines Signatursystems. Intuitiv soll natürlich verhindert werden, daß ein nicht durch Kenntnis eines Signaturschlüssels befugter Angreifer eine Signatur fälscht. Hierbei kann unterschieden werden, ob ein solcher Angriff nur als erfolgreich gilt, wenn eine Signatur zu einer *vorgegebenen* Nachricht gefälscht wurde, oder ob es reicht, eine Signatur zu *irgendeiner* Nachricht zu fälschen. Im letzteren Fall ist sofort einsichtig, daß beispielsweise das RSA-Signatursystem aus Rivest u. a. [122] anfällig für Angriffe ist.

Unabhängig davon können auch – ähnlich wie im Fall von Public-Key-Kryptosystemen – die Möglichkeiten des Angreifers bei einem Angriff variiert werden. Etwa kann dem Angreifer ein *Signaturorakel* zugestanden werden, das Klartexte nach Wahl des Angreifers signiert. Es ist klar, daß durch das Signaturorakel erhaltene Signaturen nicht als erfolgreich gefälschte Signaturen gewertet werden dürfen. Ein äußerst lesenswerter Überblick über Angriffsziele und -möglichkeiten findet sich

[25] Alternativ ist natürlich auch eine Abschwächung dahingehend denkbar, daß der Verifikationsalgorithmus nur mit überwältigender Wahrscheinlichkeit ein gewisses Ergebnis liefern muß.

in Goldwasser u. a. [77].

In der Arbeit [77] kann auch ein konkretes Signaturschema angegeben werden, welches dem stärksten dort vorgestellten Sicherheitskriterium „EF-CMA“ („existential unforgeability under adaptive chosen-message attacks“) genügt.[26] Hier seien auch die in diesem Sinne sicheren Verfahren aus Rompel [123], Cramer und Damgård [47, 48], Dwork und Naor [57] erwähnt. Der Begriff EF-CMA ist insbesondere von Interesse, da er schon äquivalent zu Komponierbarkeit des Signatursystems ist – jedoch nur bei geeigneter Wahl der diesbezüglichen Idealisierung digitaler Signaturen als ideale Funktionalität. Genau dieser Zusammenhang soll noch im Detail besprochen werden.

In jedem Fall ist es nun Zeit, den Begriff EF-CMA in geeigneter Form zu fixieren. Hierbei soll eine geeignete Formulierung dieses Begriffs aus [77] angegeben werden:

Definition 3.8. *Sei* $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *ein Public-Key-Signatursystem. Für einen Angreifer (d. h. PPT-Algorithmus) A sei folgendes Experiment definiert:*

$$\begin{array}{l} \textit{Experiment } \mathbf{Exp}_{S,A}^{\text{ef-cma}}(k)\textit{:} \\ \quad (vk, sk) \leftarrow \mathsf{KeyGen}(1^k) \\ \quad (m, s) \leftarrow A^{\mathsf{Sign}(1^k, sk, \cdot)}(1^k, vk) \\ \quad \textbf{Return } \mathsf{Verify}(1^k, vk, m, s) \end{array}$$

Hierbei wird verlangt, daß A nie Nachrichten m zurückgibt, für die A bei seinem $\mathsf{Sign}(1^k, sk, \cdot)$*-Orakel Signaturen erfragt hat.*[27]

Dann heißt P EF-CMA-sicher, wenn für alle in vorstehendem Sinne erlaubten Angreifer A die Funktion

$$\mathsf{Pr}\left[\mathbf{Exp}_{S,A}^{\text{ef-cma}}(k) \rightarrow \texttt{valid}\right]$$

vernachlässigbar in k ist.

Es darf also für EF-CMA-Sicherheit keinen Angreifer A geben, welcher in der Lage ist, eine gültige Signatur zu *irgendeiner* Nachricht zu produzieren. Dabei darf A sogar auf ein Orakel zurückgreifen, das Nachrichten nach Wahl von A signiert.

3.3.2 Eine problematische Idealisierung

In [27] konnte eine Idealisierung eines Public-Key-Signatursystems in Form einer idealen Funktionalität $\mathcal{F}_{\text{SIG}}$ angegeben werden. Diese Funktionalität wurde dann in [40] leicht abgewandelt, und es wurde ein Beweis angegeben, daß diese Variation von $\mathcal{F}_{\text{SIG}}$ gerade von den Signatursystemen – in naheliegender Weise als Protokolle aufgefaßt – realisiert wird, welche EF-CMA-sicher sind. Abbildung 3.4 ist eine

[26] Zwar ist das Schema aus [77] nicht zustandslos, paßt also nicht in Definition 3.7, es kann aber ohne Sicherheitseinbußen zustandslos gemacht werden, vgl. Goldreich [68].

[27] Wird vom Angreifer A nur verlangt, daß er nie (m, s) ausgibt, wobei s eine von $\mathsf{Sign}(1^k, sk, \cdot)$ gelieferte Signatur zu einer Anfrage m ist, erhält man den stärkeren Begriff der „non-malleability“, vgl. Stern u. a. [132].

Reproduktion der Variante aus [40].[28] Die im folgenden beschriebenen Probleme treten allerdings auch mit der älteren Idealisierung aus [27] auf.

Die Funktionalität $\mathcal{F}_{\text{SIG}}$

Protokollteilnehmer: $\mathcal{F}_{\text{SIG}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

- **Initialisierung:** Erwarte in der ersten Aktivierung von einer Partei P_i eine Eingabe $(\texttt{Signer}, sid)$. Sende dann $(\texttt{Signer}, sid)$ zum Angreifer. Von nun an, ignoriere alle weiteren $(\texttt{Signer}, sid)$-Eingaben.
- **Signatur-Generierung:** Bei Erhalt einer Eingabe $(\texttt{Sign}, sid, m)$ von P_i, gib $(\texttt{Sign}, sid, m)$ zum Angreifer. Bei Erhalt von $(\texttt{Signature}, sid, m, \sigma)$ vom Angreifer, setze $s_m = \sigma$, sende $(\texttt{Signature}, sid, m, \sigma)$ an P_i und fordere den Angreifer auf, diese Nachricht sofort auszuliefern. Speichere das Paar (m, s_m).
- **Signatur-Verifikation:** Bei Erhalt von $(\texttt{Verify}, sid, P_{i'}, m, \sigma)$ von P_j:
  1. Falls $i = i'$ (d. h. wenn die Senderidentität in der Verifikationsanfrage mit der Identität des tatsächlichen Signierers übereinstimmt): Falls m noch nie signiert wurde, setze $f = 0$. Wenn m schon zuvor signiert wurde (d. h. s_m ist definiert) und $s_m = \sigma$ gilt, setze $f = 1$. Falls m zwar zuvor signiert wurde, aber $s_m \neq \sigma$ ist, lasse den Angreifer über f entscheiden. (D. h.: Sende $(\texttt{Verify}, sid, P_j, P_{i'}, m, \sigma)$ zum Angreifer. Bei Erhalt von $\phi \in \{0, 1\}$ vom Angreifer, setze $f = \phi$.)
  2. Falls $i \neq i'$: Wenn $P_{i'}$ unkorrumpiert ist, setze $f = 0$. Laß andernfalls den Angreifer über f entscheiden wie in Schritt 1.
  3. Ist schließlich f bestimmt, sende $(\texttt{Verified}, sid, m, f)$ an P_j und fordere den Angreifer auf, diese Nachricht sofort auszuliefern.

Abbildung 3.4: Die Idealisierung eines Public-Key-Signatursystems aus [40]

Zum Verständnis sind einige Kommentare unabdingbar: Zunächst ist eine Implementierung von $\mathcal{F}_{\text{SIG}}$ durch ein Public-Key-Signatursystem, also durch lokal auszuführende Algorithmen gewünscht. Folgerichtig wird $\mathcal{F}_{\text{SIG}}$ wie $\mathcal{F}_{\text{PKE}}$ als *direkte* Funktionalität modelliert, siehe auch Abschnitt 3.2.1.

Allerdings werden ausschließlich `Sign`- und `Verify`-Eingaben direkt beantwortet; die anfängliche Bestätigung der Initialisierung von $\mathcal{F}_{\text{SIG}}$ darf vom Simulator verzögert werden, um eine Schlüsselverteilung in einem realisierenden Protokoll zu ermöglichen. Ähnlich wie im Fall der Funktionalität $\mathcal{F}_{\text{PKE}}$ darf der Simulator von $\mathcal{F}_{\text{SIG}}$ herausgegebene Signaturen wählen. Dies geschieht, um eine Verteilung von Signaturen wie in einem realen Public-Key-Signatursystem zu ermöglichen,

[28] Diese Variante wurde auch in Canetti und Krawczyk [36] beschrieben und verwendet.

und sollte nicht als „Sicherheitsloch“ betrachtet werden: Über die *Gültigkeit* einer Signatur entscheidet $\mathcal{F}_{\text{SIG}}$ unabhängig von deren Format.

Auch bedarf noch die Bedingung „[...] Wenn $P_{i'}$ unkorrumpiert ist, [...]“ in der Beschreibung von $\mathcal{F}_{\text{SIG}}$ einer Erläuterung. In Canetti und Krawczyk [36] wird nämlich als Variante des Modells aus [27] zusätzlich $\mathcal{F}$ über Korruptionen informiert. Auf diese Weise lassen sich mächtigere Funktionalitäten definieren (die dabei natürlich nicht zwangsläufig realisierbar durch reale Protokolle sein müssen). Im vorliegenden Fall ist die Betrachtung dieser Abwandlung des Modells sinnvoll und soll genutzt werden.

Generell ist $\mathcal{F}_{\text{SIG}}$ – ähnlich wie die Funktionalität $\mathcal{F}_{\text{PKE}}$, vgl. Abschnitt 3.2.1 – unsauber und mißverständlich formuliert. Etwa legt nach dieser Beschreibung die erste `Signer`-Eingabe die signierende Partei fest. Hier führt eine ähnliche Problematik wie der – für den Falle der ursprüngliche Formulierung von $\mathcal{F}_{\text{PKE}}$ – in Abschnitt 3.2.1 beschriebenen zu einer trivialen Unterscheidung von realem und idealem Modell.

Informell kann demnach kein reales Protokoll feststellen, welche Partei als erste eine `Signer`-Eingabe bekommen hat. Die $\mathcal{F}_{\text{SIG}}$-Formulierung aus [40] ist schon aus diesem Grund nicht realisierbar; jedoch ist hier offensichtlich, wie $\mathcal{F}_{\text{SIG}}$ abgewandelt werden muß, um diesem Angriff zu entgehen: Ähnlich wie im Falle von $\mathcal{F}_{\text{PKE}}$ bietet sich eine Parametrisierung der Funktionalität über *signierende Parteien* an.

Schwieriger zu handhaben ist ein Angriff nach folgendem Muster: $\mathcal{Z}$ initialisiert $\mathcal{F}_{\text{SIG}}$ zunächst mittels einer `Signer`-Eingabe von einer Partei P_i, und erfragt unmittelbar danach bei P_i die Signatur σ zu einer beliebigen Nachricht m. Sogleich danach (also bevor im realen Modell irgendwelche Nachrichten ausgeliefert wurden) läßt $\mathcal{Z}$ eine *andere* Partei P_j das Tupel (m, σ') auf Gültigkeit überprüfen. Hierbei setzt $\mathcal{Z}$ jeweils mit Wahrscheinlichkeit $1/2$ $\sigma' = \sigma$ bzw. $\sigma' = s$ für eine gültige Signatur s zur Nachricht m bezüglich eines neuen, von $\mathcal{Z}$ selbst erzeugten Schlüsselpaars.[29] Diese `Verify`-Anfrage wird im idealen Modell genau dann mit $f = 1$ beantwortet, wenn $\sigma' = \sigma$ ist. Da aber die Verteilungen von s und σ identisch sind, und niemals eine Nachricht an P_j ausgeliefert wurde, ist die Verteilung von f im realen Modell unabhängig davon, ob $\sigma' = \sigma$ ist.

Intuitiv wurde ausgenutzt, daß im idealen Modell sofort nach der Initialisierungsnachricht an einen Signierer P_i Signaturen überprüft werden können. Es wird nicht eine eventuell im realen Modell nötige Schlüsselverteilung abgewartet. Dieser Mangel kann behoben werden, indem – ähnlich wie im Falle der Funktionalität $\mathcal{F}_{\text{PKE}}$ – ein expliziter Verifikationsschlüssel eingeführt wird, welcher bei einer `Signer`-Anfrage zurückgeliefert wird, und mit welchem Verifikationsanfragen ausgestattet sein müssen. Damit kann der nicht durch einen lokalen Algorithmus faßbare Prozeß der Schlüsselverteilung „ausgelagert“ werden.

Zurückkehrend zur Beschreibung von $\mathcal{F}_{\text{SIG}}$ sei bemerkt, daß diese Beschreibung offen läßt, wie Verifikationsanfragen bei einer *mehrfach* signierten Nachricht m ge-

[29] Hier wird der Einfachheit halber ein als Protokoll aufgefaßtes Signatursystem zugrundegelegt; der Angriff läßt sich aber verallgemeinern.

handhabt werden; in diesem Fall ist die „gültige“ Signatur s_m aus Schritt 1 der Verifikationsprozedur im allgemeinen nicht eindeutig bestimmt. Zudem ist in der gegebenen Formulierung unklar, wie sich $\mathcal{F}_{\text{SIG}}$ bei `Sign-` oder `Verify`-Anfragen verhält, die *vor* der Initialisierung durch eine `Signer`-Eingabe eintreffen. Es liegt nahe, solche Anfragen einfach zu ignorieren, und insbesondere *nicht* dem Angreifer zur Beantwortung weiterzuleiten; es hat ja anschaulich noch keine Schlüsselverteilung stattgefunden.[30] In diesem Punkt liegt tatsächlich der Kern eines noch zu besprechenden Problems – eine umfassende Diskussion folgt.

Für den Augenblick und das Verständnis des noch zu gebenden Beweises der Nicht-Realisierbarkeit von $\mathcal{F}_{\text{SIG}}$ ist zunächst von Bedeutung, daß $\mathcal{F}_{\text{SIG}}$ Verifikationsanfragen für Signaturen, welche nicht durch $\mathcal{F}_{\text{SIG}}$ selbst erzeugt wurden, mit „ungültig“ (bzw. $f = 0$) beantwortet. Bei Annahme etwa eines als Protokoll aufgefaßten Signatursystems S jedoch können im Falle einer *korrumpierten* signierenden Partei (im Sinne von S) gültige Signaturen zu beliebigen Nachrichten *lokal* von $\mathcal{Z}$ generiert werden. Eine Anfrage bei einer beliebigen unkorrumpierten Partei P_j wird nun eine solche Signatur als gültig klassifizieren – im Gegensatz zu $\mathcal{F}_{\text{SIG}}$ im idealen Modell.

Für diesen Angriff ist es wichtig, daß $\mathcal{F}_{\text{SIG}}$ die korrumpierte Partei P_i als signierende Partei akzeptiert, also mit einer entsprechenden `Signer`-Eingabe von P_i initialisiert wurde. Es sei zunächst angenommen, daß die Beschreibung von $\mathcal{F}_{\text{SIG}}$ dahingehend vervollständigt sei, daß bei noch nicht erfolgter `Signer`-Initialisierung alle Verifikationsanfragen von $\mathcal{F}_{\text{SIG}}$ ignoriert werden. Dann muß ein „guter“ Simulator auch bei schon initial (also nicht-adaptiv) korrumpiertem Signierer P_i eine solche Initialisierung vornehmen, um nicht „aufzufallen“. In diesem Fall genügt also schon ein nicht-adaptiver Angriff, um zu zeigen, daß kein Signatursystem $\mathcal{F}_{\text{SIG}}$ realisiert.

Werden andererseits bei nicht-initialisiertem $\mathcal{F}_{\text{SIG}}$ alle Verifikationsanfragen zum Simulator durchgeleitet, so hat dieser die Möglichkeit, bei initial korrumpiertem P_i einfach nie eine `Signer`-Eingabe im Namen von P_i zu senden und alle Verifikationsanfragen durch die Algorithmen des Signatursystems S zu beantworten. In diesem Fall ist also ein *adaptiver* Angriff vonnöten, bei welchem $\mathcal{Z}$ zunächst – bei noch unkorrumpiertem P_i – die Initialisierung von $\mathcal{F}_{\text{SIG}}$ durch eine explizite `Signer`-Eingabe sicherstellt und erst anschließend P_i durch einen geeigneten Angreifer (etwa den Dummy-Angreifer) korrumpieren läßt. Danach kann wie im nicht-adaptiven Fall weiter verfahren werden.

Im letzteren Fall (d. h. falls bei nicht-initialisierter idealer Funktionalität der Angreifer `Sign-` und `Verify`-Anfragen beantwortet) stellt sich zudem ein anderes Problem: Nun kann ein geeigneter Angriff mit korrumpiertem Signierer sogar im idealen Modell (das ja als „hinreichend gute“ Idealisierung angenommen werden soll) erzwingen, daß die Gültigkeit von Signaturen erst zum Zeitpunkt der Überprüfung festgelegt wird. Dies ist natürlich alles andere als wünschenswert: Man denke

[30] Überdies führt eine solche Vervollständigung zu einem Verlust von intuitiven Sicherheitsgarantien; man denke nur an einen korrumpierten Signierer, der keine Initialisierung vornimmt.

an zu Protokollbeginn fixierte, aber verschlüsselte Verträge, über deren Gültigkeit erst nach Ablauf eines kompletten Protokolls entschieden wird.

Diese Diskussion soll später noch im Zusammenhang mit einer verbesserten Signaturfunktionalität weitergeführt werden. Vorher soll jedoch der oben beschriebene Angriff in einer verallgemeinerten und in [7] veröffentlichten Form vorgestellt werden.

Hierbei wird wie besprochen angenommen, daß bei nicht-initialisierter Funktionalität $\mathcal{F}_{\text{SIG}}$ alle Verifikationsanfragen ignoriert werden; im Fall, daß solche Anfragen vom Simulator beantwortet werden, kann eine schon angedeutete Variation des folgenden Angriffs zumindest die Nicht-Realisierbarkeit von $\mathcal{F}_{\text{SIG}}$ bezüglicher adaptiver Angriffe zeigen.

3.3.3 Ein Unmöglichkeitsresultat

Satz 3.9. *Es existiert kein reales Protokoll π, das die ideale Funktionalität $\mathcal{F}_{\text{SIG}}$ aus [40] für $n \geq 2$ Parteien realisiert. Dies gilt sogar bezüglich authentifizierter Kanäle und nicht-adaptiver Angreifer.*

Beweis. Sei zum Widerspruch π mit $\pi \geq \mathcal{F}_{\text{SIG}}$ angenommen. Seien weiter mit P_i und P_j zwei verschiedene Parteien fixiert. Es bezeichne $\mathcal{S}$ den Simulator, welcher Angriffe bezüglich des Dummy-Angreifers $\tilde{\mathcal{A}}$ simuliert (vgl. Abschnitt 3.1.5). Sei nun $\mathcal{Z}_1$ die wie folgt definierte Umgebung, welche im realen Modell mit π und dem Dummy-Angreifer ausgeführt wird:

1. Aktiviere P_i mit Eingabe $(\texttt{Signer}, 0)$.

2. Wähle gleichverteilt $r \in \{0,1\}^k$ und aktiviere P_i mit $(\texttt{Sign}, 0, r)$; es bezeichne $(\texttt{Signature}, 0, r, \sigma)$ die erhaltene Antwort. Wenn keine direkte Antwort kommt, halte mit Ausgabe 0 an.

3. Aktiviere P_j mit Eingabe $(\texttt{Verify}, 0, P_i, r, \sigma)$; es bezeichne $(\texttt{Verified}, 0, r, f)$ die erhaltene Antwort. Wenn keine direkte Antwort kommt, halte mit Ausgabe 0 an. Ansonsten halte mit Ausgabe $f \in \{0,1\}$ an.

Es ist klar, daß $\mathcal{Z}_1$ polynomial beschränkt ist. Nach Definition von $\mathcal{F}_{\text{SIG}}$ ist weiter $\mathsf{Pr}[\mathcal{Z}_1 \to 1 \mid \mathcal{F}_{\text{SIG}}, \mathcal{S}](k) = 1$ und deshalb $\mathsf{Pr}\left[\mathcal{Z}_1 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$ als Funktion in k überwältigend. Dabei ist zu beachten, daß aufrund der Direktheit von $\mathcal{F}_{\text{SIG}}$ Antworten auf `Sign`- und `Verify`-Eingaben sofort ausgeliefert werden.

Sei weiter $\mathcal{Z}_2$ die wie folgt definierte Umgebung, welche wie $\mathcal{Z}_1$ im realen Modell mit π und dem Dummy-Angreifer ausgeführt wird:

1. Weise den Angreifer an, P_i zu korrumpieren. Bereite eine lokale Simulation $P_i^{(s)}$ von P_i vor.

2. Aktiviere $P_i^{(s)}$ mit Eingabe $(\texttt{Signer}, 0)$.

3. Wähle gleichverteilt $r \in \{0,1\}^k$ und aktiviere $P_i^{(s)}$ mit $(\texttt{Sign}, 0, r)$; es bezeichne $(\texttt{Signature}, 0, r, \sigma)$ die erhaltene Antwort. Wenn keine direkte Antwort kommt, halte mit Ausgabe 0 an.

4. Aktiviere P_j mit Eingabe $(\texttt{Verify}, 0, P_i, r, \sigma)$; es bezeichne $(\texttt{Verified}, 0, r, f)$ die erhaltene Antwort. Wenn keine direkte Antwort kommt, halte mit Ausgabe 0 an. Ansonsten halte mit Ausgabe $f \in \{0,1\}$ an.

Es soll davon ausgegangen werden, daß $P_i^{(s)}$ wenigstens pro Aktivierung polynomial laufzeitbeschränkt im Maximum der Gesamtlänge aller erhaltenen Eingaben und k ist,[31] vgl. die Diskussion in Abschnitt 3.2.2. Damit ist $\mathcal{Z}_2$ polynomial beschränkt. Weiter ist, da P_i passiv korrumpiert wurde – also die Korruption das Verhalten von P_i nicht beeinflußt –, die Funktion

$$\mathsf{Pr}\left[\mathcal{Z}_2 \to 1 \mid \pi, \tilde{\mathcal{A}}\right] = \mathsf{Pr}\left[\mathcal{Z}_1 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$$

überwältigend in k, womit wegen $\pi \geq \mathcal{F}_{\text{SIG}}$ auch $\mathsf{Pr}[\mathcal{Z}_2 \to 1 \mid \mathcal{F}_{\text{SIG}}, \mathcal{S}]$ überwältigend in k ist. Somit wird nach Konstruktion von $\mathcal{Z}_2$ die *lokal* von $\mathcal{Z}_2$ (genauer: von $P_i^{(s)}$) erzeugte Signatur σ zu r von P_j als gültig erkannt.

Nach Definition von $\mathcal{F}_{\text{SIG}}$ hat also $\mathcal{S}$ im idealen Modell mit überwältigender Wahrscheinlichkeit eine explizite $(\texttt{Sign}, 0, r)$-Eingabe im Namen der korrumpierten Partei P_i an $\mathcal{F}_{\text{SIG}}$ gegeben.[32] Weil $\mathcal{S}$ für festes $\mathcal{Z} = \mathcal{Z}_2$ als polynomial beschränkt angenommen werden darf (siehe auch Abschnitt 3.2.2), kann aber $\mathcal{S}$ insgesamt nur polynomial viele Nachrichten r' signieren lassen. Schließlich ist aber die Menge der von $\mathcal{S}$ signierten Nachrichten unabhängig von der tatsächlichen – von $\mathcal{Z}_2$ in Schritt 3 gewählten – Nachricht $r \in \{0,1\}^k$, da $\mathcal{Z}_2$ die Signatur σ *lokal* mittels $P_i^{(s)}$ generiert und die Verifikationsanfrage in Schritt 4 direkt beantwortet werden muß.

Also ist die Wahrscheinlichkeit, daß $\mathcal{S}$ im idealen Modell die von $\mathcal{Z}_2$ gewählte Nachricht $r \in \{0,1\}^k$ signiert, vernachlässigbar in k. Demnach wird σ in Schritt 4 nur mit vernachlässigbarer Wahrscheinlichkeit als gültig erkannt, und die Funktion $\mathsf{Pr}[\mathcal{Z}_2 \to 1 \mid \mathcal{F}_{\text{SIG}}, \mathcal{S}]$ muß vernachlässigbar sein; es entsteht ein Widerspruch. □

Es sei explizit darauf hingewiesen, daß der beschriebene Angriff *nicht* ausnutzt, daß die Initialisierung im realen Modell möglicherweise der Auslieferung von Nachrichten bedarf. Es wird auch nicht benutzt, daß in der Formulierung von $\mathcal{F}_{\text{SIG}}$ aus [40] prinzipiell jede Partei Signierer sein kann. Insbesondere zeigen offensichtliche Adaptionen dieses Angriffs, daß auch eine wie in Abschnitt 3.3.2 besprochene diesbezügliche Korrektur von $\mathcal{F}_{\text{SIG}}$ nicht realisierbar ist.

Dieser Angriff kann auch auf einfache Weise abgewandelt werden, so daß eine auf die `Signer`-Initialisierung folgende Schlüsselverteilung abgewartet wird, solange diese Schlüsselverteilung mit polynomial vielen und langen Nachrichten abläuft.

[31] Dies entspricht dem sehr schwachen Begriff „weakly polynomial“ aus [11].

[32] Hier fließt ein, daß bei nicht-initialisiertem $\mathcal{F}_{\text{SIG}}$ Verifikationsanfragen ignoriert werden; vgl. die Diskussion im vorigen Abschnitt.

(Dies verbessert nicht den Angriffserfolg, sondern soll nur zeigen, daß der Erfolg dieses Angriffs nicht auf den initialen Schlüsselaustausch zurückzuführen ist.)

In [40, Claim 2] wird behauptet, daß $\mathcal{F}_{\text{SIG}}$ genau von den Signatursystemen (als reale Protokolle interpretiert) realisiert wird, welche EF-CMA-sicher sind. Es mag von Interesse sein, zu sehen, an welcher Stelle der in [40] gegebene Beweis für diese Behauptung irrt.

Der kritische Teil des besagten Beweises behandelt die Aussage $\pi_S \geq \mathcal{F}_{\text{SIG}}$ bei als EF-CMA-sicher angenommenem Public-Key-Signatursystem S und damit assoziiertem realem Protokoll π_S. Hierfür werden π_S-Angriffe auf S-Angriffe (d. h. auf Angreifer im EF-CMA-Spiel mit System S, vgl. Definition 3.8) reduziert. Genauer wird eine Umgebung $\mathcal{Z}$, welche zwischen einem realen Modell mit Angreifer $\mathcal{A}$ und einem idealen Modell mit speziell konstruiertem Angreifer $\mathcal{S} = \mathcal{S}(\mathcal{A})$ erfolgreich unterscheidet, zu einem erfolgreichen EF-CMA-Angreifer G umgeformt.

Bei der Begründung des Erfolgs von G im EF-CMA-Spiel wird behauptet, daß "[...] as long as event B does not occur, $\mathcal{Z}$'s view of an interaction with $\mathcal{A}$ and parties running the protocol is distributed identically to its view of an interaction with $\mathcal{S}$[33] and $\mathcal{F}_{\text{SIG}}$ in the ideal process". Hierbei ist wiederum B definiert als das Ereignis, daß während eines Protokollablaufs im realen Modell "[...] $ver(v, m, \sigma) = 1$ for some message m and signature σ, but the signer is uncorrupted and never signed m during the execution of the protocol", wobei *ver* den Verifikationsalgorithmus des Systems S bezeichnet.

Diese Aussage ist falsch. Im Falle einer korrumpierten signierenden Partei P_i läßt der konstruierte Simulator $\mathcal{S}$ überhaupt keine Nachrichten im Namen von P_i signieren. Trotzdem können natürlich während eines realen Protokollablaufs wie im Beweis von Satz 3.9 bei etwa passiv korrumpierten P_i Nachrichten legitim signiert werden. Da für die zitierte Aussage keinerlei weitere Begründung gegeben wurde, läßt sich der Fehler im Beweis auch nicht genauer lokalisieren.

3.3.4 Eine korrigierte Idealisierung

Das im Beweis von Satz 3.9 aufgedeckte Problem ist, wie schon informell in Abschnitt 3.3.2 besprochen, ein prinzipielles Problem. Es kann im Falle eines passiv korrumpierten Signierers nicht verhindert werden, daß Nachrichten lokal und ohne Zutun von $\mathcal{F}_{\text{SIG}}$ signiert werden.[34] Insofern dürfen also bei korrumpiertem Signierer unbekannte Signaturen nicht von $\mathcal{F}_{\text{SIG}}$ als ungültig abgetan werden. Für eine Realisierbarkeit ist – wie der Beweis von Satz 3.9 zeigt – nötig, daß in diesem Fall eine für das Signatursystem spezifische Beurteilung der Gültigkeit der Signatur erfolgt.

Es liegt nahe, diese Aufgabe in den Simulator auszulagern, da dieser im Gegensatz zu $\mathcal{F}_{\text{SIG}}$ protokollspezifisch gewählt werden kann. Eine solchermaßen korrigierte und dabei von den anderen in Abschnitt 3.3.2 angesprochenen Mängeln befreite Funktionalität $\mathcal{F}_{\text{SIG}}^{(i)}$ wurde in [7] angegeben und ist in Abbildung 3.5 beschrieben.

[33] Hier wurde ein offensichtlicher Tippfehler in [40] korrigiert.

[34] Alternativ wäre höchstens die Realisierung eines *nicht-direkten* $\mathcal{F}_{\text{SIG}}$ durch ein interaktives Protokoll denkbar.

Die Funktionalität $\mathcal{F}_{\text{SIG}}^{(i)}$

Protokollteilnehmer: $\mathcal{F}_{\text{SIG}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

- **Initialisierung:** Beim *ersten* Erhalt einer Eingabe $(\texttt{Signer}, sid)$ von P_i (und nur P_i), sende $(\texttt{Key}, sid)$ zu $\mathcal{S}$. Bei Erhalt einer Antwort $(\texttt{Key}, sid, v)$ von $\mathcal{S}$, leite diese Eingabe an P_i weitere und speichere v. Ignoriere weitere $(\texttt{Signer}, sid)$-Eingaben. Was folgt, findet nur *nach* der ersten `Signer`-Eingabe von P_i Anwendung.
- **Signatur-Generierung:** Bei Erhalt von $(\texttt{Sign}, sid, m)$ von P_i (und nur P_i), leite diese Eingabe an $\mathcal{S}$ weiter. Bei Erhalt von $(\texttt{Signature}, sid, m, \sigma)$ von $\mathcal{S}$, leite diese Nachricht an P_i weiter und speichere das Tupel $(m, \sigma, 1)$.
- **Signatur-Verifikation:** Bei Erhalt von $(\texttt{Verify}, sid, m, \sigma', v')$ von P_j, antworte mit $(\texttt{Verified}, sid, m, f)$, wobei f wie folgt bestimmt wird:
  1. Ist $v = v'$ und wurde ein Tupel (m, σ', g) gespeichert[a], so setze $f = g$.
  2. Ist $v = v'$, ist weiter P_i unkorrumpiert, und existiert kein gespeichertes Tupel (m, σ, g) für irgendwelche σ, g, so setze $f = 0$.
  3. In allen anderen Fällen (insbesondere im Fall $v \neq v'$), leite die erhaltene Eingabe an $\mathcal{S}$ weiter; extrahiere f aus einer Antwort $(\texttt{Verified}, sid, m, f)$ von $\mathcal{S}$ und speichere das Tupel (m, σ', f).

[a] Hier ist Eindeutigkeit bei Existenz garantiert.

Abbildung 3.5: Die Idealisierung eines Signatursystems aus [7]

Man beachte zunächst, daß $\mathcal{F}_{\text{SIG}}^{(i)}$ über den Index der signierenden Partei parametrisiert ist. Es sind also keine unangenehmen Mehrdeutigkeiten wie im Falle von $\mathcal{F}_{\text{SIG}}$ möglich. Irritierenderweise ist der diesbezüglich beschriebene Angriff immer noch mit der in Canetti [29] überarbeitenden Funktionalität $\mathcal{F}_{\text{SIG}}$ möglich; Abhilfe schafft erst die Funktionalität $\mathcal{F}_{\text{SIG}}$ aus Canetti [30, 31, 32], in welcher eine Parametrisierung der Funktionalität über Signierer aus der Sitzungsnummer abgeleitet wird.

Weiter wird im Falle von $\mathcal{F}_{\text{SIG}}^{(i)}$ keine initiale Schlüsselverteilung mehr idealisiert, da dies zu den schon angesprochenen Problemen führt. Stattdessen wird ähnlich wie bei $\mathcal{F}_{\text{PKE}}$ ein expliziter Verifikationsschlüssel v eingeführt, welcher bei Verifikationen immer von der überprüfenden Partei mitgeschickt werden muß. Bei falschem Verifikationsschlüssel werden *keine* Sicherheitsgarantien vergeben.

Unabhängig von der hier besprochenen Arbeit [7] fanden Korrekturen (mit der gerade angesprochenen Einschränkung) der Funktionalität $\mathcal{F}_{\text{SIG}}$ aus [40] auch in [29, 30, 31, 32] statt. In diesem Zusammenhang wird auch die gerade gegebene Idealisierung $\mathcal{F}_{\text{SIG}}^{(i)}$ aus [7] zitiert; es wird dabei kritisiert, daß $\mathcal{F}_{\text{SIG}}^{(i)}$ im Falle eines kor-

rumpierten Signierers keine *gleichlautenden* Verifikationsergebnisse für wiederholte Anfragen bezüglich derselben Signatur garantiert.

In der Tat nimmt $\mathcal{F}_{\text{SIG}}^{(i)}$ keinen Schaden, wenn gleichbleibende Verifikationsresultate auch bei korrumpiertem Signierer P_i erzwungen werden. Dies könnte geschehen, indem im ersten Unterpunkt der Beschreibung der Signaturverifikation die Bedingung „$v = v'$“ ersatzlos gestrichen wird. Der nachfolgende Beweis für die erwünschte Äquivalenz zu EF-CMA-Sicherheit bleibt dabei uneingeschränkt gültig.

Ist allerdings der Signierer korrumpiert, kann ohnehin nicht mehr von idealen Sicherheitsgarantien gesprochen werden, da die Gültigkeit von Signaturen allein vom Angreifer festgelegt wird. Dies gilt sowohl für $\mathcal{F}_{\text{SIG}}^{(i)}$ als auch für die in [29, 30, 31, 32] gegebenen Korrekturen von $\mathcal{F}_{\text{SIG}}$ und ist – wie der Beweis von Satz 3.9 aufzeigt – ein inhärentes Problem jeder (realisierbaren) direkten Idealisierung eines Public-Key-Signatursystems.

3.3.5 Ein korrigierter Äquivalenzbeweis

Nun soll die ursprünglich angestrebte Äquivalenz von $\mathcal{F}_{\text{SIG}}^{(i)}$-Sicherheit und EF-CMA-Sicherheit für ein Signatursystem gezeigt werden. Es muß dafür noch fixiert werden, wie genau ein Signatursystem $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ als reales Protokoll $\pi_S^{(i)}$ (mit signierender Partei P_i) aufgefaßt werden kann. Da mit $\mathcal{F}_{\text{SIG}}^{(i)}$ für Verifikationsanfragen auch ein Verifikationsschlüssel eingeführt wurde, kann dies in völliger Analogie zum Fall eines Public-Key-Kryptosystems P geschehen.

In Protokoll $\pi_S^{(i)}$ beantwortet P_i (und nur P_i) die erste `Signer`-Eingabe mit einem mittels $\mathsf{KeyGen}(1^k)$ generierten öffentlichen Schlüssel; der entsprechende Signaturschlüssel wird nur lokal bei P_i gespeichert, um später auf `Sign`-Anfragen eine jeweilige mittels Sign generierte Signatur zu liefern. Alle anderen Parteien beantworten nur `Verify`-Anfragen, und dies naheliegenderweise durch lokale Ausführung des Verify-Algorithmus von S.

Satz 3.10. *Sei $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ ein Public-Key-Signatursystem. Dann sind folgende Aussagen äquivalent:*

(i) $\pi_S^{(i)} \geq \mathcal{F}_{\text{SIG}}^{(i)}$ *bezüglich adaptiver Angreifer,*

(ii) $\pi_S^{(i)} \geq \mathcal{F}_{\text{SIG}}^{(i)}$ *bezüglich nicht-adaptiver Angreifer,*

(iii) S ist EF-CMA-sicher.

Beweis. (i)⇒(ii) ist trivial, da eine Umgebung $\mathcal{Z}$ über alle Korruptionen informiert wird, vgl. Abschnitt 3.1.4.

Zum Beweis von (ii)⇒(iii) muß ein EF-CMA-Angriff auf einen nicht-adaptiven $\pi_S^{(i)}$-Angriff reduziert werden. Sei also für einen EF-CMA-Angreifer A eine Umgebung $\mathcal{Z} = \mathcal{Z}(A)$ wie folgt definiert:

$\mathcal{Z}$ führt mit einer Simulation von A das Experiment $\mathbf{Exp}_{S,A}^{\text{ef-cma}}$ aus Definition 3.8 durch. Hierbei wird die initiale KeyGen-Schlüsselgenerierung durch eine `KeyGen`-

Anfrage bei P_i ersetzt. Weiter werden von A durchgeführte Aufrufe des Sign-Orakels in `Sign`-Anfragen an P_i umgewandelt.

Die Gültigkeit der von A gelieferten Signatur s zur Nachricht m wird mittels einer `Verify`-Anfrage bei P_i überprüft. Schließlich gibt $\mathcal{Z}$ genau dann eine 1 aus, wenn die von A gelieferte Signatur gültig ist, also wenn das Experiment „`valid`“ liefert. Aus technischen Gründen hält $\mathcal{Z}$ auch mit 0, sobald eine Partei korrumpiert wird – dies bedeutet nämlich zwangsläufig, daß ein Protokollablauf im idealen Modell stattfindet. (Es wird hier angenommen, daß $\mathcal{Z}$ im realen Modell mit dem Dummy-Angreifer $\tilde{\mathcal{A}}$ ausgeführt wird, welcher Korruptionen nur auf explizite Anweisung von $\mathcal{Z}$ durchführt.)

Nach Konstruktion gilt damit für beliebiges $k \in \mathbb{N}$:

$$\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \pi_S^{(i)}, \tilde{\mathcal{A}}\right](k) = \mathsf{Pr}\left[\mathbf{Exp}_{S,A}^{\text{ef-cma}}(k) \to 1\right].$$

Da A als EF-CMA-Angreifer nie eine Signatur zu einer schon signierten Nachricht ausgibt, gilt weiter $\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \mathcal{F}_{\text{SIG}}^{(i)}, \mathcal{S}\right](k) = 0$ für beliebigen, festen Simulator $\mathcal{S}$. Hier ist zu beachten, daß, sofern im idealen Modell keine Partei korrumpiert wird (was sofort zu einer 0-Ausgabe von $\mathcal{Z}$ führt), keine Nachrichten durch $\mathcal{S}$ signiert werden können.

Somit ist

$$\mathbf{Adv}_{\tilde{\mathcal{A}},\mathcal{S},\mathcal{Z}}^{\pi_S^{(i)},\mathcal{F}_{\text{SIG}}^{(i)}}(k) = \mathsf{Pr}\left[\mathbf{Exp}_{S,A}^{\text{ef-cma}}(k) \to 1\right],$$

was ¬(iii)⇒ ¬(ii) und damit (ii)⇒(iii) zeigt.

Um den Beweis zu beenden, muß noch (iii)⇒(i) gezeigt werden. Hierfür wird ein adaptiver Angriff auf $\pi_S^{(i)}$ auf einen EF-CMA-Angriff auf S reduziert. Es reicht dafür aus, Angriffe bezüglich des Dummy-Angreifers aus Abschnitt 3.1.5 zu betrachten. Zunächst sei für diesen realen Dummy-Angreifer der Simulator $\mathcal{S}_{\text{SIG}}^{(i)}$ aus Abbildung 3.6 fixiert.

Sei also eine beliebige Umgebung $\mathcal{Z}$ gegeben, welche im realen Modell mit dem Dummy-Angreifer arbeitet. Aufgrund der „lokalen“ Struktur von Protokoll $\pi_S^{(i)}$ werden Anfragen an den Simulator $\mathcal{S}_{\text{SIG}}^{(i)}$ genau wie Anfragen an den realen Angreifer $\tilde{\mathcal{A}}$ beantwortet. $\mathcal{Z}$ kann somit reales und ideales Modell nur durch Ein- und Ausgaben von unkorrumpierten Parteien unterscheiden.

Das folgende Ereignis B wird in Analogie zu [40] definiert. Sei B das Ereignis, daß zu einem Zeitpunkt, zu dem die signierende Partei P_i *nicht* korrumpiert ist, $\mathcal{Z}$ eine Partei nach Verifikation einer (im Sinne von S) gültigen Signatur zu einer noch nicht vorher signierten Nachricht fragt. Sei $\neg B$ das Ereignis, daß B nicht eintritt.[35]

Nach Konstruktion von $\mathcal{F}_{\text{SIG}}$ und $\mathcal{S}_{\text{SIG}}^{(i)}$ unterscheidet sich die Sicht von $\mathcal{Z}$ im realen und idealen Modell nicht, bis B eintritt: Ist P_i korrumpiert, werden alle Verifikationsanfragen für Signaturen, die nicht explizit durch $\mathcal{F}_{\text{SIG}}^{(i)}$ erzeugt oder schon durch $\mathcal{S}_{\text{SIG}}^{(i)}$ verifiziert wurden, an $\mathcal{S}_{\text{SIG}}^{(i)}$ weitergeleitet. In allen Fällen ist das Ergebnis

[35] Man beachte, daß hier die Eindeutigkeit der Verifikation genutzt wird.

Der Simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$

- **Kommunikation mit $\mathcal{F}_{\mathrm{SIG}}^{(i)}$:**
  - Bei Erhalt von $(\mathtt{Key}, sid)$ von $\mathcal{F}_{\mathrm{SIG}}^{(i)}$: Speichere $(v, s) \leftarrow \mathsf{KeyGen}(1^k)$ und sende $(\mathtt{Key}, sid, v)$ an $\mathcal{F}_{\mathrm{SIG}}^{(i)}$.
  - Bei Erhalt von $(\mathtt{Sign}, sid, m)$ von $\mathcal{F}_{\mathrm{SIG}}^{(i)}$: Setze $\sigma \leftarrow \mathsf{Sign}(1^k, s, m)$ und sende $(\mathtt{Signature}, sid, m, \sigma)$ an $\mathcal{F}_{\mathrm{SIG}}^{(i)}$; falls s noch ungesetzt ist, tue nichts.
  - Bei Erhalt einer Nachricht $(\mathtt{Verify}, sid, m, \sigma, v')$ von $\mathcal{F}_{\mathrm{SIG}}^{(i)}$: Setze $f \leftarrow \mathsf{Verify}(1^k, v', m, \sigma)$ und sende $(\mathtt{Verified}, sid, m, f)$ an $\mathcal{F}_{\mathrm{SIG}}^{(i)}$.
  - Liefere auf Anfrage von $\mathcal{F}_{\mathrm{SIG}}^{(i)}$ Nachrichten an die Parteien direkt aus.
- **Kommunikation mit $\mathcal{Z}$:**
  - Melde auf Anfrage, daß keine Nachrichten von Parteien gesendet wurden.
  - Befehle, eine Nachricht auszuliefern, werden gespeichert.
  - Bei einem Befehl zur Korruption von P_j: Korrumpiere zunächst die Dummy-Partei $\tilde{P}_j$, um Informationen über die Kommunikation von $\tilde{P}_j$ mit $\mathcal{Z}$ zu bekommen. Bereite dann einen P_j-Zustand (inklusive benutztem Zufallsband) vor, welcher außerdem alle an P_j ausgelieferte Nachrichten enthält. Für $i = j$, füge außerdem den Signaturschlüssel s hinzu, falls schon generiert. Liefere diesen Zustand an $\mathcal{Z}$ zurück.

Abbildung 3.6: Der Simulator $\mathcal{S}_{\mathrm{SIG}}^{(i)}$

der Verifikation dasselbe wie im realen Modell. Ist andererseits P_i unkorrumpiert, ist die Behauptung klar.

Damit folgt für jeden Sicherheitsparameter $k \in \mathbb{N}$:

$$\mathsf{Pr}\left[B \mid \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}\right] = \mathsf{Pr}\left[B \mid \pi_S^{(i)}, \tilde{\mathcal{A}}\right] \tag{3.3}$$

$$\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \neg B, \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}\right] = \mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \neg B, \pi_S^{(i)}, \tilde{\mathcal{A}}\right]. \tag{3.4}$$

Sei nun A folgender EF-CMA-Angreifer auf S: A simuliert einen Protokollablauf mit $\mathcal{Z}$, $\tilde{\mathcal{A}}$ und Protokoll $\pi_S^{(i)}$ im realen Modell. Dabei werden alle bei P_i generierten Signaturen an das Sign-Orakel von A weitergeleitet, und eine initiale Schlüsselgenerierungsanfrage mit der Eingabe vk von A beantwortet.

Tritt B ein, läßt also $\mathcal{Z}$ bei unkorrumpiertem P_i eine gültige Signatur σ zu einer noch nicht mittels P_i signierten Nachricht m verifizieren, hält A mit Ausgabe (m, σ). In allen anderen Fällen (insbesondere, sobald P_i korrumpiert werden soll)

hält A mit einer Fehlermeldung an.

Für den Erfolg von A gilt dann wegen (3.3) und (3.4):

$$\begin{aligned}
&\left|\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \pi_S^{(i)}, \tilde{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathcal{Z} \to 1 \mid \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}\right]\right| \\
&= \left|\mathsf{Pr}[B] \cdot \left(\mathsf{Pr}\left[\mathcal{Z} \to 1 \mid B, \pi_S^{(i)}, \tilde{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathcal{Z} \to 1 \mid B, \mathcal{F}_{\mathrm{SIG}}^{(i)}, \mathcal{S}_{\mathrm{SIG}}^{(i)}\right]\right)\right| \\
&\leq \mathsf{Pr}[B] = \mathsf{Pr}\left[\mathbf{Exp}_{A,S}^{\text{ef-cma}} \to \texttt{valid}\right],
\end{aligned}$$

was ¬(i)⇒ ¬(iii) und damit (iii)⇒(i) zeigt. □

3.3.6 Offene Fragen

Natürlich ist die Eigenschaft von $\mathcal{F}_{\mathrm{SIG}}^{(i)}$, im Falle eines korrumpierten Signierers nur verminderte Sicherheitsgarantien zu geben, alles andere als wünschenswert. Eine solche Eigenschaft ist für eine Idealisierung von digitalen Signaturen allerdings, wie gezeigt wurde, unvermeidbar. Hier ergibt sich eine interessante Frage: Kann unter geeigneten *Zusatzannahmen*, also in einem geeigneten hybriden Modell, eine stärkere Funktionalität wie etwa das ursprünglich vorgeschlagene $\mathcal{F}_{\mathrm{SIG}}$ realisiert werden?

Eine solche Frage ist insbesondere für den Einsatz von digitalen Signaturen in größeren Protokollen interessant. Möglich scheint beispielsweise eine Konstruktion analog zur in Abschnitt 4.2 vorgestellten Protokollkonstruktion, welche Random Oracles benutzt. Im allgemeinen muß aber darauf geachtet werden, daß die benutzte Annahme nicht zu absurd ist – es ist etwa $\mathcal{F}_{\mathrm{SIG}}$ banalerweise in einem $\mathcal{F}_{\mathrm{SIG}}$-hybriden Modell realisierbar, was natürlich ein vollkommen uninteressantes Ergebnis darstellt.

Auch könnten „key substitution attacks“ (siehe etwa Blake-Wilson und Menezes [22], Rosa [124], Menezes und Smart [108], Geiselmann und Steinwandt [67], Tan [133]) in Betracht gezogen werden: Bei einem solchen Angriff wird versucht, zu einem gegebenen Verifikationsschlüssel und einer Signatur einen *anderen* Verifikationsschlüssel zu finden, so daß die Signatur auch bezüglich dieses neuen Schlüssels gültig ist. Zum Einfangen einer derartigen Sicherheitseigenschaft müßte natürlich eine ideale Funktionalität konstruiert werden, welche mehrere Verifikationsschlüssel in einer einzigen Instanz verwaltet.

4 Weitere Idealisierungen

Nachdem im vorigen Kapitel für den Fall von Public-Key-Systemen eine intuitive Modellierung betrachtet wurde, stellt sich die Frage, ob sich auch andere kryptographische Aufgaben ähnlich elegant darstellen und untersuchen lassen. Tatsächlich konnten schon in Canetti [27] für viele Aufgaben intuitive Idealisierungen angegeben werden. Zwei spezielle Aufgabengebiete sollen in diesem Kapitel näher untersucht werden. Es handelt sich dabei um die Protokollaufgabe „Schlüsselaustausch“, die im Zusammenhang mit symmetrischen Kryptosystemen eine wichtige Rolle spielt und um den theoretisch bedeutsamen Protokollbaustein „Bit Commitment“ bzw. „Commitment“.

4.1 Schlüsselaustausch

Die Benutzung symmetrischer Kryptosysteme setzt einen den Benutzern gemeinsamen Schlüssel voraus. Im Gegensatz zu Public-Key-Kryptosystemen ist dabei tunlichst zu vermeiden, diesen Schlüssel direkt im Klartext zu übertragen, denn unter Kenntnis dieses Schlüssel lassen sich nicht nur Chiffrate generieren, sondern auch entschlüsseln (daher der Name *symmetrisches* Kryptosystem). Kann also ein gemeinsamer Schlüssel nicht von vornherein vorausgesetzt werden, wird ein *Schlüsselaustauschverfahren* benötigt.

Im folgenden soll sich der Einfachheit halber auf einen Schlüsselaustausch zwischen *zwei* Parteien beschränkt werden; eine Untersuchung von Verfahren zum „Group Key Agreement“ unter mehreren Parteien im Umfeld simulierbarer Sicherheitsdefinitionen ist beispielsweise in Steiner [130] zu finden.

Zudem sollen ohne weitere Diskussion authentifizierte Kanäle benutzt werden; diese können schließlich mit den in in Canetti [31] vorgestellten Methoden und unter Benutzung der im vorigen Kapitel ausführlich besprochenen Idealisierungen $\mathcal{F}_{\text{SIG}}$ bzw. $\mathcal{F}_{\text{SIG}}^{(i)}$ implementiert werden.

In diesem Umfeld konnte in Shoup [127] ein simulationsbasierter Sicherheitsbegriff für Schlüsselaustauschprotokolle angegeben werden – dort wird der Vergleich zu einer Idealisierung von Schlüsselaustauschprotokollen herangezogen, um die Sicherheit eines gegebenen realen Protokolls zu beurteilen. Eine ähnliche Herangehensweise wurde in Canetti und Krawczyk [35] gewählt; in beiden Arbeiten konnten hinreichende Kriterien für die Implementierung eines sicheren Kanals gegeben werden. Unklar ist dabei jedoch, in welcher Weise dieser sichere Kanal selbst komponiert werden kann; eine Verwendung in einem allgemeinen Simulierbarkeitsmodell wie [27] scheint nicht ohne weiteres möglich.

Deshalb wurde in [27] und Canetti und Krawczyk [36] eine in diesem Sinne komponierbare Idealisierung der Protokollaufgabe „Schlüsselaustausch“ in Form einer idealen Funktionalität $\mathcal{F}_{\mathrm{KE}}$ angegeben. Für die Variante aus [36] wurde dann gezeigt, wie mittels $\mathcal{F}_{\mathrm{KE}}$ sichere Kanäle – gleichsam in Form einer idealen Funktionalität $\mathcal{F}_{\mathrm{SMT}}$ zur sicheren Nachrichtenübertragung – implementiert werden können.

Bezüglich der Frage der Realisierbarkeit von $\mathcal{F}_{\mathrm{KE}}$ selbst wurden in [36] mehrere einfache und praktikable Protokolle angegeben, von welchen behauptet wurde, daß sie $\mathcal{F}_{\mathrm{KE}}$ simulierbar sicher realisieren. Diese Funktionalität $\mathcal{F}_{\mathrm{KE}}$ aus [36] soll nun näher untersucht werden. Es wird sich dabei herausstellen, daß – abgesehen von einer unsauberen Formulierung, welche strenggenommen schon eine Nicht-Realisierbarkeit von $\mathcal{F}_{\mathrm{KE}}$ zur Folge hat – ein allgemeiner Angriff auf $\mathcal{F}_{\mathrm{KE}}$ existiert, welcher sich auch auf andere ideale Funktionalitäten übertragen läßt. (Die dabei gemachten Beobachtungen übertragen sich sofort auf die ältere Variante von $\mathcal{F}_{\mathrm{KE}}$ aus [27].)

Anschließend kann eine korrigierte Idealisierung $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ angegeben werden, für welche die Realisierbarkeit mittels einfacher Protokolle gezeigt wird. Weiter werden konkrete Kriterien für Sicherheit bezüglich $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ aufgezeigt. Schließlich wird eine stärkere Idealisierung $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ motiviert, vorgestellt und bezüglich ihrer Implementierbarkeit untersucht. Es kann dabei ein allgemeines Konstruktionsverfahren für (im Sinne von $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$) sichere Schlüsselaustauschprotokolle angegeben werden.

Zunächst eine technische Vorbemerkung: Die in Abschnitt 3.1 vorgestellte Grundform des Modells aus [27] sieht vor, daß im Falle einer Korruption einer Partei der Angreifer über den aktuellen Zustand *und* alle vorigen Zustände der jeweiligen ITM informiert wird. Damit wird modelliert, daß es einer Partei nicht zuverlässig möglich ist, Daten zu löschen. Im Hinblick auf die Protokollaufgabe „Schlüsselaustausch“ ist es jedoch aus technischen Gründen wünschenswert, *löschende* Parteien zu modellieren – mehr dazu später.

Die Modellierung löschender Parteien kann nun – wie schon in [36] vorgeschlagen – ganz einfach dadurch geschehen, daß ein Angreifer bei einer Korruption nur den *gegenwärtigen* Zustand einer ITM erhält. Damit kann eine ITM eine Löschung von Daten durch Überschreiben der jeweiligen Zellen auf ihrem Arbeitsband herbeiführen.[1] Ein solches Modell mit löschenden Parteien soll nun für den speziellen Fall der Protokollaufgabe Schlüsselaustausch vorausgesetzt werden.

4.1.1 Eine problematische Idealisierung

Um einen Anfang zu machen, sei in Abbildung 4.1 die Idealisierung $\mathcal{F}_{\mathrm{KE}}$ eines Schlüsselaustausches aus [36] reproduziert.

Im Unterschied zu den schon vorgestellten idealen Funktionalitäten für Public-Key-Kryptosysteme und Signatursysteme ist $\mathcal{F}_{\mathrm{KE}}$ keine direkte Funktionalität. Es können also alle Ausgaben von $\mathcal{F}_{\mathrm{KE}}$ vom Simulator beliebig verzögert werden.

[1] Weder [27] noch [36] legen fest, ob Nachrichten auf Kommunikations- und Ein- bzw. Ausgabebändern automatisch zwischen den Aktivierungen gelöscht werden. Hiervon muß nun für eine sinnvolle Modellierung von löschenden Parteien ausgegangen werden.

Die Funktionalität $\mathcal{F}_{\text{KE}}$

Protokollteilnehmer: $\mathcal{F}_{\text{KE}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. Bei Erhalt von $(\texttt{Establish-session}, sid, P_i, P_j, role)$ von einer Partei P_i, speichere das Tupel $(sid, P_i, P_j, role)$ und sende dieses Tupel an den Angreifer. Zusätzlich, wenn ein gespeichertes Tupel $(sid, P_j, P_i, role')$ existiert (entweder mit $role' \neq role$ oder $role' = role$), dann:
   (a) Wenn P_i und P_j beide unkorrumpiert sind, wähle $\kappa \xleftarrow{R} \{0,1\}^k$, sende $(\texttt{Key}, sid, \kappa)$ an P_i und P_j, sende $(\texttt{Key}, sid, P_i, P_j)$ an den Angreifer und halte an.
   (b) Wenn entweder P_i oder P_j korrumpiert ist, sende eine Nachricht $(\texttt{Choose-value}, sid, P_i, P_j)$ an den Angreifer; erhalte einen Wert κ vom Angreifer, sende $(\texttt{Key}, sid, \kappa)$ an P_i und P_j und halte an.
2. Bei Korruption von entweder P_i oder P_j: Wenn der Sitzungsschlüssel noch nicht gesendet wurde (d. h. er wurde noch nicht auf das Ausgangs-Kommunikationsband geschrieben), versorge $\mathcal{S}$ mit dem Sitzungsschlüssel. Ansonsten gib $\mathcal{S}$ keine Information.

Abbildung 4.1: Die Idealisierung eines Schlüsselaustausches aus [36]

Dies hat zur Folge, daß durch Wahl eines geeigneten Simulators, welcher keine Ausgabe von $\mathcal{F}_{\text{KE}}$ je ausliefert, eine „stumme“ Funktionalität erzwungen wird. Eine solche stumme Funktionalität kann durch ein geeignetes, gleichermaßen stummes Protokoll realisiert werden, ohne daß damit irgendein praktischer Nutzen einhergeht. Eine eingehendere Diskussion findet in Abschnitt 4.1.2 statt; bis dahin soll von – in einem intuitiven Sinn – nicht-trivialen Realisierungen von $\mathcal{F}_{\text{KE}}$ ausgegangen werden.

Weiter ist interessant, daß $\mathcal{F}_{\text{KE}}$ keine explizite statistische Verteilung der Sitzungsschlüssel vorgibt. (In diesem Sinne ist $\mathcal{F}_{\text{KE}}$ demnach unterspezifiziert.) Natürlich ist hier eine Parametrisierung über die möglichen statistischen Verteilungen der Sitzungsschlüssel denkbar – um einem realen Protokoll, welches $\mathcal{F}_{\text{KE}}$ realisieren soll, in gewisser Weise „entgegenzukommen“. Andererseits hat dieses Vorgehen den Nachteil, daß Protokollen, welche $\mathcal{F}_{\text{KE}}$ etwa in naheliegender Weise für die Erzeugung von Schlüsseln für ein symmetrisches Kryptosystem nutzen, keine Garantien bezüglich der statistischen Verteilung der Sitzungsschlüssel gegeben werden können. Das kann – abhängig vom konkret gewählten symmetrischen Kryptosystem – problematisch sein.

Gibt $\mathcal{F}_{\text{KE}}$ jedoch die Garantie eines etwa *gleichverteilten* k-Bit-Sitzungsschlüssels, so muß natürlich ein reales, zur Realisierung von $\mathcal{F}_{\text{KE}}$ herangezogenes Schlüsselaustauschprotokoll gewährleisten, daß auch die real erzeugten Schlüssel ununterscheidbar von gleichverteilten Schlüsseln sind. Etwa im Falle eines Diffie-Hellman-

basierten Protokolls ist damit eine *Nachbearbeitung* des Rohschlüssels g^{xy} nötig; Eine solche Nachbearbeitung findet allerdings in [36] – insbesondere für das dort vorgestellte Diffie-Hellman-basierte Protokoll SIG-DH – nicht statt.

Es ist weiter zu bemerken, daß auch die Beschreibung von $\mathcal{F}_{\text{KE}}$ Unsauberkeiten aufweist: Etwa ist im zweiten Schritt der Beschreibung zunächst unklar, welche Parteien mit P_i und P_j gemeint sind – prinzipiell ermöglicht $\mathcal{F}_{\text{KE}}$ ja beliebigen Parteien einen Schlüsselaustausch. Zudem sorgt die in diesem zweiten Schritt beschriebene Fallunterscheidung „[...] Wenn der Sitzungsschlüssel noch nicht gesendet wurde, [...]“ für Verwirrung: Wenn der Sitzungsschlüssel gesendet wurde, hat $\mathcal{F}_{\text{KE}}$ nach Konstruktion schon angehalten, kann also gar nicht mehr aktiviert worden sein. Wurde umgekehrt der Sitzungsschlüssel noch nicht gesendet, kann er auch noch nicht bestimmt sein; insofern ist nicht klar, wie der Angreifer dann mit dem Sitzungsschlüssel versorgt werden kann.

Auch ist aufgrund der dynamischen Zuweisung von Rollen an Protokollteilnehmer ein trivialer Angriff möglich: Angenommen, es finden `Establish-session`-Eingaben an ein Teilnehmerpaar P_i, P_j *und* an ein Teilnehmerpaar $P_{i'}, P_{j'}$ statt. Dann legt im idealen Modell die Reihenfolge dieser Eingaben fest, welches Paar Schlüssel erhält. Im realen Modell hingegen kann die Reihenfolge dieser Eingaben von keinem Protokoll festgestellt werden, sofern etwa zwischen diesen Eingaben keine Nachrichten zwischen Parteien ausgeliefert werden. Hier tritt einmal mehr zu Tage, daß Eingaben *direkt* (d. h. ohne Verzögerungsmöglichkeiten) zur idealen Funktionalität gelangen; nur Ausgaben dürfen vom Simulator verzögert oder sogar blockiert werden.

Es ist klar, daß eine Parametrisierung von $\mathcal{F}_{\text{KE}}$ über die beiden Teilnehmer P_i und P_j des Schlüsselaustausches im konkreten Fall von $\mathcal{F}_{\text{KE}}$ Abhilfe schafft. Jedoch kämpft $\mathcal{F}_{\text{KE}}$ mit einem noch fataleren Problem: Der Sitzungsschlüssel κ wird *sofort* nach Eingang der `Establish-session`-Eingaben festgelegt, also zu Beginn des eigentlichen Protokolls; man sollte aber meinen, daß auch nach diesen Eingaben noch wenigstens eine Partei Einfluß auf den Sitzungsschlüssel hat. Ein geeigneter (im Detail noch vorzustellender) adaptiver Angriff zeigt nun, daß dieses Problem eine Nicht-Realisierbarkeit von $\mathcal{F}_{\text{KE}}$ zur Folge hat.

Dies falsifiziert insbesondere die in [36] geführten Sicherheitsbeweise der dort untersuchten Protokolle. Zwar wurde dort eine Sicherheit von $\mathcal{F}_{\text{KE}}$ auch bezüglich adaptiver Angriffe behauptet, aber in den Beweisen einfach nicht berücksichtigt. (Ähnlich wie im Falle von $\mathcal{F}_{\text{SIG}}$ läßt sich hier der Fehler in den entsprechenden Beweisen aufgrund der Struktur dieser Beweise nicht genau lokalisieren.)

4.1.2 Zwischenspiel: Nicht-triviale Protokolle

Eine wichtige technische Bemerkung ist der Vorstellung des angedeuteten Angriffs vorauszuschicken: Aus formalen Gründen ist $\mathcal{F}_{\text{KE}}$ sehr wohl realisierbar durch allgemeine Protokolle. Beispielsweise ist das triviale Protokoll, welches nie Ausgaben generiert, offensichtlich eine sichere Realisierung von $\mathcal{F}_{\text{KE}}$. Um solche „Artefakte“ der Simulierbarkeitsdefinition (genauer: der Modellierung eines asynchronen

Netzwerks) zu vermeiden, nennt [27] ein Protokoll *nicht-trivial*, sofern es Ausgabe generiert, wenn keine Partei korrumpiert wird und alle Nachrichten ausgeliefert werden.

Diese Definition ist unsauber – zum einen ist unklar, was „Ausgabe generieren" relativ zu einer gegebenen Idealisierung meint. Bei einer insbesondere *probabilistischen* Idealisierung besteht nämlich keine offensichtliche Korrespondenz zwischen Zuständen der idealen Funktionalität und Zuständen der realen Parteien – insofern kann im allgemeinen nicht entschieden werden, wann eine reale Partei denn Ausgabe „zu generieren hätte". Zum anderen ist die Formulierung „alle Nachrichten ausliefern" äußerst problematisch: Wie soll ein Angreifer bei sich anstauenden Nachrichten verfahren?[2] In jeder Aktivierung eine der angestauten Nachrichten auszuliefern könnte ihn faktisch „lähmen", da nun keine eigenen Nachrichten mehr ausgeliefert werden können.

Eine andere Definition für *nicht-trivial* wird in Canetti u. a. [39] angegeben: Hier wird in einer leicht geänderten Sicherheitsdefinition gefordert, daß ein idealer Angreifer alle Nachrichten ausliefert und keine Partei korrumpiert, wenn der entsprechende reale Angreifer alle Nachrichten ausliefert und keine Partei korrumpiert. Auch hier ist mit dem selben Argument wie oben unklar, wann ein Angreifer „alle Nachrichten ausliefert".

Eine saubere, jedoch technisch aufwendige Definition von „nicht-trivial" kann für das verwandte Mehrparteienmodell Pfitzmann und Waidner [117], Backes u. a. [11] in Backes, Hofheinz, Müller-Quade und Unruh [8] gegeben werden. Da jedoch dabei einige Eigenheiten des Modells [117, 11] ausgenutzt werden, ist eine Adaption für [27] nicht offensichtlich. Deshalb soll hier folgende Konkretisierung des Begriffs „nicht-trivial" für den speziellen Fall der Idealisierung $\mathcal{F}_{\text{KE}}$ vorgenommen werden (die folgende Definition überträgt sich dabei direkt auf die später noch zu formulierenden Varianten von $\mathcal{F}_{\text{KE}}$):

Definition 4.1. *Ein Protokoll π heiße* nicht-triviale Realisierung von $\mathcal{F}_{\text{KE}}$*, wenn π nicht nur $\mathcal{F}_{\text{KE}}$ realisiert, sondern außerdem ein Polynom $p \in \mathbb{Z}[k]$ mit $p(k) > 0$ für alle $k \in \mathbb{N}$ existiert, so daß für beliebige Parteiindizes i, j und Sitzungsnummern sid gilt: Bei Ausführungen von π mit*

- *einer Umgebung $\mathcal{Z}_{p,sid}^{(i,j)}$, welche in ihrer ersten Aktivierung P_i mit der Eingabe* (`Establish-session`, *sid*, P_i, P_j, `initiator`) *aktiviert, in ihrer zweiten Aktivierung P_j mit der Eingabe* (`Establish-session`, *sid*, P_j, P_i, `responder`) *aktiviert, in den nächsten $p(k)$ Aktivierungen mit einer Nachricht* `token` *den Angreifer aktiviert und danach schließlich anhält;*

- *einem Angreifer $\mathcal{A}$, welcher in jeder Aktivierung die lexikographisch erste zwischen zwei Parteien gesandte Nachricht ausliefert (existiert keine solche Nachricht, geht $\mathcal{A}$ in einen Wartezustand),*

[2] Aufgrund der nachrichtengetriebenen Natur des Schedulings darf ein Angreifer immer nur *eine* Nachricht pro Aktivierung ausliefern, vgl. Abschnitt 3.1.2.

geben mit überwältigender Wahrscheinlichkeit sowohl P_i als auch P_j Ausgaben der Form $(\texttt{Key}, sid, \kappa)$*, wobei nicht gefordert wird, daß κ in beiden Fällen identisch ist.*

Natürlich ist diese Definition speziell auf ein Schlüsselaustauschprotokoll zugeschnitten, jedoch sind nur Eigenschaften erfaßt, welche man schon aus intuitiver Nicht-Trivialität folgern würde:

- Das Protokoll π soll nach polynomial vielen Nachrichten mit nicht notwendig gleichen Ausgaben bei P_i und P_j terminieren. (Die Gleichheit der von P_i und P_j erzeugten Sitzungsschlüssel würde dann aus $\pi \geq \mathcal{F}_{\text{KE}}$ folgen.)
- Hierbei werden alle Nachrichten zwischen beliebigen Parteien ausgeliefert; damit sind sogar Schlüsselaustauschverfahren mit „Helferparteien“ erlaubt. (Natürlich kann Definition 4.1 bei diesbezüglichem Nicht-Gefallen so angepaßt werden, daß nur Nachrichten zwischen P_i und P_j ausgeliefert werden; auf alles Weitere hat das keinen Einfluß.)
- Weiter wird keine Partei korrumpiert. Es muß also schon bei einer einzigen korrumpierten Partei keine Ausgabe mehr garantiert sein.

4.1.3 Noch ein Unmöglichkeitsresultat

Umso erstaunlicher ist das sogleich vorzustellende Resultat von Hofheinz, Müller-Quade und Steinwandt aus [84]:

Satz 4.2. *Es existiert kein reales Protokoll π für $n \geq 2$ Parteien, welches eine nicht-triviale Realisierung der idealen Funktionalität $\mathcal{F}_{\text{KE}}$ aus [36] ist.*

Beweis. Sei zum Widerspruch angenommen, daß ein im realen Modell formuliertes Protokoll π eine nicht-triviale Realisierung von $\mathcal{F}_{\text{KE}}$ sei. Sei dann $p \in \mathbb{Z}[k]$ das zugehörige[3] Polynom aus Definition 4.1. Seien weiter mit P_i und P_j zwei verschiedene Parteien fixiert. Es bezeichne $\mathcal{S}$ den Simulator, der im idealen Modell reale Angriffe des Dummy-Angreifers $\tilde{\mathcal{A}}$ simuliert.

Zunächst soll gezeigt werden, daß bei Auslieferung aller Nachrichten auch im idealen Modell Ausgabe generiert wird. Sei dazu $\mathcal{Z}_1$ die Umgebung, die wie $\mathcal{Z}_{p,sid}^{(i,j)}$ (mit $sid = 0$) zunächst P_i und P_j mit `Establish-session`-Eingaben initialisiert und dann den Angreifer anweist, wie $\mathcal{A}$ aus Definition 4.1 alle Nachrichten zwischen den Parteien auszuliefern. Haben sowohl P_i als auch P_j nach höchstens $p(k)$ Nachrichtenauslieferungen eine `Key`-Ausgabe generiert, hält $\mathcal{Z}_1$ mit Ausgabe 1; ist dies nach $p(k)$ Nachrichtenauslieferungen nicht geschehen, hält $\mathcal{Z}_1$ mit Ausgabe 0.

Da π nicht-trivial gemäß Definition 4.1 ist, ist $\mathsf{Pr}\left[\mathcal{Z}_1 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$ überwältigend in k. Damit ist auch $\mathsf{Pr}[\mathcal{Z}_1 \to 1 \mid \mathcal{F}_{\text{KE}}, \mathcal{S}]$ überwältigend in k. Für eine kleine Abwandlung $\mathcal{Z}_2$ von $\mathcal{Z}_1$, welche für eine 1-Ausgabe zusätzlich *Gleichheit* der ausgegebenen Schlüssel fordert, ist nach Konstruktion von $\mathcal{F}_{\text{KE}}$ $\mathsf{Pr}[\mathcal{Z}_2 \to 1 \mid \mathcal{F}_{\text{KE}}, \mathcal{S}]$

[3] Natürlich kommt es hier nicht auf die spezielle Wahl von p an.

überwältigend in k, womit schließlich $\mathsf{Pr}\left[\mathcal{Z}_2 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$ als überwältigend in k gezeigt ist.

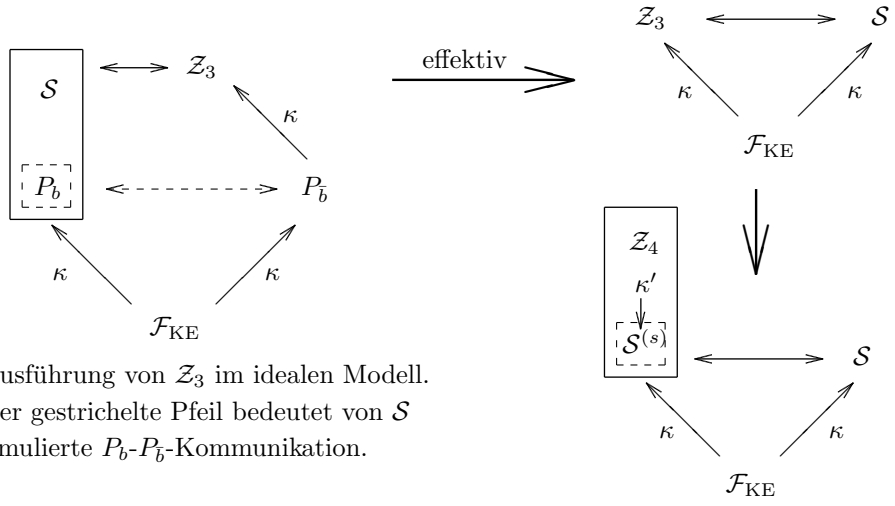
Soweit wurde nur gezeigt, daß – bei Benutzung durch $\mathcal{Z}_1$ bzw. $\mathcal{Z}_2$ – das als sicher angenommene Protokoll π mit überwältigender Wahrscheinlichkeit beiden Protokollparteien P_i und P_j zu gleichen Schlüsseln verhilft. Der *eigentliche* Angriff, der letzten Endes den für den Beweis gewünschten Widerspruch erbringt, kann nun in Form der nachfolgend beschriebenen Umgebung $\mathcal{Z}_3$ vorgestellt werden.

1. Wähle gleichverteilt $(b, \bar{b})$ aus $\{(i,j),(j,i)\}$ und $c \in \{0,1\}$.
2. Aktiviere P_i mit Eingabe (`Establish-session`, sid, P_i, P_j, `initiator`).
3. Aktiviere P_j mit Eingabe (`Establish-session`, sid, P_j, P_i, `responder`).
4. Im Falle $c = 0$ (bzw. $c = 1$): Weise den Angreifer an, P_b (bzw. alle Parteien *außer* $P_{\bar{b}}$) zu korrumpieren. Bereite Simulationen $P_\ell^{(s)}$ der korrumpierten Parteien vor.
5. Gib der internen Simulation $P_b^{(s)}$ von P_b eine passende `Establish-session`-Eingabe und laß die korrumpierten Parteien (über den Angreifer) am Protokoll π teilnehmen.
6. Laß dabei den Angreifer die ersten $p(k)$ Nachrichten zwischen Parteien in jeweils lexikographischer Reihenfolge ausliefern. (Nachrichten an korrumpierte Parteien werden natürlich nur innerhalb der Simulation ausgeliefert.)
7. Geben sowohl $P_b^{(s)}$ als auch $P_{\bar{b}}$ *gleiche* Schlüssel aus, halte mit Ausgabe 1; sonst halte mit Ausgabe 0.

Zur Veranschaulichung von $\mathcal{Z}_3$ kann die linke Seite von Abbildung 4.2 benutzt werden: Zunächst weist $\mathcal{Z}_3$ den Angreifer (also im realen Modell $\tilde{\mathcal{A}}$ und im idealen Modell $\mathcal{S}$) an, P_b zu korrumpieren.[4] Anschließend übernimmt $\mathcal{Z}_3$ die Rolle der soeben korrumpierten Partei P_b.

Im realen Modell wird dazu $\tilde{\mathcal{A}}$ angewiesen, alle Nachrichten zwischen der $\mathcal{Z}_3$-internen Simulation $P_b^{(s)}$ und der echten – unkorrumpierten – Partei $P_{\bar{b}}$ zu vermitteln. Hier geschieht also ein „normaler“ Schlüsselaustausch zwischen $P_b^{(s)}$ und $P_{\bar{b}}$, und $\mathcal{Z}_3$ darf nach den Vorbetrachtungen erwarten, von beiden Parteien den gleichen Sitzungsschlüssel präsentiert zu bekommen. Denn mit $\mathsf{Pr}\left[\mathcal{Z}_2 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$ ist auch $\mathsf{Pr}\left[\mathcal{Z}_3 \to 1 \mid \pi, \tilde{\mathcal{A}}\right]$ überwältigend in k, da ja in $\mathcal{Z}_3$ nur eine teilweise Simulation von π stattfindet. Nach Annahme muß demnach auch $\mathsf{Pr}[\mathcal{Z}_3 \to 1 \mid \mathcal{F}_{\mathrm{KE}}, \mathcal{S}]$ überwältigend in k sein.

[4] Die Unterscheidung $c = 0$ bzw. $c = 1$ deckt nur Protokolle ab, welche mehrere „Helferparteien“ benutzen, um einen Schlüssel zu erzeugen; für den einfacheren Fall, daß nur zwei Parteien P_i und P_j kommunizieren, darf von $c = 0$ ausgegangen werden.



Ausführung von $\mathcal{Z}_3$ im idealen Modell. Der gestrichelte Pfeil bedeutet von $\mathcal{S}$ simulierte P_b-$P_{\bar{b}}$-Kommunikation.

Abbildung 4.2: Beweisskizze zu Satz 4.2

Auch im idealen Modell müssen also gleiche Schlüssel ausgegeben werden – hier fällt es dem Simulator $\mathcal{S}$ zu, die Kommunikation zwischen $P_b^{(s)}$ und $P_{\bar{b}}$ zu *simulieren*. (Denn im idealen Modell findet natürlich kein echter Protokolldurchlauf von π statt.) Mit anderen Worten: $\mathcal{S}$ muß für $\mathcal{Z}_3$ (bzw. $P_b^{(s)}$) in die Rolle der realen Partei $P_{\bar{b}}$ schlüpfen, und mit $P_b^{(s)}$ einen Schlüsselaustausch durchführen. Diese Situation ist in der rechten oberen Ecke von Abbildung 4.2 dargestellt.

Wie schon angesprochen muß $\mathcal{S}$ dabei sicherstellen, daß $P_b^{(s)}$ denselben Schlüssel ausgibt wie die unkorrumpierte Partei $P_{\bar{b}}$. Nun wird aber nach Definition des idealen Modells und von $\mathcal{F}_{\mathrm{KE}}$ der Schlüssel κ, den die Dummy-Partei $P_{\bar{b}}$ schließlich ausgibt, sofort nach Eintreffen der `Establish-session`-Eingaben gewählt.[5] Damit muß $\mathcal{S}$ also die simulierte Partei $P_b^{(s)}$ davon „überzeugen“, denselben Schlüssel κ auszugeben wie die unkorrumpierte Partei $P_{\bar{b}}$. Dies muß $\mathcal{S}$ in den Rollen von P_i *und* P_j möglich sein, denn $b \in \{i, j\}$ (also die zu korrumpierende Partei) wird ja gleichverteilt von $\mathcal{Z}_3$ gewählt.

Informell können also sowohl P_b als auch $P_{\bar{b}}$ beliebige Schlüssel beim jeweils anderen erzwingen. Diese Eigenschaft führt zu einem Widerspruch, wenn man sich vorstellt, daß beide Parteien unterschiedliche Schlüssel erzwingen wollen. Die Situation ist in der rechten unteren Ecke von Abbildung 4.2 dargestellt.

Genauer benutzt die nun beschriebene Umgebung $\mathcal{Z}_4$ als Trick eine Simulation von $\mathcal{S}$ selbst, um Schlüssel zu erzwingen. Bis zu Schritt 4 verhält sich $\mathcal{Z}_4$ wie $\mathcal{Z}_3$; jedoch wird bei $\mathcal{Z}_4$ anstelle von Parteisimulationen $P_\ell^{(s)}$ eine Simulation $\mathcal{S}^{(s)}$ des Simulators $\mathcal{S}$ vorbereitet und gestartet. Hierbei wird ein komplettes ideales Modell mit Protokollumgebung $\mathcal{Z}_3^{(s)}$ simuliert, wobei $\mathcal{Z}_3^{(s)}$ allerdings im ersten Schritt die zu den Belegungen der $\mathcal{Z}_4$-eigenen Variablen $(b, \bar{b})$ und c jeweils komplementären Werte $(\bar{b}, b)$ und $1 - c$ wählt.

[5] Hat $\mathcal{Z}_3$ im idealen Modell Schritt 4 erreicht, so hat $\mathcal{F}_{\mathrm{KE}}$ bereits κ gewählt, ausgegeben und angehalten!

Damit läßt $\mathcal{Z}_3^{(s)}$ genau die Parteien korrumpieren, welche von $\mathcal{Z}_4$ nicht korrumpiert wurden; $\mathcal{S}^{(s)}$ simuliert also genau die realen Parteien, welche von $\mathcal{Z}_4$ korrumpiert wurden. Damit ist klar, in welcher Weise $\mathcal{Z}_4$ zwischen realem Angreifer und $\mathcal{S}^{(s)}$ Nachrichten „vermittelt“. Schließlich hält $\mathcal{Z}_4$ mit 1-Ausgabe genau dann, wenn die unkorrumpierte Partei $P_{\bar{b}}$ einen Schlüssel ausgibt, der mit dem Schlüssel übereinstimmt, den die simulierte ideale Funktionalität $\mathcal{F}_{\mathrm{KE}}^{(s)}$ gewählt hat.

Nach Konstruktion ist damit

$$\mathsf{Pr}\left[\mathcal{Z}_4 \to 1 \mid \pi, \tilde{\mathcal{A}}\right] = \mathsf{Pr}[\mathcal{Z}_3 \to 1 \mid \mathcal{F}_{\mathrm{KE}}, \mathcal{S}]$$

überwältigend in k. Jedoch muß $\mathsf{Pr}[\mathcal{Z}_4 \to 1 \mid \mathcal{F}_{\mathrm{KE}}, \mathcal{S}]$ sogar vernachlässigbar sein, da die hier verglichenen und von $\mathcal{F}_{\mathrm{KE}}$ bzw. $\mathcal{F}_{\mathrm{KE}}^{(s)}$ gewählten Schlüssel *unabhängig* und gleichverteilt aus $\{0,1\}^k$ gezogen wurden. Dies ist ein Widerspruch zu $\pi \geq \mathcal{F}_{\mathrm{KE}}$. □

4.1.4 Zwischenspiel: Eine mögliche Modelländerung

Der gerade gegebene Beweis nutzt wie schon besprochen aus, daß Eingaben auf dem Weg *zur* idealen Funktionalität nicht vom Simulator verzögert werden können. Tatsächlich ist ein vollkommen analoger Angriff nach dem Muster

- gib einer Partei eine Eingabe,
- laß sie dann vom Angreifer korrumpieren,
- führe das Protokoll weiter mit einer simulierten Partei, welche eine andere Eingabe bekam

auch auf andere Funktionalitäten möglich. Hier sei als Prototyp etwa die Idealisierung $\mathcal{F}_{\mathrm{SFE}}$ einer sicheren Funktionsauswertung aus [27] genannt.

Es liegt nun nahe, über eine Änderung des Modells nachzudenken, bei welcher der Simulator auch Eingaben auf dem Weg zur idealen Funktionalität verzögern kann. Eine solche Modelländerung erfolgte tatsächlich – direkt, nachdem die Arbeit [84] zur Veröffentlichung akzeptiert wurde – in der Revision [39] der Arbeit Canetti u. a. [38].

Hierbei erfährt der Simulator von allen einer idealen Funktionalität gegebenen Eingaben. Damit die ideale Funktionalität diese Eingaben aber tatsächlich erhält und mit der jeweils erhaltenen Eingabe aktiviert wird, müssen diese Eingaben (einzeln, aber in beliebiger Reihenfolge) an die Funktionalität ausgeliefert werden. Dabei läßt [39] einige Details bezüglich des Schedulings offen. Etwa ist unklar, ob der Simulator bei neuen, noch nicht an eine Funktionalität ausgelieferten Eingaben aktiviert wird; zudem ist nicht klar, wie *direkte* Funktionalitäten gehandhabt werden können.

Als zusätzliche Schwierigkeit ergibt sich außerdem, daß der Simulator nicht weiß, ob eine syntaktisch ungültige Eingabe gegeben wurde; naheliegenderweise hat der Simulator nämlich keinen Zugriff auf den Inhalt der Eingaben. Ignoriert etwa ein

reales Protokoll ungültige Eingaben (was anzunehmen ist), so kann ein Simulator im allgemeinen nur dann einen realen Protokollablauf simulieren, wenn er darüber informiert ist, wann eine ungültige Eingabe gegeben wurde.

Da ein generelles direktes Ausliefern der Eingabe natürlich nach dem vorgestellten Angriff keine Option ist, erhält der Simulator – diese Änderung erfolgte erst in einer Überarbeitung der Revision [39] – die „Header" der Eingabe. Die „Header" werden dabei etwas vage als die nicht-geheimnistragenden Teile der Eingabe klassifiziert; die genaue Definition der „Header"-Teile einer Nachricht ist bei der Spezifikation der idealen Funktionalität vorzunehmen.

Da dieses Vorgehen äußerst künstlich erscheint, soll diese Modelländerung hier nicht weiter verfolgt werden. Es kann ohnehin in [84] eine *schon im alten Modell aus [27]* realisierbare Variante $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ angegeben werden. Diese soll nun vorgestellt werden.

4.1.5 Eine korrigierte Idealisierung

Abgesehen von der Quantifizierung über die jeweiligen Protokollteilnehmer (die Notwendigkeit hierfür wurde schon in 4.1.1 begründet) ist es also nötig, auch nach Initialisierungseingaben von beiden Protokollteilnehmern noch mit Wahl und Ausgabe des Sitzungsschlüssels abzuwarten, ob während der Simulation eines realen Protokollablaufs eine Partei korrumpiert wird. Passiert dies, kann keine ideale Wahl des Sitzungsschlüssels mehr garantiert werden; es ist damit zu rechnen, daß der Sitzungsschlüssel nun auch von dem Verhalten der korrumpierten Partei abhängt.[6] Daher wird auch im Falle einer im Protokollverlauf korrumpierten Partei dem Simulator die Wahl des Sitzungsschlüssels überlassen. Insgesamt ergibt sich die in Abbildung 4.3 dargestellte korrigierte Funktionalität $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$.

Zunächst erfolgt die Wahl des Sitzungsschlüssel κ explizit *gleichverteilt* aus der Menge $\{0,1\}^k$. Mit anderen Worten: Der Sitzungsschlüssel ist einfach ein gleichverteilter k-Bitstring. Damit ist klar, daß jedes reale Protokoll, welches sich anschickt, $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ zu realisieren, auch gleichverteilte Sitzungsschlüssel garantieren muß.

Es ist weiter zu beachten, daß der Angreifer $\mathcal{S}$ nun Einfluß auf den Zeitpunkt der Wahl des Sitzungsschlüssels hat. Es kann dabei für $\mathcal{S}$ von Vorteil sein, einen späten Zeitpunkt zu wählen: Ist zum Zeitpunkt der Wahl des Schlüssels eine am Schlüsselaustausch teilnehmende Partei (d. h. P_i oder P_j) korrumpiert, so darf $\mathcal{S}$ den Sitzungsschlüssel bestimmen. Insbesondere kann ein geschickt konstruierter Simulator $\mathcal{S}$ den im Beweis von Satz 4.2 benutzten Angriff simulieren, indem $\mathcal{S}$ die Wahl des Sitzungsschlüssels bis zum Ende des simulierten Protokolldurchlaufs verzögert und sich damit die freie Wahl des Sitzungsschlüssels ermöglicht.

Trotzdem bedingt die Möglichkeit des Simulators, den Zeitpunkt der Schlüsselwahl zu verzögern, noch keinen intuitiven Verlust von Sicherheitseigenschaften des Schlüsselaustausches: $\mathcal{S}$ darf ja ohnehin die Auslieferung der Sitzungsschlüssel von $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ an die Dummy-Parteien nach Belieben verzögern. Es kann also auch $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ als

[6] Eine diesbezügliche Verstärkung der idealen Funktionalität soll noch erarbeitet werden.

Die Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$

Protokollteilnehmer: $\mathcal{F}_{\text{KE}}^{(i,j)}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. Warte auf Nachrichten $(\texttt{Ready}, sid)$ von den Parteien P_i und P_j und vom Angreifer $\mathcal{S}$. Bei Erhalt von $(\texttt{Ready}, sid)$ von P_i oder P_j: Leite diese Nachricht (einschließlich der Senderidentität) weiter an $\mathcal{S}$.

2. Verfahre nach dem Erhalt von $(\texttt{Ready}, sid)$-Nachrichten von P_i, P_j und $\mathcal{S}$ (in beliebiger Reihenfolge) wie folgt:

   (a) Sind P_i und P_j beide unkorrumpiert, wähle κ gleichverteilt aus $\{0,1\}^k$.

   (b) Sende andernfalls eine Nachricht $(\texttt{Choose-Key}, sid)$ an $\mathcal{S}$. Beim Erhalt einer Antwort $(\texttt{Key}, sid, \kappa)$ mit $\kappa \in \{0,1\}^k$ von $\mathcal{S}$: Extrahiere κ aus dieser Nachricht.

   Sobald κ gesetzt ist, sende $(\texttt{Key}, sid, \kappa)$ an P_i und P_j, sende $(\texttt{Key}, sid)$ an $\mathcal{S}$, und halte dann an.

Abbildung 4.3: Die korrigierte Idealisierung $\mathcal{F}_{\text{KE}}^{(i,j)}$ aus [84]

hinreichend ideale Modellierung der Protokollaufgabe „Schlüsselaustausch“ aufgefaßt werden (wobei in diesem Zusammenhang allerdings auch die in Abschnitt 4.1.8 vorzustellende Verschärfung $\mathcal{F}_{\text{KE+}}^{(i,j)}$ von $\mathcal{F}_{\text{KE}}^{(i,j)}$, welche selbst bei korrumpierten Parteien noch Sicherheitsgarantien vergibt, nicht unerwähnt bleiben sollte).

4.1.6 Ein konkretes Sicherheitskriterium

Diese Motivation vorausgeschickt, sollen nun konkrete Eigenschaften eines Schlüsselaustauschprotokolls untersucht werden, welche die Realisierung von $\mathcal{F}_{\text{KE}}^{(i,j)}$ sicherstellen. Dazu wird auf Anforderungen zurückgegriffen, welche in ähnlicher Form auch schon in [36] zu einem ähnlichen Zweck verwandt wurden. Die vorliegende Konkretisierung und Abänderung stammt aus [84].

Definition 4.3. *Ein reales Protokoll $\pi^{(i,j)}$, welches über die Indizes von zwei Parteien P_i und P_j parametrisiert ist, heißt* universell komponierbares Schlüsselaustauschprotokoll*, wenn folgende Bedingungen erfüllt sind:*

- *$\pi^{(i,j)}$ bietet syntaktisch die gleiche Schnittstelle wie $\mathcal{F}_{\text{KE}}^{(i,j)}$ (in bezug auf die Kommunikation mit der Protokollumgebung $\mathcal{Z}$),*
- *es werden in $\pi^{(i,j)}$ nur Nachrichten zwischen P_i und P_j gesendet,*
- *sobald eine Partei in $\pi^{(i,j)}$ Ausgabe generiert, löscht sie alle internen Variablen (d. h. sie löscht ihr Arbeitsband und nimmt einen fest definierten Zustand ein),*

- *sobald in einem Protokolldurchlauf die erste Partei P_ℓ ($\ell \in \{i,j\}$) Ausgabe generiert, hat die jeweils andere Partei $P_{\bar{\ell}}$ – sofern diese nicht korrumpiert ist – alle Informationen bis auf den noch auszugebenden Sitzungsschlüssel gelöscht; weiter sendet danach P_ℓ nur eine feste „Bestätigungsnachricht“ an $P_{\bar{\ell}}$,*

- *bei Auslieferung aller Nachrichten zwischen P_i und P_j (für beliebige Reihenfolgen der Auslieferung), und wenn weder P_i noch P_j korrumpiert werden, generieren P_i und P_j schließlich gleiche Ausgaben, welche auch unter Kenntnis aller P_i-P_j-Kommunikation ununterscheidbar von der Gleichverteilung auf $\{0,1\}^k$ sind.*[7]

Die vorletzte Bedingung kann als Spezialfall der „`ack`-Eigenschaft“ aus [36] aufgefaßt werden, wohingegen die letzte Bedingung eine Variante der „SK-Sicherheit“ aus [36] darstellt.

Beispiel 4.4. Ein Beispiel für ein universell komponierbares Schlüsselaustauschprotokoll ist die Abwandlung $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ (dargestellt in Abbildung 4.4) des Protokolls SIG-DH aus [36]. Hierbei ist $\mathcal{G} = \{G_\nu\}_\nu$ eine Familie von zyklischen Gruppen von Primzahlordnung (jeweils explizit durch einen Erzeuger g gegeben) mit entsprechenden Verteilungen auf den Gruppenelementen, so daß das Decisional-Diffie-Hellman-Problem (DDH-Problem) schwer ist.[8] Für jede Gruppe $\langle g\rangle \in \mathcal{G}$ bezeichne $\mathcal{H}_{\langle g\rangle} = \{H_{\langle g\rangle,\nu}\}_\nu$ eine Familie von paarweise unabhängigen Hashfunktionen, welche benutzt wird, um „Rohschlüssel“ g^{xy} auf Bitstrings $H_{\langle g\rangle,\nu}(g^{xy})$ abzubilden. Da $\mathcal{F}^{(i,j)}_{\text{KE}}$ den Schlüssel κ als gleichverteilten k-Bitstring wählt, wird hier ein Vorgehen gemäß [127, Abschnitt 5.3.2] gewählt (siehe auch [130]).

Genauer wird eine Wahl der Parameter angenommen, für welche die Schwierigkeit des DDH-Problems in $\mathcal{G}$ und das „Entropy Smoothing Theorem“ (siehe etwa LUBY [104, Kapitel 8]) die polynomiale Ununterscheidbarkeit der Verteilungen $\{g, g^x, g^y, \nu, H_{\langle g\rangle,\nu}(g^{xy})\}_k$ bzw. $\{g, g^x, g^y, \nu, \kappa\}_k$ (für einen gleichverteilten k-Bitstring κ) implizieren. (Hierbei wird eine geeignete Verteilung von ν angenommen.) Es wird also insbesondere unterstellt, daß für jede mit einem Sicherheitsparameter $k \in \mathbb{N}$ assoziierte Gruppe $\langle g\rangle \in \mathcal{G}$ die Ausgabelänge der Hashfunktion $H_{\langle g\rangle,\nu}$ für alle ν jeweils k ist.

Während das Protokoll SIG-DH aus [36] die Beschreibung einer geeigneten Gruppe G als „initiale Information“ veröffentlicht annimmt (was im dort benutzten Modell [27] nur durch eine geeignete hybride Funktionalität formal gefaßt werden kann – in [36] aber nicht getan wird), wird im abgebildeten Protokoll $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ die Beschreibung $\text{D}(g)$ einer geeigneten Gruppe $G = \langle g\rangle$ explizit zwischen den Parteien übertragen. Um hierbei „falsche“, die Sicherheit gefährdende Wahlen von G durch eine korrumpierte Partei P_i zu vermeiden, wird weiter unterstellt, daß bei gegebenem

[7] Damit ist gemeint: Kein PPT-Algorithmus A kann mit nicht-vernachlässigbarer Wahrscheinlichkeit die Protokollausgabe einer Partei von einem gleichverteilt gezogenen $\kappa \in \{0,1\}^k$ unterscheiden, selbst wenn A als Eingabe auch eine entsprechende Protokollkommunikation erhält.

[8] Es wird also unterstellt, daß die Verteilungen (g^x, g^y, g^{xy}) und (g^x, g^y, g^z) für zufällig gewählte $x, y, z \in G_\nu$ polynomial ununterscheidbar sind.

Das Protokoll $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$

Dies sind Instruktionen für zwei Parteien P_i und P_j zum Ausführen eines Schlüsselaustausches. Bevor sie diesen Instruktionen folgt, erwartet eine Partei eine initiale (`Ready`, sid)-Eingabe.

1. P_i wählt abhängig vom Sicherheitsparameter k eine Gruppe $\langle g\rangle \in \mathcal{G}$ und danach zufällig $x \in \{1,\ldots,|\langle g\rangle|-1\}$. P_i berechnet weiter $\alpha = g^x \in \langle g\rangle$ und sendet $(sid, \mathrm{D}(g), \alpha)$ an P_j, wobei $\mathrm{D}(g)$ eine Beschreibung der Gruppe $\langle g\rangle$ ist, welche auch g spezifiziert.

2. Bei Erhalt von $(sid, \mathrm{D}(g), \alpha)$ von P_i mit $\alpha \in \langle g\rangle$ und für k akzeptablem $\mathrm{D}(g)$ wählt P_j zufällig $y \in \{1,\ldots,|\langle g\rangle|-1\}$ und einen Index ν in die Familie $\mathcal{H}_{\langle g\rangle}$. Dann berechnet P_j die Werte $\beta = g^y \in \langle g\rangle$, $\gamma = \alpha^y \in \langle g\rangle$ und damit $\kappa = H_{\langle g\rangle,\nu}(\gamma) \in \{0,1\}^k$, sendet (sid, β, ν) an P_i und löscht alle lokalen Variablen bis auf κ.

3. Bei Erhalt von (sid, β, ν) von P_j berechnet P_i den Wert $\gamma = \beta^x \in \langle g\rangle$ und damit $\kappa = H_{\langle g\rangle,\nu}(\gamma)$. Dann sendet P_i die Nachricht $(sid, \texttt{Done})$ an P_j, gibt $(\texttt{Key}, sid, \kappa)$ aus, löscht alle lokalen Variablen (inklusive κ) und hält an.

4. Bei Erhalt von $(sid, \texttt{Done})$ von P_i gibt P_j den Wert $(\texttt{Key}, sid, \kappa)$ aus, löscht alle lokalen Variablen (inklusive κ) und hält an.

Abbildung 4.4: Das Schlüsselaustauschprotokoll $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$

Sicherheitsparameter k und gegebener Beschreibung $\mathrm{D}(g)$ effizient verifiziert werden kann, ob $\langle g\rangle \in \mathcal{G}$ ist und ob $\langle g\rangle$ für den gegebenen Sicherheitsparameter k eine akzeptable Wahl darstellt (welche also die Schwierigkeit des DDH-Problems sicherstellt).[9] Diese etwas kompliziertere, aber flexiblere Formulierung scheint angesichts praktisch vorgeschlagener Protokolle wie IKEv2 [93] oder JFKi, JFKr [1] akzeptabel. Man beachte, daß bei der hiesigen Modellierung „Initialisierungsannahmen“ über gewählte Gruppen wie in [36] vermieden werden.

Nach Konstruktion und der obigen Diskussion ist klar, daß $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ allen Anforderungen an ein universell komponierbares Schlüsselaustauschprotokoll genügt.

Wie zu sehen war, wurde ein erheblicher technischer Aufwand getrieben, um Gleichverteilung des Sitzungsschlüssels $\kappa = H_{\langle g\rangle,\nu}(g^{xy})$ zu gewährleisten. Im speziellen Fall zyklischer Gruppen $G \subset (\mathbb{Z}/p\mathbb{Z})^\times$ kann hier allerdings einfacher vorgegangen werden: Es reicht ein einfaches Abschneiden der Bitdarstellungen von Elementen $g^{xy} \in G$. In jedem Fall jedoch muß eine Gleichverteilung der Sitzungsschlüssel sichergestellt werden, was – wie schon angedeutet – in [36] für das Protokoll SIG-DH

[9] Dies ist zum Beispiel in einfacher Weise möglich, falls aus einem Sicherheitsparameter k deterministisch und effizient $\langle g\rangle$ abgeleitet werden kann.

nicht geschieht.

Man beachte, daß das Protokoll SIG-DH aus [36] explizite Signaturen zur Nachrichtenauthentifizierung verwendet, da in [36] ein „halb-authentifiziertes“ Netzwerk angenommen wird, welches nur für die jeweils erste über einen Kanal gesendete Nachricht Authentifikation garantiert. Da diese Annahme etwas technisch erscheint, und überdies die Authentifikation von Kanälen schon in Abschnitt 3.3 diskutiert wurde, wird hier von authentifizierten Kanälen ausgegangen; damit ist keine explizite Verwendung eines Signatursystems nötig.

Am Beispiel des Protokolls $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ wird nun deutlich, weshalb es sinnvoll ist, für einen etwaigen Simulierbarkeitsbeweis *löschende* Parteien vorauszusetzen: Bei einem realen Angriff, bei dem erst nach Protokollende beide Parteien korrumpiert werden, gerät ein Simulator bei nicht-löschenden Parteien in die Verlegenheit, sich schon im Verlauf der Protokollsimulation Exponenten x und y und einen Index ν ausdenken zu müssen, so daß $\kappa = H_{\langle g\rangle,\nu}(g^{xy})$ ist. (Denn durch die spätere Korruption von P_i und P_j müssen mit dem simulierten Protokollablauf stimmige x, y und ν aufgedeckt werden – was bei löschenden Parteien nun eben nicht vonnöten ist.) Problematisch hierbei ist, daß der Simulator sich auf x, y und ν festlegen muß, *bevor* er überhaupt κ kennt. Dieses Argument zeigt, daß keine Hoffnung besteht, mit $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ die Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$ zu realisieren, sofern die Parteien als nicht-löschend angenommen werden.

Ein ähnliches Problem tritt – bei Annahme nicht-löschender Parteien – auch mit dem sogleich vorzustellenden Protokoll auf, kann dort allerdings prinzipiell durch die technisch aufwendige Methode der „non-committing encryption“, vgl. CANETTI U. A. [33], prinzipiell gehandhabt werden.

Beispiel 4.5. Als weiteres Beispiel dient das Protokoll $\text{PKKE}_P^{(i,j)}$ aus Abbildung 4.5, welches über ein als semantisch sicher angenommenes Public-Key-Kryptosystem $P = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ parametrisiert ist, das in der Lage ist, k-Bitstrings zu verschlüsseln. (D. h., es wird $\{0,1\}^k \subseteq \mathcal{M}_{pk}$ für alle möglichen $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ unterstellt.) Die semantische Sicherheit von P garantiert dabei, daß eine Verschlüsselung des Sitzungsschlüssels κ keine durch einen PPT-Algorithmus verwertbare Information über κ preisgibt.[10] Demnach erfüllt auch $\text{PKKE}_P^{(i,j)}$ alle Bedingungen eines universell komponierbaren Schlüsselaustauschprotokolls.

4.1.7 Ein Hinlänglichkeitsbeweis

Entscheidend ist natürlich, in welcher Beziehung der Begriff des „universell komponierbaren Schlüsselaustauschprotokolls“ zur idealen Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$ steht. Daß der Name „universell komponierbares Schlüsselaustauschprotokoll“ tatsächlich gerechtfertigt ist, zeigt folgender Satz:

[10] Genauer kann hier eine Reduktion auf den zu semantischer Sicherheit äquivalenten Sicherheitsbegriff IND-CPA (vgl. etwa BELLARE U. A. [17]) erfolgen.

Das Protokoll $\text{PKKE}_P^{(i,j)}$

Dies sind Instruktionen für zwei Parteien P_i und P_j zum Ausführen eines Schlüsselaustausches. Bevor sie diesen Instruktionen folgt, erwartet eine Partei eine initiale $(\texttt{Ready}, sid)$-Eingabe.

1. P_i generiert ein Schlüsselpaar $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, speichert sk und sendet (sid, pk) an P_j.
2. Bei Erhalt von (sid, pk) von P_i wählt P_j gleichverteilt $\kappa \in \{0,1\}^k$ und verschlüsselt $c \leftarrow \mathsf{Encrypt}(1^k, pk, \kappa)$. P_j sendet dann (sid, c) an P_i und löscht alle lokalen Variablen bis auf κ.
3. Bei Erhalt von (sid, c) von P_j entschlüsselt P_i zunächst den Sitzungsschlüssel $\kappa \leftarrow \mathsf{Decrypt}(1^k, sk, c)$. P_i gibt dann $(\texttt{Key}, sid, \kappa)$ aus, sendet $(sid, \texttt{Done})$ an P_j, löscht alle lokalen Variablen (inklusive κ) und hält an.
4. Bei Erhalt von $(sid, \texttt{Done})$ von P_i gibt P_j den Wert $(\texttt{Key}, sid, \kappa)$ aus, löscht alle lokalen Variablen (inklusive κ) und hält an.

Abbildung 4.5: Das Schlüsselaustauschprotokoll $\text{PKKE}_P^{(i,j)}$

Satz 4.6. *Jedes universell komponierbare Schlüsselaustauschprotokoll $\pi^{(i,j)}$ ist eine nicht-triviale Realisierung von $\mathcal{F}_{\text{KE}}^{(i,j)}$ bezüglich authentifizierter Kanäle und adaptiver Angreifer.*

Beweis. Man betrachte zunächst den in Abbildung 4.6 dargestellten Simulator $\mathcal{S}_\pi^{(i,j)}$, welcher Angriffe des Dummy-Angreifers $\tilde{\mathcal{A}}$ simuliert.

Weiter sei eine Umgebung $\mathcal{Z}$ fixiert, welche versucht, zwischen $\pi^{(i,j)}$ und $\tilde{\mathcal{A}}$ einerseits, und $\mathcal{F}_{\text{KE}}^{(i,j)}$ und $\mathcal{S}_\pi^{(i,j)}$ andererseits zu unterscheiden. Da nur die Parteien P_i und P_j tatsächlich am Protokoll beteiligt sind, darf ohne Beschränkung der Allgemeinheit angenommen werden, daß $\mathcal{Z}$ nie Anfragen zur Korruption einer Partei P_ℓ mit $\ell \notin \{i, j\}$ stellt.

In einem ersten Schritt soll eine Modifikation $\mathcal{Z}_1$ von $\mathcal{Z}$ betrachtet werden, welche, sobald P_i oder P_j korrumpiert werden soll *bevor* eine Partei Ausgabe generiert hat, mit gleichverteilt gezogener Ausgabe $b \in \{0,1\}$ anhält. Nun ist für $k \in \mathbb{N}$

$$\mathbf{Adv}_{\tilde{\mathcal{A}},\mathcal{S}_\pi^{(i,j)},\mathcal{Z}}^{\pi^{(i,j)},\mathcal{F}_{\text{KE}}^{(i,j)}}(k) = \mathbf{Adv}_{\tilde{\mathcal{A}},\mathcal{S}_\pi^{(i,j)},\mathcal{Z}_1}^{\pi^{(i,j)},\mathcal{F}_{\text{KE}}^{(i,j)}}(k), \tag{4.1}$$

was wie folgt eingesehen werden kann: Zum einen ist die Sicht von $\mathcal{Z}$ im realen und idealen Modell nach Konstruktion von $\mathcal{S}_\pi^{(i,j)}$ identisch verteilt *bevor* eine Partei Ausgabe generiert. Wird aber eine Partei P_ℓ korrumpiert, bevor eine Ausgabe erfolgt ist, so liefert $\mathcal{S}_\pi^{(i,j)}$ einen mit der bisherigen Simulation konsistenten Zustand von P_ℓ und darf außerdem die Ausgabe der jeweils anderen Partei der bisherigen

Der Simulator $\mathcal{S}_\pi^{(i,j)}$

- **Kommunikation mit $\mathcal{F}_{\text{KE}}^{(i,j)}$:**
  - Bei Erhalt von $(\texttt{Ready}, sid)$ mit initialem Sender $\tilde{P}_\ell$ ($\ell \in \{i,j\}$) von $\mathcal{F}_{\text{KE}}^{(i,j)}$: Leite diese Nachricht an $P_\ell^{(s)}$ weiter.
  - Sobald die erste simulierte Partei $P_\ell^{(s)}$, für welche die Dummy-Partei $\tilde{P}_\ell$ *nicht* korrumpiert ist, Ausgabe generiert (die κ genannt sei): Speichere κ und sende $(\texttt{Ready}, sid)$ an $\mathcal{F}_{\text{KE}}^{(i,j)}$.
  - Bei Erhalt von $(\texttt{Choose-Key}, sid)$ von $\mathcal{F}_{\text{KE}}^{(i,j)}$ (was bedeutet, daß $\mathcal{S}_\pi^{(i,j)}$ schon κ gespeichert hat): Sende $(\texttt{Key}, sid, \kappa)$ an $\mathcal{F}_{\text{KE}}^{(i,j)}$.
  - Ausgaben von $\mathcal{F}_{\text{KE}}^{(i,j)}$ werden ausgeliefert, wenn die entsprechende simulierte Partei $P_\ell^{(i,j)}$ Ausgabe generiert.
- **Kommunikation mit $\mathcal{Z}$:**
  - Melde auf Anfrage nach gesendeten Nachrichten genau die Nachrichten, die zwischen den simulierten $P_i^{(s)}$ und $P_j^{(s)}$ gesendet wurden.
  - Speichere jeden Befehl, eine Nachricht auszuliefern. Falls der Empfänger P_i oder P_j ist, liefere diese Nachricht an $P_i^{(s)}$ bzw. $P_j^{(s)}$ aus.
  - Bei einem Befehl zur Korruption einer Partei P_ℓ: Korrumpiere zunächst die Dummy-Partei $\tilde{P}_\ell$ und bereite dann einen Zustand von P_ℓ vor, der auch alle an P_ℓ gelieferten Nachrichten enthält. Ist $\ell \in \{i,j\}$, sende, sofern nicht schon geschehen, $(\texttt{Ready}, sid)$ im Namen von $\tilde{P}_\ell$ an $\mathcal{F}_{\text{KE}}^{(i,j)}$, sobald *sid* bekannt ist. Liefere schließlich den präparierten P_ℓ-Zustand an $\mathcal{Z}$, verfahre jedoch bei $\ell \in \{i,j\}$ dazu wie folgt:
    (a) Wenn $\mathcal{S}_\pi^{(i,j)}$ noch kein `Ready`-Signal an $\mathcal{F}_{\text{KE}}^{(i,j)}$ gesendet hat (wenn also noch keine simulierte Partei mit unkorrumpierter entsprechender Dummy-Partei Ausgabe generiert hat), liefere den Zustand von $P_\ell^{(s)}$ als Zustand von P_ℓ an $\mathcal{Z}$.
    (b) Andernfalls hat $\mathcal{S}_\pi^{(i,j)}$ ein `Ready`-Signal gesendet, und demnach hat $\mathcal{F}_{\text{KE}}^{(i,j)}$ den Sitzungsschlüssel κ ausgegeben; κ kann von $\mathcal{S}_\pi^{(i,j)}$ also durch Ausliefern an die korrumpierte Dummy-Partei $\tilde{P}_\ell$ gewonnen werden. Nach Annahme über $\pi^{(i,j)}$ muß nun κ genau dann in den Zustand von P_ℓ integriert werden, wenn $P_\ell^{(s)}$ noch keine Ausgabe generiert hat. Liefere also den Zustand von $P_\ell^{(s)}$ (ggf. mit ersetztem κ) als Zustand von P_ℓ an $\mathcal{Z}$ zurück.

Abbildung 4.6: Der Simulator $\mathcal{F}_\pi^{(i,j)}$

Simulation anpassen. Damit ist die Sicht von $\mathcal{Z}$ im realen und idealen Modell identisch verteilt, falls Korruptionen stattfinden, bevor eine Ausgabe erfolgt. Damit ist (4.1) gezeigt.

Eine Modifikation $\mathcal{Z}_2$ von $\mathcal{Z}_1$ kommt nun auch ohne Korruptionen *nach* erfolgter Ausgabe κ einer Partei aus: Anstatt die Korruption von P_ℓ zu erfragen, beantwortet $\mathcal{Z}_2$ diese Anfragen für sich selbst durch einen selbst-präparierten Zustand von P_ℓ. Dieser Zustand ist entweder leer, oder enthält nur κ – je nachdem, ob P_ℓ schon Ausgabe generiert hat oder nicht. Nach Annahme über $\pi^{(i,j)}$ und $\mathcal{S}_\pi^{(i,j)}$ ist dies genau der vom Angreifer bei einer echten Korruption von P_ℓ erhaltene Zustand, es sei denn, im realen Modell geben die Parteien unterschiedliche Ausgaben. Dieser Fall tritt aber nur mit vernachlässigbarer Wahrscheinlichkeit auf, so daß

$$\mathbf{Adv}^{\pi^{(i,j)},\mathcal{F}_{\mathrm{KE}}^{(i,j)}}_{\tilde{\mathcal{A}},\mathcal{S}_\pi^{(i,j)},\mathcal{Z}_1}(k) - \mathbf{Adv}^{\pi^{(i,j)},\mathcal{F}_{\mathrm{KE}}^{(i,j)}}_{\tilde{\mathcal{A}},\mathcal{S}_\pi^{(i,j)},\mathcal{Z}_2}(k) \tag{4.2}$$

für die beschriebene Umgebung $\mathcal{Z}_2$, welche überhaupt keine Korruptionen erfragt, vernachlässigbar in k ist. Kombination von (4.1) und (4.2) zeigt, daß sich ohne Beschränkung der Allgemeinheit auf Umgebungen $\mathcal{Z}$ beschränkt werden darf, welche *nie* eine Partei korrumpieren lassen.

Für solche Umgebungen $\mathcal{Z}$ liefert $\mathcal{S}_\pi^{(i,j)}$ eine Simulation des realen Protokolls π, welche sich *allein* in der Ausgabe unterscheidet: Im idealen Modell werden die Ausgaben der Parteien einfach durch gleiche und gleichverteilt gezogene $\kappa \in \{0,1\}^k$ ersetzt. Nun ist aber $\pi^{(i,j)}$ als „universell komponierbares Schlüsselaustauschprotokoll“ gemäß Definition 4.3 angenommen, und somit kann $\pi^{(i,j)} \geq \mathcal{F}_{\mathrm{KE}}^{(i,j)}$ direkt aus der Ununterscheidbarkeit der Protokollausgaben von gleichverteilten k-Bitstrings abgeleitet werden.

Die Nicht-Trivialität dieser Realisierung folgt schließlich auch aus Definition 4.3. □

4.1.8 Eine stärkere Idealisierung

Die vorstehend besprochene Korrektur $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ von $\mathcal{F}_{\mathrm{KE}}$ gibt – wie auch $\mathcal{F}_{\mathrm{KE}}$ selbst – im Falle auch nur einer korrumpierten Partei keine Garantien mehr über die Verteilung des Sitzungsschlüssel. Tatsächlich wird in diesem Fall dem Simulator ermöglicht, den Sitzungsschlüssel zu wählen. Damit ist beispielsweise eine Simulation von universell komponierbaren Schlüsselaustauschprotokollen möglich; dies zeigt ja gerade Satz 4.6. Angesichts der gegebenen Beispielprotokolle (insbesondere angesichts des Protokolls aus Beispiel 4.5) drängt sich nun die Frage auf, ob man nicht sogar von stärkeren Garantien ausgehen darf: Etwa scheint das Protokoll $\textsc{pkke}_P^{(i,j)}$ auch im Falle eines korrumpierten Initiators (d. h. im Falle einer korrumpierten Partei P_i) noch einen gleichverteilt gewählten Sitzungsschlüssel zu garantieren.

Dieser Gedanke wird durch die in [84] vorgeschlagene verstärkte Idealisierung $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ eines Schlüsselaustauschprotokolls aufgegriffen. Genauer garantiert $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ auch im Falle eines korrumpierten Initiators einen gleichverteilt gewählten Schlüssel, vgl. Abbildung 4.7. (Ein Vergleich zwischen der Funktionalität $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ aus Ab-

bildung 4.3 und $\mathcal{F}_{\text{KE+}}^{(i,j)}$ zeigt, daß dies auch der einzige Unterschied der beiden Funktionalitäten ist.)

Die Funktionalität $\mathcal{F}_{\text{KE+}}^{(i,j)}$

Protokollteilnehmer: $\mathcal{F}_{\text{KE}}^{(i,j)}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. Warte auf Nachrichten $(\texttt{Ready}, sid)$ von den Parteien P_i und P_j, und vom Angreifer $\mathcal{S}$. Bei Erhalt von $(\texttt{Ready}, sid)$ von P_i oder P_j, leite diese Nachricht (einschließlich der Senderidentität) weiter an $\mathcal{S}$.
2. Nach dem Erhalt von $(\texttt{Ready}, sid)$-Nachrichten von P_i, P_j und $\mathcal{S}$ (in beliebiger Reihenfolge), verfahre wie folgt:
   (a) Ist P_j unkorrumpiert, wähle κ gleichverteilt aus $\{0,1\}^k$.
   (b) Andernfalls, sende eine Nachricht $(\texttt{Choose-Key}, sid)$ an $\mathcal{S}$. Beim Erhalt einer Antwort $(\texttt{Key}, sid, \kappa)$ von $\mathcal{S}$: Extrahiere κ aus dieser Nachricht.

   Sobald κ gesetzt ist, sende $(\texttt{Key}, sid, \kappa)$ an P_i und P_j, sende $(\texttt{Key}, sid)$ an $\mathcal{S}$, und halte dann an.

Abbildung 4.7: Die verstärkte Idealisierung $\mathcal{F}_{\text{KE+}}^{(i,j)}$ aus [84]

Unglücklicherweise realisiert keines der Protokolle $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ bzw. $\text{PKKE}_{P}^{(i,j)}$ (zumindest für bestimmte semantisch sichere Systeme P) diese Funktionalität $\mathcal{F}_{\text{KE+}}^{(i,j)}$, obwohl eine solche Realisierung natürlich höchst intuitiv wäre. Das gilt auch für die „seitenverkehrten“ Protokolle $\text{DH}_{\mathcal{G},\mathcal{H}}^{(j,i)}$ bzw. $\text{PKKE}_{P}^{(j,i)}$. Um dies einzusehen, betrachte man folgende Umgebung $\mathcal{Z}$, welche im realen Modell mit dem Dummy-Angreifer und Protokoll $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ auszuführen ist:

$\mathcal{Z}$ läßt das Protokoll normal verlaufen, korrumpiert jedoch P_i direkt nachdem die (einzige) Nachricht von P_j an P_i gesendet wurde. Damit kann $\mathcal{Z}$ aus dem Zustand der korrumpierten Partei P_i und den gesendeten Nachrichten einen Sitzungsschlüssel κ_i extrahieren. $\mathcal{Z}$ läßt dann eine Bestätigungsnachricht im Namen von P_i an P_j übermitteln, so daß auch P_j eine Ausgabe κ_j generiert. (Generiert P_j keine Ausgabe, findet offenbar eine Ausführung im idealen Modell statt, und $\mathcal{Z}$ hält mit Ausgabe 0.) Schließlich hält $\mathcal{Z}$ mit Ausgabe 1 bzw. 0 genau dann, wenn $\kappa_i = \kappa_j$ bzw. $\kappa_i \neq \kappa_j$ ist.

Nun wird $\mathcal{Z}$ im realen Modell immer mit 1-Ausgabe halten, während sich der Simulator schon durch die *vor* der Korruption übermittelten Nachrichten auf κ_i festlegen mußte. Die Wahl von κ_i ist dabei nach Konstruktion von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ unabhängig von dem von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ gewählten Sitzungsschlüssel κ_j. Damit gibt $\mathcal{Z}$ im idealen Modell nur mit exponentiell kleiner Wahrscheinlichkeit 1 aus; somit ist $\mathcal{Z}$ ein guter Unterscheider, und es gilt $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)} \not\geq \mathcal{F}_{\text{KE+}}^{(i,j)}$.

Ein ähnliches Argument findet im Falle des Protokolls $\text{PKKE}_P^{(i,j)}$ Anwendung, falls das zugrundegelegte Public-Key-Kryptosystem nicht schon die Eigenschaft „nicht-festlegender Chiffrate“ hat (siehe etwa [33]). Leider sind jedoch nur sehr aufwendige derartige Systeme bekannt; da zudem etwa für RSA- oder ElGamal-basierte Systeme ein Chiffrat mit zugehörigem öffentlichen Schlüssel schon *eindeutig* den zugehörigen Klartext festlegt, gilt für viele praktisch relevante semantisch sichere Public-Key-Kryptosysteme P mit obigem Argument $\text{PKKE}_P^{(i,j)} \not\geq \mathcal{F}_{\text{KE+}}^{(i,j)}$.

Dieser Zustand mag als unbefriedigend empfunden werden. Insbesondere das Protokoll $\text{PKKE}_P^{(i,j)}$, in welchem ja lediglich ein Schlüssel von P_j nach P_i transportiert wird, scheint intuitiv gegen eine Korruption von P_i immun zu sein. Deshalb wurde in [84] eine weitere Funktionalität $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ angegeben, welche ähnliche Sicherheitsgarantien wie $\mathcal{F}_{\text{KE+}}^{(i,j)}$ vergibt, dabei jedoch insbesondere von den Protokollen $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ und $\text{PKKE}_P^{(i,j)}$ realisierbar ist.

Dabei ist $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ über ein *Non-Information-Orakel* parametrisiert (vgl. [36]), welches *abhängig* vom betrachteten realen Protokoll ist. Für eine genauere (leider aufgrund der Natur von Non-Information-Orakeln etwas technische) Besprechung dieser Ergebnisse sei hier auf [84] verwiesen.

4.1.9 Ein allgemeines Konstruktionsverfahren

Alternativ kann jedes Protokoll $\pi^{(i,j)}$, welches die „schwächere“ Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$ realisiert, schon zu einem Protokoll umgebaut werden, welches $\mathcal{F}_{\text{KE+}}^{(i,j)}$ realisiert. Insbesondere gilt dies also für – im Sinne von Definition 4.3 – universell komponierbare Schlüsselaustauschprotokolle. Der Umbau eines solchen Ausgangsprotokolls $\pi^{(i,j)}$ kann nun elegant im $\mathcal{F}_{\text{KE}}^{(i,j)}$-hybriden Modell formuliert werden; das tatsächliche umgebaute Protokoll kann dann wegen des Kompositionstheorems 3.2 *ohne Sicherheitsverlust* durch bloßes Einsetzen von $\pi^{(i,j)}$ gewonnen werden.

Man betrachte konkret das „Rahmenprotokoll“ $\text{PAD}^{(i,j)}$ aus Abbildung 4.8, welches im $\mathcal{F}_{\text{KE}}^{(i,j)}$-hybriden Modell formuliert ist, also Instanzen der idealen Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$ nutzt. Die Idee von $\text{PAD}^{(i,j)}$ ist simpel: Nach einem erfolgten Schlüsselaustausch mittels einer $\mathcal{F}_{\text{KE}}^{(i,j)}$-Instanz (welche an P_i und P_j einen vorläufigen Sitzungsschlüssel $\bar{\kappa}$ liefert) schickt P_j einen gleichverteilt gezogenen k-Bitstring ψ an P_i. Dieser wird anschließend auf den vorläufigen Schlüssel $\bar{\kappa}$ addiert, um den endgültigen Schlüssel $\kappa = \bar{\kappa} \oplus \psi$ zu erzeugen. Auf diese Weise kann auch im Falle einer korrumpierten Partei P_i die Gleichverteilung des endgültigen Schlüssels κ sichergestellt werden, sofern P_j unkorrumpiert ist.

Genauer ergibt sich folgende Aussage:

Satz 4.7. *Das im $\mathcal{F}_{\text{KE}}^{(i,j)}$-hybriden Modell formulierte Protokoll $\text{PAD}^{(i,j)}$ aus Abbildung 4.8 ist eine nicht-triviale Realisierung von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ bezüglich authentifizierter Kanäle und adaptiver Angreifer.*

Beweis. Es reicht, für jeden Hybrid-Modell-Angreifer $\mathcal{H}$ einen Simulator $\mathcal{S} = \mathcal{S}_{\mathcal{H}}$ anzugeben, so daß keine Umgebung zwischen $\mathcal{H}$ und Protokoll $\text{PAD}^{(i,j)}$ einerseits

Das Protokoll $\text{PAD}^{(i,j)}$

Dies sind Instruktionen für zwei Parteien P_i und P_j zum Ausführen eines Schlüsselaustausches. Bevor sie diesen Instruktionen folgt, erwartet eine Partei eine initiale $(\texttt{Ready}, sid)$-Eingabe. Die Ausführung dieses Protokolls findet im $\mathcal{F}_{\text{KE}}^{(i,j)}$-hybriden Modell statt, d. h. die Parteien haben Zugriff auf beliebig viele Instanzen der idealen Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$.

1. P_i und P_j senden beide direkt nach Erhalt der $(\texttt{Ready}, sid)$-Eingabe eine Eingabe $(\texttt{Ready}, 0)$ an die $\mathcal{F}_{\text{KE}}^{(i,j)}$-Instanz mit Sitzungsnummer 0.

2. Nach dem Erhalt eines Schlüssels $\bar{\kappa} \in \{0,1\}^k$ von $\mathcal{F}_{\text{KE}}^{(i,j)}$ wählt P_j gleichverteilt ein $\psi \in \{0,1\}^k$. P_j sendet dann (sid, ψ) an P_i, berechnet $\kappa = \bar{\kappa} \oplus \psi$ und löscht alle lokalen Variablen bis auf κ.

3. Sobald P_i sowohl eine Nachricht (sid, ψ) (mit $\psi \in \{0,1\}^k$) von P_j als auch einen vorläufigen Schlüssel $\bar{\kappa} \in \{0,1\}^k$ von $\mathcal{F}_{\text{KE}}^{(i,j)}$ erhalten hat, berechnet P_i den Sitzungsschlüssel $\kappa = \bar{\kappa} \oplus \psi$. P_i sendet dann $(sid, \texttt{Done})$ an P_j, gibt $(\texttt{Key}, sid, \kappa)$ aus, löscht danach alle lokalen Variablen und hält.

4. Bei Erhalt von $(sid, \texttt{Done})$ von P_i gibt P_j den Wert $(\texttt{Key}, sid, \kappa)$ aus, löscht alle lokalen Variablen und hält an.

Abbildung 4.8: Das Schlüsselaustauschprotokoll $\text{PAD}^{(i,j)}$

und $\mathcal{S}$ und $\mathcal{F}_{\text{KE+}}^{(i,j)}$ andererseits unterscheiden kann. Sei also ein solches $\mathcal{H}$ gegeben; dann simuliert $\mathcal{S}$ intern einen hybriden Protokollablauf von $\text{PAD}^{(i,j)}$ mit Parteien $P_1^{(s)}, \ldots, P_n^{(s)}$, einem (simulierten) Angreifer $\mathcal{H}$ und (soweit benötigt) Instanzen der idealen Funktionalität $\mathcal{F}_{\text{KE}}^{(i,j)}$. Kommunikation von $\mathcal{H}$ mit der Umgebung wird von $\mathcal{S}$ zur nicht-simulierten Umgebung $\mathcal{Z}$ weitergeleitet; das umfaßt auch Nachrichten, die $\mathcal{S}$ von $\mathcal{Z}$ erhält.

Nach Konstruktion ist damit die Sicht von $\mathcal{Z}$ im idealen Modell identisch mit der Sicht im realen Modell, bis auf die folgenden Ausnahmen: Zum einen wird $\mathcal{Z}$ im realen Modell durch spezielle (modellinterne) Benachrichtigungen über alle Korruptionen informiert, die $\mathcal{H}$ vornimmt, siehe auch Abschnitt 3.1.2. Eine solche Benachrichtigung kann natürlich auch von $\mathcal{S}$ herbeigeführt werden, falls etwa der simulierte Angreifer $\mathcal{H}$ eine simulierte Partei $P_\ell^{(s)}$ korrumpiert: $\mathcal{S}$ muß nur die entsprechende Dummy-Partei $\tilde{P}_\ell$ korrumpieren. (Aus Gründen, die später noch klar werden, sendet $\mathcal{S}$ außerdem bei einer solchen Korruption einer Partei $\tilde{P}_\ell$ mit $\ell \in \{i,j\}$ eine $(sid, \texttt{Ready})$-Nachricht im Namen von $\tilde{P}_\ell$ an $\mathcal{F}_{\text{KE+}}^{(i,j)}$.)

Weiter ist natürlich das Ein-/Ausgabeverhalten der idealen Dummy-Parteien nicht notwendig identisch mit dem der von $\mathcal{S}$ simulierten Parteien. Es sollen zunächst die Eingaben, die $\mathcal{Z}$ Parteien gibt, betrachtet werden. Tatsächlich ignoriert das hybride Protokoll alle $\mathcal{Z}$-Eingaben bis auf die jeweils erste $(sid, \texttt{Ready})$-Einga-

be an P_i bzw. P_j. Genau über diese Eingaben wird aber $\mathcal{S}$ von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ informiert, so daß diese Eingaben auch den simulierten Parteien $P_i^{(s)}$ bzw. $P_j^{(s)}$ gegeben werden können. Natürlich müssen auch die ignorierten Eingaben im Falle einer späteren Korruption einer Partei in deren Zustand integriert werden; es sollte aber offensichtlich sein, wie dies zu geschehen hat.

Es verbleibt noch, die *Ausgaben* der Parteien im idealen Modell bzw. in der Simulation von $\mathcal{S}$ zu betrachten. $\mathcal{S}$ verhält sich diesbezüglich wie folgt:

- Wenn die simulierte Partei $P_j^{(s)}$ eine Nachricht (sid, ψ) mit $\psi \in \{0,1\}^k$ an $P_i^{(s)}$ sendet und zu diesem Zeitpunkt $P_i^{(s)}$, aber nicht $P_j^{(s)}$ korrumpiert ist, muß der „Pad“ ψ so abgeändert werden, daß er mit dem tatsächlich von $\tilde{P}_j$ an $\mathcal{Z}$ ausgegebenen Schlüssel κ konsistent ist. Zunächst sendet also $\mathcal{S}$ eine $(sid, \texttt{Ready})$-Nachricht an $\mathcal{F}_{\text{KE+}}^{(i,j)}$, und liefert den von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ daraufhin[11] erzeugten Schlüssel κ sogleich an die korrumpierte Partei $\tilde{P}_i$ (d. h. effektiv an sich selbst) aus. $\mathcal{S}$ setzt dann $\bar{\psi} = \kappa \oplus \bar{\kappa}$ (wobei $\bar{\kappa}$ den Schlüssel bezeichnet, den die simulierte $\mathcal{F}_{\text{KE}}^{(i,j)}$-Instanz mit Sitzungsnummer 0 an $P_i^{(s)}$ und $P_j^{(s)}$ gesendet hat) und modifiziert den internen Zustand der simulierten Partei $P_j^{(s)}$ so, daß anstelle von ψ nun $\bar{\psi}$ an $P_i^{(s)}$ gesendet wird; der entsprechende noch im Speicher von $P_j^{(s)}$ verbleibende Sitzungsschlüssel ist damit $\bar{\kappa} \oplus \bar{\psi} = \kappa$.

  Wichtig hierbei ist, daß $\bar{\psi}$ genau wie ψ ein gleichverteilter k-Bitstring ist, da $\bar{\kappa}$ gleichverteilt ist; intuitiv sieht $\bar{\psi}$ damit für den Angreifer $\mathcal{H}$ „genauso aus“ wie ψ. Das bedeutet, daß sich dieser Eingriff nicht in der Sicht von $\mathcal{H}$ bzw. $\mathcal{Z}$ niederschlägt.

- Generiert eine simulierte Partei (etwa $P_\ell^{(s)}$ mit $\ell \in \{i,j\}$) eine Ausgabe, so muß auch die nicht-simulierte Dummy-Partei $\tilde{P}_\ell$ Ausgabe generieren. Dazu muß $\mathcal{S}$ aber nur eine Ausgabe von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ an $\tilde{P}_\ell$ ausliefern; hat $\mathcal{F}_{\text{KE+}}^{(i,j)}$ den Sitzungsschlüssel noch nicht gewählt, muß $\mathcal{S}$ vorher noch eine $(sid, \texttt{Ready})$-Nachricht an $\mathcal{F}_{\text{KE+}}^{(i,j)}$ schicken.[12] (Wurde zu diesem Zeitpunkt $\tilde{P}_j$ schon korrumpiert, wird $\mathcal{S}$ hierbei auch nach einem Schlüssel gefragt; $\mathcal{S}$ wählt dann den von $P_\ell^{(s)}$ ausgegebenen Schlüssel.)

- Der einzige Fall, in dem sich die Ausgaben der in $\mathcal{S}$ simulierten Parteien von den tatsächlichen Ausgaben der jeweiligen Dummy-Parteien unterscheiden können, tritt auf, wenn zum Zeitpunkt des Sendens der (sid, ψ)-Nachricht von $P_j^{(s)}$ an $P_i^{(s)}$ *keine* der beiden Parteien korrumpiert ist.

  Wird nun anschließend weder $P_i^{(s)}$ noch $P_j^{(s)}$ korrumpiert, stellt dies kein Problem dar, da $\mathcal{H}$ keine Information über den von $\mathcal{F}_{\text{KE}}^{(i,j)}$ gewählten vorläufigen

[11] Hier fließt ein, daß $\mathcal{S}$ bei Korruption von P_i oder P_j automatisch eine $(sid, \texttt{Ready})$-Nachricht im Namen der korrumpierten Partei an $\mathcal{F}_{\text{KE+}}^{(i,j)}$ sendet.

[12] Man beachte, daß die Wahl und Ausgabe eines Sitzungsschlüssels von $\mathcal{F}_{\text{KE+}}^{(i,j)}$ immer auch mit einer Nachricht an $\mathcal{S}$ einhergeht, welche die sofortige Wiederaktivierung von $\mathcal{S}$ sicherstellt.

Sitzungsschlüssel $\bar{\kappa}$ besitzt. Auch falls $P_j^{(s)}$ korrumpiert wird, kann $\mathcal{S}$ entweder den noch von $\tilde{P}_i$ auszugebenden Sitzungsschlüssel in Konsistenz mit der internen Simulation bestimmen, oder $\tilde{P}_i$ hat schon eine Ausgabe κ generiert, und $\mathcal{S}$ kann den internen Zustand von $P_j^{(s)}$ nach dieser Korruption entsprechend anpassen. (Hat $P_j^{(s)}$ schon Ausgabe generiert, ist dieser Zustand einfach leer – hat $P_j^{(s)}$ noch keine Ausgabe generiert, kann $\mathcal{S}$ den Schlüssel κ durch Auslieferung an die nun korrumpierte Partei $\tilde{P}_j$ erfahren.)

Es muß damit lediglich noch der Fall einer Korruption von $P_i^{(s)}$ *nachdem* die (sid, ψ)-Nachricht an $P_i^{(s)}$ gesendet wurde betrachtet werden. Wurde diese Nachricht zum Zeitpunkt der Korruption schon an $P_i^{(s)}$ ausgeliefert, hat $P_i^{(s)}$ nur noch einen „leeren“ und damit mit der Simulation konsistenten Zustand.

Findet die Korruption aber *vor* Auslieferung der (sid, ψ)-Nachricht statt, muß im Zustand von $P_i^{(s)}$ der vorläufige, von $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ an $P_i^{(s)}$ gelieferte Sitzungsschlüssel $\bar{\kappa}$ durch einen mit der Ausgabe κ von $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ konsistenten Schlüssel $\bar{\kappa}' = \kappa \oplus \psi$ ersetzt werden. (Hierbei muß gegebenenfalls eine Wahl von κ durch eine $(sid, \texttt{Ready})$-Nachricht an $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ herbeigeführt werden – $\mathcal{S}$ kann sich dann κ wieder durch Auslieferung an die korrumpierte Dummy-Partei $\tilde{P}_i$ zugänglich machen.)

Da der so ersetzte vorläufige Schlüssels $\bar{\kappa}' \in \{0,1\}^k$ gleichverteilt ist, beeinflußt dies nicht die Sicht von $\mathcal{H}$ oder $\mathcal{Z}$.

Insgesamt gewährleistet $\mathcal{S}$ also nach dieser Diskussion eine Sicht für $\mathcal{Z}$, die identisch zu der im hybriden Modell ist. Es ist klar, daß die Konstruktion von $\mathcal{S}$ nicht von der Wahl von $\mathcal{Z}$ abhängt. Damit ist $\mathrm{PAD}^{(i,j)} \geq \mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ gezeigt; die Nicht-Trivialität dieses konkreten Protokolls ist dabei offensichtlich. □

Wie schon angesprochen garantiert das Kompositionstheorem 3.2 dabei, daß ein beliebiges, $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ realisierendes Protokoll durch Einsetzen in $\mathrm{PAD}^{(i,j)}$ zu einem „stärkeren“ Protokoll umgebaut werden kann, welches seinerseits sogar $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ realisiert. Das gilt also insbesondere für die schon besprochenen universell komponierbaren Schlüsselaustauschprotokolle $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ und $\mathrm{PKKE}_{P}^{(i,j)}$. Die Modularität dieses Vorgehens kann somit als Beleg für die Eleganz des Ansatzes aus [27] gewertet werden.

4.1.10 Offene Fragen

Die Funktionalität $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ gibt schon größere Sicherheitsgarantien als $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$; auch im Falle einer korrumpiertem Partei P_i wird ein ideal gezogener, also nicht vom Simulator beeinflußter Sitzungsschlüssel garantiert. Nun liegt es nahe, eine weitere Verstärkung (die hier etwa als $\mathcal{F}_{\mathrm{KE++}}^{(i,j)}$ bezeichnet werde) dieser Funktionalität zu betrachten, welche in *jedem* Fall (also sowohl bei korrumpiertem P_i als auch bei korrumpierten P_j) ideal gezogene Sitzungsschlüssel garantiert.

Eine solche Funktionalität wird offenbar nicht von einem der angegebenen Protokolle realisiert – selbst im Falle eines Diffie-Hellman-basierten Protokolls kann eine Partei P_j, welche als letzte Einfluß auf den Sitzungsschlüssel hat, durch wiederholtes Ändern ihrer Zufallswahlen einen Sitzungsschlüssel erzwingen, welcher etwa mit logarithmisch (in k) vielen Nullbits beginnt.

Hier würde sich eigentlich die Benutzung eines Commitment-Verfahrens (siehe auch Abschnitt 4.2) anbieten, mit welchem sich P_j als erstes auf die später gesendete Nachricht festlegt. Leider ist auch ein solches Protokoll – obwohl es intuitiv auch im Falle einer korrumpierten Partei P_j ideal verteilte Schlüssel garantiert – nicht simulierbar sicher. (Der Grund hierfür ist, daß keine Benutzung eines Commitment-Verfahrens simuliert werden kann; diese Problematik wird im nächsten Abschnitt besprochen.)

Insgesamt bleibt also die interessante Frage nach einer Realisierung einer Funktionalität wie $\mathcal{F}_{\mathrm{KE++}}^{(i,j)}$ durch ein *reales* Protokoll (also keines, welches etwa in einem Commitment-hybriden Modell formuliert ist) offen.

4.2 Commitment-Verfahren

Als weitere Protokollaufgabe soll nun die Aufgabe *Commitment* vorgestellt werden. Ein Commitment-Verfahren wurde schon implizit in Blum [24] genutzt, um es zwei Parteien zu ermöglichen, einen geheimen Münzwurf *telefonisch* zu realisieren. Dabei wird eine Gleichverteilung des Wurfergebnisses auch dann garantiert, wenn eine der beiden Parteien unehrlich ist. Dieses erstaunliche Ergebnis wird wie folgt erzielt: Zunächst legt sich Partei P_1 bei Partei P_2 auf ein gleichverteilt gewähltes Bit b_1 fest, *ohne* daß P_2 eine verwertbare Information über b_1 erhält. Intuitiv sendet P_1 also P_2 einen geschlossenen Tresor, in dem sich b_1 befindet.

Anschließend zieht P_2 gleichverteilt ein Bit b_2 und sendet b_2 (im Klartext) an P_1. Nachdem sich auf diese Weise beide Parteien jeweils auf ein Bit festgelegt haben, deckt P_1 schließlich b_1 gegenüber P_2 auf. Intuitiv sendet P_1 an P_2 einen Schlüssel für den im ersten Schritt gesendeten Tresor, so daß P_2 an das Bit b_1 gelangt. Protokollausgabe (d. h. Ergebnis des Münzwurfs) ist dann bei beiden Parteien $b = b_1 \oplus b_2 \in \{0,1\}$.

Intuitiv ist klar, daß die Protokollausgabe b gleichverteilt ist, sofern nur einer der beiden Teilnehmer eine „ehrliche“, also gleichverteilte Wahl seines jeweiligen Bits b_i vornimmt. Im Falle einer korrumpierten Partei P_1 (und einer ehrlichen Partei P_2) folgt dies daraus, daß sich P_1 schon im ersten Schritt auf b_1 festlegt, also b_1 unabhängig von b_2 wählen muß.

Entscheidendes Werkzeug bei diesem Protokoll ist natürlich ein Verfahren, welches es P_1 ermöglicht, sich bei P_2 auf ein Bit b_1 festzulegen, *ohne* dabei b_1 selbst zu veröffentlichen. Ein solches Verfahren wird *Bit-Commitment-Verfahren* genannt. Ermöglicht das Verfahren einer Partei sogar, sich auf einen Bitstring s festzulegen, spricht man von einem *String-Commitment-Verfahren*.

Bit-Commitment-Verfahren wurden auch beispielsweise in Goldreich u. a. [73]

benutzt, um Protokolle für allgemeine sichere Funktionsauswertungen zu konstruieren; es konnte also für *jede* durch einen PPT-Algorithmus berechenbare Funktion f mit n Eingaben und n Ausgaben ein Protokoll für n Parteien angegeben werden, welches die Auswertung von f an den Stellen der jeweiligen Parteieingaben realisiert. Dabei erhält jede Partei nur „ihre“ Ausgabe; selbst korrumpierte und sich nicht an das Protokoll haltende Parteien erfahren nicht mehr als ihre jeweilige Ausgabe und können auch die Ausgaben der anderen Parteien nicht weiter beeinflussen, als ihnen dies ohnehin durch Abändern ihrer eigenen Eingabe möglich wäre.

Nach dieser kurzen Motivation soll nun aber zunächst der Begriff des Commitment-Verfahrens fixiert werden:

Definition 4.8. *Ein* (String-)Commitment-Verfahren *sind zwei PPT-Algorithmen* Commit, Verify, *welche für alle* $k \in \mathbb{N}$ *und* $m, com, dec \in \{0,1\}^*$ *die folgenden syntaktischen Bedingungen erfüllen:*

- $\mathsf{Commit}(1^k, s) \in \{0,1\}^* \times \{0,1\}^*$,
- $\mathsf{Verify}(1^k, com, dec) \in \{0,1\}^* \cup \{\perp\}$.

Es wird Korrektheit *in dem Sinne gefordert, daß für alle* $k \in \mathbb{N}$, $s \in \{0,1\}^*$ *und alle möglichen* $(com, dec) \leftarrow \mathsf{Commit}(1^k, s)$ *gilt:* $\mathsf{Verify}(1^k, com, dec) = s$.

Zunächst soll bemerkt werden, daß Definition 4.8 offenbar nur *nicht-interaktive* Commitment-Verfahren modelliert; es ist durchaus üblich, auch *interaktive* Verfahren zu betrachten, bei denen die Commit- bzw. die Aufdeck-Phase eine *Interaktion* zwischen den beiden Protokollteilnehmern umfaßt. Diese Verallgemeinerung führt jedoch zu einer komplizierteren Definition von (klassischen) Sicherheitseigenschaften eines Commitment-Verfahrens und soll hier deshalb nicht weiter verfolgt werden.

Analog zur Definition von Public-Key-Krypto- oder Public-Key-Signatursystemen könnte man auch in Definition 4.8 vernachlässigbar wenige Fehler erlauben. Auch könnte man Einschränkungen der Menge der erlaubten Bitstrings s zulassen. Beide Verallgemeinerungen von Definition 4.8 sind jedoch hier nicht von Interesse.

Aus Definition 4.8 läßt sich leicht eine Definition eines *Bit-Commitment-Verfahrens* (im Gegensatz zum mit Definition 4.8 erfaßten *String Commitment*) gewinnen; man muß nur den erlaubten Wertebereich von s auf $\{0,1\}$ einschränken. In diesem Sinne kann jedes Commitment-Verfahren auch als Bit-Commitment-Verfahren angesehen werden – die sogleich zu betrachtenden Sicherheitskriterien übertragen sich ähnlich leicht.

Folgende etablierte Sicherheitskriterien modellieren die schon angesprochenen intuitiven Anforderungen an ein Commitment-Verfahren (siehe etwa Goldreich [70] für weitergehende Diskussionen und eine Formulierung bezüglich interaktiver Commitment-Verfahren):

Definition 4.9. *Ein Commitment-Verfahren* $C = (\mathsf{Commit}, \mathsf{Verify})$ *heißt* bindend*, wenn für jeden PPT-Algorithmus* A *die Funktion*

$$\mathsf{Pr}\left[(c, d_1, d_2) \leftarrow A(1^k) : \ \bot \neq \mathsf{Verify}(1^k, c, d_1) \neq \mathsf{Verify}(1^k, c, d_2) \neq \bot\right]$$

vernachlässigbar in k *ist. Gilt dies sogar für unbeschränkte Algorithmen* A*, so heißt* C uneingeschränkt bindend*.*

Ein bindendes Commitment-Verfahren garantiert also, daß kein polynomial beschränkter Algorithmus ein Commitment erzeugen kann, welches er auf zwei Arten aufdecken kann. Die gewünschte „verbergende“ Eigenschaft eines Commitment-Verfahrens läßt sich mittels einer IND-CCA-ähnlichen Definition fassen:

Definition 4.10. *Sei* $C = (\mathsf{Commit}, \mathsf{Verify})$ *ein Commitment-Verfahren. Dann sei für ein Paar* $A = (A_1, A_2)$ *von Algorithmen,* $b \in \{0, 1\}$ *und* $k \in \mathbb{N}$

$$\mathsf{P}^b_{C,A}(k) := \mathsf{Pr}\left[(s_0, s_1, h) \leftarrow A_1(1^k); (c, \cdot) \leftarrow \mathsf{Commit}(1^k, s_b) : \ A_2(1^k, c, h) = 1\right].^{13}$$

Es heißt C verbergend*, falls für jedes Paar* $A = (A_1, A_2)$ *von PPT-Algorithmen die Funktion*

$$\mathbf{Adv}^C_A(k) := \mathsf{P}^1_{C,A}(k) - \mathsf{P}^0_{C,A}(k)$$

vernachlässigbar in k *ist. Gilt dies für Paare* (A_1, A_2)*, so daß* A_1 *ausschließlich Tupel* (s_0, s_1, h) *mit* $|s_0| = |s_1|$ *ausgibt, heißt* C inhaltsverbergend*. Ist* $\mathbf{Adv}^C_A$ *sogar für Paare* (A_1, A_2) *von unbeschränkten Algorithmen vernachlässigbar, so heißt* C uneingeschränkt verbergend *bzw.* uneingeschränkt inhaltsverbergend*.*

Es ist bei einem verbergenden Commitment-Verfahren also keinem polynomial beschränkten Algorithmus möglich, zwei Commitments auf vorher selbst ausgewählte Bitstrings zu unterscheiden. Die Abschwächung „inhaltsverbergend“ ist insbesondere für uneingeschränkt bindende Commitment-Verfahren interessant: Hier zeigt ein einfaches Abzählargument, daß die *Länge* des Bitstrings, auf den sich festgelegt wird, nicht vollständig verborgen werden kann.

An dieser Stelle soll auf einen Fehler in Hofheinz und Müller-Quade [83] hingewiesen werden: Dort wurde eine schwächere Definition von „verbergend“ benutzt, bei der kein expliziter Algorithmus A_1 Bitstrings s_0 und s_1 wählt, sondern bei der lediglich die polynomiale Ununterscheidbarkeit (vgl. Definition A.10) der ersten Komponenten von $\mathsf{Commit}(1^k, s_0)$ bzw. $\mathsf{Commit}(1^k, s_1)$ für beliebige, aber *feste* $s_0, s_1 \in \{0, 1\}^*$ gefordert wird. Für eine solche Definition von „verbergend“ gilt der nachfolgende Satz 4.12 (der auch in [83] behauptet wurde) nicht.

Es ist einfach einzusehen, daß kein Verfahren C zugleich uneingeschränkt bindend *und* uneingeschränkt (inhalts-)verbergend sein kann. Andererseits gibt es sehr wohl Commitment-Verfahren, welche uneingeschränkt bindend sind (etwa das ursprünglich vorgeschlagene Verfahren aus [24]), und auch solche, die uneingeschränkt (inhalts-)verbergend sind, vgl. Naor und Yung [109], Damgård u. a. [52].

[13] Der Bitstring h dient hier nur zur Übermittlung von Statusinformationen von der ersten Phase des Angriffs (von A_1 ausgeführt) zur zweiten (von A_2 ausgeführt).

4.2.1 Simulierbare Commitments

Nun stellt sich die Frage, wie ein geeigneter *simulierbarkeitsbasierter* Sicherheitsbegriff für ein Commitment-Verfahren auszusehen hat. Eine diesbezügliche Formulierung von Commitment-Verfahren wäre höchst erstrebenswert, um die Analyse größerer, Commitments benutzender Protokolle zu modularisieren. Etwa könnte damit die allgemeine Protokollkonstruktion aus [73] simulierbar aufgebaut und analysiert werden.[14]

Genauer erhebt sich die Frage nach einer geeigneten Idealisierung, also einer idealen Funktionalität, welche intuitiv widerspiegelt, welche Garantien ein Commitment-Protokoll gibt. Die Formulierung einer solchen idealen Funktionalität geschah in Canetti und Fischlin [34]; unglücklicherweise konnte schon dort gezeigt werden, daß diese Funktionalität durch überhaupt kein reales Protokoll realisiert werden kann. Dies ist kein Problem der speziellen Formulierung aus [34], welches leicht behoben werden könnte. Das auftretende Problem soll nun anhand der in Abbildung 4.9 reproduzierten[15] Funktionalität $\mathcal{F}_{\text{MCOM}}$ aus [34] besprochen werden.

Die Funktionalität $\mathcal{F}_{\text{MCOM}}$

Protokollteilnehmer: $\mathcal{F}_{\text{MCOM}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

- **Commit-Phase**: Bei Erhalt einer Eingabe $(\texttt{Commit}, sid, ssid, P_i, P_j, b)$ mit $b \in \{0,1\}$ von P_i: Speichere das Tupel $(ssid, P_i, P_j, b)$, und sende die Nachricht $(\texttt{Receipt}, sid, ssid, P_i, P_j)$ an P_j und $\mathcal{S}$. Ignoriere alle weiteren `Commit`-Nachrichten mit derselben *ssid* von P_i an P_j.
- **Aufdeck-Phase**: Bei Erhalt einer Eingabe $(\texttt{Reveal}, sid, ssid)$ von P_i: Wenn schon ein Tupel $(ssid, P_i, P_j, b)$ gespeichert wurde, sende die Nachricht $(\texttt{Reveal}, sid, ssid, P_i, P_j, b)$ an P_j und $\mathcal{S}$. Ansonsten tue nichts.

Abbildung 4.9: Die Idealisierung $\mathcal{F}_{\text{MCOM}}$ aus [34] bzw. [39]

Zunächst ist zu bemerken, daß die Funktionalität mehrere Commitment-Vorgänge mittels „Untersitzungsnummern“ (engl. „subsession IDs“) *ssid* verwaltet. Davon abgesehen ist $\mathcal{F}_{\text{MCOM}}$ denkbar einfach gestaltet: Eine Partei P_i kann sich bei einer Partei P_j auf ein Bit b festlegen, indem sie eine entsprechende `Commit`-Nachricht an $\mathcal{F}_{\text{MCOM}}$ schickt, woraufhin P_j eine Bestätigung über diesen Vorgang erhält, *nicht* jedoch das Bit b. (Damit agiert also $\mathcal{F}_{\text{MCOM}}$ als „Treuhänder“ für das Bit b.) Erst später, auf eine entsprechende `Reveal`-Anfrage von P_i hin, deckt $\mathcal{F}_{\text{MCOM}}$ das Bit b gegenüber P_j auf.

Es ist unmittelbar klar, daß jedes Commitment-Verfahren, welches – als Protokoll interpretiert – $\mathcal{F}_{\text{MCOM}}$ realisiert, schon bindend und verbergend ist: Es müssen nur

[14] Ohne zu weit vorzugreifen sei hier bemerkt, daß genau dies in [39] geschah und ein bemerkenswertes Machbarkeitsresultat abgeleitet werden konnte.

[15] Es wird hier die Darstellung aus [39] gewählt, da die dort verwendeten Variablennamen auch für die später noch vorzustellenden Funktionalitäten und Protokolle benutzt werden.

die entsprechenden Angreifer aus den Definitionen 4.9 bzw. 4.10 auf Umgebungen $\mathcal{Z}$ reduziert werden, welche im realen Modell Angriffe mittels des Dummy-Angreifers ausführen.

Strenggenommen ist allerdings $\mathcal{F}_{\text{MCOM}}$ im Modell [27] schon allein deshalb nicht durch ein reales Protokoll realisierbar, weil $\mathcal{F}_{\text{MCOM}}$ jede `Commit`-Eingabe *sofort* akzeptiert. Hier tritt ein ähnliches Problem wie im Falle der Schlüsselaustauschfunktionalität $\mathcal{F}_{\text{KE}}$ aus [36] auf, siehe auch Abschnitt 4.1.1, wenn folgender Angriff durchgeführt wird: $\mathcal{Z}$ gibt P_i eine `Commit`-Eingabe für ein Bit $b = 0$. Direkt danach wird P_i korrumpiert, und verhält sich gegenüber P_j so, als ob ein reales Commitment auf $b = 1$ stattfindet. Ideal findet damit ein Commitment auf 0 statt, wohingegen im realen Modell ein Commitment auf 1 geschieht. Späteres Aufdecken führt zur Unterscheidung.

Dieses Problem kann aber mit ähnlichen Mitteln wie den in Abschnitt 4.1.5 beschriebenen durch eine geeignete Anpassung der Funktionalität behoben werden. (Alternativ löst die in Abschnitt 4.1.4 besprochene Modelländerung in der Revision [39] von [38] das generelle Problem der zu frühen Eingaben – auch für die Funktionalität $\mathcal{F}_{\text{MCOM}}$.)

Nicht so einfach behebbar ist folgendes Problem, welches schon in [34] erkannt wurde: Man nehme an, P_i sei schon zu Anfang des Protokollablaufs korrumpiert worden, und die Umgebung $\mathcal{Z}$ selbst führe – etwa mittels des Dummy-Angreifers und einer $\mathcal{Z}$-internen Simulation von P_i – ein Commitment bei P_j auf ein gleichverteilt gezogenes Bit b durch. Für eine auch bei späterem Aufdecken stimmige Simulation muß ein Simulator $\mathcal{S}$ nun in der Lage sein, aus dem von $\mathcal{Z}$ erzeugten Commitment das entsprechende Bit b zu extrahieren; $\mathcal{S}$ muß also das Commitment brechen. Eine $\mathcal{Z}$-interne Simulation eines solchen Simulators $\mathcal{S}$ kann nun benutzt werden, um bei anfänglich korrumpiertem P_j aus einem realen Commitment der unkorrumpierten Partei P_i das entsprechende Bit b zu extrahieren; dies ist nach Konstruktion im idealen Modell nicht möglich.

Die Betrachtung zeigt, daß hier ein prinzipielles Problem von Idealisierungen der Protokollaufgabe „Commitment“ vorliegt. Andererseits ist eine solche Idealisierung äußerst attraktiv: In [39] konnte gezeigt werden, daß allein unter Benutzung von Commitment-Bausteinen fast[16] jede Zweiparteienfunktionalität realisiert werden kann. Diese Aussage kann in [39] unter geeigneten Zusatzannahmen sogar auf Mehrparteienfunktionalitäten ausgedehnt werden; essentiell bleibt in jedem Fall aber die Benutzung eines idealisierten Commitment-Verfahrens.

Dieses Dilemma wurde in [34] wie folgt gelöst: Es wird eine Realisierung von $\mathcal{F}_{\text{MCOM}}$ im $\mathcal{F}_{\text{CRS}}$-hybriden Modell angegeben. Hierbei stellt $\mathcal{F}_{\text{CRS}}$ eine – aufgrund des Kompositionstheorems 3.2 zwangsläufig nicht durch ein reales Protokoll realisierbare – „Hilfsfunktionalität“ dar, welche das Vorhandensein eines öffentlichen und ideal gezogenen Zufallsstrings modelliert.[17]

[16] Die Aussage wurde in [39] für alle nicht-direkten Funktionalitäten gezeigt, welche Korruptionsbenachrichtigungen ignorieren. Diese Beschränkung ist angesichts der Möglichkeit passiver Korruptionen (Korruptionen, die am Verhalten einer Partei nichts ändern) sinnvoll.

[17] Bei einer geeigneten Komplexitätsannahme kann sich hier ein gleichverteilt gezogener k-Bit-

Entscheidend dabei ist, daß ein Simulator $\mathcal{S}$ im idealen Modell nun die Freiheit hat, den simulierten Zufallsstring *selbst* zu wählen, da $\mathcal{S}$ ja insbesondere Instanzen von $\mathcal{F}_{\text{CRS}}$ simuliert. Etwa könnte der Zufallsstring im realen Modell den öffentlichen Schlüssel eines Public-Key-Kryptosystems darstellen, zu dem der geheime Schlüssel völlig unbekannt ist; ein Simulator kann nun aber für seine Simulation des hybriden Protokolls einen öffentlichen Schlüssel wählen, zu dem er den geheimen Schlüssel kennt.

Diese Art von Simulationsvorteil nutzend konnte in [34] ein konkretes Protokoll angegeben werden, welches (von dem Problem des zu frühen Akzeptierens der Eingaben abgesehen) die Funktionalität $\mathcal{F}_{\text{MCOM}}$ realisiert. Dieses Protokoll ist jedoch sehr aufwendig und zudem – wie auch das später vorgeschlagene Protokoll aus [39] – sehr anfällig gegenüber Abschwächungen der Annahme über den Zufallsstring. Mit anderen Worten: Gesteht man dem realen Angreifer zu, die Wahl des Zufallsstrings zu beeinflussen oder im Extremfall sogar komplett zu bestimmen, so verlieren die betrachteten Protokolle nach Konstruktion sowohl die Eigenschaft des Verbergens als auch die des Bindens.

4.2.2 Eine andere Annahme

Thema der von Hofheinz und Müller-Quade in [83] veröffentlichten und nachfolgend vorgestellten Arbeit ist es nun, diese Nachteile durch Nutzen einer *anderen* „Helferfunktionalität“ als $\mathcal{F}_{\text{CRS}}$ zu beheben. Genauer sollen einfache Mechanismen angegeben werden, um ein klassisch sicheres (also verbergendes und bindendes) Commitment-Verfahren in ein simulierbar sicheres Protokoll umzuwandeln. Hierbei wird mit einem *Random Oracle* die etablierte Idealisierung einer Hashfunktion (siehe etwa Bellare und Rogaway [18]) genutzt.

Genauer wird sowohl unkorrumpierten Parteien als auch einem Angreifer ein *Orakel* (eben das „Random Oracle“) zur Verfügung gestellt, welches sich wie eine zufällig gewählte Funktion $\{0,1\}^* \to \{0,1\}^k$ verhält, also wie eine zufällig gewählte Hashfunktion. Entscheidend dabei ist zum einen, daß diese Funktion bei jedem Protokolldurchlauf neu gewählt wird (also nicht „vorhersehbar“ ist), und zum anderen, daß nur ein Black-Box-Zugriff auf diese Funktion erlaubt ist. Die Funktion kann also insbesondere nicht von einem Angreifer ausgewertet werden, *ohne* das Orakel explizit zu befragen. Mit dieser Idealisierung sind die vorzustellenden Protokolle simulierbar sicher – jedoch bleiben diese Protokolle klassisch sicher (d. h. bindend und verbergend), wenn das Random Oracle durch eine konkrete (also feste) kryptographische Hashfunktion ersetzt wird.

Damit kann in gewisser Weise doppelte Sicherheit garantiert werden: Unter der Annahme eines Random Oracles eignen sich die in [83] entwickelten Protokolle zur modularen Protokollkonstruktion im Modell aus [27], und selbst bei Aufgabe dieser starken Idealisierung ist der Sicherheitsverlust „nur“ der ohnehin unvermeidliche Verlust der Simulierbarkeit.

---

string vorgestellt werden.

Als erstes muß dazu das Hilfsmittel des Random Oracles im Modell von [27] formalisiert werden. Dies geschieht natürlich durch eine geeignete Funktionalität $\mathcal{F}_{\text{RO}}$, welche in Abbildung 4.10 dargestellt ist.

Die Funktionalität $\mathcal{F}_{\text{RO}}$

Protokollteilnehmer: $\mathcal{F}_{\text{RO}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

1. $\mathcal{F}_{\text{RO}}$ verwaltet eine interne Liste L (welche initial leer ist) von Paaren von Bitstrings.
2. Bei Erhalt einer Eingabe (sid, s) mit $s \in \{0,1\}^*$ von einer Partei P_i oder von $\mathcal{S}$, verfahre wie folgt:
   - Existiert ein Paar $(s, \tilde{h})$ in L, setze $h := \tilde{h}$.
   - Wähle andernfalls gleichverteilt $h \in \{0,1\}^k$ und speichere (s, h) in L.

   Sende schließlich der aktivierenden Maschine (sid, s, h) zurück und, falls diese Maschine eine Partei ist, weise den Angreifer an, diese Nachricht direkt auszuliefern.

Abbildung 4.10: Die Idealisierung $\mathcal{F}_{\text{RO}}$ eines Random Oracles aus [83]

Da diese Funktionalität die Idealisierung einer Hashfunktion darstellt, ist sie als direkte Funktionalität modelliert. Es ist klar, daß $\mathcal{F}_{\text{RO}}$ prinzipiell durch kein reales Protokoll realisiert werden kann; alles andere würde nach den gemachten Bemerkungen auch kein hinreichendes Hilfsmittel für eine Realisierung einer Commitment-Funktionalität darstellen.

Zwar kann durch Auswerten des Random Oracles an einer festen Stelle auch die Funktionalität $\mathcal{F}_{\text{CRS}}$ (bei Gleichverteilung des durch $\mathcal{F}_{\text{CRS}}$ gezogenen Zufallsstrings) realisiert werden; ein solcher „Umweg“ über $\mathcal{F}_{\text{RO}}$ beseitigt jedoch nicht die oben angesprochenen Nachteile der Commitment-Protokolle im $\mathcal{F}_{\text{CRS}}$-hybriden Modell.

Als letzte Vorbereitung soll noch eine für weitere Betrachtungen geeignete Idealisierung von Commitment-Verfahren gegeben werden: Die Funktionalität $\mathcal{F}_{\text{SCOM}}$ aus [83] (siehe Abbildung 4.11) behandelt zum einen das schon angesprochene technische Problem der Funktionalität $\mathcal{F}_{\text{MCOM}}$ aus Abbildung 4.9, und zum anderen ermöglicht sie Commitments auf beliebige Bitstrings (und nicht nur einzelne Bits, wie im Falle von $\mathcal{F}_{\text{MCOM}}$).

Man beachte, daß $\mathcal{F}_{\text{SCOM}}$ dem Simulator erlaubt, eine Auswahl über die zu akzeptierende Eingabe zu treffen, falls mehrere `Commit`-Eingaben vorliegen. Dabei wird natürlich nicht der jeweilige Bitstring preisgegeben, auf den sich eine Partei festlegen möchte; $\mathcal{S}$ erhält lediglich Existenzinformationen über `Commit`-Anfragen. Diese Abschwächung der Funktionalität erlaubt es dem Simulator, im Falle einer adaptiv korrumpierten Partei eine vormals getätigte `Commit`-Anfrage zu „überstimmen“, falls nicht schon eine Bestätigung der Festlegung ausgegeben wurde. Damit

Die Funktionalität $\mathcal{F}_{\text{SCOM}}$

Protokollteilnehmer: $\mathcal{F}_{\text{SCOM}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$.

- **Commit-Phase**:
  1. Bei Erhalt einer Eingabe $(\texttt{Commit}, sid, ssid, P_i, P_j, s)$ mit $s \in \{0,1\}^*$ von P_i, sende eine Benachrichtigung $(\texttt{Request}, sid, ssid, P_i, P_j)$ an $\mathcal{S}$.
  2. Bei Erhalt von $(\texttt{Ready}, sid, ssid, P_i, P_j, \ell)$ vom Angreifer, und wenn schon wenigstens ℓ $(\texttt{Commit}, sid, ssid, P_i, P_j, m)$-Nachrichten von P_i empfangen wurden (mit evtl. unterschiedlichen $m \in \{0,1\}^*$): Speichere das Tupel $(ssid, P_i, P_j, m)$ für das m aus der ℓ-ten dieser Eingaben und sende $(\texttt{Receipt}, sid, ssid, P_i, P_j)$ an P_j. Ignoriere alle weiteren `Commit`- und `Ready`-Nachrichten mit diesen Werten von $ssid$, P_i und P_j.
- **Aufdeck-Phase**: Bei Erhalt einer Eingabe $(\texttt{Reveal}, sid, ssid)$ von P_i: Wenn schon ein Tupel $(ssid, P_i, P_j, b)$ gespeichert wurde, sende die Nachricht $(\texttt{Reveal}, sid, ssid, P_i, P_j, b)$ an P_j und $\mathcal{S}$. Ansonsten tue nichts.

Abbildung 4.11: Die Idealisierung $\mathcal{F}_{\text{SCOM}}$ aus [83]

ist – ohne intuitive Sicherheitseinbußen – das technische Problem des zu frühen Festlegens behoben.

4.2.3 Eine allgemeine Protokollkonstruktion

Das Protokoll HC_C aus Abbildung 4.12 nutzt seinerseits ein bindendes und verbergendes Commitment-Verfahren $C = (\mathsf{Commit}, \mathsf{Verify})$. Aus C entsteht so unter Benutzung eines Random Oracles (also im $\mathcal{F}_{\text{RO}}$-hybriden Modell) ein Protokoll, welches die ideale Funktionalität $\mathcal{F}_{\text{SCOM}}$ realisiert:

Satz 4.11. *Sei $C = (\mathsf{Commit}, \mathsf{Verify})$ ein bindendes und inhaltsverbergendes Commitment-Verfahren. Dann realisiert das Protokoll HC_C die ideale Funktionalität $\mathcal{F}_{\text{SCOM}}$ bezüglich authentifizierter Kanäle und adaptiver Angreifer.*

Beweis. Es muß für jeden hybriden Angreifer $\mathcal{H}$, welcher Protokoll HC_C attackiert, ein Simulator $\mathcal{S} = \mathcal{S}(\mathcal{H})$ angegeben werden, der solche Angriffe im $\mathcal{F}_{\text{SCOM}}$-idealen Modell simuliert.

Sei also ein solches $\mathcal{H}$ gegeben. Das zugehörige $\mathcal{S}$ simuliert intern einen kompletten Protokollablauf von HC_C mit simulierten Parteien $P_1^{(s)}, \ldots, P_n^{(s)}$, einem simulierten Angreifer $\mathcal{H}$ und – soweit benötigt – simulierten $\mathcal{F}_{\text{RO}}$-Instanzen. Wie im Beweis von Satz 4.7 wird alle Kommunikation von $\mathcal{H}$ mit der Umgebung zur nicht-simulierten Umgebung $\mathcal{Z}$ durchgestellt.

Das Protokoll HC_C

Dies sind Instruktionen für Parteien $P_1, \ldots, P_n$ zum Ausführen eines Commitment-Protokolls im $\mathcal{F}_{\text{RO}}$-hybriden Modell. Im folgenden bezeichne $\mathcal{O}_{sid}(s)$ die Antwort der $\mathcal{F}_{\text{RO}}$-Instanz mit Sitzungsnummer sid auf die Anfrage s. Weiter sei $C = (\mathsf{Commit}, \mathsf{Verify})$ ein bindendes und inhaltsverbergendes Commitment-Verfahren. Sei schließlich $p \in \mathbb{Z}[k]$ eine obere Schranke für die Anzahl der von Commit bei Eingaben $(1^k, h)$ (mit $h \in \{0,1\}^k$) benutzten Zufallsbits.

1. Bei einer Eingabe $(\texttt{Commit}, sid, ssid, P_i, P_j, s)$ mit $s \in \{0,1\}^*$ berechnet[a] eine Partei P_i

$$(com_1, \cdot) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}_{sid}(ssid, i, j, s, r_1); r_2)$$
$$(com_2, dec_2) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}_{sid}(r_2))$$

   für gleichverteilt gezogene $r_1 \in \{0,1\}^k$ und $r_2 \in \{0,1\}^{p(k)}$. P_i sendet dann $(sid, ssid, com_1, com_2)$ an P_j und speichert $(ssid, j, s, r_1, r_2, dec_2)$. Weitere $(\texttt{Commit}, sid, ssid, P_i, P_j, \cdot)$-Eingaben werden ignoriert.

2. Bei Erhalt von $(sid, ssid, com_1, com_2)$ mit $com_1, com_2 \in \{0,1\}^*$ von einer Partei P_i speichert eine Partei P_j das Tupel $(ssid, i, com_1, com_2)$ und gibt $(\texttt{Receipt}, sid, ssid, P_i, P_j)$ aus. Jede weitere Nachricht mit derselben $ssid$ von P_i wird ignoriert.

3. Bei Erhalt einer Eingabe $(\texttt{Reveal}, sid, ssid, P_j)$, und falls P_i schon ein Tupel $(ssid, j, s, r_1, r_2, dec_2)$ (für irgendwelche[b] s, r_1, r_2, dec_2) gespeichert hat, sendet P_i die Nachricht $(sid, ssid, s, r_1, r_2, dec_2)$ an P_j. Weitere $(\texttt{Reveal}, sid, ssid, P_j)$-Eingaben (mit diesen $sid, ssid, P_j$) werden ignoriert.

4. Bei Erhalt von $(sid, ssid, s, r_1, r_2, dec_2)$ mit $s, r_1, r_2, dec_2 \in \{0,1\}^*$ von einer Partei P_i, und falls P_j schon vorher eine $(sid, ssid, com_1, com_2)$-Nachricht von P_i erhalten hat, berechnet eine Partei P_j zunächst $o_2 \leftarrow \mathsf{Verify}(1^k, com_2, dec_2)$. Ist $o_2 = \mathcal{O}_{sid}(r_2)$, und ist weiter com_1 die erste Komponente von $\mathsf{Commit}(1^k, \mathcal{O}_{sid}(ssid, i, j, s, r_1); r_2)$, so gibt P_j den Wert $(\texttt{Reveal}, sid, ssid, P_i, P_j, s)$ aus und ignoriert alle weiteren $(sid, ssid, \ldots)$-Nachrichten von P_i. Andernfalls tut P_j nichts.

[a] Hier sei auf Notation A.7 verwiesen.
[b] Bei Existenz eines solchen Tupels sind s, r_1, r_2 und dec_2 eindeutig bestimmt.

Abbildung 4.12: Das Commitment-Protokoll HC_C

Für eine erfolgreiche Simulation bleibt nur noch sicherzustellen, daß Ein- und Ausgaben sowie Korruptionen im idealen Modell konsistent mit der $\mathcal{S}$-internen Simulation eines hybriden Protokollablaufes sind. (Man beachte, daß $\mathcal{Z}$ modellinterne Korruptionsbenachrichtigungen erhält.) Im einzelnen geschieht dies wie folgt:

- Korrumpiert die $\mathcal{S}$-interne Simulation von $\mathcal{H}$ eine Partei $P_i^{(s)}$, so korrumpiert $\mathcal{S}$ zunächst die Dummy-Partei $\tilde{P}_i$ und integriert ignorierte $\tilde{P}_i$-Eingaben in den Zustand von $P_i^{(s)}$.

- Bei Erhalt von $(\texttt{Receipt}, sid, ssid, P_i, P_j)$ von $\mathcal{F}_{\text{SCOM}}$ zu einem Zeitpunkt, zu dem $\tilde{P}_i$ (und damit $P_i^{(s)}$) unkorrumpiert ist, muß $\mathcal{S}$ den Versand eines Commitments von $P_i^{(s)}$ an $P_j^{(s)}$ simulieren. Hierfür wählt $\mathcal{S}$ gleichverteilt $o_1 \in \{0,1\}^k$, $r_2, r_3 \in \{0,1\}^{p(k)}$ und berechnet

$$(com_1, dec_1) \leftarrow \mathsf{Commit}(1^k, o_1; r_2)$$
$$(com_2, dec_2) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}_{sid}(r_2); r_3).$$

  (Dabei wird das simulierte Random Oracle mit der Sitzungsnummer *sid* befragt.) Dann simuliert $\mathcal{S}$ eine Nachricht $(sid, ssid, com_1, com_2)$ von $P_i^{(s)}$ an $P_j^{(s)}$ und speichert diese Nachricht zusammen mit r_2, r_3 und o_1. Wichtig hierbei ist, daß $\mathcal{S}$ so ein Commitment generiert hat, welches später – von $\mathcal{S}$ – als Commitment auf einen beliebigen Bitstring s aufgedeckt werden kann; hierzu muß $\mathcal{S}$ nur das simulierte Random Oracle $\mathcal{F}_{\text{RO}}$ mit der Sitzungsnummer *sid* manipulieren. (Wie dies zu geschehen hat, wird bei der Behandlung von `Reveal`-Eingaben illustriert.)

- Liefert die Simulation von $\mathcal{H}$ eine Nachricht $(sid, ssid, com_1, com_2)$ von $P_i^{(s)}$ an $P_j^{(s)}$ aus, und hat $P_j^{(s)}$ noch keine Nachricht mit dieser *ssid* von P_i erhalten, so muß $\mathcal{S}$ eine `Receipt`-Ausgabe von $\tilde{P}_j$ erzwingen. Ist $\tilde{P}_i$ (und damit $P_i^{(s)}$) unkorrumpiert, so kann dies durch eine `Ready`-Nachricht von $\mathcal{S}$ an $\mathcal{F}_{\text{SCOM}}$ und Auslieferung einer dann gesendeten `Receipt`-Ausgabe an $\tilde{P}_j$ erfolgen. (Im Falle von authentifizierten Kanälen und bei unkorrumpiertem $P_i^{(s)}$ kann eine solche Nachricht nur durch eine explizite Eingabe an $\tilde{P}_i$ entstanden sein.)

  Ist andererseits $P_i^{(s)}$ korrumpiert, dann muß $\mathcal{S}$ zunächst den Bitstring s, auf den sich mit der Nachricht festgelegt wird, extrahieren, um sich dann im Namen der korrumpierten Dummy-Partei $\tilde{P}_i$ auf s festlegen zu können.[18] Dies geschieht durch Betrachtung aller (durch Parteien oder das simulierte $\mathcal{H}$ gemachten) Anfragen an $\mathcal{O}_{sid}$; genauer kommen damit nur polynomial viele s-r_1-r_2-Kombinationen in Frage, für die jeweils

$$com_1 \stackrel{?}{=} \mathsf{Commit}(1^k, \mathcal{O}_{sid}(ssid, i, j, s, r_1); r_2)$$

[18] Diese Verletzung der „Verbergend"-Eigenschaft des Commitments gelingt natürlich nur, weil $\mathcal{S}$ Einsicht in die Random Oracles seiner Simulation hat.

überprüft werden muß. Kann auf diese Weise kein s gewonnen werden, liegt nach Konstruktion mit überwältigender Wahrscheinlichkeit ein ungültiges Commitment vor, welches nicht aufgedeckt werden kann.[19] In diesem Fall kann also gefahrlos $s = 0$ gesetzt werden. Ist s nicht eindeutig bestimmt, muß eine Kollision von $\mathcal{O}_{sid}$-Ausgaben vorliegen, welche nur mit vernachlässigbarer Wahrscheinlichkeit auftritt; auch hier kann $s = 0$ gesetzt werden.

Sobald s bestimmt ist, sendet $\mathcal{S}$ im Namen der korrumpierten Dummy-Partei $\tilde{P}_i$ eine Eingabe $(\texttt{Commit}, sid, ssid, P_i, P_j, s)$ an $\mathcal{F}_{\text{SCOM}}$. Anschließend sendet $\mathcal{S}$ die Nachricht $(\texttt{Ready}, sid, ssid, P_i, P_j, \ell)$ an $\mathcal{F}_{\text{SCOM}}$, wobei ℓ die Anzahl der insgesamt erhaltenen $(\texttt{Request}, sid, ssid, P_i, P_j)$-Benachrichtigungen ist; $\mathcal{S}$ legt sich damit also (im Namen von $\tilde{P}_i$) auf s fest.

- Bei Erhalt von $(\texttt{Reveal}, sid, ssid, P_i, P_j, s)$ von $\mathcal{F}_{\text{SCOM}}$ (was impliziert, daß $\tilde{P}_i$ und damit $P_i^{(s)}$ immer noch unkorrumpiert ist) läßt $\mathcal{S}$ die simulierte Partei $P_i^{(s)}$ ein Commitment auf s bei P_j und mit Sitzungs- und Untersitzungsnummer sid bzw. $ssid$ berechnen. Dabei erzwingt $\mathcal{S}$ durch Manipulation von Zufallsbändern der simulierten Maschinen, daß

  - die simulierte Funktionalität $\mathcal{F}_{\text{RO}}$ mit Sitzungsnummer sid auf die Anfrage $(ssid, i, j, s, r_1)$ von $P_i^{(s)}$ mit dem entsprechenden $o_1 \in \{0,1\}^k$ antwortet, welches von $\mathcal{S}$ gespeichert wurde,[20]
  - $P_i^{(s)}$ bei diesem Commitment die von $\mathcal{S}$ benutzten und gespeicherten Werte r_2 und r_3 benutzt.

  Die so erzeugte Commitment-Nachricht an $P_j^{(s)}$ wird verworfen; der Prozeß diente nur dazu, den Zustand von $P_i^{(s)}$ und von $\mathcal{F}_{\text{RO}}$ den tatsächlich von $\mathcal{Z}$ gegebenen Eingaben anzupassen. (Genauer muß $P_i^{(s)}$s interner Zustand noch dahingehend manipuliert werden, daß bei späteren Korruption das beschriebene Commitment zeitlich direkt vor Senden der entsprechenden Nachricht an $P_j^{(s)}$ vorzufinden ist, s. o.)

  Schließlich wird $P_i^{(s)}$ noch eine $(\texttt{Reveal}, sid, ssid, P_i, P_j, s)$-Eingabe gegeben, da eine solche Eingabe auch im idealen Modell stattgefunden hat. Die zugehörige `Reveal`-Nachricht von $\mathcal{F}_{\text{SCOM}}$ an $\tilde{P}_j$ wird von $\mathcal{S}$ ausgeliefert, sobald das simulierte $P_j^{(s)}$ diese Ausgabe generiert.

- Dieses Verfahren wird auch im Falle einer Korruption einer Partei $P_i^{(s)}$ für alle von $\mathcal{S}$ für $P_i^{(s)}$ generierten, aber noch nicht aufgedeckten Commitments

[19] Hier ist neben der „Bindend“-Eigenschaft von C entscheidend, daß die Untersitzungsnummer $ssid$ und die Identitäten i und j mithaschiert wurden, so daß eine Wiederverwendung von Hashwerten in anderen Untersitzungen nicht möglich ist.

[20] Dies ist nur möglich, falls $\mathcal{F}_{\text{RO}}$ nicht schon vorher für $(ssid, i, j, s, r_1)$ befragt wurde; das allerdings passiert nach Wahl von r_1 nur mit vernachlässigbarer Wahrscheinlichkeit.

angewendet. (Man beachte, daß $\mathcal{S}$ im Falle einer Korruption von $\tilde{P}_i$ auch alle Eingaben von $\tilde{P}_i$ erfährt.)

- Generiert eine Partei $P_j^{(s)}$ schließlich `Reveal`-Ausgabe, muß die Dummy-Partei $\tilde{P}_j$ eine gleichlautende Ausgabe generieren. Die obigen Anweisungen stellen sicher, daß hierfür $\mathcal{S}$ lediglich die entsprechende `Reveal`-Nachricht von $\mathcal{F}_{\text{SCOM}}$ an $\tilde{P}_j$ ausliefern muß, falls der Committer $\tilde{P}_i$ zum Zeitpunkt der `Reveal`-Eingabe unkorrumpiert war.

  Falls aber $\tilde{P}_i$ korrumpiert ist, so muß $\mathcal{S}$ diese `Reveal`-Eingabe im Namen von $\tilde{P}_i$ geben; dies ist ohne weiteres möglich, da $\mathcal{S}$ den entsprechenden Bitstring s entweder bei Korruption von $\tilde{P}_i$ oder durch Extraktion aus einem Commitment (siehe oben) erfahren hat.[21]

Nach dieser Besprechung garantiert $\mathcal{S}$ im idealen Modell eine Sicht für $\mathcal{Z}$, welche *identisch* zur Sicht von $\mathcal{Z}$ im $\mathcal{F}_{\text{RO}}$-hybriden Modell ist, unter der Bedingung, daß sich nie eine `Reveal`-Ausgabe einer in $\mathcal{S}$ simulierten Partei $P_j^{(s)}$ von der entsprechenden Ausgabe von $\tilde{P}_j$ unterscheidet. Dieses Ereignis tritt aber nur ein, wenn $\mathcal{S}$ nicht in der Lage ist, einen Bitstring s eindeutig aus einem Commitment zu extrahieren, oder wenn $\mathcal{S}$ ein selbst-generiertes Commitment nicht aufdecken kann.

Beide Ereignisse treten – wie oben begründet – nur mit vernachlässigbarer Wahrscheinlichkeit ein, womit die Behauptung folgt. □

4.2.4 Erhalt klassischer Sicherheitseigenschaften

Das Protokoll HC_C gestattet somit die Umwandlung von klassisch sicheren Commitment-Verfahren in simulierbar sichere Protokolle. Tatsächlich kann HC_C selbst wieder als Commitment-Verfahren betrachtet werden: Hierzu müssen nur alle Sitzungsnummern und Parteiidentitäten weggelassen werden. Es ergibt sich somit für ein Commitment-Verfahren $C = (\mathsf{Commit}, \mathsf{Verify})$ und ein Random Oracle $\mathcal{O}$ die algorithmische Entsprechung $\text{HC}_C^{\mathcal{O}} = (\mathsf{Commit}_C^{\mathcal{O}}, \mathsf{Verify}_C^{\mathcal{O}})$ des Protokolls HC_C wie folgt:

$$\mathsf{Commit}_C^{\mathcal{O}}(1^k, s) = ((com_1, com_2), (s, r_1, r_2, dec_2))$$

mit gleichverteilten $r_1 \in \{0,1\}^k$, $r_2 \in \{0,1\}^{p(k)}$ (für das $p \in \mathbb{Z}[k]$ aus Protokoll HC_C) und

$$\begin{aligned}(com_1, \cdot) &\leftarrow \mathsf{Commit}(1^k, \mathcal{O}(s, r_1); r_2),\\ (com_2, dec_2) &\leftarrow \mathsf{Commit}(1^k, \mathcal{O}(r_2)).\end{aligned}$$

Schließlich ergibt sich

$$v \leftarrow \mathsf{Verify}_C^{\mathcal{O}}\left(1^k, (com_1, com_2), (s, r_1, r_2, dec_2)\right)$$

[21] Hier wird die „Bindend"-Eigenschaft von C und die „Kollisionsresistenz" von $\mathcal{F}_{\text{RO}}$ genutzt.

mit $v = s$ genau dann, wenn für $o_2 \leftarrow \mathsf{Verify}(1^k, com_2, dec_2)$ gilt, daß $o_2 = \mathcal{O}(r_2)$ und com_1 die erste Komponente von $\mathsf{Commit}(1^k, \mathcal{O}(s, r_1); r_2)$ ist. Andernfalls wird $v = \perp$ gesetzt.

Nun ist leicht einzusehen, daß das so beschrieben Verfahren $\text{HC}_C^{\mathcal{O}}$ ein bindendes und (inhalts-)verbergendes Commitment-Verfahren ist, sofern C diese Eigenschaften besitzt. Das ist auch nicht weiter verwunderlich, denn das Protokoll HC_C realisiert ja schon $\mathcal{F}_{\text{SCOM}}$. Viel interessanter ist die folgende Frage: Bleibt eine gewisse „Restsicherheit“ von $\text{HC}_C^{\mathcal{O}}$ erhalten, wenn das Random Oracle $\mathcal{O}$ durch eine kryptographische Hashfunktion ersetzt wird? Simulierbare Sicherheit kann nun nicht mehr vorliegen; es ist aber nicht auszuschließen, daß das neue Verfahren noch die klassischen Sicherheitskriterien „bindend“ und „verbergend“ erfüllt.

Tatsächlich gilt folgender Satz, wenn für eine Familie $\mathcal{H} = \{h_k\}_{k \in \mathbb{N}}$ von kollisionsresistenten Hashfunktionen (vgl. Definition A.9) mit $\text{HC}_C^{\mathcal{H}}$ das soeben beschriebene Verfahren $\text{HC}_C^{\mathcal{O}}$ bezeichne, in dem jedoch alle Auswertungen von $\mathcal{O}$ durch Auswertungen von h_k ersetzt werden:

Satz 4.12. *Sei $\mathcal{H} = \{h_k\}_{k \in \mathbb{N}}$ eine Familie von kollisionsresistenten Hashfunktionen, und sei C ein bindendes und inhaltsverbergendes Commitment-Verfahren. Dann ist $\text{HC}_C^{\mathcal{H}}$ ein bindendes und sogar verbergendes Commitment-Verfahren. Ist C uneingeschränkt inhaltsverbergend, so ist $\text{HC}_C^{\mathcal{H}}$ uneingeschränkt verbergend.*

Beweis. Zunächst vererbt sich die Eigenschaft, überhaupt ein Commitment-Verfahren gemäß Definition 4.8 zu sein, direkt von C auf $\text{HC}_C^{\mathcal{H}}$. Zum Nachweis der „Bindend“-Eigenschaft von $\text{HC}_C^{\mathcal{H}}$ sei ein PPT-Algorithmus A angenommen, welcher mit nicht-vernachlässigbarer Wahrscheinlichkeit ein $\text{HC}_C^{\mathcal{H}}$-Commitment zusammen mit zwei gültigen Decommitments auf verschiedene Bitstrings findet. Dann findet A nach Konstruktion von $\text{HC}_C^{\mathcal{H}}$ entweder eine $\mathcal{H}$-Kollision oder zwei verschiedene C-Decommitments zu einem C-Commitment. Beides widerspricht den gemachten Annahmen.

Es bleibt zu zeigen, daß $\text{HC}_C^{\mathcal{H}}$ verbergend ist. Sei also $C = (\mathsf{Commit}, \mathsf{Verify})$. Sei weiter $\overline{\mathsf{Commit}}_C^{\mathcal{H}}$ die Abwandlung von $\mathsf{Commit}_C^{\mathcal{H}}$, welche anstelle eines C-Commitments auf $h_k(r_2)$ eines auf 1^k berechnet. Genauer ist also

$$((com_1, com_2), (s, r_1, r_2)) \leftarrow \overline{\mathsf{Commit}}_C^{\mathcal{H}}(1^k, s)$$

mit

$$\begin{aligned}(com_1, \cdot) &\leftarrow \mathsf{Commit}(1^k, h_k(s, r_1)),\\ (com_2, \cdot) &\leftarrow \mathsf{Commit}(1^k, 1^k)\end{aligned}$$

und gleichverteiltem $r_1 \in \{0,1\}^k$. Bei offensichtlicher Definition eines geeigneten PPT-Algorithmus $\overline{\mathsf{Verify}}_C^{\mathcal{H}}$ ergibt sich somit ein Commitment-Verfahren $\overline{\text{HC}}_C^{\mathcal{H}}$.

Sei nun $A = (A_1, A_2)$ nun ein Paar von PPT-Algorithmen, welches die „Verbergend“-Eigenschaft von $\text{HC}_C^{\mathcal{H}}$ bricht (vgl. Definition 4.10). Sei in der Notation von Definition 4.10 genauer $\mathbf{Adv}_A^{\text{HC}_C^{\mathcal{H}}}$ nicht-vernachlässigbar in k.

Sei dabei zunächst angenommen, daß auch die Funktion $\mathbf{Adv}_A^{\overline{\mathrm{HC}}_C^{\mathcal{H}}}$ nicht-vernachlässigbar in k ist. Sei dann $A' = (A'_1, A'_2)$ ein Paar von PPT-Algorithmen, so daß einerseits $(h_k(s_0, r_1), h_k(s_1, r_1), h) \leftarrow A'_1(1^k)$ für $(s_0, s_1, h) \leftarrow A_1(1^k)$ und gleichverteiltes $r_1 \in \{0,1\}^k$ ist, und andererseits $A'_2(1^k, c, h)$ seine Ausgabe als Ergebnis von $A_2(1^k, (c, \mathsf{Commit}(1^k, 1^k)), h)$ bestimmt. Damit ist $\mathbf{Adv}_A^{\overline{\mathrm{HC}}_C^{\mathcal{H}}} = \mathbf{Adv}_{A'}^{C}$, was der Annahme über C widerspricht.

Es muß also $\mathbf{Adv}_A^{\overline{\mathrm{HC}}_C^{\mathcal{H}}}$ vernachlässigbar in k sein, so daß – in der Notation von Definition 4.10 – die Funktion

$$\mathbf{Adv}_A^{\mathrm{HC}_C^{\mathcal{H}}}(k) - \mathbf{Adv}_A^{\overline{\mathrm{HC}}_C^{\mathcal{H}}}(k) = \left(\mathsf{P}^1_{\mathrm{HC}_C^{\mathcal{H}},A}(k) - \mathsf{P}^1_{\overline{\mathrm{HC}}_C^{\mathcal{H}},A}(k)\right) - \left(\mathsf{P}^0_{\mathrm{HC}_C^{\mathcal{H}},A}(k) - \mathsf{P}^0_{\overline{\mathrm{HC}}_C^{\mathcal{H}},A}(k)\right)$$

nicht-vernachlässigbar in k ist. Demnach ist wenigstens einer der beiden umklammerten Terme rechts vom Gleichheitszeichen nicht-vernachlässigbar. Sei genauer etwa $\mathsf{P}^i_{\mathrm{HC}_C^{\mathcal{H}},A}(k) - \mathsf{P}^i_{\overline{\mathrm{HC}}_C^{\mathcal{H}},A}(k)$ nicht-vernachlässigbar ($i \in \{0,1\}$).

Sei $A' = (A'_1, A'_2)$ ein Paar von PPT-Algorithmen mit $(r_2, 1^k, r_2) \leftarrow A'_1(1^k)$ für gleichverteiltes $r_2 \in \{0,1\}^{p(k)}$, und sei weiter die Ausgabe b von $A'_2(1^k, c, r_2)$ durch $A_2(1^k, (com_1, c), h)$ gegeben, wobei hierfür com_1 die erste Komponente von $\mathsf{Commit}(1^k, h_k(s_i, r_1); r_2)$ für gleichverteiltes $r_1 \in \{0,1\}^k$ und $(s_0, s_1, h) \leftarrow A_1(1^k)$ sei. Damit ist $\mathbf{Adv}^{C}_{A'}(k) = \mathsf{P}^i_{\mathrm{HC}_C^{\mathcal{H}},A}(k) - \mathsf{P}^i_{\overline{\mathrm{HC}}_C^{\mathcal{H}},A}(k)$ nicht-vernachlässigbar in k, was auch der Annahme über C widerspricht.

Da diese Reduktionen auch für Paare (A_1, A_2) von nicht-PPT-Algorithmen anwendbar sind, zeigt dies die Behauptung auch im uneingeschränkten Fall. □

Der entscheidende Vorteil von $\mathrm{HC}_C^{\mathcal{H}}$ gegenüber C ist dabei, daß $\mathrm{HC}_C^{\mathcal{H}}$ bei Idealisierung der benutzten Hashfunktion durch ein Random Oracle tatsächlich sogar simulierbare Sicherheit garantiert; dies ist bei C nicht notwendig der Fall.

4.2.5 Eine weitere Protokollkonstruktion

Der Übergang von einem Commitment-Verfahren C zu einem Verfahren $\mathrm{HC}_C^{\mathcal{H}}$ erhält leider keine *uneingeschränkte* „Bindend"-Eigenschaft von C. Diese geht beim Haschieren des Bitstrings s, r_1 verloren; hier gibt es für gewisse s eine Vielzahl von möglichen Urbildern, die durch eine geeignete vollständige Suche gefunden werden können. Auf diese Weise gelingt es einem *unbeschränkten* Algorithmus A, mit einem Commitment und *mehreren* gültigen Decommitments aufzuwarten.

Sollte gewünscht sein, auch eine etwaige uneingeschränkte „Bindend"-Eigenschaft von C beizubehalten, ist eine etwas aufwendigere Aufarbeitung des Bitstrings s, auf den sich festgelegt werden soll, nötig.[22] Eine Beschreibung einer solchen Konstruktion soll nun geschehen.

Dabei ist zunächst zu beachten, daß ein uneingeschränkt bindendes Commitment-Verfahren schon allein aus einem Abzählargument heraus die *Länge* von s

[22] Es reicht hier für ein simulierbar sicheres Verfahren nicht aus, zu einem Hashwert auch noch s selbst hinzuzufügen; das würde den Simulator der Möglichkeit berauben, Commitments zu erzeugen, welche später nach Belieben aufgedeckt werden können.

nicht völlig verbergen kann. Für eine Betrachtung simulierbarer Sicherheit der anzugebenden Protokollkonstruktion spiegelt sich dies in einer leicht geänderten idealen Funktionalität $\mathcal{F}_{\text{BSCOM}}$ wider, welche sich von $\mathcal{F}_{\text{SCOM}}$ nur dadurch unterscheidet, daß der Simulator $\mathcal{S}$ bei einem zu erfolgenden Commitment die Länge des jeweiligen Bitstrings s erfährt. Genauer enthalten die entsprechenden `Request`-Nachrichten an $\mathcal{S}$ auch die Länge $|s|$ von s. (Natürlich erhält $\mathcal{S}$ ansonsten *keine* Information über s.)

Als technisches Hilfsmittel für eine geeignete Vorverarbeitung von s sei nun ein PPT-Algorithmus $F_{\mathcal{H}}$ vorgestellt, welcher über eine Familie $\mathcal{H} = \{h_k\}_{k\in\mathbb{N}}$ von kollisionsresistenten Hashfunktionen parametrisiert ist. Bei einer Eingabe $b \in \{0,1\}$ zieht $F_{\mathcal{H}}$ zunächst gleichverteilt $r \in \{0,1\}^{k+1}$ und setzt $r_0 = h_k(r)$ und $r_1 = h_k(r_0) = h_k(h_k((r))$. Ist dann $h_k(r_1) \neq r_0$, so gibt $F_{\mathcal{H}}$ die Ausgabe $r_b r_{1-b} \in \{0,1\}^{2k}$. Im Falle $h_k(r_1) = r_0$ allerdings gibt $F_{\mathcal{H}}$ die Ausgabe $b^{2k} \in \{0,1\}^{2k}$ aus.[23]

Für längere Eingaben $s \in \{0,1\}^*$ (wobei s die Bitdarstellung $s = s_1 \cdots s_{|s|}$ habe) berechnet $F_{\mathcal{H}}$ seine Ausgabe $F_{\mathcal{H}}(s_1) \cdots F_{\mathcal{H}}(s_{|s|}) \in \{0,1\}^{2k|s|}$ bitweise und mit jeweils neuen Zufallswahlen. Eine Formulierung $F_{\mathcal{O}}$ bezüglich eines Random Oracles $\mathcal{O}$ ist dann offensichtlich.

Es kann nun $F_{\mathcal{H}}$ durch einen PPT-Algorithmus $F_{\mathcal{H}}^{-1}$ invertiert werden: Im Spezialfall einer Eingabe lr mit $l, r \in \{0,1\}^k$ wertet $F_{\mathcal{H}}^{-1}$ die Prädikate $P_0 = [h_k(l) = r]$ und $P_1 = [h_k(r) = l]$ aus. Gilt dann $P_0 \neq P_1$, so gibt $F_{\mathcal{H}}^{-1}$ ein Bit b aus, welches anzeigt, ob P_1 gilt. Ist aber $P_0 = P_1$, so gibt $F_{\mathcal{H}}^{-1}$ das erste Bit von l aus.

Für Eingaben $f_1 \cdots f_n$ für ein $n \in \mathbb{N}$ und gewisse $f_i \in \{0,1\}^{2k}$ berechnet $F_{\mathcal{H}}^{-1}$ seine Ausgabe $F_{\mathcal{H}}^{-1}(f_1) \cdots F_{\mathcal{H}}^{-1}(f_n) \in \{0,1\}^n$ blockweise. Eingaben anderer Form werden ohne Ausgabe ignoriert. Damit ist klar, daß $F_{\mathcal{H}}^{-1}(F_{\mathcal{H}}(s)) = s$ für alle $s \in \{0,1\}^*$ und Sicherheitsparameter $k \in \mathbb{N}$ gilt. Eine analoge Konstruktion $F_{\mathcal{O}}^{-1}$ mit $F_{\mathcal{O}}^{-1}(F_{\mathcal{O}}(s)) = s$ für alle s, k ist auch für den Algorithmus $F_{\mathcal{O}}$ in offensichtlicher Weise möglich.[24]

Mit diesem Algorithmus $F_{\mathcal{H}}$ zur Vorverarbeitung der Commitment-Nachrichten ausgestattet kann nun ein Protokollkonstruktion angegeben werden, welche auch eine eventuelle uneingeschränkte „Bindend"-Eigenschaft des ursprünglichen Protokolls erhält. Genauer ist das Protokoll BHC_C aus Abbildung 4.13 genau wie schon HC_C über ein bindendes und verbergendes Commitment-Schema C parametrisiert. Es soll nun zunächst – im Random-Oracle-Modell – die simulierbare Sicherheit nachgewiesen werden:

Satz 4.13. *Sei $C = (\mathsf{Commit}, \mathsf{Verify})$ ein bindendes und inhaltsverbergendes Commitment-Verfahren. Dann realisiert das Protokoll BHC_C die ideale Funktionalität $\mathcal{F}_{\text{BSCOM}}$ bezüglich authentifizierter Kanäle und adaptiver Angreifer.*

Beweis. Der Beweis folgt dem von Satz 4.11, und es werden hier nur die notwendigen Änderungen des Simulators $\mathcal{S}$ aus dem dortigen Beweis angegeben. Genauer ist

[23] Hier hat $F_{\mathcal{H}}$ also wegen $h_k(r) = r_0 = h_k(r_1)$ für $r \neq r_1$ eine h_k-Kollision gefunden, weshalb dieser Fall nur mit vernachlässigbarer Wahrscheinlichkeit auftritt.

[24] In [83] wurde ein leicht anderer Algorithmus $F_{\mathcal{H}}$ gewählt, für dessen Benutzung eine etwas stärkere Annahme über $\mathcal{H}$ nötig ist.

Das Protokoll BHC_C

Dies sind Instruktionen für Parteien $P_1, \ldots, P_n$ zum Ausführen eines Commitment-Protokolls im $\mathcal{F}_{\text{RO}}$-hybriden Modell. Es bezeichne wieder $\mathcal{O}_{sid}(s)$ die Antwort der $\mathcal{F}_{\text{RO}}$-Instanz mit Sitzungsnummer *sid* auf die Anfrage s. Weiter sei $C = (\mathsf{Commit}, \mathsf{Verify})$ ein bindendes und inhaltsverbergendes Commitment-Verfahren. Schließlich sei $q \in \mathbb{Z}[k, \ell]$ eine obere Schranke für die Anzahl der von Commit bei Eingaben $(1^k, h)$ (mit $h \in \{0,1\}^\ell$) benutzten Zufallsbits.

1. Bei einer Eingabe $(\texttt{Commit}, sid, ssid, P_i, P_j, s)$ mit $s \in \{0,1\}^*$ berechnet eine Partei P_i zunächst $f \leftarrow F_{\mathcal{O}_{sid}}(s)$ und damit

$$(com_1, \cdot) \leftarrow \mathsf{Commit}(1^k, (\mathcal{O}_{sid}(ssid, i, j, f), f); r_2)$$
$$(com_2, dec_2) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}_{sid}(r_2))$$

   für gleichverteilt gezogenes $r_2 \in \{0,1\}^{q(k,k+|f|)}$. P_i sendet dann die Nachricht $(sid, ssid, com_1, com_2)$ an P_j und speichert $(ssid, j, f, r_2, dec_2)$. Weitere $(\texttt{Commit}, sid, ssid, P_i, P_j, \cdot)$-Eingaben werden ignoriert.

2. Bei Erhalt von $(sid, ssid, com_1, com_2)$ mit $com_1, com_2 \in \{0,1\}^*$ von einer Partei P_i speichert eine Partei P_j das Tupel $(ssid, i, com_1, com_2)$ und gibt $(\texttt{Receipt}, sid, ssid, P_i, P_j)$ aus. Jede weitere Nachricht mit derselben *ssid* von P_i wird ignoriert.

3. Bei Erhalt einer Eingabe $(\texttt{Reveal}, sid, ssid, P_j)$, und falls P_i ein Tupel $(ssid, j, f, r_2, dec_2)$ (für irgendwelche[a] s, r_2, dec_2) gespeichert hat, sendet P_i die Nachricht $(sid, ssid, f, r_2, dec_2)$ an P_j. Weitere $(\texttt{Reveal}, sid, ssid, P_j)$-Eingaben (mit diesen $sid, ssid, P_j$) werden ignoriert.

4. Bei Erhalt von $(sid, ssid, s, r_2, dec_2)$ mit $s, r_2, dec_2 \in \{0,1\}^*$ von einer Partei P_i, und falls P_j schon vorher eine $(sid, ssid, com_1, com_2)$-Nachricht von P_i erhalten hat, berechnet eine Partei P_j zunächst $o_2 \leftarrow \mathsf{Verify}(1^k, com_2, dec_2)$. Ist $o_2 = \mathcal{O}_{sid}(r_2)$, und ist weiter com_1 die erste Komponente von $\mathsf{Commit}(1^k, (\mathcal{O}_{sid}(ssid, i, j, f), f); r_2)$, so gibt P_j den Wert $(\texttt{Reveal}, sid, ssid, P_i, P_j, F^{-1}_{\mathcal{O}_{sid}}(s))$ aus und ignoriert alle weiteren $(sid, ssid, \ldots)$-Nachrichten von P_i. Andernfalls tut P_j nichts.

[a] f, r_2 und dec_2 sind dann eindeutig bestimmt.

Abbildung 4.13: Das Commitment-Protokoll BHC_C

es nur nötig, zu beschreiben, wie $\mathcal{S}$ etwaige im Namen einer korrumpierten Partei gemachte Commitments aufdecken kann und wie $\mathcal{S}$ Commitments generiert, welche später – von $\mathcal{S}$ – in beliebiger Weise aufgedeckt werden können.

Um ein später beliebig aufdeckbares Commitment auf eine Nachricht der Länge $|s|$ zu generieren,[25] verfährt $\mathcal{S}$ wie folgt: $\mathcal{S}$ wählt gleichverteilt $f \in \{0,1\}^{2k|s|}$, berechnet $o_1 \leftarrow \mathcal{O}_{sid}(ssid, i, j, f)$ und generiert ein Commitment (com_1, com_2) mittels

$$(com_1, \cdot) \leftarrow \mathsf{Commit}(1^k, (o_1, f); r_2)$$
$$(com_2, dec_2) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}_{sid}(r_2); r_3)$$

für gleichverteilte $r_2 \in \{0,1\}^{q(k,k+|f|)}$ und $r_3 \in \{0,1\}^{q(k,k)}$. (Siehe Abbildung 4.13 für eine Definition von q.) Später kann ein solches Commitment dann durch Manipulation einer $\mathcal{F}_{\mathrm{RO}}$-Instanz (d. h. durch geeignete Wahl von Urbildern der k-Bit-Blöcke in f) in beliebiger Weise aufgedeckt werden, *sofern* $\mathcal{F}_{\mathrm{RO}}$ nicht schon für einen der k-Bit-Blöcke in f befragt wurde. Die Wahrscheinlichkeit hierfür ist aber vernachlässigbar, da C als verbergend angenommen wurde. Weiter ist die Verteilung eines solchen f *nach* Aufdecken eines Commitments polynomial ununterscheidbar von der Verteilung „legitim“ generierter f (vgl. Definition A.10), so daß diese Manipulation die Sicht der Umgebung $\mathcal{Z}$ nur in vernachlässigbarer Weise beeinflußt. (Da C inhaltsverbergend ist, ist die Verteilung von f *vor* dem Aufdecken nicht relevant.)

Das Aufdecken eines legitim generierten Commitments kann wie im Fall des Protokolls $\textsc{hc}_C$ geschehen, da aus f mittels $F_{\mathcal{O}}^{-1}$ schon der zugehörige Bitstring s gewonnen werden kann. Man beachte wieder, daß nicht unter Benutzung von $\mathcal{F}_{\mathrm{RO}}$ generierte Commitments später nur mit vernachlässigbarer Wahrscheinlichkeit aufgedeckt werden können, weshalb für nicht extrahierbare Commitments ein beliebiges s (etwa $s = 0$) gewählt werden darf.

Mit diesen Änderungen kann der Beweis von Satz 4.11 übernommen werden. □

Weiter kann $\textsc{bhc}_C$ – genau wie $\textsc{hc}_C$ – durch Weglassen der Sitzungsnummern und Parteiidentitäten als Commitment-Verfahren betrachtet werden. Es ergibt sich also auch hier für ein Commitment-Verfahren $C = (\mathsf{Commit}, \mathsf{Verify})$ und ein Random Oracle $\mathcal{O}$ die algorithmische Entsprechung $\textsc{bhc}_C^{\mathcal{O}} = (\mathsf{Commit}_C^{\mathcal{O}}, \mathsf{Verify}_C^{\mathcal{O}})$ des Protokolls $\textsc{bhc}_C$ wie folgt:

$$\mathsf{Commit}_C^{\mathcal{O}}(1^k, s) = ((com_1, com_2), (f, r_2, dec_2))$$

mit

$$f \leftarrow F_{\mathcal{O}}(s),$$
$$(com_1, \cdot) \leftarrow \mathsf{Commit}(1^k, (\mathcal{O}(f), f); r_2),$$
$$(com_2, dec_2) \leftarrow \mathsf{Commit}(1^k, \mathcal{O}(r_2)).$$

[25] Hier wird benutzt, daß $\mathcal{S}$ von $\mathcal{F}_{\mathrm{BSCOM}}$ im Falle einer `Commit`-Anfrage über die Länge der entsprechenden Nachricht s in Kenntnis gesetzt wird.

für gleichverteiltes $r_2 \in \{0,1\}^{q(k,k+|f|)}$. Schließlich ergibt sich

$$v \leftarrow \mathsf{Verify}_C^{\mathcal{O}}\left(1^k, (com_1, com_2), (f, r_2, dec_2)\right)$$

mit $v = F_{\mathcal{O}}^{-1}(f)$ genau dann, wenn für $o_2 \leftarrow \mathsf{Verify}(1^k, com_2, dec_2)$ gilt, daß $o_2 = \mathcal{O}(r_2)$ und com_1 die erste Komponente von $\mathsf{Commit}(1^k, \mathcal{O}(f), f; r_2)$ ist. Andernfalls wird $v = \perp$ gesetzt.

Wieder sei für eine Familie $\mathcal{H} = \{h_k\}_{k\in\mathbb{N}}$ von kollisionsresistenten Hashfunktionen mit $\text{BHC}_C^{\mathcal{H}}$ das soeben beschriebene Verfahren $\text{BHC}_C^{\mathcal{O}}$ bezeichnet, in dem jedoch alle Auswertungen von $\mathcal{O}$ durch Auswertungen von h_k ersetzt werden. Entscheidend für die hiesigen Betrachtungen ist natürlich, daß eine uneingeschränkte „Bindend"-Eigenschaft von C auch durch $\text{BHC}_C^{\mathcal{H}}$ erhalten wird. Das soll sogleich gezeigt werden:

Satz 4.14. *Sei $\mathcal{H} = \{h_k\}_{k\in\mathbb{N}}$ eine Familie von kollisionsresistenten Hashfunktionen, und sei C ein bindendes und inhaltsverbergendes Commitment-Verfahren. Dann ist auch $\text{BHC}_C^{\mathcal{H}}$ ein bindendes und inhaltsverbergendes Commitment-Verfahren. Ist C sogar verbergend, uneingeschränkt (inhalts-)verbergend oder uneingeschränkt bindend, so hat auch $\text{BHC}_C^{\mathcal{H}}$ die jeweilige Eigenschaft.*

Beweis. Daß $\text{BHC}_C^{\mathcal{H}}$ ein Commitment-Verfahren ist, folgt aus $F_{\mathcal{H}}^{-1} \circ F_{\mathcal{H}} = \text{id}_{\{0,1\}^*}$. Die „Bindend"-Eigenschaft von $\text{BHC}_C^{\mathcal{H}}$ folgt wie im Beweis von Satz 4.12. Allerdings gibt hier jeder Algorithmus, welcher die „Bindend"-Eigenschaft von $\text{BHC}_C^{\mathcal{H}}$ bricht, wegen $F_{\mathcal{H}}^{-1} \circ F_{\mathcal{H}} = \text{id}_{\{0,1\}^*}$ Anlaß zu einem Algorithmus, welcher in jedem Fall die „Bindend"-Eigenschaft von C bricht. Hier wird also die Kollisionsresistenz von $\mathcal{H}$ nicht benötigt, was auch eine Reduktion von unbeschränkten Angreifern ermöglicht.

Auch der Nachweis der „Verbergend"-Eigenschaft von $\text{BHC}_C^{\mathcal{H}}$ verläuft analog zum Beweis von Satz 4.12. Zu beachten ist nur, daß die entsprechenden Reduktionen nur dann Angreifer A_1' auf die „Verbergend"-Eigenschaft von C liefern, welche nur dann Bitstrings s_0, s_1 *gleicher* Länge ausgeben, wenn die entsprechenden A_1-Angreifer auf $\text{BHC}_C^{\mathcal{H}}$ diese Eigenschaft haben. □

4.2.6 Offene Fragen

Neben den etablierten Eigenschaften „bindend" und „verbergend" für ein Commitment-Verfahren existiert auch noch die etwas neuere Eigenschaft der „non-malleability", vgl. Dolev u. a. [55]. Diese fordert ganz ähnlich wie die entsprechende Eigenschaft für Public-Key-Kryptosysteme, daß es keinem PPT-Algorithmus möglich ist, aus einem gegebenen Commitment ein neues zu generieren, so daß die zugehörigen Klartexte in einer PPT-berechenbaren Relation stehen. (Natürlich ist die Charakterisierung in dieser Form noch nicht präzise, da ja ein Commitment noch nicht unbedingt einen einzigen Klartext festlegt; eine genaue Definition kann in [55] gefunden werden.)

Zwar kann relativ einfach eingesehen werden, daß „non-malleability" schon durch simulierbare Sicherheit impliziert wird (insbesondere sind die Verfahren $\text{HC}_C^{\mathcal{O}}$ und

$\textsc{bhc}_C^{\mathcal{O}}$ non-malleable); es ist aber augenblicklich unklar, ob sich eine „non-malleability“-Eigenschaft eines Commitment-Verfahrens C auch auf $\textsc{hc}_C^{\mathcal{H}}$ oder $\textsc{bhc}_C^{\mathcal{H}}$ überträgt.

5 Netzwerkmodellierungen

Das im vorangehenden benutzte Modell aus Canetti [27] verwendet ein asynchrones und nachrichtengetriebenes Scheduling. Es ist also zu jedem Zeitpunkt immer nur eine Partei aktiv, und die Auslieferung von Nachrichten kann von einem Angreifer beliebig verzögert werden. Damit ist es einer Partei prinzipiell nicht möglich, zu beurteilen, ob eine andere Partei eine Nachricht nie gesandt hat, oder ob nur die Auslieferung dieser Nachricht bislang verzögert wurde.

Weiter kann eine dritte Partei erst recht nicht beurteilen, ob ein Kommunikationskanal zwischen zwei anderen Parteien nicht benutzbar ist, weil eine der beiden Parteien ihn einfach ignoriert, oder ob der Kanal vom Angreifer durch fehlende Auslieferung blockiert wird.

Diese Situation führt dazu, daß eine Partei durch Nicht-Senden einer Nachricht prinzipiell ein größeres Protokoll zum Stillstand bringen kann. Das ist weder theoretisch befriedigend noch in allen Fällen praktisch motivierbar. In diesem Kapitel soll deshalb eine Abänderung der Modelle aus [27], Canetti u. a. [39] vorgeschlagen werden, mit welcher sich Kommunikationskanäle mit *garantierter* Auslieferung modellieren lassen. Für dieses Modell soll dann als interessantes Ergebnis – und weitere Motivation für das Modell selbst – hergeleitet werden, daß sich gewisse Protokollaufgaben bei Annahme garantierter Nachrichtenauslieferung nicht realisieren lassen; dies gilt selbst dann, wenn man Protokolle erlaubt, welche als Bausteine Instanzen der Primitiven „Oblivious Transfer“ und „Broadcast“ verwenden.

Besonders interessant dabei: Im asynchronen Modell aus [39] lassen sich insbesondere im sogleich vorzustellenden abgeänderten Modell nicht realisierbare Protokollaufgaben *sehr wohl* realisieren. In der Tat lassen sich im Modell [39] – mit kleinen Einschränkungen – alle als ideale Funktionalitäten formulierbaren Protokollaufgaben realisieren; dies folgt aus einem Ergebnis in [39].

Bevor diese Zusammenhänge genauer beleuchtet werden können, soll jedoch das abgeänderte Berechnungsmodell für Protokollausführungen von Hofheinz und Müller-Quade aus [82] vorgestellt werden. Wichtig dabei ist, daß dieses Modell – im Gegensatz zu den bisher besprochenen Modellen – eine unmittelbare Nachrichtenauslieferung zwischen Parteien oder idealen Funktionalitäten und Parteien garantiert. In diesem Sinne sind also die Aktionsmöglichkeiten des realen *und* die des idealen Angreifers eingeschränkt; beide haben weniger Kontrolle über das zugrundegelegte Netzwerk als im asynchronen Fall.

Ein solches *synchrones* Modell wurde schon in den Arbeiten Canetti [26], Pfitzmann und Waidner [116] betrachtet. Jedoch stellt die Arbeit [26] kein hinreichend allgemeines Kompositionstheorem zur Verfügung; weiter ist für die Arbeit [116] keine direkte Verbindung zur asynchronen Modellierung [27] offen-

sichtlich. Dies ist insbesondere für eine Beziehung zu den zahlreichen im asynchronen Modell [27] formulierten Ergebnissen (insbesondere der Machbarkeitsaussage aus [39]) von Belang.

Die Grundidee des nun vorzustellenden Modells ist dabei vergleichbar mit der Grundmotivation der Arbeiten [116, 26], Pfitzmann und Waidner [117], [27]. Es wird also wieder ein reales Protokoll mit einer Idealisierung der jeweiligen Protokollaufgabe verglichen; die genauen Änderungen gegenüber dem schon diskutierten Modell [27] sollen nun besprochen werden.

5.1 Das Berechnungsmodell

Die in den vorgehenden Kapiteln aufgedeckten Unstimmigkeiten (siehe insbesondere Abschnitt 3.2.2) im Modell [27] sollen natürlich in dem nun besprochenen Modell beseitigt werden, soweit möglich.

Deshalb wird gefordert, daß – solange nicht explizit anders vermerkt – für *jede* im folgenden betrachtete ITM M ein festes, nur maschinenabhängiges Polynom $p \in \mathbb{Z}[k]$ existiert, so daß M jede Aktivierung (unabhängig vom Inhalt der Bänder) nach höchstens $p(k)$ Turing-Schritten beendet und entweder in einen Halte- oder einen Wartezustand übergeht. Zusätzlich wird für alle im folgenden betrachteten Umgebungen $\mathcal{Z}$ und Angreifer $\mathcal{A}$ (bzw. Simulatoren $\mathcal{S}$) gefordert, daß sie nach polynomial vielen Aktivierungen anhalten; genauer wird für diese Maschinen die Existenz eines jeweiligen, nur maschinenabhängigen Polynoms $q \in \mathbb{Z}[k]$ gefordert, so daß die Maschine nach höchstens $q(k)$ Aktivierungen in einen Haltezustand übergeht.

Somit können prinzipiell ideale Funktionalitäten und Protokolle formuliert werden, welche nicht nach einer willkürlichen und im voraus festzulegenden Anzahl von Aktivierungen ihren Dienst einstellen. Hier ist aber zu beachten, daß andererseits jeder Simulator nach polynomial vielen Runden gehalten haben muß; denn genauer werden nur Benutzungen und Angriffe betrachtet, welche einen polynomialen Zeitraum nicht überschreiten. Für einen reibungslosen Beweis des Kompositionstheorems wird aber immer noch eine polynomiale Beschränkung aller Maschinen *pro Aktivierung* angenommen.[1]

Um den problematischen Effekt der Formulierung aus [27] zu vermeiden, daß eine hinreichend (polynomial) mächtige Umgebung einen Angreifer – insbesondere also einen Simulator – durch ständige Aktivierungen schon zu Anfang eines Protokollablaufs zum Anhalten zwingen kann (vgl. die Diskussion in Abschnitt 3.2.2), wird folgender Mechanismus eingeführt: Jeder Angreifer (also insbesondere jeder Simulator) darf jederzeit in einen speziellen Wartezustand übergehen, aus welchem er nicht *von der Umgebung* aktiviert werden kann. Jede Nachricht von der Umgebung an den Angreifer wird dann einfach verworfen – der Simulator *blockiert* somit

[1] Eine diesbezügliche Verallgemeinerung, welche immer noch sichere Komposition von Protokollen erlaubt, wird in Hofheinz, Müller-Quade und Unruh [87] erreicht.

Nachrichten von der Umgebung. Andererseits wird der Angreifer noch durch alle anderen Ereignisse (etwa Nachrichten von idealen Funktionalitäten) aktiviert.

Ein ähnlicher Mechanismus wurde auch in Backes [5] für das Modell [117] zu einem ähnlichen Zweck verwendet. Neben der technischen Notwendigkeit kann hier auch als praktische Motivation angeführt werden, daß es einer Maschine – etwa bei durch getrennte Netzwerkkarten realisierten Anbindungen – möglich sein kann, Nachrichten aus gewissen Verbindungen „abzublocken“.

Das Kommunikationsmodell verwirklicht eine *synchrone* Kommunikation der Parteien und idealen Funktionalitäten: Die Berechnung verläuft in *Runden*, wobei in jeder Runde die Parteien gleichzeitig aktiviert werden. Nachrichten zwischen den Parteien oder zwischen Parteien und idealen Funktionalitäten werden zwischen diesen Berechnungsschritten automatisch ausgeliefert; der Angreifer darf zwischen den Parteien verschickte Nachrichten lesen, hat jedoch keinen Einfluß auf deren Zustellung. Zwar darf der Angreifer Nachrichten an Parteien oder ideale Funktionalitäten senden; diese werden jedoch automatisch mit einem Absenderfeld versehen, welches den Angreifer als Absender identifiziert. Ein ähnlicher Mechanismus ist auch für alle Kommunikation zwischen den Parteien vorgesehen, so daß in diesem Sinne eine authentifizierte Kommunikation mit garantierter Auslieferung implementiert wird. Insbesondere ist es dem Angreifer nicht möglich, Absenderadressen zu „fälschen“. Um die Eingangs-Kommunikationsbänder dabei nicht zu blockieren (etwa durch eine übermäßig lange Nachricht des Angreifers an eine Partei), wird eine Stückelung des Inhalts aller Partei-Eingangs-Kommunikationsbänder nach Absender (engl. „interleaving“) vorgenommen. Weiter wird nach jeder Aktivierung einer ITM M deren Eingangs-Kommunikationsband automatisch gelöscht, damit auch hier keine unerwarteten Effekte durch zu lange Nachrichten auftreten können.

5.1.1 Reales und hybrides Modell

Mit diesen Vorbemerkungen ausgestattet kann nun eine detaillierte Beschreibung der Berechnungen in einem realen oder hybriden Modell vorgenommen werden. Tatsächlich kann das reale Modell (in welchem ja nur ein Protokoll ohne „Hilfsfunktionalitäten“ ausgeführt wird) als Spezialfall eines hybriden Modells aufgefaßt werden, so daß es reicht, eine Beschreibung des $\{\mathcal{F}_i\}$-hybriden Modells für eine endliche Menge $\{\mathcal{F}_i\}$ von ITMs – den idealen Funktionalitäten – zu geben. Ein reales Protokoll kann dann im $\emptyset$-hybriden Modell betrachtet werden.

Protokollteilnehmer sind also die $\mathcal{F}_i$, eine Umgebung $\mathcal{Z}$, ein Angreifer $\mathcal{A}$ und natürlich die Parteien $P_1, \ldots, P_n$. $\mathcal{Z}$ darf wieder – wenn aktiviert – die Eingabebänder der Parteien beschreiben und deren Ausgabebänder lesen. Der Angreifer $\mathcal{A}$ darf wieder alle Kommunikations-Ausgangsbänder der Parteien lesen, und außerdem – durch Übergang in einen speziellen Zustand – eine Partei P_i korrumpieren. Im letzteren Fall wird nicht nur der Zustand und der Inhalt der Bänder von P_i, sondern auch die Folge aller Vorgängerzustände und -bandinhalte[2] auf das Ein-

[2] Aus naheliegenden Gründen sind hier nur die bislang gelesenen Zellen des Zufallsbands gemeint.

gangs-Kommunikationsband von $\mathcal{A}$ geschrieben. Anschließend darf $\mathcal{A}$ im Namen von P_i kommunizieren, also Nachrichten auf das Ausgangs-Kommunikationsband von P_i schreiben sowie das Eingangs-Kommunikationsband von P_i lesen. Weiter erhalten alle $\mathcal{F}_i$ und $\mathcal{Z}$ automatisch eine Nachricht, daß P_i korrumpiert wurde.

Genauer läßt sich ein Protokollablauf im $\{\mathcal{F}_i\}$-hybriden Modell durch folgenden Algorithmus beschreiben, wobei initial alle Bänder der Maschinen bis auf das Sicherheitsparameterband (welches den für alle Maschinen gleichen Sicherheitsparameter $k \in \mathbb{N}$ in unärer Codierung enthalte), das Zufallsband und das Identitätsband leer seien.

1. **Angriffsphase**: Dies ist ein nachrichtengetriebenes Zusammenspiel zwischen $\mathcal{Z}$, $\mathcal{A}$ und den $\mathcal{F}_i$, nur können die $\mathcal{F}_i$ und $\mathcal{Z}$ nicht direkt kommunizieren.

   Zunächst wird $\mathcal{Z}$ mit der Eingabe „`round-start`“ aktiviert. Nachdem $\mathcal{Z}$ diese Aktivierung beendet hat, werden alle Nachrichten, welche $\mathcal{Z}$ an $\mathcal{A}$ geschrieben hat, ausgeliefert[3]. Hat $\mathcal{Z}$ keine Nachricht an $\mathcal{A}$ geschrieben oder angehalten, ist der Protokollablauf beendet, und der Inhalt der ersten Zelle des Ausgabebands von $\mathcal{Z}$ wird als die (binäre) Ausgabe des Ablaufs interpretiert.

   Andernfalls wird $\mathcal{A}$ als nächstes aktiviert, es sei denn, $\mathcal{A}$ blockiert Nachrichten von $\mathcal{Z}$ (vgl. die Diskussion weiter oben; in diesem Fall wird $\mathcal{Z}$ als nächstes aktiviert). Hat $\mathcal{A}$ während seiner Aktivierung eine Nachricht an $\mathcal{Z}$ geschrieben, wird $\mathcal{Z}$ nach Auslieferung *aller* dieser Nachrichten wieder aktiviert. Andernfalls wird die erste Funktionalität $\mathcal{F}_i$ (in Ordnung der Maschinenidentitäten), welcher $\mathcal{A}$ eine Nachricht geschrieben hat – gegebenenfalls im Namen einer korrumpierten Partei P_i –, nach Auslieferung aller dieser Nachrichten an $\mathcal{F}_i$ aktiviert. Nachdem diese Funktionalität $\mathcal{F}_i$ dann ihre Aktivierung beendet hat, werden alle Nachrichten, die $\mathcal{F}_i$ an $\mathcal{A}$ geschrieben hat, ausgeliefert, und $\mathcal{A}$ wird wieder aktiviert.

   Sollte $\mathcal{A}$ in einer Aktivierung aber weder Nachrichten an $\mathcal{A}$ noch an ein $\mathcal{F}_i$ geschrieben haben, beginnen die Berechnungsphasen:

2. **Parteiberechnungen**: Alle Nachrichten, welche eine Partei P_i an eine andere Partei P_j geschrieben hat, werden ausgeliefert. Dann werden alle Parteien *gleichzeitig* aktiviert. Nachdem alle P_i ihre Aktivierung beendet haben, werden Nachrichten von den P_i an $\mathcal{A}$ bzw. an die $\mathcal{F}_i$ ausgeliefert.

3. **Funktionalitätenberechnungen**: Alle $\mathcal{F}_i$ werden *gleichzeitig* mit Eingabe „`computation`“ aktiviert.[4] Nachdem alle $\mathcal{F}_i$ ihre Aktivierung beendet haben, werden Nachrichten von den $\mathcal{F}_i$ an $\mathcal{A}$ bzw. die Parteien P_i ausgeliefert. Hiernach wird wieder mit Schritt 1 (der Angriffsphase) fortgefahren.

[3] Mit „Ausliefern“ ist gemeint: Die Nachrichten werden automatisch vom Ausgangs-Kommunikationsband der Umgebung $\mathcal{Z}$ auf das Eingangs-Kommunikationsband von $\mathcal{A}$ verschoben.

[4] Durch diese spezielle Eingabe kann ein $\mathcal{F}_i$ eine Aktivierung durch $\mathcal{A}$ von einer „regulären“ Aktivierung in dieser Phase unterscheiden.

Man beachte, daß der Angreifer $\mathcal{A}$ keinen Zugriff auf die Kommunikation der Parteien mit den idealen Funktionalitäten hat. Andererseits darf ein Angreifer die von einer korrumpierten Partei gesandten Nachrichten abhängig von den Nachrichten wählen, welche in *derselben* Runde von anderen Parteien gesandt wurden. Da alle beteiligten ITMs eine Aktivierung nach polynomial vielen Schritten beenden und zudem $\mathcal{A}$ und $\mathcal{Z}$ nach polynomial vielen Aktivierungen anhalten, umfaßt ein Protokollablauf immer nur polynomial viele Schritte.

Weiter haben die Parteien (und der Angreifer) Zugriff auf nur *eine* Instanz jeder Funktionalität $\mathcal{F}_i$. Dies stellt keine echte Einschränkung dar, da eine einzelne Funktionalität im Sinne von Canetti und Rabin [41] parallelisiert werden kann. Wie das genau im Falle des hiesigen Modells geschieht, soll in Abschnitt 5.1.4 beschrieben werden.

Der Begriff des *Protokolls* soll nun gefaßt werden als eine Menge $P_1, \ldots, P_n$ von ITMs (den Parteien). Meistens wird mit einem Protokoll π direkt auch ein geeignetes hybrides Modell (also eine Menge $\{\mathcal{F}_i\}$ von idealen Funktionalitäten) assoziiert, damit eine Formulierung von π überhaupt Sinn hat. Die Zufallsvariable der Ausgabe einer Umgebung $\mathcal{Z}$ bei Ausführung im $\{\mathcal{F}_i\}$-hybriden Modell mit einem Angreifer $\mathcal{A}$ und einem Protokoll $\pi = \{P_1, \ldots, P_n\}$ bei Sicherheitsparameter k heiße $\mathcal{Z}(\{\mathcal{F}_i\}, \pi, \mathcal{A}, k)$. Damit ergibt sich folgende Sicherheitsdefinition, welche zwei hybride Protokolle vergleicht:

Definition 5.1. *Sei π ein n-Parteien-Protokoll im $\{\mathcal{F}_i\}$-hybriden Modell, und sei τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\}$-hybriden Modell.* π realisiert τ *(geschrieben $\pi \geq_{\mathsf{s}} \tau$) genau dann, wenn für jeden Angreifer $\mathcal{A}$ ein Angreifer $\mathcal{S}$ (der* Simulator*) existiert, so daß für alle $\mathcal{Z}$ die Funktion*

$$\mathbf{Adv}^{\pi,\tau}_{\mathcal{A},\mathcal{S},\mathcal{Z}}(k) := |\mathsf{Pr}[\mathcal{Z}(\{\mathcal{F}_i\}, \pi, \mathcal{A}, k) = 1] - \mathsf{Pr}[\mathcal{Z}(\{\mathcal{G}_j\}, \tau, \mathcal{S}, k) = 1]|$$

vernachlässigbar in k ist. Gilt dies sogar für in jeder Aktivierung unbeschränkte Umgebungen $\mathcal{Z}$, welche aber immer noch nach höchstens polynomial vielen Aktivierungen halten, so sagt man, daß π uneingeschränkt *τ realisiert (geschrieben $\pi \gg_{\mathsf{s}} \tau$).*

Man beachte, daß für den uneingeschränkten Fall nur die Beschränkung der Umgebung, aber nicht die des Angreifers fallengelassen wird. Damit impliziert $\pi \gg_{\mathsf{s}} \tau$ schon $\pi \geq_{\mathsf{s}} \tau$, was nicht immer der Fall wäre, wenn „$\gg_{\mathsf{s}}$“ auch Sicherheit bezüglich unbeschränkter Angreifer fordern würde. Zudem wurde für „$\gg_{\mathsf{s}}$“ die Beschränkung einer polynomialen Anzahl von Aktivierungen von $\mathcal{Z}$ *nicht* fallengelassen. Damit soll erfaßt werden, daß „uneingeschränkte Sicherheit“ nur einen Grad der Unterscheidbarkeit von Protokollen angibt, aber keine Aussage über „übermäßig lange“ Benutzungen macht.

5.1.2 Das ideale Modell

Das ideale Modell kann nun als Modifikation des hybriden Modells formuliert werden. Genauer liegt es nahe, das $\mathcal{F}$-ideale Modell für eine einzelne ideale Funktio-

nalität $\mathcal{F}$ als das $\{\mathcal{F}\}$-hybride Modell bezüglich einer Menge $D(\mathcal{F}) = \{\tilde{P}_1, \ldots, \tilde{P}_n\}$ spezieller Dummy-Parteien zu formulieren. (Die Anzahl n der Parteien wird dabei implizit durch $\mathcal{F}$ selbst festgelegt.)

Jede Dummy-Partei sendet wieder jede Eingabe als Nachricht an $\mathcal{F}$ und kopiert Antworten von $\mathcal{F}$ auf ihr Ausgabeband. Eine polynomiale Beschränktheit der Dummy-Partei *pro Aktivierung* kann dabei von der jeweiligen polynomialen Beschränkung von $\mathcal{F}$ abgeleitet werden.

Als einziger struktureller Unterschied zum „gewöhnlichen" $\{\mathcal{F}\}$-hybriden Modell wird schließlich für das $\mathcal{F}$-ideale Modell noch ein zusätzlicher Parteiberechnungsschritt direkt *nach* den Funktionalitätenberechnungen eingeführt; hierdurch ist es möglich, Funktionalitäten zu modellieren, welche *unmittelbar* Ausgabe geben. Man denke etwa an eine Funktionalität, welche letztendlich durch Algorithmenausführungen implementiert werden soll.

Die Ausgabe einer Umgebung $\mathcal{Z}$, welche in einem solchen $\mathcal{F}$-idealen Modell bei Sicherheitsparameter $k \in \mathbb{N}$ und mit einem Angreifer $\mathcal{S}$ ausgeführt wird, sei mit $\mathcal{Z}(\mathcal{F}, \mathcal{S}, k)$ bezeichnet. Hiermit läßt sich formulieren, was es für ein hybrides Protokoll heißt, eine ideale Funktionalität zu realisieren:

Definition 5.2. *Sei π ein n-Parteien-Protokoll im $\{\mathcal{F}_i\}$-hybriden Modell, und sei $\mathcal{F}$ eine ideale Funktionalität für n Parteien.* π realisiert $\mathcal{F}$ *(geschrieben $\pi \geq_{\mathsf{s}} \mathcal{F}$) genau dann, wenn für jeden Angreifer $\mathcal{A}$ ein Angreifer $\mathcal{S}$ existiert, so daß die Funktion*

$$\mathbf{Adv}^{\pi,\mathcal{F}}_{\mathcal{A},\mathcal{S},\mathcal{Z}}(k) := |\mathsf{Pr}[\mathcal{Z}(\{\mathcal{F}_i\}, \pi, \mathcal{A}, k) = 1] - \mathsf{Pr}[\mathcal{Z}(\mathcal{F}, \mathcal{S}, k) = 1]|$$

vernachlässigbar in k ist. Gilt dies sogar für in jeder Aktivierung unbeschränkte Umgebungen $\mathcal{Z}$, welche aber immer noch nach höchstens polynomial vielen Aktivierungen halten, so sagt man, daß π uneingeschränkt $\mathcal{F}$ *realisiert (geschrieben $\pi \gg_{\mathsf{s}} \mathcal{F}$).*

Es ist klar, daß die Relationen „$\geq_{\mathsf{s}}$" und „$\gg_{\mathsf{s}}$" transitiv auf n-Parteien-Protokollen sind. Weiter folgt aus $\pi \geq_{\mathsf{s}} \tau$ und $\tau \geq_{\mathsf{s}} \mathcal{F}$ schon $\pi \geq_{\mathsf{s}} \mathcal{F}$; die analoge Aussage gilt für die Relation „$\gg_{\mathsf{s}}$".

5.1.3 Eine Familie von Dummy-Angreifern

Für alles Weitere soll nun ein wichtiges technisches Hilfsmittel zur Verfügung gestellt werden. Genauer soll – wie schon im Modell von [27] – ein *Dummy-Angreifer* formuliert werden, welcher es erlaubt, beliebige Angriffe effektiv in die Umgebung $\mathcal{Z}$ auszulagern. Aus den schon in Abschnitt 3.2.2 besprochenen Gründen kann allerdings kein einzelner Angreifer in diesem Sinne vollständig sein; es muß also eine *Familie* von in ihrer Gesamtheit vollständigen Dummy-Angreifern angegeben werden.

Es heiße also eine Menge $\mathfrak{A}$ von Angreifern *vollständig*, falls geänderte Definitionen 5.1 und 5.2, in denen jeweils nur über Angreifer $\mathcal{A} \in \mathfrak{A}$ quantifiziert wird

(jedoch immer noch beliebige Simulatoren $\mathcal{S}$ zugelassen werden), schon zu den ursprünglichen Definitionen 5.1 bzw. 5.2 äquivalent sind.

Man betrachte nun die spezielle Angreifermenge $\mathfrak{A} := \{\mathcal{A}_p \mid p \in \mathbb{Z}[k], \forall k \in \mathbb{Z} : p(k) > 0\}$. Hierbei bezeichnet $\mathcal{A}_p$ einen Dummy-Angreifer, welcher in jeder Aktivierung seine aktuelle Sicht[5] an $\mathcal{Z}$ übermittelt und zusätzlich spezielle Befehle von $\mathcal{Z}$ befolgt. Genauer veranlaßt ein „`corrupt` P_i"-Befehl $\mathcal{A}_p$ dazu, die Partei P_i zu korrumpieren und deren Zustandsgeschichte (nebst aller Bänder) an $\mathcal{Z}$ zu übermitteln. „`write` m" bringt $\mathcal{A}_p$ dazu, die Nachricht m auf sein Ausgangs-Kommunikationsband zu schreiben; „`write` m `as` P_i" ist das analoge Kommando, um m im Namen einer korrumpierten Partei P_i schreiben zu lassen. In beiden Fällen unterläßt $\mathcal{A}_p$ es, eine Sicht an $\mathcal{Z}$ zu übermitteln, sofern Nachrichten an eine ideale Funktionalität geschrieben wurden, welche dadurch anschließend ausgeliefert werden. Schließlich veranlaßt „`next-round`" $\mathcal{A}_p$ dazu, direkt und ohne eine Nachricht geschrieben zu haben zu terminieren.

Eine polynomiale Beschränkung wird durch $\mathcal{A}_p$-interne Zähler herbeigeführt, welche sicherstellen, daß $\mathcal{A}_p$ nie länger als $p(k)$ Turing-Schritte in einer Aktivierung ausführt, und daß $\mathcal{A}_p$ nach $p(k)$ Aktivierungen anhält. Man beachte, daß kein $\mathcal{A}_p$ je blockiert.

Lemma 5.3. *Die beschriebene Menge $\mathfrak{A}$ von Dummy-Angreifern ist vollständig.*

Es sei darauf hingewiesen, daß die Beweise zu Lemma 5.3, Satz 5.4 und Satz 5.5 grundsätzlich den Beweisideen aus [116, 27, 41] folgen.

Beweis von Lemma 5.3. Seien n-Parteien-Protokolle π und τ im $\{\mathcal{F}_i\}$- bzw. im $\{\mathcal{G}_j\}$-hybriden Modell gegeben, so daß $\pi \geq_{\mathtt{s}} \tau$ in bezug auf die Angreifermenge $\mathfrak{A}$ gilt. Es muß gezeigt werden, daß hieraus schon $\pi \geq_{\mathtt{s}} \tau$ folgt. Sei dazu $\overline{\mathcal{A}}$ ein beliebiger Angreifer und sei $p \in \mathbb{Z}[k]$ ein Polynom, welches die Gesamtlaufzeit von $\overline{\mathcal{A}}$ nach oben beschränkt. Wegen $\pi \geq_{\mathtt{s}} \tau$ in bezug auf $\mathfrak{A}$ existiert ein Simulator $\mathcal{S}_p$, welcher Angriffe von $\mathcal{A}_p$ simuliert. Aus $\mathcal{S}_p$ wird nun ein Simulator $\overline{\mathcal{S}}$ für $\overline{\mathcal{A}}$ konstruiert.

Intern simuliert $\overline{\mathcal{S}}$ sowohl $\overline{\mathcal{A}}$ als auch $\mathcal{S}_p$. $\overline{\mathcal{S}}$ blockiert, wenn $\overline{\mathcal{A}}$ blockiert oder hält, und $\overline{\mathcal{S}}$ hält, sobald $\overline{\mathcal{A}}$ und $\mathcal{S}_p$ gehalten haben. In jeder Aktivierung aktiviert $\overline{\mathcal{S}}$ zunächst $\mathcal{S}_p$, wobei alle Nachrichten von den $\mathcal{G}_j$ an $\mathcal{S}_p$ weitergeleitet werden; zusätzlich wird – soweit nötig – eine leere Nachricht von $\mathcal{Z}$ für $\mathcal{S}_p$ simuliert.

Wenn $\mathcal{S}_p$ dann eine Sicht zurückliefert, wird $\overline{\mathcal{A}}$ mit dieser Sicht aktiviert; alle Kommunikation von $\overline{\mathcal{S}}$ mit $\mathcal{Z}$ wird zu $\overline{\mathcal{A}}$ durchgestellt. Korruptionen durch $\overline{\mathcal{A}}$ und Kommunikation von $\overline{\mathcal{A}}$ mit den Parteien werden in passende Kommandos an $\mathcal{S}_p$ umgewandelt.

Seinerseits führt $\overline{\mathcal{S}}$ alle Korruptionen von $\mathcal{S}_p$ aus und stellt zu $\mathcal{S}_p$ jede Kommunikation mit den $\mathcal{G}_j$ und die Sicht von $\overline{\mathcal{S}}$ auf die Parteien durch.

Sei nun $\overline{\mathcal{Z}}$ eine Umgebung. Sei $\mathcal{Z}$ die Umgebung, welche intern $\overline{\mathcal{Z}}$ und $\overline{\mathcal{A}}$ simuliert. Interaktionen von $\overline{\mathcal{Z}}$ mit den Parteien werden dabei von $\mathcal{Z}$ weitergeleitet.

[5] Damit ist gemeint: Der Inhalt der Ausgangs-Kommunikationsbänder aller Parteien, und der Inhalt der Eingangs-Kommunikationsbänder der korrumpierten Parteien.

Sichten für $\overline{\mathcal{A}}$ werden von $\mathcal{Z}$ aus Angreifer-Sichten des mit $\mathcal{Z}$ interagierenden Angreifers $\mathcal{A}$ extrahiert. Weiter werden Korruptionen und Kommunikation von $\overline{\mathcal{A}}$ in Form entsprechender Dummy-Angreifer-Kommandos an $\mathcal{A}$ weitergeleitet. Wenn $\overline{\mathcal{A}}$ eine Aktivierung ohne Schreiben einer Nachricht an die Umgebung oder eine Funktionalität beendet (und somit die Berechnungsphase beginnen soll), formuliert $\mathcal{Z}$ ein entsprechendes Kommando an $\mathcal{A}$. Schließlich hält $\mathcal{Z}$ (mit der Ausgabe von $\overline{\mathcal{Z}}$), wenn $\overline{\mathcal{Z}}$ hält.

Nach Annahme über p ist damit für alle $k \in \mathbb{N}$

$$\mathcal{Z}(\{\mathcal{F}_i\}, \pi, \mathcal{A}_p, k) = \overline{\mathcal{Z}}(\{\mathcal{F}_i\}, \pi, \overline{\mathcal{A}}, k)$$

und

$$\mathcal{Z}(\{\mathcal{G}_j\}, \tau, \mathcal{S}_p, k) = \overline{\mathcal{Z}}(\{\mathcal{G}_j\}, \tau, \overline{\mathcal{S}}, k).$$

Das zeigt $\pi \geq_{\mathsf{s}} \tau$, da $\overline{\mathcal{Z}}$ und $\overline{\mathcal{A}}$ beliebig waren. Der Beweis ist auch für die Fälle „$\gg_{\mathsf{s}}$“ und „$\pi \geq_{\mathsf{s}} \mathcal{F}$“ (bzw. „$\pi \gg_{\mathsf{s}} \mathcal{F}$“) gültig. □

5.1.4 Komposition

Für einen simulierbarkeitsbasierten Sicherheitsbegriff wie den gerade gegebenen ist natürlich wichtig, wie sich Protokollsicherheit unter Komposition von Protokollen verhält. Die folgenden Betrachtungen und Resultate zeigen, daß der gegebene Begriff in diesem Sinne „gutartig“ ist.

Es werden im Stil von Unruh [136] zwei Kompositionstheoreme angegeben. Das erste der beiden sichert zu, daß eine einzelne Funktionalität in einem größeren Protokoll ersetzt werden kann durch ein Protokoll, welches diese Funktionalität realisiert. Das zweite Kompositionstheorem sichert zu, daß die Parallelisierung eines Protokolls und einer Funktionalität keinen Sicherheitsverlust mit sich bringt.

Wie für das Modell von [27] muß zunächst geklärt werden, was es genau heißt, eine Funktionalität in einem größeren Protokoll zu ersetzen. Sei dazu π ein n-Parteien-Protokoll, welches im $\{\mathcal{F}_i\}$-hybriden Modell formuliert ist. Sei τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\}$-hybriden Modell, so daß $\pi \geq_{\mathsf{s}} \mathcal{F}$ oder $\pi \gg_{\mathsf{s}} \mathcal{F}$ für ein $\mathcal{F} \in \{\mathcal{F}_i\}$ gelte und $\{\mathcal{F}_i\} \cap \{\mathcal{G}_j\} = \emptyset$ ist.[6]

Dann bezeichnet π^τ die folgende kanonische Modifikation von π. Sei P_i^π bzw. P_i^τ die jeweilige Partei aus π bzw. τ. Dann führt die Partei P_i aus π^τ zunächst das Programm von P_i^π und dann das Programm von P_i^τ aus. Von P_i^π bzw. P_i^τ an Parteien versandte Nachrichten werden dabei durch ein vorangestelltes π bzw. τ identifiziert. Zusätzlich wird jede Nachricht von P_i^π an $\mathcal{F}$ als Eingabe an P_i^τ weitergeleitet. Analog werden Ausgaben von P_i^τ in Nachrichten von $\mathcal{F}$ an P_i^π umgewandelt. Damit ist π^τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\} \cup (\{\mathcal{F}_i\} \setminus \{\mathcal{F}\})$-hybriden Modell, in welchem intuitiv alle $\mathcal{F}$-Anfragen an eine Instanz des Unterprotokolls τ weitergeleitet werden.

[6] Der Schnitt ist hier bezüglich der Identitäten der ITMs $\mathcal{F}_i$ bzw. $\mathcal{G}_j$ zu verstehen; durch Umbenennen der Funktionalitäten kann die Bedingung also immer erzwungen werden.

Satz 5.4. *Sei π ein n-Parteien-Protokoll im $\{\mathcal{F}_i\}$-hybriden Modell. Sei τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\}$-hybriden Modell mit $\{\mathcal{F}_i\} \cap \{\mathcal{G}_j\} = \emptyset$ und $\pi \geq_{\mathsf{s}} \mathcal{F}$ (bzw. $\pi \gg_{\mathsf{s}} \mathcal{F}$) für ein $\mathcal{F} \in \{\mathcal{F}_i\}$. Dann gilt $\pi^\tau \geq_{\mathsf{s}} \pi$ (bzw. $\pi^\tau \gg_{\mathsf{s}} \pi$) für das oben beschriebene komponierte Protokoll π^τ.*

Beweis. Es reicht nach Lemma 5.3, zu jedem Dummy-Angreifer $\mathcal{A}_p$ einen Simulator $\mathcal{S} = \mathcal{S}_p$ anzugeben, so daß für jede Umgebung $\mathcal{Z}$ die Funktion $\mathbf{Adv}^{\pi,\tau}_{\mathcal{A}_p,\mathcal{S},\mathcal{Z}}(k)$ vernachlässigbar in k ist. Die folgende Beschreibung von $\mathcal{S}$ nimmt stillschweigend an, daß $\mathcal{S}$ die Komplexitätsbeschränkungen von $\mathcal{A}_p$ respektiert. (D. h. $\mathcal{S}$ hält, wenn $\mathcal{A}_p$ gehalten hätte und kürzt seine an $\mathcal{Z}$ gelieferten Sichten p entsprechend.)

Sei also $\tau \geq_{\mathsf{s}} \mathcal{F}$. Dann existiert ein Simulator $\overline{\mathcal{S}}$, so daß $\mathbf{Adv}^{\pi,\tau}_{\overline{\mathcal{A}}_p,\overline{\mathcal{S}},\overline{\mathcal{Z}}}(k)$ vernachlässigbar in k ist. (Hier bezeichnet $\overline{\mathcal{A}}_p$ genau den Dummy-Angreifer $\mathcal{A}_p$; die Notation soll nur zur Identifikation des jeweils angegriffenen Protokolls dienen.)

Der Simulator $\mathcal{S}$ simuliert intern $\overline{\mathcal{S}}$ und agiert wie $\mathcal{A}_p$, jedoch mit den folgenden Ausnahmen. Zunächst blockiert $\mathcal{S}$, sobald $\mathcal{A}_p$ gehalten hätte. Weiter wird in jeder $\mathcal{S}$-Aktivierung auch $\overline{\mathcal{S}}$ genau einmal aktiviert, gegebenenfalls mit (gefälschten) $\mathcal{Z}$-Kommandos, siehe unten. Kommunikation von $\overline{\mathcal{S}}$ mit $\mathcal{F}$ wird durch $\mathcal{S}$ an $\mathcal{F}$ durchgestellt. Die in der von $\overline{\mathcal{S}}$ gelieferten Sicht enthaltenen Nachrichten (von Parteien bzw. den $\mathcal{G}_i$) werden in die später an $\mathcal{Z}$ zurückgelieferte Sicht von $\mathcal{S}$ integriert.

Anweisungen, eine Partei P_i zu korrumpieren, werden von $\mathcal{S}$ an $\overline{\mathcal{S}}$ weitergeleitet. Will $\overline{\mathcal{S}}$ dann P_i korrumpieren, so stellt $\mathcal{S}$ für $\overline{\mathcal{S}}$ einen durch Korruption von P_i erhaltenen Zustand zusammen, in dem nur τ-Eingaben berücksichtigt werden. Der daraufhin von $\overline{\mathcal{S}}$ zurückgelieferte P_i-Zustand wird dann in den echten P_i-Zustand integriert. Anweisungen an $\mathcal{S}$, Nachrichten mit vorangestelltem τ auszuliefern, werden an $\overline{\mathcal{S}}$ weitergeleitet; ebenso jede „`next-round`“-Anweisung. Auf diese Weise simuliert $\mathcal{S}$ nur den tatsächlich durch $\mathcal{F}$ idealisierten Teil des Protokolls π.

Sei nun $\mathcal{Z}$ eine Umgebung, welche versucht, π^τ und $\mathcal{A}_p$ von π und dem konstruierten $\mathcal{S}$ zu trennen. Ein solches $\mathcal{Z}$ gibt Anlaß zu einer Umgebung $\overline{\mathcal{Z}}$, welche τ und $\overline{\mathcal{A}}_p$ von $\mathcal{F}$ und $\overline{\mathcal{S}}$ mit demselben Erfolg unterscheidet. Hierfür simuliert $\overline{\mathcal{Z}}$ intern $\mathcal{Z}$ und den Angreifer $\mathcal{A}_p$, die Funktionalitäten $\{\mathcal{F}_i\} \setminus \{\mathcal{F}\}$ und die Parteien P_i^π aus Protokoll π. Die nicht-simulierten Maschinen, mit denen $\overline{\mathcal{Z}}$ dabei interagiert, seien hier mit $\overline{\mathcal{A}}$ und $\overline{P}_i$ bezeichnet.

Das Scheduling innerhalb der Simulation von $\overline{\mathcal{Z}}$ wird dabei mit dem tatsächlichen Scheduling synchronisiert, wobei alle Nachrichten, die in der Simulation an $\mathcal{F}$ gesendet werden, durch entsprechende Eingaben an die $\overline{P}_i$ bzw. Anweisungen an $\overline{\mathcal{A}}$ ersetzt werden. Analog werden von $\overline{\mathcal{A}}$ gemeldete Nachrichten in die Simulation integriert. Schließlich hält $\overline{\mathcal{Z}}$ (mit der Ausgabe von $\mathcal{Z}$), wenn $\mathcal{Z}$ hält. Damit ist

$$\mathcal{Z}(\{\mathcal{G}_j\} \cup \{\mathcal{F}_i\} \setminus \{\mathcal{F}\}, \pi^\tau, \mathcal{A}_p, k) = \overline{\mathcal{Z}}(\{\mathcal{G}_j\}, \tau, \overline{\mathcal{A}}_p, k)$$

und

$$\mathcal{Z}(\{\mathcal{F}_j\}, \pi, \mathcal{S}, k) = \overline{\mathcal{Z}}(\mathcal{F}, \overline{\mathcal{S}}, k)$$

für alle $k \in \mathbb{N}$, was $\pi^\tau \geq_{\mathsf{s}} \pi$ zeigt. Dieselben Reduktionen zeigen auch $\pi^\tau \gg_{\mathsf{s}} \pi$, wenn von $\tau \gg_{\mathsf{s}} \mathcal{F}$ ausgegangen wird. □

Nun ist zu klären, wie eine Parallelisierung eines Protokolls bzw. einer idealen Funktionalität auszusehen hat. Angesichts der polynomialen Beschränkung eines Protokolls reicht es hier, eine Parallelisierung in polynomial viele Instanzen anzugeben. Sei also $\mathcal{F}$ eine ideale Funktionalität und $p \in \mathbb{Z}[k]$ ein Polynom mit $p(k) > 0$ für alle k. Dann sei $\mathcal{F}||p$ die *p-Parallelisierung* von $\mathcal{F}$, welche intern $p(k)$ Simulationen $\mathcal{F}^1, \ldots, \mathcal{F}^{p(k)}$ von $\mathcal{F}$ verwaltet.

Jede „`computation`"-Eingabe wird an alle diese Simulationen weitergeleitet, genauso wie Korruptionsbenachrichtigungen. (Diese Korruptionsbenachrichtigungen ziehen jedoch – wie bei $\mathcal{F}||p$ selbst – keine automatische Aktivierung eines $\mathcal{F}^j$ nach sich.) Jede andere Nachricht wird nur an ein $\mathcal{F}^j$ weitergeleitet, wenn sie von der Form (j, m) ist; der Präfix j wird bei der Weiterleitung gestrichen. Analog werden alle Ausgaben m einer Funktionalität $\mathcal{F}^j$ als (j, m) ausgegeben. Eine $\mathcal{F}$-Instanz wird bei einer Aktivierung von $\mathcal{F}||p$ nur aktiviert, wenn sie eine Nachricht oder eine Eingabe bekommen hat.

Analog kann die Parallelisierung $\pi||p$ eines Protokolls π definiert werden. Hier simuliert jede Partei P_i intern $p(k)$ Instanzen $P_i^1, \ldots, P_i^{p(k)}$ der Partei P_i^π aus π. Wie bei $\mathcal{F}||p$ werden alle Ausgaben und gesendeten Nachrichten mit der jeweiligen Instanznummer versehen; Eingaben und eingehende Nachrichten werden an die durch eine solche Instanznummer j benannte Simulation P_i^j weitergeleitet. Syntaktisch ist also $\pi||p$ im $\{\mathcal{F}_i||p\}$-hybriden Modell formuliert, falls π im $\{\mathcal{F}\}$-hybriden Modell formuliert ist.

Wie schon angedeutet, sind ähnliche Parallelisierungstechniken schon in [41] und [136] betrachtet worden. Es konnte dabei in beiden Fällen ein dem nächsten Satz ähnliches Resultat abgeleitet werden:

Satz 5.5. *Sei $p \in \mathbb{Z}[k]$ mit $p(k) > 0$ für alle $k \in \mathbb{Z}$. Sei weiter π ein n-Parteien-Protokoll im $\{\mathcal{F}_i\}$-hybriden Modell und τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\}$-hybriden Modell. Dann impliziert $\pi \geq_{\mathsf{s}} \tau$ schon $\pi||p \geq_{\mathsf{s}} \tau||p$. Weiter impliziert $\pi \geq_{\mathsf{s}} \mathcal{F}$ schon $\pi||p \geq_{\mathsf{s}} \mathcal{F}||p$. Diese Aussagen gelten auch für uneingeschränkte Sicherheit.*

Beweis. Sei $p \in \mathbb{Z}[k]$ ($p(k) > 0$) und $\pi \geq_{\mathsf{s}} \tau$. Dann gibt es für jeden Dummy-Angreifer $\overline{\mathcal{A}}_q$ einen Simulator $\overline{\mathcal{S}}_q$, so daß für alle $\overline{\mathcal{Z}}$ die Funktion $\mathbf{Adv}^{\pi,\tau}_{\overline{\mathcal{A}}_q,\overline{\mathcal{S}}_q,\overline{\mathcal{Z}}}(k)$ vernachlässigbar in k ist. Sei $\mathcal{A}_q$ ein beliebiger Dummy-Angreifer auf $\pi||p$. Es soll nun ein Simulator $\mathcal{S}$ konstruiert werden, welcher mit $\tau||p$ Angriffe von $\mathcal{A}_q$ simuliert. Wie im Beweis von Satz 5.4 wird angenommen, daß $\mathcal{S}$ die Komplexitätsbeschränkungen von $\mathcal{A}_q$ respektiert. Weiter wird $\mathcal{S}$ wieder blockieren, sobald $\mathcal{A}_q$ angehalten hätte.

Intern verwaltet $\mathcal{S}$ genau $p(k)$ Simulationen $\overline{\mathcal{S}}^j$ $(1 \leq j \leq p(k))$ von $\overline{\mathcal{S}}_q$ und hält, sobald alle diese Simulationen angehalten haben. Die Sicht jedes dieser $\overline{\mathcal{S}}^j$ ist die gefilterte Sicht von $\mathcal{S}$ selbst, so daß nur Nachrichten mit vorangestelltem j betrachtet werden (der Präfix j wird natürlich für die Sicht von $\overline{\mathcal{S}}^j$ entfernt). Es ist dann klar, wie Nachrichtenauslieferungen der $\overline{\mathcal{S}}^j$ gehandhabt werden.

Bei einer Anweisung von $\mathcal{Z}$, eine Partei P_i zu korrumpieren, korrumpiert $\mathcal{S}$ zunächst P_i und weist dann alle $\overline{\mathcal{S}}^j$-Instanzen an, P_i zu korrumpieren. Will dann etwa

$\overline{\mathcal{S}}^j$ tatsächlich P_i korrumpieren, so wird an $\overline{\mathcal{S}}^j$ ein entsprechend auf die P_i-interne P_i^j-Simulation eingeschränkter Zustand von P_i zurückgeliefert.

Schließlich werden „`next-round`"-Anweisungen an alle $\overline{\mathcal{S}}^j$ weitergeleitet. Weiter wird, sofern keine Nachricht an eine ideale Funktionalität zu schreiben oder ein „`next-round`"-Kommando von $\mathcal{Z}$ auszuführen ist, für $\mathcal{Z}$ eine Sicht von $\mathcal{S}$ aus allen Sichten, die die $\overline{\mathcal{S}}^j$ auf Anfrage liefern, generiert.

Sei nun eine Umgebung $\mathcal{Z}$ gegeben, für welche $\mathbf{Adv}^{\pi||p,\tau||p}_{\mathcal{A}_q,\mathcal{S},\mathcal{Z}}(k)$ nicht-vernachlässigbar in k ist. Um den ersten Teil des Satzes zu zeigen, muß hieraus ein Widerspruch hergeleitet werden. Sei also für $\ell \in \mathbb{N}$ das folgende, im $\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}$-hybriden Modell formulierte Protokoll $H_\ell = \{P_1^\ell, \ldots, P_n^\ell\}$ definiert: Die Partei P_i^ℓ unterhält $\min\{\ell, p(k)\}$ interne Simulationen der Partei P_i^π aus Protokoll π und $\max\{p(k) - \ell, 0\}$ interne Simulationen der Partei P_i^τ aus Protokoll τ. Nachrichtenbehandlung und internes Scheduling passiert wie im Falle der p-Parallelisierung eines Protokolls – genauer werden die P_i^π-Simulationen mit den Präfixen $1, \ldots, \min\{\ell, p(k)\}$ identifiziert.

Sei weiter $\mathcal{H}_\ell$ der folgende Angreifer auf H_ℓ: Der Aufbau von $\mathcal{H}_\ell$ gleicht dem von $\mathcal{S}$, jedoch simuliert $\mathcal{H}_\ell$ nur $\max\{p(k) - \ell, 0\}$ Instanzen des Simulators $\overline{\mathcal{S}}_q$ und dafür $\min\{\ell, p(k)\}$ Instanzen des Dummy-Angreifers $\overline{\mathcal{A}}_q$. Dabei haben die $\overline{\mathcal{A}}_q$-Simulationen Zugriff auf die jeweiligen Protokollstränge (d. h. Nachrichtenpräfixe) von π in H_ℓ Zugriff. Damit ist für alle $k \in \mathbb{Z}$

$$\mathcal{Z}(\{\mathcal{F}_i||p\}, \pi||p, \mathcal{A}_q, k) = \mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_{p(k)}, \mathcal{H}_{p(k)}, k)$$

und

$$\mathcal{Z}(\{\mathcal{G}_j||p\}, \tau||p, \mathcal{S}, k) = \mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_0, \mathcal{H}_0, k).$$

Damit muß eine Familie $\{m_k\}_{k\in\mathbb{N}}$ mit $m_k \in \{1, \ldots, p(k)\}$ existieren, so daß die Funktion

$$\begin{aligned} &\mathsf{Pr}[\mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_{m_k-1}, \mathcal{H}_{m_k-1}, k) = 1] \\ &-\mathsf{Pr}[\mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_{m_k}, \mathcal{H}_{m_k}, k) = 1] \end{aligned} \tag{5.1}$$

nicht-vernachlässigbar in k ist.

Sei $\overline{\mathcal{Z}}$ die Umgebung, welche zunächst gleichverteilt ein $\ell \in \{1, \ldots, p(k)\}$ wählt und dann einen kompletten Protokolldurchlauf von H_ℓ mit $\mathcal{Z}$, Funktionalitäten $\{\mathcal{F}_i\} \cup \{\mathcal{G}_j\}$ und einem Angreifer $\mathcal{H}_\ell$ simuliert. Wieder wird das Scheduling in der Simulation mit dem tatsächlichen Scheduling synchronisiert; allerdings werden Eingaben und Nachrichten, welche einen Präfix ℓ haben (also im ℓ-ten Protokollstrang innerhalb von H_ℓ vorkommen), als Eingaben an die nicht-simulierten, mit $\overline{\mathcal{Z}}$ interagierenden Parteien $\overline{P}_i$ weitergeleitet bzw. in entsprechende Anweisungen an den mit $\overline{\mathcal{Z}}$ laufenden Angreifer $\overline{\mathcal{A}}$ umgewandelt. Analog werden nicht-simulierte Ausgaben der $\overline{P}_i$ und vom Angreifer $\overline{\mathcal{A}}$ gemeldete Nachrichten nach Vorstellen eines Präfix ℓ in die Simulation integriert. $\overline{\mathcal{Z}}$ hält schließlich (mit der Ausgabe von $\mathcal{Z}$), wenn das simulierte $\mathcal{Z}$ hält.

Damit ist nach Konstruktion für alle $k \in \mathbb{Z}$

$$\overline{\mathcal{Z}}(\{\mathcal{G}_j\}, \tau, \mathcal{S}_q, k) = \mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_{\ell-1}, \mathcal{H}_{\ell-1}, k)$$

und

$$\overline{\mathcal{Z}}(\{\mathcal{F}_i\}, \pi, \mathcal{A}_q, k) = \mathcal{Z}(\{\mathcal{F}_i||p\} \cup \{\mathcal{G}_j||p\}, H_{\ell}, \mathcal{H}_{\ell}, k).$$

Da $\ell = m_k$ mit Wahrscheinlichkeit $1/p(k)$ ist, führt die Nicht-Vernachlässigbarkeit der Funktion aus (5.1) auf einen Widerspruch zu $\pi \geq_{\mathsf{s}} \tau$. Diese Reduktion ist auch im Falle uneingeschränkter Sicherheit anwendbar.

Das zeigt die erste Behauptung. Die zweite Behauptung folgt analog, sofern nur das im idealen Modell leicht geänderte Scheduling (d. h. die zweifache Aktivierung der Dummy-Parteien in einer Runde) in der Konstruktion des Protokolls H_ℓ berücksichtigt wird. □

Als Kombination der beiden Kompositionstheoreme ergibt sich folgende Anwendung, welche das Analogon zum „universellen“ Kompositionstheorem aus [27] darstellt:

Korollar 5.6. *Sei $p \in \mathbb{Z}[k]$ mit $p(k) > 0$ für alle $k \in \mathbb{Z}$. Sei weiter π ein n-Parteien-Protokoll im $\{\mathcal{F}_i\}$-hybriden Modell, und sei τ ein n-Parteien-Protokoll im $\{\mathcal{G}_j\}$-hybriden Modell mit $\{\mathcal{F}_i\} \cap \{\mathcal{G}_j\} = \emptyset$ und $\pi \geq_{\mathsf{s}} \mathcal{F}$ (bzw. $\pi \gtrapprox_{\mathsf{s}} \mathcal{F}$) für ein $\mathcal{F}||p \in \{\mathcal{F}_i\}$. Dann gilt $\pi^{\tau||p} \geq_{\mathsf{s}} \pi$ (bzw. $\pi^{\tau||p} \gtrapprox_{\mathsf{s}} \pi$).*

Intuitiv besagt Korollar 5.6, daß in einem größeren Protokoll die *Parallelisierung* einer idealen Funktionalität durch ein parallelisiertes Protokoll ersetzt werden darf, auch wenn nur bekannt ist, daß eine *einfache* Version des Protokolls eine einfache Version der idealen Funktionalität realisiert.

5.1.5 Schalenkonstruktionen

Das hier vorgestellte Modell erlaubt es einem Angreifer nicht, Nachrichten zu verzögern; das gilt selbst für Nachrichten, die von oder an eine ideale Funktionalität gesandt wurden. Damit können Funktionalitäten zwar in sehr detaillierter Art und Weise formuliert werden, jedoch mag in vielen praktischen Fällen eine Abstraktion von gewissen Auslieferungsdetails wünschenswert sein. Genauer kann gewünscht sein, dem Angreifer – bis zu einem gewissen Grad – die Auslieferung der Nachricht zu überlassen.

Hierzu soll eine Idee aus Backes [6] aufgegriffen werden: Parteien eines Protokolls und Funktionalitäten werden dazu mit „Schalen“ ausgerüstet, welche ein Rahmenprogramm für die Partei bzw. Funktionalität darstellen. (Eine solche „Schale“ wurde in [6] benutzt, um in einem synchronen Modell formulierte Maschinen in einem asynchronen Modell darzustellen.)

Genauer soll für eine ITM M – wobei man hier an eine Partei oder ideale Funktionalität denken sollte – die $(p_{\mathsf{recv}}, p_{\mathsf{send}}, \mathsf{clk})$-*Schale* $[M](p_{\mathsf{recv}}, p_{\mathsf{send}}, \mathsf{clk})$ (oft auch

$[M]$ genannt, wenn der Kontext klar ist) von M für $p_{\mathsf{recv}}, p_{\mathsf{send}} \in \mathbb{Z}[k] \cup \{\infty\}$ und $\mathsf{clk} \in \{\mathsf{async}, \mathsf{sync}\}$ definiert werden.

Intern simuliert $[M]$ die ursprüngliche Maschine M und leitet Ein- und Ausgaben sowie jede Kommunikation mit $\mathcal{A}$ direkt an bzw. von M weiter. Andererseits wird über jede eingehende Nachricht m mit Absender $S \neq \mathcal{A}$ zunächst $\mathcal{A}$ durch eine Nachricht „`receive-request`(j, S)“ informiert, wobei j einfach eine von $[M]$ vergebene laufende Nummer darstellt. Erhält $[M]$ dann von $\mathcal{A}$ eine Nachricht „`receive-permission`(j)“, so wird nicht diese Nachricht, sondern die ursprüngliche Nachricht m von S zu M durchgestellt.

Auf diese Weise wird aber eine Nachricht nie länger als $p_{\mathsf{recv}}(k)$ Runden (also Aktivierungen bzw. „`computation`“-Eingaben für Parteien bzw. ideale Funktionalitäten M) zurückgehalten. Gegebenenfalls findet eine automatische Auslieferung von m statt.

Analog wird $\mathcal{A}$ über die Existenz und den Empfänger ausgehender Nachrichten von M mittels einer „`send-request`(j, R)“-Nachricht informiert. Ein tatsächliches Senden der Nachricht von $[M]$ findet dann nach einer „`send-permission`(j)“-Nachricht von $\mathcal{A}$ oder automatisch nach $p_{\mathsf{send}}(k)$ Runden statt.

Ist $\mathsf{clk} = \mathsf{sync}$, so wird die interne M-Simulation von $[M]$ bei *jeder* Aktivierung von $[M]$ aktiviert; eine etwaige „`computation`“-Eingabe wird dabei weitergeleitet. Für $\mathsf{clk} = \mathsf{async}$ hingegen wird M nur bei eingehenden Nachrichten und Eingaben $in \neq$ `computation` aktiviert. („`computation`“-Eingaben werden also durch $[M]$ unterdrückt.) Dabei wird M zunächst für jede eingegangene Eingabe einmal aktiviert und anschließend für jede eingegangene Nachricht; die Nachrichten werden dabei in der Reihenfolge der Senderidentitäten abgearbeitet.

Schließlich hält $[M]$ an, sobald M angehalten hat und alle von M gesendeten Nachrichten von $[M]$ abgeschickt wurden. Eine polynomiale Beschränkung von $[M]$ *pro Aktivierung* kann durch geeignete Kürzung der betrachteten Nachrichten herbeigeführt werden. (Für Details hierzu sei auf [82] verwiesen.)

Mit einer solchen Schalenkonstruktion können beispielsweise Netzwerke modelliert werden, in denen zwar eine Nachrichtenauslieferung garantiert wird, aber in denen – aus Sicht einer Protokollmaschine – keine a-priori-Schranken für die tatsächliche Güte der betrachteten Kommunikationsleitungen vorliegen. Zudem können ideale Funktionalitäten betrachtet werden, welche in einem asynchronen und nachrichtengetriebenen Modell wie [27] formuliert wurden – es muß eine solche Funktionalität nur mit einer geeigneten Schale versehen werden.

5.1.6 Verbindungen zu anderen Modellen

Im in den vorigen Kapiteln eingehend besprochenen Modell [27] konnten zahlreiche konkrete Protokolle bzw. Protokollkonstruktionen untersucht werden. Die in dieser Arbeit behandelten Protokollaufgaben stellen dabei nur eine kleine Auswahl dar. Aus diesem Grund wäre es sehr interessant, bereits bestehende Resultate, welche im Modell aus [27] (bzw. Abschnitt 3.1) formuliert wurden, auch im in diesem Kapitel behandelten Modell formulieren zu können. Hierbei könnte Gebrauch von den

gerade vorgestellten Schalenkonstruktionen gemacht werden, um asynchron formulierten Maschinen einen asynchronen Protokollablauf vorzugaukeln. (Mit einem ähnlichen Resultat konnte beispielsweise eine Verbindung zwischen dem synchronen Modell [116] und seiner asynchronen Ausprägung [117] hergestellt werden.)

Insbesondere zwei Details erschweren die Aufgabe für das asynchrone Modell aus Abschnitt 3.1: die dortigen Maschinen werden aufgrund der nachrichtengetriebenen Natur des dortigen Schedulings jeweils nur mit *einer* Nachricht auf einmal aktiviert. Zudem wird, falls eine Maschine keine Nachrichten sendet oder Eingaben gibt, im Modell aus Abschnitt 3.1 die Umgebung $\mathcal{Z}$ aktiviert. Im analogen Fall wird im in diesem Kapitel vorgestellten Modell allerdings der Angreifer $\mathcal{A}$ aktiviert.

Dennoch kann in einem eingeschränkten Sinn das asynchrone Scheduling aus Abschnitt 3.1 synchron formuliert werden. Hierbei werden Sicherheitseigenschaften erhalten, was als Beleg dafür gewertet werden kann, daß Sicherheit im hier vorgestellten synchronen Modell keine zu strikte Forderung ist.

Zunächst muß sich hier auf strikt polynomial beschränkte Umgebungen und Angreifer aus dem asynchronen Modell aus Abschnitt 3.1 beschränkt werden. Dies ist die in [27] vorgeschlagene Komplexitätsklasse dieser Maschinen, jedoch soll nicht verschwiegen werden, daß diese Forderung nicht unproblematisch ist, vgl. Abschnitt 3.2.2.

Weiter soll von einer idealen Funktionalität $\mathcal{F}$ im Sinne des asynchronen Modells aus Abschnitt 3.1 gesagt werden, daß sie *nicht mit dem Angreifer kommuniziert*, wenn sie (a) nie eine Nachricht an den Angreifer sendet und (b) alle Nachrichten vom Angreifer ignoriert (d. h. $\mathcal{F}$ geht sofort in einen unveränderten Wartezustand über, wenn eine Nachricht vom Angreifer eintrifft).

Sei also für ein reales Protokoll $\pi = \{P_1, \ldots, P_n\}$ im Sinne von Abschnitt 3.1 mit $[\pi]_{\mathrm{UC}} = \{P'_1, \ldots, P'_n\}$ das Protokoll mit $P'_i = [P_i](\infty, 0, \mathsf{async})$ bezeichnet. Mit anderen Worten: $[\pi]_{\mathrm{UC}}$ ist eine synchrone Variante von π, in der effektiv nur der Angreifer Nachrichten ausliefern kann.

Ideale Funktionalitäten in der ursprünglichen Formulierung von [27] werden nicht über Korruptionen informiert. Eine Schale im hier vorgestellten synchronen Modell, welche zur Einbettung einer asynchron formulierten idealen Funktionalität $\mathcal{F}$ gedacht ist, darf also solche Korruptionsinformationen nicht an die interne Simulation von $\mathcal{F}$ weiterleiten. Sei also $[\mathcal{F}]_{\mathrm{UC}}$ identisch mit $[\mathcal{F}](0, \infty, \mathsf{async})$, außer daß $[\mathcal{F}]_{\mathrm{UC}}$ keine Korruptionsbenachrichtigungen an $\mathcal{F}$ weitergibt. Intuitiv können so asynchron formulierte Funktionalitäten in ein synchrones Modell eingebettet werden, indem einfach die Auslieferung einer Nachricht an eine Partei dem Angreifer überlassen wird.

Nun soll auch eine Einbettung von Funktionalitäten im Sinne von [39], welche Korruptionsbenachrichtigungen *erwarten*, und welche zudem in einem leicht geänderten Scheduling (vgl. Abschnitt 4.1.4) formuliert sind, ermöglicht werden. Dazu sei $[\mathcal{F}]_{\mathrm{UC}'}$ identisch zu $[\mathcal{F}](\infty, \infty, \mathsf{async})$, bis auf die Tatsache, daß im Falle eingehender Nachrichten von einer Partei zusätzlich zu der Benachrichtigung über die eingegangene Nachricht auch die „Header“ dieser Nachricht an $\mathcal{A}$ gegeben werden.

Es sei hier daran erinnert, daß der Angreifer im geänderten Scheduling von [39]

auch Nachrichten auf dem Weg *zur* idealen Funktionalität verzögern bzw. blockieren darf. Um dennoch ungültige Eingaben zu identifizieren, erhält er im Modell von [39] die „Header“ solcher Nachrichten. Für eine eingehendere Diskussion sei auf Abschnitt 4.1.4 verwiesen.

Hiermit kann das folgende Resultat formuliert werden:

Satz 5.7. *Sei $\mathcal{F}$ eine ideale Funktionalität, welche nicht mit dem Angreifer kommuniziert. Wenn $\pi \geq \mathcal{F}$ bezüglich authentifizierter Kanäle für ein reales Protokoll π im Modell von [27] gilt, dann gilt $[\pi]_{\mathrm{UC}} \geq_{\mathtt{s}} [\mathcal{F}]_{\mathrm{UC}}$ im hier vorgestellten synchronen Modell. Gilt $\pi \geq \mathcal{F}$ bezüglich authentifizierter Kanäle für ein reales Protokoll π im Modell von [39], so ist $[\pi]_{\mathrm{UC}} \geq_{\mathtt{s}} [\mathcal{F}]_{\mathrm{UC}'}$.*

Beweis. Sei $\pi \geq \mathcal{F}$ bezüglich authentifizierter Kanäle im asynchronen Modell aus Abschnitt 3.1 für eine Funktionalität $\mathcal{F}$, welche nicht mit dem Angreifer kommuniziert. Für $p \in \mathbb{Z}[k]$ mit $p(k) > 0$ für alle $k \in \mathbb{Z}$ bezeichne $\tilde{\mathcal{A}}_p$ die p-Restriktion des Dummy-Angreifers aus Abschnitt 3.1.5: $\tilde{\mathcal{A}}_p$ simuliert $\tilde{\mathcal{A}}$, erzwingt dabei aber durch vorzeitigen Abbruch bzw. vorzeitiges Anhalten, daß $\tilde{\mathcal{A}}_p$ pro Aktivierung nie länger als $p(k)$ Turing-Schritte läuft und nach $p(k)$ Aktivierungen anhält.

Nach Annahme gibt es für jedes $q \in \mathbb{Z}[k]$ ($q(k) > 0$) einen Simulator $\tilde{\mathcal{S}}_q$, so daß für jede Umgebung $\tilde{\mathcal{Z}}$ im Sinne von Abschnitt 3.1 die Funktion $\mathbf{Adv}^{\pi,\mathcal{F}}_{\tilde{\mathcal{A}}_q,\tilde{\mathcal{S}}_q,\tilde{\mathcal{Z}}}(k)$ vernachlässigbar in k ist (vgl. Abschnitt 3.1.4). Nach Annahme über $\mathcal{F}$ darf dabei ohne Beschränkung der Allgemeinheit davon ausgegangen werden, daß kein $\tilde{\mathcal{S}}_q$ je eine Nachricht an $\mathcal{F}$ sendet.

Sei nun $p \in \mathbb{Z}[k]$ ($p(k) > 0$) beliebig, und sei $\mathcal{A}_p$ der p-beschränkte Dummy-Angreifer im synchronen Modell (siehe Abschnitt 5.1.3). Es wird nun ein entsprechender Simulator $\mathcal{S}$ im synchronen Modell beschrieben; es sei dabei wieder angenommen, daß $\mathcal{S}$ die p-Beschränkung von $\mathcal{A}_p$ respektiert (vgl. auch den Beweis zu Lemma 5.3). Intern simuliert $\mathcal{S}$ eine Instanz von $\tilde{\mathcal{S}}_q$ für ein hinreichend großes Polynom $q \in \mathbb{Z}[k]$ (siehe unten). Dabei verfährt $\mathcal{S}$ wie folgt:

- Für $\tilde{\mathcal{S}}_q$ sind alle Nachrichten sichtbar, über welche $\mathcal{S}$ von $[\mathcal{F}]$ durch entsprechende „`send-request`“-Nachrichten informiert wurde. Will $\tilde{\mathcal{S}}_q$ eine dieser Nachrichten ausliefern, sendet $\mathcal{S}$ die jeweilige „`send-permission`“-Nachricht an $[\mathcal{F}]$.

- „`write`“-Anweisungen von $\mathcal{Z}$ werden von $\mathcal{S}$ gespeichert. Wenn der Sender korrumpiert und der Empfänger unkorrumpiert ist, wird in der jeweils nächsten Runde eine „`receive-request`“-Nachricht vom Empfänger der Nachricht an den Angreifer simuliert. Sobald $\mathcal{S}$ ein „`next-round`“-Kommando von $\mathcal{Z}$ empfängt, werden alle gültigen gespeicherten „`receive-permission`“-Nachrichten in Anfragen an $\tilde{\mathcal{S}}_q$ übersetzt, die entsprechende Nachricht an eine Partei auszuliefern.

- Korruptionsanfragen von $\mathcal{S}$ werden an $\tilde{\mathcal{S}}_q$ weitergeleitet. Will $\tilde{\mathcal{S}}$ dann eine Partei P_i korrumpieren, korrumpiert $\mathcal{S}$ zunächst die Dummy-Partei $\tilde{P}_i$ und

versorgt $\tilde{\mathcal{S}}_q$ dann mit der Zustandsfolge und den Bandinhalten von $\tilde{P}_i$. Der dann von $\tilde{\mathcal{S}}_q$ zurückgelieferte P_i-Zustand wird vor der Rückgabe an $\mathcal{Z}$ dann wie folgt angepaßt: Es wird eine Schale $[P_i]$ hinzugefügt, und alle vorher erhaltenen „`send-request`"- bzw. „`receive-request`"-Nachrichten, sowie von $\mathcal{S}$ gesandte „`receive-permission`"-Nachrichten werden in den Zustand von $[P_i]$ integriert.

- Eine von $\mathcal{S}$ an $\mathcal{Z}$ zu liefernde Sicht wird durch eine Anfrage bei $\tilde{\mathcal{S}}_q$ nach zwischen Parteien zu simulierenden Nachrichten konstruiert; diese werden dann als entsprechende „`send-request`"-Nachrichten und – in der nächsten Runde – als „`receive-request`"-Nachrichten gemeldet.

Damit kann q nun nachträglich so gewählt werden, daß $\tilde{\mathcal{S}}_q$ bei einer solchen Simulation nie hält und genügend große Sichten für $\mathcal{S}$ liefern kann. (Ein solches q muß es geben, denn es kommuniziert kein $\tilde{\mathcal{S}}_q$ mit $\mathcal{F}$.)

Sei $\mathcal{Z}$ eine Umgebung im synchronen Modell. Sei dann $\tilde{\mathcal{Z}}$ die Umgebung im Sinne von Abschnitt 3.1, welche intern $\mathcal{Z}$ simuliert und mit der Ausgabe von $\mathcal{Z}$ hält, wenn $\mathcal{Z}$ hält. Die Idee hinter $\tilde{\mathcal{Z}}$ ist, dem simulierten $\mathcal{Z}$ eine Sicht wie bei einer Ausführung im synchronen Modell zu geben. Dazu verfährt $\tilde{\mathcal{Z}}$ wie folgt:

- Bei der ersten Aktivierung von $\mathcal{Z}$ erhält $\mathcal{Z}$ eine „`round-start`"-Eingabe.

- Jede Eingabe, welche $\mathcal{Z}$ den Parteien gibt, wird von $\tilde{\mathcal{Z}}$ gespeichert, bis $\mathcal{Z}$ dem Angreifer eine „`next-round`"-Nachricht schickt. Dann aktiviert $\mathcal{Z}$ – in Reihenfolge der Parteiidentitäten – alle Parteien, welche von $\mathcal{Z}$ Eingaben bekommen haben, nacheinander mit diesen Eingaben. Danach wird $\mathcal{Z}$ wieder mit einer „`round-start`"-Eingabe aktiviert, und es werden für $\mathcal{Z}$ alle Ausgaben sichtbar, welche von den Parteien generiert wurden.

- Jede von „`next-round`" verschiedene Nachricht von $\mathcal{Z}$ an den Angreifer behandelt $\tilde{\mathcal{Z}}$ wie $\mathcal{S}$: Genauer wird für $\mathcal{Z}$ eine Antwort durch eine Anfrage von $\tilde{\mathcal{Z}}$ beim jeweiligen Angreifer generiert; die dabei erhaltenen Nachrichten werden wie in der Beschreibung von $\mathcal{S}$ umformatiert. „`write`"-Anweisungen werden gespeichert und beim nächsten „`next-round`"-Kommando an den Angreifer weitergeleitet. (Anweisungen, eine „`receive-permission`"-Nachricht zu schreiben, werden dabei wie beschrieben umgewandelt.) Korruptionskommandos werden an den Angreifer weitergeleitet und der zurückgelieferte Zustand mit einer Schale versehen. Korruptionsbenachrichtigungen werden von $\tilde{\mathcal{Z}}$ an $\mathcal{Z}$ weitergeleitet.

Durch das Zwischenspeichern aller Nachrichten bis zur nächsten Runde simuliert $\tilde{\mathcal{Z}}$ für $\mathcal{Z}$ gerade ein Scheduling und eine Nachrichtenauslieferung wie bei einer Ausführung im synchronen Modell. Hierbei ist entscheidend, daß nach Annahme keine Kommunikation zwischen $\mathcal{S}$ und $\mathcal{F}$ stattfindet: Andernfalls könnte $\mathcal{F}$ während einer Kommunikation zwischen $\mathcal{Z}$ und $\tilde{\mathcal{S}}_q$ aktiviert werden, oder $\tilde{\mathcal{S}}_q$ könnte in der simulierten Parteiberechnungsphase aktiviert werden.

Genauer hat für alle $k \in \mathbb{N}$ die Ausgabe von $\tilde{\mathcal{Z}}$ bei Protokollabläufen mit $\tilde{\mathcal{A}}_q$ und π die gleiche Verteilung wie $\mathcal{Z}(\emptyset, [\pi], \mathcal{A}_p, k)$. Weiter hat die Ausgabe von $\tilde{\mathcal{Z}}$ bei Protokollabläufen mit $\tilde{\mathcal{S}}_q$ die gleiche Verteilung wie $\mathcal{Z}([\mathcal{F}], \mathcal{S}, k)$. Damit folgt $[\pi] \geq_{\mathsf{s}} [\mathcal{F}]$ im synchronen Modell. Für Protokolle und Funktionalitäten im Modell [39] gelingt ein analoger Beweis, wobei nur offensichtliche Anpassungen bezüglich der Auslieferung von Nachrichten *an* die ideale Funktionalität vorzunehmen sind. □

5.1.7 Offene Fragen

Nun stellt sich die Frage, ob auch eine ähnliche Einbettung der anderen Modelle [116, 26, 117], Backes u. a. [11] möglich ist. Das synchrone Modell aus [26] fokussiert Protokolle zur sicheren Funktionsauswertung und betrachtet nur Umgebungen, welche nicht in einen Protokollablauf eingreifen; insofern scheint eine Beziehung zu diesem Modell nicht offensichtlich.

Die Arbeiten [116] bzw. [117, 11] betrachten synchrone bzw. asynchrone Netzwerkmodellierungen, in denen die betrachteten Kommunikations- und Schedulingmodelle nicht unerheblich von den hier und in [27] betrachteten Modellen abweichen. Auch hier erscheint eine Verbindung nicht offensichtlich; andererseits wäre eine solche Verbindung äußerst gewinnbringend, da dadurch möglicherweise viele in den Modellen [116, 117, 11] gewonnene *strukturelle* Resultate (z. B. [116], Backes u. a. [10], Backes und Pfitzmann [9]) übertragen werden könnten.

Auch ist eine Untersuchung des Gültigkeitsbereichs der Umkehrung von Satz 5.7 interessant. Folgt (für eine gewisse Klasse von Protokollen und Funktionalitäten) schon Sicherheit im Sinne von [27] aus dem hier vorgestellten Sicherheitsbegriff im synchronen Modell? Es kann auch eine Verallgemeinerung von Satz 5.7 dahingehend angestrebt werden, daß auch gewisse Funktionalitäten betrachtet werden, welche mit dem Angreifer kommunizieren; die meisten schon vorgestellten konkreten Funktionalitäten informieren den Angreifer in der Tat über gewisse Eingaben. (Dies mag für viele Funktionalitäten im Sinne von [39] schon gar nicht mehr nötig sein, da hier der Angreifer ohnehin Benachrichtigungen über die Header der gemachten Eingaben erhält, vgl. Abschnitt 4.1.4.)

Weiter deckt Satz 5.7 nur *reale*, aber keine *hybriden* Realisierungen einer Funktionalität $\mathcal{F}$ ab. (Insbesondere von Interesse sind hier etwa das in [39] benutzte $\mathcal{F}_{\text{CRS}}$-hybride Modell oder das in Abschnitt 4.2 benutzte $\mathcal{F}_{\text{RO}}$-hybride Modell.) Diesbezügliche Erweiterungen der Aussage von Satz 5.7 könnten also eine weitere Nähe zu den Modellen [27, 39] aufdecken.

5.2 Ein Unvollständigkeitsresultat

Der verbleibende Teil dieses Kapitels soll sich nun damit beschäftigen, strukturelle Unterschiede zwischen dem soeben besprochenen synchronen Modell und den asynchronenen Modellen aus [27, 39] aufzuzeigen. Genauer soll eine konkrete Protokollaufgabe angegeben werden, welche im hiesigen synchronen Modell selbst von

beliebig vielen „Oblivious-Transfer“- und „Broadcast“-Bausteinen nicht realisiert werden kann. Andererseits kann im asynchronen Modell von [39] mittels „Oblivious Transfer“ und „Broadcast“ (fast[7]) jede Funktionalität realisiert werden – insbesondere die im synchronen Modell *nicht* realisierbare Protokollaufgabe.

Wie der Beweis zeigen wird, liegt das Problem für diese Diskrepanz nicht in einer künstlichen Formulierung der trennenden Protokollaufgabe. Vielmehr ist der tiefere Grund für die Trennung, daß im Falle von *Konflikten* zwischen zwei Parteien, im Verlaufe derer klar wird, daß eine der beiden Parteien korrumpiert ist und sich nicht an das Protokoll hält, ein Protokoll im allgemeinen nicht fortgesetzt werden kann. In einem asynchronen Modell ist dies nicht tragisch, da dann das Protokoll gefahrlos abgebrochen werden kann – für eine Simulation eines solchen Abbruchs unterläßt es der Simulator dann einfach, Ausgaben der idealen Funktionalität auszuliefern.

Andererseits muß in einem synchronen Modell (oder auch nur einem Modell mit einem verläßlichen Netzwerk, welches etwa durch die schon bekannten Schalenkonstruktionen modelliert werden kann) auch im Falle korrumpierter Parteien eine Protokollausgabe garantiert werden. Das reale (oder hybride) Protokoll muß also in jedem Fall fortgesetzt werden. Hierzu kann es nötig sein, den Betrüger (d. h. die korrumpierte Partei) zu *identifizieren*. Der nachfolgende Beweis zeigt, daß dies im allgemeinen nicht möglich ist.

5.2.1 Die betrachteten Hilfsfunktionalitäten

Es soll zunächst mit der Vorstellung der benutzten Protokollbausteine begonnen werden. Als erstes ist in Abbildung 5.1 eine Idealisierung eines Broadcast-Kanals dargestellt. Dessen Funktion braucht nicht weiter kommentiert zu werden; es soll nur darauf hingewiesen werden, daß für eine polynomiale Laufzeitbeschränkung per Aktivierung eine Parametrisierung über eine polynomiale Maximallänge der Eingaben nötig ist.

Die Funktionalität $\mathcal{F}_{\mathrm{BC}}^{\ell}$

Protokollteilnehmer: $\mathcal{F}_{\mathrm{BC}}^{\ell}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$. $\mathcal{F}_{\mathrm{BC}}^{\ell}$ ist parametrisiert über ein Polynom $\ell \in \mathbb{Z}[k]$. Pro Aktivierung wird nur die erste erhaltene Nachricht betrachtet.

- Bei Erhalt einer Nachricht $m \in \{0,1\}^l$ mit $1 \leq l \leq \ell(k)$ von einer Partei P_i: Sende (P_i, m) an alle Parteien und an $\mathcal{S}$.

Abbildung 5.1: Die Idealisierung $\mathcal{F}_{\mathrm{BC}}^{\ell}$ eines Broadcast-Kanals aus [82]

Weiter soll der Baustein „Oblivious Transfer“ betrachtet werden. Er ermöglicht es einer Partei P_i, einer Partei P_j zwei Bits x_1 und x_2 anzubieten; P_j darf dann

[7] Wie schon angedeutet, müssen hier direkte Funktionalitäten und solche Funktionalitäten ausgeschlossen werden, welche von ihrem Wissen um Korruptionen Gebrauch machen.

$m \in \{1,2\}$ wählen und erhält x_m, *nicht* aber x_{3-m}. Umgekehrt erfährt P_i nicht, welches Bit P_j gewählt hat.[8] In Abbildung 5.2 ist eine Formulierung von Oblivious Transfer als ideale Funktionalität angegeben.

Die Funktionalität $\mathcal{F}_{\mathrm{OT}}$

Protokollteilnehmer: $\mathcal{F}_{\mathrm{OT}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$. Pro Aktivierung wird nur die erste erhaltene Nachricht betrachtet.

1. Bei Erhalt von $(\texttt{Sender}, P_i, P_j, x_1, x_2)$ mit $x_1, x_2 \in \{0,1\}$ von P_i: Speichere diese Nachricht und sende $(\texttt{Sender}, P_i, P_j)$ an $\mathcal{S}$; ignoriere jede weitere Nachricht für diesen Wert von (P_i, P_j).
2. Bei Erhalt von $(\texttt{Receiver}, P_i, P_j, m)$ mit $m \in \{1,2\}$ von P_j: Speichere diese Nachricht und sende $(\texttt{Receiver}, P_i, P_j)$ an $\mathcal{S}$; ignoriere jede weitere Nachricht für diesen Wert von (P_i, P_j).
3. Wenn zu irgendeinem Zeitpunkt für irgendwelche P_i, P_j sowohl eine $(\texttt{Sender}, P_i, P_j, x_1, x_2)$-Nachricht als auch eine $(\texttt{Receiver}, P_i, P_j, m)$-Nachricht gespeichert sind: Sende (einmalig) $(\texttt{Transferred}, P_i, P_j, x_m)$ an P_j und $(\texttt{Transferred}, P_i, P_j)$ an $\mathcal{S}$.

Abbildung 5.2: Die Idealisierung $\mathcal{F}_{\mathrm{OT}}$ von Oblivious Transfer aus [82]

5.2.2 Vollständigkeit im asynchronen Modell

Bezüglich klassisch sicherer Funktionsauswertungen existieren eine Reihe von Protokollkonstruktionen (etwa Kilian [94], Goldwasser und Levin [74], Crépeau u. a. [51]), welche ausschließlich Oblivious-Transfer-Bausteine (und gegebenenfalls einen Broadcast-Kanal) verwenden; die angegebenen Konstruktionen benötigen für Sicherheitsaussagen dabei noch nicht einmal komplexitätstheoretische Annahmen wie etwa die Existenz von Einwegfunktionen.

Auf der anderen Seite wird im simulierbarkeitsbasierten Modell von [39] eine generelle Realisierbarkeitsaussage für als ITMs formulierte idealisierte Funktionalitäten angegeben. Hierzu wird die Protokollkonstruktion von Goldreich u. a. aus [73] in erweiterter Form benutzt.

Genauer werden für in einem solchermaßen konstruierten Protokoll die Protokollbausteine „Global Bit Commitment“[9] und „Broadcast“ benötigt. Auf diese Weise sind diese beiden Bausteine für das asynchrone Modell aus [39] – unter geeigneten komplexitätstheoretischen Annahmen und im schon angesprochenen Sinne nur „fast“ – *vollständig*.

[8] Gelegentlich – etwa in der ursprünglichen Fassung aus Rabin [120] – wird Oblivious Transfer als verrauschter Kanal modelliert.

[9] Dieser sogleich genauer zu untersuchende Protokollbaustein wird in [39] „One-to-Many-Commitment“ genannt.

Es soll kurz skizziert werden, warum in einem asynchronen Modell wie [39] schon mit mehreren Oblivious-Transfer-Bausteinen und einem Broadcast-Kanal die Funktionalität „Global Bit Commitment“ implementiert werden kann. Zunächst sei die Protokollaufgabe „Global Bit Commitment“ informell als Verallgemeinerung von „Bit Commitment“ (siehe auch Abschnitt 4.2) beschrieben, die es einer Partei erlaubt, sich bei *allen anderen* (und nicht nur bei *einer anderen*) Partei auf ein Bit festzulegen.

Abgesehen davon stimmt „Global Bit Commitment“ mit „Bit Commitment“ überein; es ergibt sich eine ideale Funktionalität $\mathcal{F}_{\text{GCOM}}$, wie sie in Abbildung 5.3 dargestellt ist. (Der Einfachheit halber verwaltet diese Funktionalität im Gegensatz zu den Funktionalitäten aus Abschnitt 4.2 nur ein Commitment pro Partei; die Aussage des nächsten Abschnitts wird dadurch nur gestärkt.)

Die Funktionalität $\mathcal{F}_{\text{GCOM}}$

Protokollteilnehmer: $\mathcal{F}_{\text{GCOM}}$, Umgebung $\mathcal{Z}$, Parteien $P_1, \ldots, P_n$ und Angreifer $\mathcal{S}$. In jeder Aktivierung wird nur die erste erhaltene Nachricht betrachtet.

- **Commit-Phase**: Bei Erhalt einer Nachricht $(\texttt{Commit}, b)$ mit $b \in \{0, 1\}$ von einer Partei P_i: Speichere das Tupel (P_i, b) und sende $(\texttt{Receipt}, P_i)$ an alle Parteien und $\mathcal{S}$. Ignoriere weitere `Commit`-Nachrichten von P_i.
- **Aufdeck-Phase**: Bei Erhalt einer Nachricht (`Reveal`) von einer Partei P_i: Wenn ein Tupel (P_i, b) gespeichert wurde, sende die Nachricht $(\texttt{Reveal}, P_i, b)$ an alle Parteien und $\mathcal{S}$. Ansonsten tue nichts.

Abbildung 5.3: Die ideale Funktionalität $\mathcal{F}_{\text{GCOM}}$ aus [82]

Betrachtet man die gerade vorgestellten Funktionalitäten im asynchronen Modell von [39], so kann schon im $\mathcal{F}_{\text{OT}}$-hybriden Modell die Funktionalität $\mathcal{F}_{\text{MCOM}}$ aus [39] (vgl. auch Abschnitt 4.2.1) realisiert werden. Um ein Commitment auf ein Bit b zu generieren, müssen lediglich k $\mathcal{F}_{\text{OT}}$-Instanzen mit „$(\texttt{Sender}, P_i, P_j, b \oplus r_l, r_l)$“-Eingaben für gleichverteilte Bits $r_1, \ldots, r_k$ aufgerufen werden. Aufdecken eines Commitments erfolgt durch Senden von b und aller r_l an P_j. Die Simulierbarkeit dieses Protokolls ist leicht einzusehen.

Andererseits kann in einem *asynchronen* Modell wie [39] schon „Global Commitment“ mit einem Broadcast-Kanal und Instanzen von $\mathcal{F}_{\text{MCOM}}$ realisiert werden. Ein Commitment bei allen Parteien kann durch einzelne Commitments mittels $\mathcal{F}_{\text{MCOM}}$ erfolgen; lediglich beim Aufdecken sendet jede Partei ihr (vorläufiges) Aufdeckergebnis mittels Broadcast an alle anderen Parteien. Die Generierung einer Ausgabe erfolgt jedoch erst, wenn ein Aufdeckergebnis von *allen* anderen Parteien bestätigt wurde.

Dieses Protokoll kann asynchron simuliert werden, da ein Simulator „`Reveal`“-Ausgaben von $\mathcal{F}_{\text{GCOM}}$ beliebig lange verzögern kann. Allerdings kann schon eine korrumpierte Partei das gesamte Protokoll zum Abbruch bringen, wenn ihr bei-

spielsweise die Ausgabe des Protokolls nicht gefällt. (Da dies asynchron prinzipiell immer möglich ist, darf ein solches Protokoll nicht als trivial betrachtet werden; siehe auch die Diskussion in Abschnitt 4.1.2.)

In diesem Sinne kann also – im asynchronen Modell [39] – mittels $\mathcal{F}_{\mathrm{OT}}$ und $\mathcal{F}_{\mathrm{BC}}^{\ell}$ (mit geeignetem ℓ) schon $\mathcal{F}_{\mathrm{GCOM}}$ realisiert werden. In Kombination mit dem Ergebnis aus [39] sind damit die Funktionalitäten $\mathcal{F}_{\mathrm{OT}}$ und $\mathcal{F}_{\mathrm{BC}}^{\ell}$ zusammengenommen (fast) vollständig.

5.2.3 Unvollständigkeit im synchronen Modell

Man beachte, daß – wenn im hiesigen synchronen Modell betrachtet – noch mit dem Problem der zu frühen Festlegung auf Eingaben umgegangen werden muß (vgl. auch Abschnitt 4.1.1). Ein Weg, dieses Problem zu lösen, ist eine Änderung der idealen Funktionalität wie etwa im Falle der Commitment-Funktionalität $\mathcal{F}_{\mathrm{SCOM}}$ aus Abschnitt 4.2.2. Um jedoch die gewünschte Nähe zu den Funktionalitäten aus [39] herzustellen, kann man das dortige Scheduling – wie in Satz 5.7 – durch eine passende Schale herstellen, welche im synchronen Modell die gleiche Situation herbeiführt wie im Modell von [39].

Genauer sei $[\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}(p_{\mathsf{recv}}, p_{\mathsf{send}})$ identisch zu $[\mathcal{F}](p_{\mathsf{recv}}, p_{\mathsf{send}}, \mathsf{async})$, bis auf die Tatsache, daß im Falle eingehender Nachrichten von einer Partei zusätzlich zu der Benachrichtigung über die eingegangene Nachricht auch die „Header“ dieser Nachricht an $\mathcal{A}$ gegeben werden. Der Header einer „$(\mathtt{Sender}, P_i, P_j, x_1, x_2)$“-Nachricht mit $x_1, x_2 \in \{0,1\}$ bestehe dabei aus „$(\mathtt{Sender}, P_i, P_j)$“ und der Header einer „$(\mathtt{Receiver}, P_i, P_j, m)$“-Nachricht mit $m \in \{1,2\}$ aus „$(\mathtt{Receiver}, P_i, P_j)$“. Alle anderen Nachrichten haben einen Header der Form „$(\mathtt{Invalid})$“. Damit ist diese Konstruktion wegen $[\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}(\infty,\infty) = [\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}$ eine Verallgemeinerung der für die Formulierung von Satz 5.7 beschriebenen Schale $[\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}$.

Seien weiter $[\mathcal{F}_{\mathrm{BC}}]_{\mathrm{UC}'}(p_{\mathsf{recv}}, p_{\mathsf{send}})$ und $[\mathcal{F}_{\mathrm{GCOM}}]_{\mathrm{UC}'}(p_{\mathsf{recv}}, p_{\mathsf{send}})$ die analogen Konstruktionen für die Funktionalitäten $\mathcal{F}_{\mathrm{BC}}$ bzw. $\mathcal{F}_{\mathrm{GCOM}}$.

Dann kann die Unvollständigkeit von Broadcast und Oblivious Transfer im hier vorgestellten synchronen Modell wie folgt allgemein ausgedrückt werden:

Satz 5.8. *Seien $\ell, p_{\mathrm{BC}}, p_{\mathrm{OT}}, p_{\mathsf{recv}}, p_{\mathsf{send}} \in \mathbb{Z}[k]$ alle > 0 für alle $k \in \mathbb{Z}$. Dann ist selbst im $\{[\mathcal{F}_{\mathrm{BC}}^{\ell}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{BC}}, [\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{OT}}\}$-hybriden synchronen Modell die Funktionalität $[\mathcal{F}_{\mathrm{GCOM}}]_{\mathrm{UC}'}(p_{\mathsf{recv}}, p_{\mathsf{send}})$ für $n \geq 3$ Parteien nicht realisierbar.*

Bevor ein Beweis angegeben wird, soll die Aussage des Satzes noch kommentiert werden. Satz 5.8 sagt aus, daß selbst eine beliebige (polynomiale) Anzahl von Oblivious-Transfer- und Broadcast-Instanzen, welche zudem alle Anfragen *sofort* beantworten, noch nicht hinreicht, um eine einzige Instanz von „Global Bit Commitment“ zu implementieren. Diese Aussage steht in krassem Gegensatz zur Vollständigkeit von Oblivious Transfer und Broadcast im asynchronen Modell.

Beweis von Satz 5.8. Seien also Polynome $\ell, p_{\mathrm{BC}}, p_{\mathrm{OT}}, p_{\mathsf{recv}}, p_{\mathsf{send}} \in \mathbb{Z}[k]$ gegeben, für welche $\pi \geq_{\mathsf{s}} [\mathcal{F}_{\mathrm{GCOM}}]_{\mathrm{UC}'}(p_{\mathsf{recv}}, p_{\mathsf{send}})$ mit einem 3-Parteien-Protokoll π gelte, wel-

ches im $\{[\mathcal{F}_{\mathrm{BC}}^{\ell}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{BC}}, [\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{OT}}\}$-hybriden Modell formuliert sei. (Der folgende Beweis überträgt sich auf ein n-Parteien-Protokoll mit $n > 3$.)

Zur Herstellung eines Widerspruchs werden nun Umgebungen $\mathcal{Z}_1, \ldots, \mathcal{Z}_5$ angegeben, so daß eine dieser Umgebungen ein erfolgreicher Unterscheider zwischen π und $\mathcal{A}_q$ (für hinreichend großes $q \in \mathbb{Z}[k]$) einerseits und $[\mathcal{F}_{\mathrm{GCOM}}]$ und dem entsprechenden $\mathcal{S}_q$ andererseits sein *muß*. Für jedes dieser $\mathcal{Z}_i$ wird angenommen, daß $\mathcal{Z}_i$ automatisch eine geeignete Beschneidung weitergeleiteter Nachrichten und Ein- bzw. Ausgaben vornimmt, so daß $\mathcal{Z}_i$ polynomial beschränkt ist. Da die Wahl der $\mathcal{Z}_i$ unabhängig von q ist, sei schon vorweg ein konkretes geeignetes q angenommen. Alle folgenden Betrachtungen beziehen sich auf ein solches, hinreichend großes q. Außerdem sei als platzsparende Abkürzung

$$\mathcal{H} := \{[\mathcal{F}_{\mathrm{BC}}^{\ell}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{BC}}, [\mathcal{F}_{\mathrm{OT}}]_{\mathrm{UC}'}(0,0)||p_{\mathrm{OT}}\}$$

vereinbart. Eine kurze Übersicht über den gesamten Beweis ist in Abbildung 5.4 zu finden.

Es sei nun $\mathcal{Z}_1$ beschrieben. $\mathcal{Z}_1$ läßt zu Anfang des Protokolls vom Angreifer die Partei P_2 korrumpieren. $\mathcal{Z}_1$ verwaltet dann intern eine Simulation $\overline{P}_2$ von P_2, welche über den Angreifer am Protokoll teilnimmt; lediglich Nachrichten, welche $\overline{P}_2$ direkt an P_1 schicken will, werden nicht weitergeleitet, genauso wie Nachrichten, die P_1 an P_2 schickt.

$\mathcal{Z}_1$ gibt nun der unkorrumpierten Partei P_1 eine Eingabe „$(\texttt{Commit}, r)$“ für ein gleichverteiltes $r \in \{0,1\}$. Dann wartet $\mathcal{Z}_1$ insgesamt $p_{\mathsf{recv}}(k) + p_{\mathsf{send}}(k)$ Runden – erhält dabei aber die Simulation $\overline{P}_2$ aufrecht – auf eine mögliche „$\texttt{Receipt}$“-Ausgabe von P_3. Kommt diese Ausgabe, gibt $\mathcal{Z}_1$ anschließend P_1 eine „$(\texttt{Reveal})$“-Eingabe und wartet wieder $p_{\mathsf{recv}}(k) + p_{\mathsf{send}}(k)$ Runden auf eine „$(\texttt{Reveal}, P_1, \overline{r})$“-Ausgabe von P_3. Kommt diese Ausgabe, hält $\mathcal{Z}_1$ mit Ausgabe $r \oplus \overline{r}$. In allen anderen Fällen hält $\mathcal{Z}_1$ mit 1-Ausgabe.

Nach Definition und Wahl der Schalenparameter von $\mathcal{F}_{\mathrm{GCOM}}$ ist

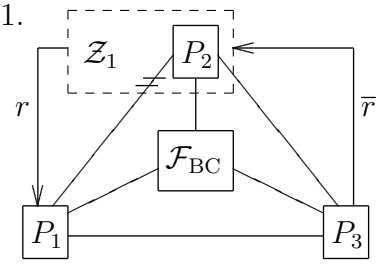
$$\mathsf{Pr}[\mathcal{Z}_1([\mathcal{F}_{\mathrm{GCOM}}], \mathcal{S}_q, k) = 0] = 1,$$

womit $\mathsf{Pr}[\mathcal{Z}_1(\mathcal{H}, \pi, \mathcal{A}_q, k) = 0]$ überwältigend in k sein muß. Deshalb generiert also in Protokoll π und in einem Protokollablauf mit $\mathcal{Z}_1$ die Partei P_3 mit überwältigender Wahrscheinlichkeit schließlich eine „$(\texttt{Reveal}, P_1, r)$“-Ausgabe.

Sei $\mathcal{Z}_2$ identisch mit $\mathcal{Z}_1$, nur daß $\mathcal{Z}_2$ initial P_1 korrumpieren läßt und intern eine Simulation $\overline{P}_1$ von P_1 (mit „frischem“ Zufallsband) über den Simulator am Protokoll teilnehmen läßt. Hierbei wird wieder alle direkte $\overline{P}_1$-P_2-Kommunikation blockiert; die „$\texttt{Commit}$“-Eingabe bekommt die lokale Simulation $\overline{P}_1$ von P_1. In einem Protokollablauf von Protokoll π sind damit die Sicht von P_1 und $\overline{P}_2$ bei einem Ablauf mit $\mathcal{Z}_1$ identisch mit den jeweiligen Sichten von $\overline{P}_1$ bzw. P_2 bei einem Ablauf mit $\mathcal{Z}_2$. Insgesamt ist also auch die Sicht von P_3 identisch in Abläufen mit π und $\mathcal{Z}_1$ bzw. $\mathcal{Z}_2$, und es ist

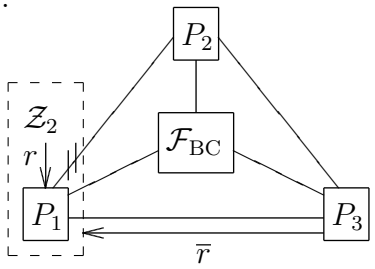
$$\mathsf{Pr}[\mathcal{Z}_2(\mathcal{H}, \pi, \mathcal{A}_q, k) = 0]$$

1.



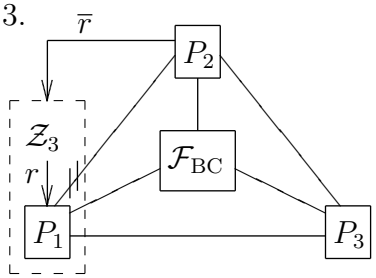
$r = \overline{r}$ ideal $\Rightarrow$ $r = \overline{r}$ in π

2.



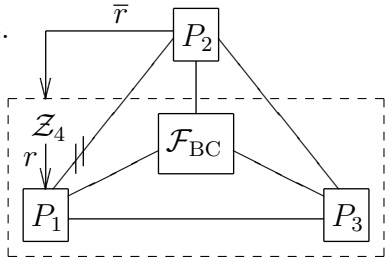
$r = \overline{r}$ in π $\Rightarrow$ $r = \overline{r}$ ideal

3.



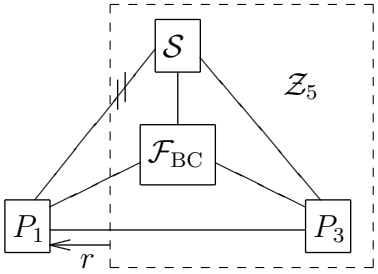
$r = \overline{r}$ ideal $\Rightarrow$ $r = \overline{r}$ in π

4.



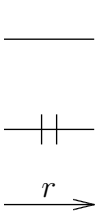
$r = \overline{r}$ in π $\Rightarrow$ $r = \overline{r}$ ideal
$\Rightarrow$ $\mathcal{S}$ extrahiert r in Commit-Phase

5.



$\mathcal{S}$ spielt P_2 und extrahiert r
$\Rightarrow$ Widerspruch zu idealer Definition

Notation:



P_i-P_j-Komm. (inkl. OT)
oder Komm. mit $\mathcal{F}_{\mathrm{BC}}$

blockierte Kommunikation

r Ein- oder Ausgaben

gestr. Linien umranden Umgebungen

Abbildung 5.4: Kurzfassung des Beweises von Satz 5.8

demnach überwältigend in k. Also muß auch $\mathsf{Pr}[\mathcal{Z}_2([\mathcal{F}_{\mathrm{GCOM}}], \mathcal{S}_q, k) = 0]$ überwältigend in k sein. Nach Definition der idealen Funktionalität gibt damit also bei Abläufen mit $\mathcal{Z}_2$ nicht nur P_3, sondern schließlich auch P_2 eine „$(\texttt{Reveal}, P_1, r)$“-Ausgabe. Mit einer offensichtlichen Modifikation $\mathcal{Z}_3$ von $\mathcal{Z}_2$ (welche ihre Ausgabe von der Ausgabe von P_2 statt der von P_3 abhängig macht) kann dann gezeigt werden, daß auch in einem Protokollablauf mit π die Partei P_2 mit überwältigender Wahrscheinlichkeit schließlich solche „$(\texttt{Reveal}, P_1, r)$“-Ausgabe generiert.

Sei nun $\mathcal{Z}_4$ identisch mit $\mathcal{Z}_3$, nur daß $\mathcal{Z}_4$ zusätzlich zu P_1 initial auch P_3 korrumpieren läßt. Wie für P_1 verwaltet $\mathcal{Z}_4$ intern eine Simulation $\overline{P}_3$ von P_3 (wieder mit „frischem“ Zufallsband). Allerdings wird die Kommunikation von $\overline{P}_3$ ohne Einschränkungen über den Angreifer bzw. geeignete $\mathcal{Z}_4$-interne Simulationen eines Klartext- bzw. Oblivious-Transfer-Kanals zwischen $\overline{P}_1$ und $\overline{P}_3$ weitergeleitet. Demnach ist also für alle $k \in \mathbb{N}$

$$\mathsf{Pr}[\mathcal{Z}_3(\mathcal{H}, \pi, \mathcal{A}_q, k) = 0] = \mathsf{Pr}[\mathcal{Z}_4(\mathcal{H}, \pi, \mathcal{A}_q, k) = 0]\,,$$

weshalb $\mathsf{Pr}[\mathcal{Z}_4([\mathcal{F}_{\mathrm{GCOM}}], \mathcal{S}_q, k) = 0]$ überwältigend in k sein muß. Nach Konstruktion von $\mathcal{F}_{\mathrm{GCOM}}$ und $\mathcal{Z}_4$ muß demnach $\mathcal{S}_q$ die Partei P_2 dazu bringen, beizeiten eine „$\texttt{Receipt}$“-Ausgabe zu generieren. Hierfür muß $\mathcal{S}_q$ im Namen der korrumpierten Partei P_1 eine „$(\texttt{Commit}, r)$“-Eingabe an $[\mathcal{F}_{\mathrm{GCOM}}]$ schicken, wobei – um spätere Unterscheidung durch $\mathcal{Z}_4$ zu vermeiden – r das auch von $\mathcal{Z}_4$ gewählte Bit sein muß. Mit anderen Worten: $\mathcal{S}_q$ muß das von $\overline{P}_1$ gemachte Commitment *brechen*, und zwar *bevor* $\mathcal{Z}_4$ ein Aufdecken des Commitments befiehlt.

Dies kann wie folgt genutzt werden, um den erwünschten Widerspruch herbeizuführen. Die Umgebung $\mathcal{Z}_5$ läßt anfangs P_2 und P_3 korrumpieren und verwaltet intern Simulationen $\overline{P}_3$ und $\overline{\mathcal{S}}_q$ von P_3 bzw. $\mathcal{S}_q$. Hierbei wird die *Simulation* $\overline{\mathcal{S}}_q$ anfangs angewiesen, P_1 und P_3 zu korrumpieren; verlangt $\overline{\mathcal{S}}_q$ tatsächlich Korruption dieser Parteien, versorgt $\mathcal{Z}_5$ die Simulation $\overline{\mathcal{S}}_q$ mit dem initialen Zustand der jeweiligen Dummy-Parteien. Der leitende Gedanke bei den folgenden Überlegungen ist, der Simulation $\overline{\mathcal{S}}_q$ den Eindruck eines Protokollablaufs mit $\mathcal{Z}_4$ zu vermitteln.

Hierzu verfährt $\mathcal{Z}_5$ folgendermaßen:

- Alle nicht via Broadcast gesendete Kommunikation zwischen $\overline{P}_3$ und P_1 (gegebenenfalls also auch $[\mathcal{F}_{\mathrm{OT}}]||p_{\mathrm{OT}}$-Benutzungen) wird durch den Angreifer $\mathcal{A}$, mit dem $\mathcal{Z}_5$ interagiert, von und zur unkorrumpierten Partei P_1 weitergeleitet. Man beachte, daß $\overline{\mathcal{S}}_q$ in einem Protokollablauf mit $\mathcal{Z}_4$ auf solche Kommunikation keinen Zugriff hat.

- Alle nicht via Broadcast gesendete Kommunikation zwischen $\overline{P}_3$ und P_2 wird in Form geeigneter „$\texttt{write}$“-Kommandos an $\overline{\mathcal{S}}_q$ weitergeleitet. (Wie schon angedeutet übernimmt intuitiv $\overline{\mathcal{S}}_q$ die Rolle von P_2.)

- Alle nicht via Broadcast gesendete Kommunikation zwischen P_1 und P_2 wird *nicht* weitergeleitet – dies ist der entscheidende Punkt in diesem Beweis. Man beachte, daß zwar sowohl $\overline{\mathcal{S}}_q$ als auch der nicht-simulierte Angreifer $\mathcal{A}$

erwarten, Kontrolle über diese Kommunikationskanäle zu haben. Allerdings kann diese Kommunikation blockiert werden, da alle bisherigen Ergebnisse bezüglich der Umgebungen $\mathcal{Z}_1, \ldots, \mathcal{Z}_4$ hergeleitet wurden, welche die P_1-P_2-Kommunikation ohnehin unterdrückt haben.

- $\overline{\mathcal{S}}_q$ wird angewiesen, sofort jede Nachricht von P_2 an $[\mathcal{F}_{\mathrm{BC}}]||p_{\mathrm{BC}}$ auszuliefern; „Ausliefern“ meint hier Senden der jeweiligen „`receive-permission`“-Nachricht an die ideale Funktionalität. Nach Definition von $\mathcal{F}_{\mathrm{BC}}$ lernt der Angreifer – und damit auch $\mathcal{Z}_5$ – die mittels Broadcast zu sendende Nachricht daraufhin. Auf diese Weise durch P_2 gesendete Nachrichten werden von $\mathcal{Z}_5$ sofort an $\mathcal{A}$ weitergeleitet, um die jeweilige Nachricht im Namen von P_2 mittels $[\mathcal{F}_{\mathrm{BC}}]||p_{\mathrm{BC}}$ zu senden. (Natürlich wird auch $\mathcal{A}$ angewiesen, diese Nachricht sofort an $[\mathcal{F}_{\mathrm{BC}}]||p_{\mathrm{BC}}$ auszuliefern.)

- In analoger Weise werden von P_1 via Broadcast versandte Nachrichten, welche von $\mathcal{A}$ gemeldet werden, an $\overline{\mathcal{S}}_q$ und $\overline{P}_3$ weitergeleitet. Schließlich werden Nachrichten, welche $\overline{P}_3$ via Broadcast versenden will, an $\mathcal{A}$ *und* an $\overline{\mathcal{S}}_q$ weitergeleitet.

$\mathcal{Z}_5$ selbst gibt – genau wie die zuvor beschriebenen $\mathcal{Z}_i$ – der unkorrumpierten Partei P_1 anfänglich eine „$(\texttt{Commit}, r)$“-Eingabe, wartet jedoch dann darauf, daß $\overline{\mathcal{S}}_q$ eine „$(\texttt{Commit}, \overline{r})$“-Nachricht im Namen von P_1 an $[\mathcal{F}_{\mathrm{GCOM}}]$ schreibt. Falls $\overline{\mathcal{S}}_q$ selbst nach $p_{\mathsf{recv}}(k) + p_{\mathsf{send}}(k)$ Runden keine solche Nachricht schreibt, hält $\mathcal{Z}_5$ mit Ausgabe r, andernfalls hält $\mathcal{Z}_5$ mit Ausgabe $r \oplus \overline{r}$.

Die gerade gegebenen Kommunikationsregeln stellen sicher, daß die Sicht von $\overline{P}_3$ und damit die Sicht von $\overline{\mathcal{S}}_q$ bei einem Protokollablauf mit $\mathcal{Z}_5$ und π identisch ist mit den Sichten von $\overline{P}_3$ bzw. $\mathcal{S}_q$ bei einem Protokollablauf mit $\mathcal{Z}_4$ im $[\mathcal{F}_{\mathrm{GCOM}}]$-idealen Modell. Damit ist $\mathsf{Pr}[\mathcal{Z}_5(\mathcal{H}, \pi, \mathcal{A}_q, k) = 0]$ also überwältigend in k. Andererseits muß

$$\mathsf{Pr}[\mathcal{Z}_5([\mathcal{F}_{\mathrm{GCOM}}], \mathcal{S}_q, k) = 0] = \frac{1}{2}$$

sein, da im idealen Modell die Bits r und $\overline{r}$ nach Definition von $\mathcal{F}_{\mathrm{GCOM}}$ statistisch unabhängig sind und r gleichverteilt ist. Insgesamt unterscheidet $\mathcal{Z}_5$ also erfolgreich Ausführungen mit π und $\mathcal{A}_p$ von Ausführungen mit $\mathcal{S}_q$ und $[\mathcal{F}_{\mathrm{GCOM}}]$ – der gewünschte Widerspruch.

Für den n-Parteienfall mit $n > 3$ findet die gleiche Argumentation Anwendung, wenn hier die Partei P_3 als die Gesamtheit aller von P_1 und P_2 verschiedenen Parteien betrachtet wird. Man beachte, daß die garantierte Auslieferungseigenschaft des Modells schon im ersten Beweisschritt benutzt wird: Ohne eine solche Eigenschaft könnte nicht garantiert werden, daß selbst bei einem *blockierten* Kanal zwischen P_1 und P_2 das Protokoll zu einer Ausgabe kommt. Ein Simulator im idealen Modell, welcher nicht an „Auslieferungspflichten“ gebunden ist, kann in solchen Konfliktfällen immer die Auslieferung der „`Receipt`“- bzw. „`Reveal`“-Ausgaben unterdrücken.

Schließlich soll noch erwähnt werden, daß der letzte Beweisschritt eine Adaption des in Abschnitt 4.2.1 skizzierten Arguments aus Canetti und Fischlin [34] ist,

welches Bit Commitment (für zwei Parteien) *ohne* Hilfsfunktionalitäten als nicht simulierbar realisierbar zeigt. Der gesamt Vorbau des Beweises diente also letztlich nur dazu, die Situation auf ein Zwei-Parteien-Commitment zu reduzieren. □

Die Stärke des gegebenen Beweises soll noch durch folgende Bemerkung untermauert werden: Man könnte einwerfen, daß eine Funktionalität $\mathcal{F}_{\text{GCOM}}$ ohnehin schwer zu realisieren ist, da eine Simulation immer noch gültig sein muß, selbst wenn realer und idealer Angreifer angehalten haben. (Man beachte, daß im synchronen Modell aus gutem Grund eine strikte polynomiale Beschränkung der Angreifer gefordert wurde.) Eine hinreichend langlebige Umgebung könnte so zunächst polynomial viele Runden „abwarten“, bis realer und idealer Angreifer gehalten haben müssen, und erst dann mit dem eigentlichen Protokollablauf beginnen.

Diese Problematik läßt sich durch Funktionalitäten behandeln, welche nach polynomial vielen *Runden* halten bzw. keine Dienste mehr garantieren. Auf diese Weise kann eine polynomiale Laufzeit eines Simulators gewählt werden, so daß der Simulator erst anhält, wenn auch die ideale Funktionalität gehalten hat; das beschriebene Problem tritt nicht mehr auf.

Nun gilt der Beweis von Satz 5.8 auch für solchermaßen „künstlich beschränkte“ Versionen von $\mathcal{F}_{\text{GCOM}}$; das im Beweis ausgenutzte Problem ist kein Laufzeitproblem, sondern eine prinzipielle Eigenschaft von verläßlichen Netzwerken. (Eine weitergehende Untersuchung und Klassifizierung ähnlicher Eigenschaften verläßlicher Netzwerke ist in Backes, Hofheinz, Müller-Quade und Unruh [8] zu finden.)

5.2.4 Offene Fragen

Wie gerade angedeutet ist nach einem etwas verstörenden Resultat wie der soeben gegeben Trennung von synchronen und asynchronen Netzwerken vor allem eine weitergehende Untersuchung interessant. Vor allem könnte gehofft werden, eine teilweise „Hierarchie“ von Netzwerken aufzubauen, und (Nicht-)Implikationen von induzierten Sicherheitsbegriffen zu finden. Diesbezügliche Resultate konnten in [8] gefunden werden.

Auch könnten durch die Suche nach einer vollständigen Kombination von Funktionalitäten im gerade gegebenen synchronen Modell neue strukturelle Einsichten in die Verbindung von synchronen und asynchronen Sicherheitsbegriffen gewonnen werden. Ist überhaupt eine allgemeine simulierbare Protokollkonstruktion in einem synchronen Modell möglich?

6 Thesen und Ausblick

Einerseits zeigen die in Kapitel 2 exemplarisch für Public-Key-Systeme gemachten Betrachtungen, daß für die Beurteilung der Sicherheit eines kryptographischen Systems notwendig ein formaler Rahmen geschaffen werden muß. Es müssen also Annahmen etwa über den Umfang möglicher Angriffe oder die Güte des Kommunikationsnetzwerks getroffen werden.

Andererseits hängen die bei einem solchem Vorgehen erzielten Ergebnisse in äußerst sensibler Weise von den getroffenen Annahmen ab; hier sei insbesondere auf die Abschnitte 4.1.4, 3.2.2 und speziell auf Kapitel 5 hingewiesen.

Allerdings ist nicht nur aus ästhetischen Gründen eine Abstraktion – etwa in Form eines geeigneten Kalküls – dynamischer Protokollabläufe wünschenswert. Eine weitere Motivation bietet hier die gegenwärtige Komplexität von Sicherheitsbeweisen für reaktive Systeme. Insbesondere problematisch ist dabei, daß mit dieser Komplexität auch eine Fehleranfälligkeit einhergeht, wie die Kapitel 3 und 4 anhand verschiedener in der Literatur zu findender Beweise aufzeigen.

Damit ergibt sich das Dilemma, daß zwar aufgrund der komplexen Natur dynamischer Protokollabläufe eine abstrahierende Behandlung erforderlich ist; jedoch stellt sich hierbei die Herausforderung, eine Abstraktion zu finden, welche feinauflösend bezüglich der gemachten Annahmen ist.

Als Teil einer diesbezüglichen Bestandsaufnahme können die jüngeren Arbeiten Backes, Hofheinz, Müller-Quade und Unruh [8], Hofheinz und Unruh [90], Hofheinz, Müller-Quade und Unruh [87], Hofheinz und Unruh [91] gesehen werden: In [8] werden für verschiedene Typen von Netzwerken konkrete Modellierungen angegeben und die induzierten Sicherheitsbegriffe anhand konkreter Protokollaufgaben getrennt. In ähnlicher Weise untersucht [90] den Zusammenhang zwischen verschiedenen etablierten Typen simulierbarer Sicherheit.

Weiter zeigt [91] die Nicht-Komponierbarkeit der Formulierung „statistischer Sicherheit“ von Pfitzmann und Waidner aus [117]. Damit wird insbesondere der Beweis des Kompositionstheorems aus [117] für den Fall statistischer Sicherheit falsifiziert. Die Situation kann aber in [91] entspannt werden, indem eine verbesserte, Komponierbarkeit garantierende Formulierung statistischer Sicherheit angegeben wird. Die Arbeit [87] schließlich definiert und untersucht einen universellen und intuitiven Begriff für „beschränkte Systeme“ und „beschränkte Angriffe“ – daß das Finden einer schlüssigen diesbezüglichen Formalisierung nicht-trivial ist, zeigt Abschnitt 3.2.2.

A Hilfsmittel

Größtenteils Goldreich [70] folgend, werden hier die benutzte Notation und die verwendeten technischen Hilfsmittel vorgestellt.

Notation A.1. *Bei Wahl eines Elements $x \in M$ wird von* zufälliger Wahl *gesprochen, wenn die dabei unterstellte Verteilung auf M klar ist. Bei Unklarheit wird die entsprechende Verteilung (z. B. Gleichverteilung für endliche M) explizit angegeben.*

Notation A.2. $\mathsf{Pr}[X \mid Y]$ *bezeichnet die* bedingte Wahrscheinlichkeit von X unter der Hypothese Y.

Notation A.3. $\mathsf{Pr}[A : X]$ *bezeichnet die* Wahrscheinlichkeit von X nach Ausführung des Algorithmus A.

Definition A.4. *Eine Funktion $f : \mathbb{N} \to \mathbb{R}$ heißt* vernachlässigbar*, wenn für jedes $c \in \mathbb{N}$ ein $k_c \in \mathbb{N}$ existiert, dergestalt daß für alle $k > k_c$ schon $|f(k)| < k^{-c}$ ist. Eine nicht vernachlässigbare Funktion heißt* nicht-vernachlässigbar.

Definition A.5. *Eine Funktion $f : \mathbb{N} \to \mathbb{R}$ heißt* überwältigend*, wenn $1 - f$ vernachlässigbar ist. Eine nicht überwältigende Funktion heißt* nicht-überwältigend.

Definition A.6. *Ein Algorithmus A heißt* PPT *(probabilistic polynomial time), wenn er zum einen probabilistisch und zum anderen (strikt) laufzeitbeschränkt durch ein festes Polynom in der Länge seiner Eingaben ist. Ein PPT-Algorithmus A heißt* PPT in der ersten Eingabe*, falls seine Laufzeit sogar durch ein festes Polynom in der Länge seiner ersten Eingabe beschränkt ist.*

Notation A.7. *Für einen PPT-Algorithmus A und $r \in \{0,1\}^*$ bezeichne $A(x;r)$ die Ausführung von A mit expliziten Zufallswahlen r (d. h. die von A benutzten Zufallsbits werden der Reihe nach aus r gelesen). Dabei wird r mit Nullen aufgefüllt, falls nötig.*

Notation A.8. *Für einen Algorithmus A mit Unterprogrammaufrufen, welche nicht in A selbst implementiert sind, und einen Algorithmus B, welcher diese Unterprogrammaufrufe implementiert, bezeichne A^B den komponierten Algorithmus, welcher A ausführt und dabei alle nicht in A selbst implementierten Unterprogrammaufrufe an B weiterleitet.*

Definition A.9. *Eine Familie* $\{h_k\}_{k\in\mathbb{N}}$ *von Funktionen* $h_k : \{0,1\}^* \to \{0,1\}^k$ *heißt* Familie von kollisionsresistenten Hashfunktionen*, wenn zum einen die Funktion*

$$\begin{aligned} h: \ \{1\}^* \times \{0,1\}^* &\longrightarrow \{0,1\}^* \\ (1^k, x) &\mapsto h_k(x) \end{aligned}$$

berechenbar durch einen PPT-Algorithmus A_h *ist, und andererseits für jeden PPT-Algorithmus* B *die Funktion*

$$\mathsf{Pr}\left[(x_1, x_2) \leftarrow B(1^k) : \ x_1 \neq x_2 \wedge h_k(x_1) = h_k(x_2)\right]$$

vernachlässigbar in k *ist.*

Definition A.10. *Zwei Familien* $\{x_k\}_{k\in\mathbb{N}}$ *und* $\{y_k\}_{k\in\mathbb{N}}$ *von Zufallsvariablen* x_k, y_k *über* $\{0,1\}^*$ *heißen* polynomial ununterscheidbar*, falls für alle Algorithmen* A*, welche PPT in ihrer ersten Eingabe sind, die Funktion*

$$\left|\mathsf{Pr}\left[A(1^k, x_k) = 1\right] - \mathsf{Pr}\left[A(1^k, y_k) = 1\right]\right|$$

vernachlässigbar in k *ist.*

Literaturverzeichnis

[1] Aiello, William ; Bellovin, Steven M. ; Blaze, Matt ; Canetti, Ran ; Ionnadis, John ; Keromytis, Angelos D. ; Reingold, Omer: Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols. In: *9th ACM Conference on Computer and Communications Security, Proceedings of CCS 2002*, ACM Press, 2002, S. 48–58

[2] Anshel, Iris ; Anshel, Michael ; Fisher, Benji ; Goldfeld, Dorian: New Key Agreement Protocols in Braid Group Cryptography. In: Naccache, David (Hrsg.): *Topics in Cryptology, Proceedings of CT-RSA 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2020), S. 13–27

[3] Anshel, Iris ; Anshel, Michael ; Goldfeld, Dorian: An Algebraic Method for Public-Key Cryptography. In: *Mathematical Research Letters* 6 (1999), S. 287–291

[4] Artin, Emil: Theorie der Zöpfe. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 4 (1925), S. 47–72

[5] Backes, Michael: *Cryptographically Sound Analysis of Security Protocols*, Universität des Saarlandes, Dissertation, 2002

[6] Backes, Michael: Unifying Simulatability Definitions in Cryptographic Systems under Different Timing Assumptions. In: Amadio, Roberto (Hrsg.) ; Lugiez, Denis (Hrsg.): *Concurrency Theory, Proceedings of CONCUR 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2761), S. 350–365

[7] Backes, Michael ; Hofheinz, Dennis: How to Break and Repair a Universally Composable Signature Functionality. In: Zhang, Kan (Hrsg.) ; Zheng, Yuliang (Hrsg.): *Information Security, Proceedings of ISC 2004*, Springer-Verlag, 2004 (Lecture Notes in Computer Science 3225), S. 61–72

[8] Backes, Michael ; Hofheinz, Dennis ; Müller-Quade, Jörn ; Unruh, Dominique: On Fairness in Simulatability-based Cryptographic Systems. In: *3rd ACM Workshop on Formal Methods in Security Engineering, Proceedings of FMSE 2005*, ACM Press, 2005. – Erscheint

[9] Backes, Michael ; Pfitzmann, Birgit: Intransitive Non-Interference for Cryptographic Purposes. In: *IEEE Symposium on Security and Privacy, Proceedings of SSP 2003*, IEEE Computer Society, 2003, S. 140–152

[10] Backes, Michael ; Pfitzmann, Birgit ; Steiner, Michael ; Waidner, Michael: Polynomial Fairness and Liveness. In: *15th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2002*, IEEE Computer Society, 2002, S. 160–174

[11] Backes, Michael ; Pfitzmann, Birgit ; Waidner, Michael: *Secure Asynchronous Reactive Systems*. IACR ePrint Archive. März 2004

[12] Banks, William D. ; Lieman, Daniel ; Shparlinski, Igor E. ; To, Van T.: Cryptographic Applications of Sparse Polynomials over Finite Rings. In: Won, Dongho (Hrsg.): *Information Security and Cryptology, Proceedings of ICISC 2000*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2015), S. 206–220

[13] Bao, Feng ; Beth, Thomas ; Deng, Robert H. ; Schnorr, Claus ; Steinwandt, Rainer ; Wu, Hongjun: Cryptanalysis of SPIFIII and ENROOTII. In: *Security through Analysis and Verification*. Schloss Dagstuhl, 2000 (Dagstuhl Seminar Report 294), S. 7. – Abstract

[14] Bao, Feng ; Deng, Robert H. ; Schnorr, Claus ; Steinwandt, Rainer ; Wu, Hongjun: Cryptanalysis of Two Sparse Polynomial Based Public Key Cryptosystems. In: Kim, Kwangjo (Hrsg.): *Public Key Cryptography, Proceedings of PKC 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 1992), S. 153–164

[15] Barkee, Boo ; Can, Dee C. ; Ecks, Julia ; Moriarty, Theo ; Ree, R. F.: Why you cannot even hope to use Gröbner Bases in Public Key Cryptography: An open letter to a scientist who failed and a challenge to those who have not yet failed. In: *Journal of Symbolic Computation* 18 (1994), Dezember, Nr. 6, S. 497–501

[16] Bellare, Mihir ; Desai, Anand ; Jokipii, Eron ; Rogaway, Phillip: A Concrete Security Treatment of Symmetric Encryption. In: *38th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 1997*, IEEE Computer Society, 1997, S. 394–403

[17] Bellare, Mihir ; Desai, Anand ; Pointcheval, David ; Rogaway, Phillip: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, Hugo (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '98*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1462), S. 26–45

[18] Bellare, Mihir ; Rogaway, Phillip: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: *1st ACM Conference on Computer and Communications Security, Proceedings of CCS 1993*, ACM Press, 1993, S. 62–73

[19] Bellare, Mihir ; Rogaway, Phillip: Optimal Asymmetric Encryption—How to Encrypt with RSA. In: Santis, Alfredo de (Hrsg.): *Advances in Cryptology, Proceedings of EUROCRYPT '94*, Springer-Verlag, 1995 (Lecture Notes in Computer Science 950), S. 92–111

[20] Birman, Joan S.: *Braids, Links, And Mapping Class Groups.* Princeton University Press, 1974 (Annals of Mathematics Studies 82)

[21] Birman, Joan S. ; Ko, Ki H. ; Lee, Sang J.: The Infimum, Supremum, and Geodesic Length of a Braid Conjugacy Class. In: *Advances in Mathematics* 164 (2001), S. 41–56

[22] Blake-Wilson, Simon ; Menezes, Alfred: Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol. In: Imai, Hideki (Hrsg.) ; Zheng, Yuliang (Hrsg.): *Public Key Cryptography, Proceedings of PKC '99*, Springer-Verlag, 1999 (Lecture Notes in Computer Science 1560), S. 154–170

[23] Bleichenbacher, Daniel: Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, Hugo (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '98*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1462), S. 1–12

[24] Blum, Manuel: Coin Flipping by Telephone. In: Gersho, Allen (Hrsg.): *Advances in Cryptology, A report on CRYPTO 81*, University of California, Electrical and Computer Engineering, 1982 (ECE Report 82-04), S. 11–15

[25] Bohli, Jens-Matthias ; González Vasco, María I. ; Martínez, Consuelo ; Steinwandt, Rainer: *Weak Keys in MST_1*. IACR ePrint Archive. Mai 2002

[26] Canetti, Ran: Security and Composition of Multi-party Cryptographic Protocols. In: *Journal of Cryptology* 3 (2000), Nr. 1, S. 143–202

[27] Canetti, Ran: *Universally Composable Security: A New Paradigm for Cryptographic Protocols.* Electronic Colloquium on Computational Complexity. Oktober 2001. – Volle Fassung von [28]

[28] Canetti, Ran: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, IEEE Computer Society, 2001, S. 136–145

[29] Canetti, Ran: *On Universally Composable Notions of Security for Signature, Certification and Authentication.* IACR ePrint Archive. November 2003

[30] Canetti, Ran: *Universally Composable Signature, Certification and Authentication*. IACR ePrint Archive. Juni 2004

[31] Canetti, Ran: Universally Composable Signature, Certification and Authentication. In: *17th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2004*, IEEE Computer Society, 2004, S. 219–235

[32] Canetti, Ran: *Universally Composable Signature, Certification and Authentication*. IACR ePrint Archive. August 2004

[33] Canetti, Ran ; Feige, Uri ; Goldreich, Oded ; Naor, Moni: Adaptively Secure Multi-party Computation. In: *Twenty-Eighth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1995*, ACM Press, 1996, S. 639–648

[34] Canetti, Ran ; Fischlin, Marc: Universally Composable Commitments. In: Kilian, Joe (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2139), S. 19–40

[35] Canetti, Ran ; Krawczyk, Hugo: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, Birgit (Hrsg.): *Advances in Cryptology, Proceedings of EUROCRYPT 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2045), S. 453–474

[36] Canetti, Ran ; Krawczyk, Hugo: Universally Composable Notions of Key Exchange and Secure Channels. In: Knudsen, Lars R. (Hrsg.): *Advances in Cryptology, Proceedings of EUROCRYPT 2002*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2332), S. 337–351

[37] Canetti, Ran ; Krawczyk, Hugo ; Nielsen, Jesper B.: Relaxing Chosen-Ciphertext Security. In: Boneh, Dan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2729), S. 565–582

[38] Canetti, Ran ; Lindell, Yehuda ; Ostrovsky, Rafail ; Sahai, Amit: Universally Composable Two-Party and Multi-party Secure Computation. In: *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, ACM Press, 2002, S. 494–503. – Extended abstract

[39] Canetti, Ran ; Lindell, Yehuda ; Ostrovsky, Rafail ; Sahai, Amit: *Universally Composable Two-Party and Multi-Party Secure Computation*. IACR ePrint Archive. Juli 2003. – Volle und überarbeitete Fassung von [38]

[40] Canetti, Ran ; Rabin, Tal: *Universal Composition with Joint State*. IACR ePrint Archive. April 2002. – Volle Fassung von [42]

[41] Canetti, Ran ; Rabin, Tal: Universal Composition with Joint State. In: Boneh, Dan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2729), S. 265–281

[42] Canetti, Ran ; Rabin, Tal: Universal Composition with Joint State. In: Boneh, Dan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2729), S. 265–281

[43] Cha, Jae C.: *CBraid: a C++ library for computations in braid groups*. Online verfügbar unter `http://knot.kaist.ac.kr/~jccha/cbraid/`. Dezember 2001

[44] Cha, Jae C. ; Ko, Ki H. ; Lee, Sang J. ; Han, Jae W. ; Cheon, Jung H.: An Efficient Implementation of Braid Groups. In: Boyd, Colin (Hrsg.): *Advances in Cryptology, Proceedings of ASIACRYPT 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2248), S. 144–156

[45] Cheon, Jung H. ; Jun, Byungheup: A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem. In: Boneh, Dan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2729), S. 212–225

[46] Courtois, Nicolas T.: The security of Hidden Field Equations (HFE). In: Naccache, David (Hrsg.): *Topics in Cryptology, Proceedings of CT-RSA 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2020), S. 266–281

[47] Cramer, Ronald ; Damgård, Ivan: Secure Signature Schemes based on Interactive Protocols. In: Coppersmith, Don (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '95*, Springer-Verlag, 1995 (Lecture Notes in Computer Science 963), S. 297–310

[48] Cramer, Ronald ; Damgård, Ivan: New Generation of Secure and Practical RSA-Based Signatures. In: Koblitz, Neal (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '96*, Springer-Verlag, 1996 (Lecture Notes in Computer Science 1109), S. 173–185

[49] Cramer, Ronald ; Shoup, Victor: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, Hugo (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '98*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1462), S. 13–25

[50] Cramer, Ronald ; Shoup, Victor: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Chosen Ciphertext Attack. In: *SIAM Journal on Computing* 33 (2003), Nr. 1, S. 167–226

[51] Crépeau, Claude ; Graaf, Jeroen van de ; Tapp, Alain: Committed Oblivious Transfer and Private Multi-Party Computation. In: Coppersmith, Don (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '95*, Springer-Verlag, 1995 (Lecture Notes in Computer Science 963), S. 110–123

[52] Damgård, Ivan B. ; Pedersen, Torben P. ; Pfitzmann, Birgit: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. In: Stinson, Douglas R. (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '93*, Springer-Verlag, 1994 (Lecture Notes in Computer Science 773), S. 250–265

[53] Dehornoy, Patrick: Braid-based cryptography. In: Myasnikov, Alexei G. (Hrsg.): *Group Theory, Statistics, and Cryptography*, ACM Press, 2004 (Contemporary Mathematics 360), S. 5–33

[54] Diffie, Whitfield ; Hellman, Martin E.: New Directions in Cryptography. In: *IEEE Transactions on Information Theory* IT-22 (1976), November, Nr. 6, S. 644–654

[55] Dolev, Danny ; Dwork, Cynthia ; Naor, Moni: Non-Malleable Cryptography. In: *Twenty-Third Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1991*, ACM Press, 1991, S. 542–552. – Extended abstract

[56] National Institute of Standards and Technology (Hrsg.): *Digital Signature Standard*. Oktober 2001. – Federal Information Processing Standard Publication 186-2

[57] Dwork, Cynthia ; Naor, Moni: An Efficient Existentially Unforgeable Signature Scheme and its Applications. In: *Journal of Cryptology* 11 (1998), Nr. 3, S. 187–208

[58] ElGamal, Taher: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakley, G. R. (Hrsg.) ; Chaum, David (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '84*, Springer-Verlag, 1985 (Lecture Notes in Computer Science 196), S. 10–18

[59] Elrifai, Elsayed A. ; Morton, H. R.: Algorithms for Positive Braids. In: *Quarterly Journal of Mathematics* 45 (1994), S. 479–497

[60] Faugère, Jean-Charles ; Joux, Antoine: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, Dan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2729), S. 44–60

[61] Fellows, Michael ; Koblitz, Neal: Combinatorial Cryptosystems Galore! In: Mullen, Gary L. (Hrsg.) ; Shiue, Jau-Shyong (Hrsg.): *Finite Fields:*

Theory, Applications and Algorithms, Proceedings of Second International Conference on Finite Fields, ACM Press, 1994 (Contemporary Mathematics 168), S. 51–61

[62] Fujisaki, Eiichiro ; Okamoto, Tatsuaki: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, Hideki (Hrsg.) ; Zheng, Yuliang (Hrsg.): *Public Key Cryptography, Proceedings of PKC '99*, Springer-Verlag, 1999 (Lecture Notes in Computer Science 1560), S. 53–68

[63] Fujisaki, Eiichiro ; Okamoto, Tatsuaki ; Pointcheval, David ; Stern, Jacques: RSA-OAEP is Secure under the RSA Assumption. In: Kilian, Joe (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2139), S. 260–274

[64] Garside, Frank A.: The Braid Group and Other Groups. In: *Quarterly Journal of Mathematics* 20 (1969), S. 235–254

[65] Garzon, Max ; Zalcstein, Yechezkel: The complexity of Grigorchuk groups with application to cryptography. In: *Theoretical Computer Science* 88 (1991), Nr. 1, S. 83–98

[66] Gebhardt, Volker: *A New Approach to the Conjugacy Problem in Garside Groups*. lanl.arXiv.org ePrint Archive. Juni 2003

[67] Geiselmann, Willi ; Steinwandt, Rainer: *A Key Substitution Attack on SFLASH^{v3}*. IACR ePrint Archive. November 2003

[68] Goldreich, Oded: Two Remarks Concerning The Goldwasser-Micali-Rivest Signature Scheme. In: Odlyzko, Andrew M. (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '86*, Springer-Verlag, 1987 (Lecture Notes in Computer Science 263), S. 104–110

[69] Goldreich, Oded: A Uniform-Complexity Treatment of Encryption and Zero-Knowledge. In: *Journal of Cryptology* 6 (1993), Nr. 1, S. 21–53

[70] Goldreich, Oded: *Foundations of Cryptography – Volume 1 (Basic Tools)*. Cambridge University Press, August 2001

[71] Goldreich, Oded: *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, Mai 2004

[72] Goldreich, Oded ; Krawczyk, Hugo: On the Composition of Zero-Knowledge Proof Systems. In: Paterson, Mike (Hrsg.): *Automata, Languages and Programming, 17th International Colloquium, Proceedings of ICALP90*, Springer-Verlag, 1990 (Lecture Notes in Computer Science 443), S. 268–282

[73] Goldreich, Oded ; Micali, Silvio ; Wigderson, Avi: How to Play any Mental Game—A Completeness Theorem for Protocols With Honest Majority. In: *Nineteenth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1987*, ACM Press, 1987, S. 218–229. – Extended abstract

[74] Goldwasser, Shafi ; Levin, Leonid A.: Fair Computation of General Functions in Presence of Immoral Majority. In: Menezes, Alfred (Hrsg.) ; Vanstone, Scott A. (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '90*, Springer-Verlag, 1991 (Lecture Notes in Computer Science 537), S. 77–93

[75] Goldwasser, Shafi ; Micali, Silvio: Probabilistic Encryption. In: *Journal of Computer and System Sciences* 28 (1984), April, Nr. 2, S. 270–299

[76] Goldwasser, Shafi ; Micali, Silvio ; Rackoff, Charles: The Knowledge Complexity of Interactive Proof Systems. In: *SIAM Journal on Computing* 18 (1989), Nr. 1, S. 186–208

[77] Goldwasser, Shafi ; Micali, Silvio ; Rivest, Ronald L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. In: *SIAM Journal on Computing* 17 (1988), Nr. 2, S. 281–308

[78] González Vasco, María I. ; Hofheinz, Dennis ; Martínez, Consuelo ; Steinwandt, Rainer: On the security of two public key cryptosystems using non-abelian groups. In: *Designs, Codes and Cryptography* 32 (2004), Nr. 1–3, S. 207–216

[79] Grant, David ; Krastev, Kate ; Lieman, Daniel ; Shparlinski, Igor: A Public Key Cryptosystem Based On Sparse Polynomials. In: Buchmann, Johannes (Hrsg.) ; Høholdt, Tom (Hrsg.) ; Stichtenoth, Henning (Hrsg.) ; Tapia-Recillas, Horacio (Hrsg.): *International Conference on Coding Theory, Cryptography and Related Areas, Proceedings of ICCC 1998*, 2000, S. 114–121

[80] Hoffstein, Jeffrey ; Pipher, Jill ; Silverman, Joseph H.: NTRU: A Ring-Based Public Key Cryptosystem. In: Buhler, Joe (Hrsg.): *Third International Symposium on Algorithmic Number Theory, Proceedings of ANTS 1997*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1423), S. 267–288

[81] Hofheinz, Dennis: *Angriffe auf das Public-Key-System Polly Cracker*. Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe. 2001. – Studienarbeit

[82] Hofheinz, Dennis ; Müller-Quade, Jörn: A Synchronous Model for Multi-Party Computation and the Incompleteness of Oblivious Transfer. In: Sabelfeld, Andrei (Hrsg.): *LICS '04 and ICALP '04 Affiliated Workshop on*

Foundations of Computer Security, Proceedings of FCS 2004, Turku Centre for Computer Science, 2004 (TUCS General Publications 31), S. 117–130

[83] Hofheinz, Dennis ; Müller-Quade, Jörn: Universally Composable Commitments Using Random Oracles. In: Naor, Moni (Hrsg.): *Theory of Cryptography, Proceedings of TCC 2004*, Springer-Verlag, 2004 (Lecture Notes in Computer Science 2951), S. 58–76

[84] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: Initiator-Resilient Universally Composable Key Exchange. In: Snekkenes, Einar (Hrsg.) ; Gollmann, Dieter (Hrsg.): *Computer Security, Proceedings of ESORICS 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2808), S. 61–84

[85] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: *On Modeling IND-CCA Security in Cryptographic Protocols*. IACR ePrint Archive. Februar 2003. – Volle Fassung von [86]

[86] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: On Modeling IND-CCA Security in Cryptographic Protocols. In: *4th Central European Conference on Cryptology, Proceedings of WARTACRYPT 2004*, 2004, S. 47–49. – Extended abstract, volle Fassung ist [85]

[87] Hofheinz, Dennis ; Müller-Quade, Jörn ; Unruh, Dominique: Polynomial Runtime in Simulatability Definitions. In: *18th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2005*, IEEE Computer Society, 2005. – Erscheint

[88] Hofheinz, Dennis ; Steinwandt, Rainer: A "Differential" Attack on Polly Cracker. In: *IEEE International Symposium on Information Theory, Proceedings of ISIT 2002*, IEEE Computer Society, 2002, S. 211. – Extended abstract

[89] Hofheinz, Dennis ; Steinwandt, Rainer: A Practical Attack on Some Braid Group Based Cryptographic Primitives. In: Desmedt, Yvo (Hrsg.): *Public Key Cryptography, Proceedings of PKC 2003*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2567), S. 187–198

[90] Hofheinz, Dennis ; Unruh, Dominique: Comparing Two Notions of Simulatability. In: Kilian, Joe (Hrsg.): *Theory of Cryptography, Proceedings of TCC 2005*, Springer-Verlag, 2005 (Lecture Notes in Computer Science 3378), S. 86–103

[91] Hofheinz, Dennis ; Unruh, Dominique: On the Notion of Statistical Security in Simulatability Definitions. In: *Information Security, Proceedings of ISC 2005*, Springer-Verlag, 2005 (Lecture Notes in Computer Science). – Erscheint

[92] Hughes, Jim: A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem. In: Batten, Lynn (Hrsg.) ; Seberry, Jennifer (Hrsg.): *Information Security and Privacy, Proceedings of ACISP 2002*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2384), S. 176–189

[93] IPSEC Working Group (Hrsg.): *Internet Key Exchange (IKEv2) Protocol*. März 2003

[94] Kilian, Joe: Founding cryptography on oblivious transfer. In: *Twentieth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1988*, ACM Press, 1988, S. 20–31

[95] Kipnis, Aviad ; Shamir, Adi: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, Michael (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '99*, Springer-Verlag, 1999 (Lecture Notes in Computer Science 1666), S. 19–30

[96] Ko, Ki H. ; Cho, Mi S. ; Choi, Doo H. ; Han, Jae W.: *Provably Secure Public-key Encryption Scheme Based on Braid Groups*. 2001. – Unveröffentlicht

[97] Ko, Ki H. ; Lee, Sang J. ; Cheon, Jung H. ; Han, Jae W. ; Kang, Ju-sung ; Park, Choonsik: New Public-Key Cryptosystem Using Braid Groups. In: Bellare, Mihir (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2000*, Springer-Verlag, 2000 (Lecture Notes in Computer Science 1880), S. 166–183

[98] Koblitz, Neal: *Algorithms and Computations in Mathematics*. Bd. 3: *Algebraic Aspects of Cryptography*. Springer-Verlag, 1998

[99] Kocher, Paul C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In: Koblitz, Neal (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '96*, Springer-Verlag, 1996 (Lecture Notes in Computer Science 1109), S. 104–113

[100] Kocher, Paul C. ; Jaffe, Joshua ; Jun, Benjamin: Differential Power Analysis. In: Wiener, Michael (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '99*, Springer-Verlag, 1999 (Lecture Notes in Computer Science 1666), S. 388–397

[101] Lee, Eonkyung ; Park, Je H.: Cryptanalysis of the Public-Key Encryption Based on Braid Groups. In: Biham, Eli (Hrsg.): *Advances in Cryptology, Proceedings of EUROCRYPT 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2656), S. 477–490

[102] Lee, Sang J. ; Lee, Eonkyung: Potential Weaknesses of the Commutator Key Agreement Protocol Based On Braid Groups. In: Knudsen, Lars R. (Hrsg.):

Advances in Cryptology, Proceedings of EUROCRYPT 2002, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2332), S. 14–28

[103] Levy-dit-Vehel, Françoise ; Perret, Ludovic: A Polly Cracker system based on Satisfiability. In: *Progress in Computer Science and Applied Logic* 23 (2004), S. 177–192

[104] Luby, Michael: *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996 (Princeton Computer Science Notes)

[105] Ly, Le V.: *Polly Two – A Public-Key Cryptosystem based on Polly Cracker*, Ruhr-Universität Bochum, Dissertation, 2002

[106] Magliveras, S. S. ; Stinson, D. R. ; Trung, Tran V.: New approaches to designing public key cryptosystems using one-way functions and trapdoors in finite groups. In: *Journal of Cryptology* 15 (2002), Nr. 4, S. 285–297

[107] Maurer, Ueli: Secure Multi-party Computation Made Simple. In: Cimato, Stelvio (Hrsg.) ; Galdi, Clemente (Hrsg.) ; Persiano, Guiseppe (Hrsg.): *IEEE Symposium on Security and Privacy, Proceedings of SSP 2002*, IEEE Computer Society, 2002, S. 14–28

[108] Menezes, Alfred ; Smart, Nigel: Security of Signature Schemes in a Multi-User Setting. In: *Designs, Codes and Cryptography* 33 (2004), November, Nr. 3, S. 261–274

[109] Naor, Moni ; Yung, Moti: Universal One-Way Hash Functions and their Cryptographic Applications. In: *Twenty-First Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1989*, ACM Press, 1989, S. 33–43

[110] Naor, Moni ; Yung, Moti: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: *Twenty-Second Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1990*, ACM Press, 1990, S. 427–437

[111] Paeng, Seong-Hun ; Ha, Kil-Chan ; Kim, Jae H.: *Improved public key cryptosystem using finite non abelian groups*. IACR ePrint Archive. August 2001

[112] Paeng, Seong-Hun ; Ha, Kil-Chan ; Kim, Jae H.: New Public Key Cryptosystem Using Finite Non Abelian Groups. In: Kilian, Joe (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2001*, Springer-Verlag, 2001 (Lecture Notes in Computer Science 2139), S. 470–485

[113] Patarin, Jacques: Hidden Fields Equations (HFE) and Isomorphism of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, Ueli (Hrsg.): *Advances in Cryptology, Proceedings of EUROCRYPT '96*, Springer-Verlag, 1996 (Lecture Notes in Computer Science 1070), S. 33–48

[114] Patarin, Jacques ; Goubin, Louis ; Courtois, Nicolas: C^*_{+-} and HM: Variations around two schemes of T. Matsumoto and H. Imai. In: Ohta, Kazuo (Hrsg.) ; Pei, Dingyi (Hrsg.): *Advances in Cryptology, Proceedings of ASIACRYPT '98*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1514), S. 35–49

[115] Pfitzmann, Birgit ; Schunter, Matthias ; Waidner, Michael: Secure Reactive Systems / IBM Zurich Research Laboratory. 2000 (RZ 3206). – Forschungsbericht

[116] Pfitzmann, Birgit ; Waidner, Michael: Composition and Integrity Preservation of Secure Reactive Systems. In: *7th ACM Conference on Computer and Communications Security, Proceedings of CCS 2000*, ACM Press, 2000, S. 245–254

[117] Pfitzmann, Birgit ; Waidner, Michael: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: *IEEE Symposium on Security and Privacy, Proceedings of SSP 2001*, IEEE Computer Society, 2001, S. 184–200

[118] RSA Laboratories (Hrsg.): *PKCS #1: RSA Encryption Standard, Version 1.5*. 1993

[119] RSA Laboratories (Hrsg.): *PKCS #1: RSA Cryptography Standard, Version 2.1*. 2002

[120] Rabin, Michael O.: How to exchange secrets by Oblivious Transfer / Aiken Computation Laboratory, Harvard University. 1981 (TR-81). – Forschungsbericht

[121] Rackoff, Charles ; Simon, Daniel R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, Joan (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO '91*, Springer-Verlag, 1992 (Lecture Notes in Computer Science 576), S. 433–444

[122] Rivest, Ronald L. ; Shamir, Adi ; Adleman, Leonard M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the ACM* 21 (1978), Februar, Nr. 2, S. 120–126

[123] Rompel, John: One-Way Functions are Necessary and Sufficient for Secure Signatures. In: *Twenty-Second Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1990*, ACM Press, 1990, S. 387–394

[124] Rosa, Tomáš: *Key-collisions in (EC)DSA: Attacking Non-repudiation*. IACR ePrint Archive. August 2002

[125] Shor, Peter W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: *35th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 1994*, IEEE Computer Society, 1994, S. 124–134

[126] Shoup, Victor: Why Chosen Ciphertext Security Matters / IBM Zurich Research Laboratory. November 1998 (RZ 3076). – Forschungsbericht

[127] Shoup, Victor: *On Formal Models for Secure Key Exchange*. IACR ePrint Archive. November 1999

[128] Sibert, Hervé: *Algorithmique des groupes de tresses*, Université de Caen, Dissertation, 2003

[129] Netscape Communications (Hrsg.): *SSL 3.0 Specification*. November 1996

[130] Steiner, Michael: *Secure Group Key Agreement*, Universität des Saarlandes, Dissertation, 2002

[131] Steinwandt, Rainer ; Geiselmann, Willi: Cryptanalysis of Polly Cracker. In: *IEEE Transactions on Information Theory* 48 (2002), Nr. 11, S. 2990–2991

[132] Stern, Jacques ; Pointcheval, David ; Malone-Lee, John ; Smart, Nigel P.: Flaws in Applying Proof Methodologies to Signature Schemes. In: Yung, Moti (Hrsg.): *Advances in Cryptology, Proceedings of CRYPTO 2002*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2442), S. 93–110

[133] Tan, Chik-How: Key Substitution Attacks on Some Provably Secure Signature Schemes. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E87-A (2004), Januar, Nr. 1

[134] Tobias, Christian: Security Analysis of the MOR Cryptosystem. In: Desmedt, Yvo (Hrsg.): *Public Key Cryptography, Proceedings of PKC 2003*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2567), S. 175–186

[135] Tsiounis, Yiannis ; Yung, Moti: On the Security of ElGamal Based Encryption. In: Imai, Hideki (Hrsg.) ; Zheng, Yuliang (Hrsg.): *Public Key Cryptography, Proceedings of PKC '98*, Springer-Verlag, 1998 (Lecture Notes in Computer Science 1431), S. 117–134

[136] Unruh, Dominique: *Formale Sicherheit in der Quantenkryptologie*. Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe. 2002. – Studienarbeit

[137] Watanabe, Yodai ; Shikata, Junji ; Imai, Hideki: Equivalence between Semantic Security and Indistinguishability against Chosen Ciphertext Attacks. In: Desmedt, Yvo (Hrsg.): *Public Key Cryptography, Proceedings of*

PKC 2003, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2567), S. 71–84

Lebenslauf

21. Februar 1979	geboren in Siegen
	Eltern: Barbara Hofheinz, geb. Lippelt und Rudi Hofheinz
1985 bis 1988	Besuch der Grundschule in Niederroßbach
1988 bis 1997	Besuch des Wilhelm-von-Oranien-Gymnasiums in Dillenburg
Juni 1997	Abschluß mit Abitur
1997 bis 1998	Zivildienst an der Otfried-Preußler-Schule für praktisch Bildbare in Dillenburg
1998 bis 2002	Studium der Informatik an der Universität Fridericiana zu Karlsruhe
März 2002	Abschluß mit Diplom
seit April 2002	Mitarbeiter am Institut für Algorithmen und Kognitive Systeme der Universität Fridericiana zu Karlsruhe

Eigene Arbeiten

Artikel in Zeitschriften

[1] González Vasco, María I. ; Hofheinz, Dennis ; Martínez, Consuelo ; Steinwandt, Rainer: On the security of two public key cryptosystems using non-abelian groups. In: *Designs, Codes and Cryptography* 32 (2004), Nr. 1–3, S. 207–216

[2] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: On Modeling IND-CCA Security in Cryptographic Protocols. In: *Tatra Mountains Mathematical Publications* (2005). – Erscheint

Beiträge zu Konferenzen/Workshops

[3] Hofheinz, Dennis ; Steinwandt, Rainer: A "Differential" Attack on Polly Cracker. In: *IEEE International Symposium on Information Theory, Proceedings of ISIT 2002*, IEEE Computer Society, 2002, S. 211. – Extended abstract

[4] Hofheinz, Dennis ; Steinwandt, Rainer: A Practical Attack on Some Braid Group Based Cryptographic Primitives. In: Desmedt, Yvo (Hrsg.): *Public Key Cryptography, Proceedings of PKC 2003*, Springer-Verlag, 2002 (Lecture Notes in Computer Science 2567), S. 187–198

[5] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: Initiator-Resilient Universally Composable Key Exchange. In: Snekkenes, Einar (Hrsg.) ; Gollmann, Dieter (Hrsg.): *Computer Security, Proceedings of ESORICS 2003*, Springer-Verlag, 2003 (Lecture Notes in Computer Science 2808), S. 61–84

[6] Hofheinz, Dennis ; Müller-Quade, Jörn: Universally Composable Commitments Using Random Oracles. In: Naor, Moni (Hrsg.): *Theory of Cryptography, Proceedings of TCC 2004*, Springer-Verlag, 2004 (Lecture Notes in Computer Science 2951), S. 58–76

[7] Hofheinz, Dennis ; Müller-Quade, Jörn ; Steinwandt, Rainer: On Modeling IND-CCA Security in Cryptographic Protocols. In: *4th Central European Conference on Cryptology, Proceedings of WARTACRYPT 2004*, 2004, S. 47–49. – Extended abstract

[8] Hofheinz, Dennis ; Müller-Quade, Jörn: A Synchronous Model for Multi-Party Computation and the Incompleteness of Oblivious Transfer. In: Sabelfeld, Andrei (Hrsg.): *LICS '04 and ICALP '04 Affiliated Workshop on Foundations of Computer Security, Proceedings of FCS 2004*, Turku Centre for Computer Science, 2004 (TUCS General Publications 31), S. 117–130

[9] Backes, Michael ; Hofheinz, Dennis: How to Break and Repair a Universally Composable Signature Functionality. In: Zhang, Kan (Hrsg.) ; Zheng, Yuliang (Hrsg.): *Information Security, Proceedings of ISC 2004*, Springer-Verlag, 2004 (Lecture Notes in Computer Science 3225), S. 61–72

[10] Hofheinz, Dennis ; Unruh, Dominique: Comparing Two Notions of Simulatability. In: Kilian, Joe (Hrsg.): *Theory of Cryptography, Proceedings of TCC 2005*, Springer-Verlag, 2005 (Lecture Notes in Computer Science 3378), S. 86–103

[11] Hofheinz, Dennis ; Müller-Quade, Jörn ; Unruh, Dominique: Universally Composable Zero-Knowledge Arguments and Commitments from Signature Cards. In: *5th Central European Conference on Cryptology, Proceedings of MoraviaCrypt 2005*, 2005. – Extended abstract

[12] Hofheinz, Dennis ; Müller-Quade, Jörn ; Unruh, Dominique: Polynomial Runtime in Simulatability Definitions. In: *18th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2005*, IEEE Computer Society, 2005. – Erscheint

[13] Hofheinz, Dennis ; Unruh, Dominique: On the Notion of Statistical Security in Simulatability Definitions. In: *Information Security, Proceedings of ISC 2005*, Springer-Verlag, 2005 (Lecture Notes in Computer Science). – Erscheint

[14] Backes, Michael ; Hofheinz, Dennis ; Müller-Quade, Jörn ; Unruh, Dominique: On Fairness in Simulatability-based Cryptographic Systems. In: *3rd ACM Workshop on Formal Methods in Security Engineering, Proceedings of FMSE 2005*, ACM Press, 2005. – Erscheint

Eingeladene Vorträge

[15] Hofheinz, Dennis: *From hard problems to cryptographic security.* Canadian Mathematical Society Winter 2004 Meeting, eingeladener Vortrag. Dezember 2004

[16] Hofheinz, Dennis: *An attack on a group-based cryptographic scheme.* 2nd Joint Meeting of AMS, DMV, and ÖMV, eingeladener Vortrag. Juni 2005

Sonstiges

[17] Hofheinz, Dennis: *Angriffe auf das Public-Key-System Polly Cracker.* Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe. 2001. – Studienarbeit

[18] Hofheinz, Dennis: *Ein Seitenkanalangriff auf das Signaturverfahren QUARTZ.* Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe. 2002. – Diplomarbeit

[19] Hofheinz, Dennis ; Müller-Quade, Jörn: *A Paradox of Quantum Universal Composability.* Posterbeitrag zum 4th European QIPC Workshop. 2003

Abkürzungsverzeichnis

BHC_C	Uneingeschränkt bindende Protokollkonstruktion für String Commitment, Seite 120
CRS	Common reference string, Hilfsannahme bei Mehrparteienberechnungen
DDH-Problem	Decisional-Diffie-Hellman-Problem, Seite 94
$\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$	Diffie-Hellman-basiertes Schlüsselaustauschprotokoll, Seite 95
EF-CMA	Existential unforgeability under adaptive chosen-message attacks, Sicherheitsforderung an ein Public-Key-Signatursystem, Seite 71
$\mathcal{F}_{\mathrm{AUTH}}$	Idealisierung einer authentifizierten Nachrichtenübertragung aus [27]
$\mathcal{F}_{\mathrm{BC}}^{\ell}$	Idealisierung eines Broadcast-Kanals, Seite 142
$\mathcal{F}_{\mathrm{BSCOM}}$	Idealisierung eines mehrfach verwendbaren, uneingeschränkt bindenden String Commitments, Seite 119
$\mathcal{F}_{\mathrm{CRS}}$	Idealisierung eines common reference string aus [27, 34]
$\mathcal{F}_{\mathrm{GCOM}}$	Idealisierung eines Bit Commitment mit mehreren Empfängern, Seite 144
$\mathcal{F}_{\mathrm{KE++}}^{(i,j)}$	Hypothetische, extrem verstärkte Idealisierung eines Schlüsselaustauschprotokolls, Seite 104
$\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$	Verstärkte Idealisierung eines Schlüsselaustauschprotokolls, Seite 100
$\mathcal{F}_{\mathrm{KE}}^{(i,j)}$	Korrigierte Idealisierung eines Schlüsselaustauschprotokolls, Seite 93
$\mathcal{F}_{\mathrm{KE}}$	Idealisierung eines Schlüsselaustauschprotokolls aus [36], Seite 85
$\mathcal{F}_{\mathrm{MCOM}}$	Idealisierung eines mehrfach benutzbaren Bit Commitment, Seite 108
$\mathcal{F}_{\text{M-SMT}}^{(i)}$	Angepaßte Idealisierung sicherer Nachrichtenübertragung mit mehreren Sendern, Seite 60
$\mathcal{F}_{\text{M-SMT}}$	Idealisierung sicherer Nachrichtenübertragung mit mehreren Sendern aus [27]

$\mathcal{F}_{\mathrm{OT}}$	Idealisierung von Oblivious Transfer, Seite 143
$\mathcal{F}_{\mathrm{PKE}}^{(i)}$	Korrigierte Idealisierung eines Public-Key-Kryptosystems, Seite 57
$\mathcal{F}_{\mathrm{PKE}}$	Idealisierung eines Public-Key-Kryptosystems, Seite 55
$\mathcal{F}_{\mathrm{RO}}$	Idealisierung eines Random Oracles bzw. einer Hashfunktion, Seite 111
$\mathcal{F}_{\mathrm{RPKE}}$	Abgeschwächte Idealisierung eines Public-Key-Kryptosystems aus [37]
$\mathcal{F}_{\mathrm{SCOM}}$	Idealisierung eines mehrfach benutzbaren String Commitments, Seite 112
$\mathcal{F}_{\mathrm{SFE}}$	Idealisierung einer allgemeinen Funktionsauswertung aus [27]
$\mathcal{F}_{\mathrm{SIG}}^{(i)}$	Korrigierte Idealisierung eines Public-Key-Signatursystems, Seite 78
$\mathcal{F}_{\mathrm{SIG}}$	Idealisierung eines Public-Key-Signatursystems aus [40], Seite 72
$\mathcal{F}_{\mathrm{SMT}}$	Idealisierung sicherer Nachrichtenübertragung aus [27]
HC_C	Uneingeschränkt verbergende Protokollkonstruktion für String Commitment, Seite 113
IKEv2	Schlüsselaustauschprotokoll aus [93]
IND-CCA	Indistinguishability of ciphertexts under adaptive chosen-ciphertext attacks, Sicherheitsforderung an ein Public-Key-Kryptosystem, Seite 62
IND-CPA	Indistinguishability of ciphertexts under chosen-plaintext attacks, Sicherheitsforderung an ein Public-Key-Kryptosystem
ITM	Interaktive Turingmaschine, Seite 41
JFKi, JFKr	Schlüsselaustauschprotokolle aus [1]
MOR	Public-Key-Kryptosystem aus [112, 111]
HFE	Klasse von Public-Key-Kryptosystemen aus [113, 114], basierend auf Hidden Field Equations
MST	Public-Key-Kryptosystem von Magliveras u. a., vgl. [106]
NP	Klasse nichtdeterministisch in Polynomialzeit berechenbarer Entscheidungsprobleme
NTRU	Public-Key-Krypto- und -Signatursystem aus [80]
OAEP	Optimal asymmetric encryption padding für Public-Key-Kryptosysteme aus [19]
$\mathrm{PAD}^{(i,j)}$	Konstruktion für Schlüsselaustauschprotokolle, Seite 102

PKCS #1	Public key cryptography standard der RSA Laboratories, siehe [118]
PKKE$^{(i,j)}$	Public-Key-gestütztes Schlüsselaustauschprotokoll, Seite 97
PPT	Probabilistic polynomial time, Komplexitätsklasse von Algorithmen, Seite 153
ROR-CCA	Real-or-random indistinguishability under adaptive chosen-ciphertext attacks, Sicherheitsforderung an ein Public-Key-Kryptosystem, Seite 61
ROR-CPA	Real-or-random indistinguishability under chosen-plaintext attacks, Sicherheitsforderung an ein Public-Key-Kryptosystem
RSAES-OAEP	RSA encryption scheme with optimal asymmetric encryption padding, siehe [119]
RSA	Public-Key-Krypto- und -Signatursystem von Rivest u. a. aus [122]
SIG-DH	Diffie-Hellman-basiertes Schlüsselaustauschprotokoll aus [36]
SK-Sicherheit	Session-Key-Sicherheit, Sicherheitsforderung an Schlüsselaustauschprotokolle aus [36]
SSL	Secure sockets layer, Verfahren zur Verschlüsselung von Netzwerkkommunikation aus [129]

Stichwortverzeichnis

— **U** —

— **V** —

— **W** —

— **Z** —