Protokollkomposition und Komplexität

Dominique Unruh

Universität des Saarlandes
Saarbrücken, Deutschland
unruh@cs.uni-sb.de

Abstract: Um dem Bedarf der modernen Kryptographie an rigorosen und mächtigen Sicherheitsdefinitionen zu genügen, sind in den letzten Jahren sogenannte simulationsbasierte Sicherheitsmodelle mit Umgebung populär geworden. Diese kommen jedoch in vielen Varianten, und es ist nicht offensichtlich, in welcher Relation diese Varianten stehen, das heißt welche Definitionen welche implizieren und welche Begriffe äquivalent sind.

Wir untersuchen die verschiedenen Varianten und geben eine vollständige Klassifikation, indem wir alle Implikationen und Trennungen zwischen den Begriffen aufzeigen. Die Resultate werden hier nur kurz skizziert, eine vollständige Darstellung findet sich in der Dissertation [Unr07].

1 Einleitung

Eine verbreitete Ansicht besagt, daß es sich bei der Kryptographie im wesentlichen um die Lehre des Ver- und Entschlüsselns geheimer Daten handelt. Noch vor dreißig Jahren wäre diese Ansicht völlig korrekt gewesen, und auch heute bleiben die Verschlüsselungsverfahren ein zentrales Forschungsgebiet der Kryptographie. Darüber hinaus haben sich jedoch eine Vielzahl weiterer Anwendungen aufgetan, wie zum Beispiel digitale Signaturen, elektronische Wahlverfahren, elektronisches Geld, digitales Rechtemanagement, sicherer Schlüsselaustausch (bei dem zwei Parteien einen geheimen Schlüssel über einen öffentlichen Kanal austauschen, ohne daß eine dritte, sie belauschende Partei diesen erfährt), Münzwurf (bei dem zwei Parteien gemeinsam ein Zufallsbit wählen, ohne daß eine der Parteien allein das Bit beeinflussen könnte) u. v. m. Kurz gesagt, die Kryptographie befaßt sich mit Kommunikationsprozessen, bei denen mindestens eine der beteiligten Parteien unehrlich oder zumindest nicht vertrauenswürdig ist. Nun ist es aber nicht genug, neue Protokolle für die oben genannten Aufgabenstellungen zu finden, gerade bei sicherheitskritischen Anwendungen ist es nötig, die Sicherheit dieser Protokolle auch noch zu beweisen. Hierzu aber ist es unumgänglich, zunächst formal zu definieren, was wir unter der Sicherheit eines Protokolls verstehen. Dies mag einfach klingen, doch stellt es sich in vielen Fällen als schwierig heraus, alle gewünschten Sicherheitseigenschaften zu erfassen, ohne dabei wichtige, aber nicht offensichtliche Kriterien außer acht zu lassen.

Eine der zentralen Fragen der exakten, d. h. die Sicherheit von Protokollen *beweisenden* Kryptologie ist daher auch heute noch die Suche nach geeigneten Definitionen der Sicher-

heit. Dabei unterscheiden wir zwischen anwendungsspezifischen Definitionen, wie z. B. der Definition eines sicheren Schlüsselaustauschprotokolls, sowie allgemeinen Definitionen, die in allgemeinster Weise festlegen, was wir unter einem sicheren Protokoll für eine vorgegebene Aufgabenstellung verstehen. Diese allgemeinen Sicherheitsmodelle erreichten ihren (vorläufigen) Höhepunkt mit der Einführung von *simulationsbasierten Sicherheitsmodellen mit Umgebung*, die 2001 unabhängig von Pfitzmann und Waidner [PW01] und von Canetti [Can01] vorgestellt wurden. Diese Modelle haben zwei große Vorteile: Zum einen erlauben sie es, die zu implementierende Aufgabenstellung in einfacher Weise durch Entwurf einer sogenannten *idealen Funktionalität* zu modellieren, daß heißt durch Modellierung einer hypothetischen Maschine, die per Definition genau das tut, was vom Protokoll erwartet wird; und zum anderen ermöglichen diese Modelle sichere Komposition: Es ist möglich, ein komplexes Protokoll zunächst unter Benutzung verschiedener Primitiven zu entwerfen, und dann diese Primitiven durch sie implementierende Protokolle zu ersetzen, was einen modularen Protokollentwurf *und -beweis* ermöglicht.

Insbesondere die sichere Komposition ist in modernen Anwendungen von größter Wichtigkeit. Zum einen ist der Beweis der Sicherheit selbst einfacher kryptographischer Protokolle üblicherweise sehr komplex, so daß ein monolithischer Sicherheitsbeweis eines Systems mittlerer Komplexität praktisch nicht mehr durchführbar ist. Nur die Möglichkeit, mithilfe von Kompositionsresultaten die Komponenten des Systems einzeln zu analysieren, macht größere Systeme der kryptographischen Analyse zugänglich. Zum anderen ist beim Design eines Protokolls oft nicht der genaue Anwendungskontext bekannt. Insbesondere bei Anwendungen im Internet spielen viele verschiedene kryptographische Applikationen zusammen, und es ist nötig, daß nicht nur jede Applikation für sich allein sicher ist, sondern daß auch im ihrem Zusammenspiel keine Sicherheitslücken entstehen.

2 Simulationsbasierte Sicherheit

Bevor wir unsere Ergebnisse vorstellen können, skizzieren wir im folgenden zunächst grob die verwendeten Sicherheitsbegriffe. Um das Verständnis zu erleichtern, diskutieren wir aber zuerst Sicherheitsbegriffe für eine spezielle Anwendung, die *sichere Funktionsauswertung*. Bei dieser ist eine Funktion f in n Argumenten gegeben, und n Parteien wollen $f(x_1, \ldots, x_n)$ berechnen, wobei die i-te Partei die Eingabe x_i beiträgt. Diese Aufgabe wird dadurch erschwert, daß die Parteien einander mißtrauen (oder anders gesagt, wir gehen davon aus, daß einige Parteien unehrlich sind). Wir erwarten intuitiv zwei Sicherheitseigenschaften von einem Protokoll π zur sicheren Auswertung von f: Zum einen wollen wir *Korrektheit*, daß also das Ergebnis der Berechnung von den unehrlichen Parteien nicht verfälscht werden kann. Und zum anderen erwarten wir *Geheimhaltung*, keine Partei soll mehr über die Eingaben der anderen lernen, als sich aus dem Funktionsergebnis $f(x_1, \ldots, x_n)$ folgern läßt. Obwohl es sich hier um zwei scheinbar unabhängige Anforderungen handelt, hat sich herausgestellt, daß es sehr schwierig ist, sie getrennt zu fassen. Stattdessen verfolgt man den sogenannten Simulationsansatz: Wir betrachten zwei Spiele, das *reale* und das *ideale Modell*. Im realen Modell wird das zu untersuchende Protokoll π ausgeführt, wobei ein Angreifer eine Anzahl von Parteien korrumpieren und die Ein- und Ausgaben der kor-

rumpierten Parteien modifizieren bzw. lernen darf. Außerdem – und das ist der wesentliche Unterschied zum idealen Modell – darf der Angreifer das Verhalten der korrumpierten Parteien kontrollieren, das heißt er erhält alle an sie gesandten Nachrichten und darf in ihrem Namen Nachrichten verschicken. Nach Protokollende gibt der Angreifer eine Ausgabe, die o. B. d. A. alle Informationen enthält, die der Angreifer im Protokollverlauf gelernt hat.

Im idealen Modell hingegen gibt es statt eines Angreifers einen Simulator, der zwar auch Parteien korrumpieren und ihre Ein-/Ausgabe kontrollieren darf, aber die Ausgaben der ehrlichen Parteien werden durch eine ideale (d. h. vom Simulator nicht beeinflußbare) Auswertung der Funktion f bestimmt, die als Argumente die Eingaben der ehrlichen Parteien und die vom Simulator modifizierten Eingaben der korrumpierten Parteien erhält. Auch der Simulator gibt eine Ausgabe, bei der er versucht, die Ausgabe des Angreifers so genau wie möglich nachzuahmen.

Wir sagen dann, das Protokoll π sei eine sichere Funktionsauswertung von f, wenn es für jeden Angreifer einen Simulator gibt, so daß für beliebige Partei-Eingaben die Ausgabe der ehrlichen Parteien und des Angreifers im realen Modell ununterscheidbar ist von der Ausgabe des Simulators und der ehrlichen Parteien im idealen Modell.

Wir erkennen nun, daß dieser Sicherheitsbegriff sowohl Korrektheit als auch Geheimhaltung impliziert. Die Korrektheit ergibt sich daraus, daß der Simulator es schaffen muß, daß die ehrlichen Parteiausgaben im idealen Modell denen im realen entsprechen. Da der Simulator aber ideal keine inkorrekten Ausgaben erzwingen kann, ist ihm das nur möglich, wenn auch im realen bereits Korrektheit gegeben ist. Analog muß der Simulator die Ausgabe des Angreifers nachahmen, dazu muß er in gewissem Sinne „ebensoviel wissen wie der Angreifer." Wieder ist dies nur möglich, wenn der Angreifer nichts lernt, was man nicht auch im Idealfall lernen kann.

Die Erklärungen der letzten Absätze illustrieren das zentrale Paradigma simulationsbasierter Sicherheitsmodelle: *Weil im idealen Modell nichts „Schlimmes" passieren kann (per Definition), und weil im idealen Modell alles passiert, was im realen passiert, kann folglich auch im realen Modell nicht „Schlimmes" passieren.* Diese Grundmotivation zieht sich, mit verschiedenen konkreten Interpretationen von „schlimm", durch alle simulationsbasierten Sicherheitsdefinitionen.

Basierend auf diesem Ansatz sind die unabhängig von Pfitzmann und Waidner [PW01] und Canetti [Can01] eingeführten simulationsbasierten Sicherheitsmodellen mit Umgebung eine natürliche Konstruktion: Auch bei diesen unterscheiden wir zwischen einem realen Protokoll und seiner Idealisierung, der idealen Funktionalität. Anders als bei der sicheren Funktionsauswertung handelt es sich bei letzterer aber nicht um eine einfache Funktion, sondern um eine hypothetische Maschine, die das gewünschte Verhalten des Protokolls beschreibt. So könnte man zum Beispiel ein elektronisches geheimes Wahlverfahren durch eine ideale Funktionalität modellieren, die von jedem Teilnehmer dessen Stimme als Eingabe erwartet und das Ergebnis der Wahl als Ausgabe liefert. Insbesondere gibt diese Funktionalität keine darüber hinausgehenden Informationen aus (wie einzelne Stimmen) und läßt keine Manipulation der Wahl zu.

Weiterhin wird bei der Ausführung des realen Protokolls ein Angreifer angenommen, der einzelne Parteien korrumpieren kann, und bei der Ausführung der Funktionalität ein Simu-

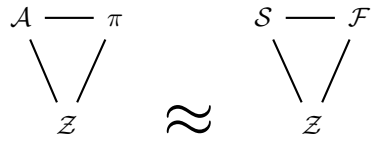


Abbildung 1: Das reale Protokoll π wird so sicher wie die ideale Funktionalität $\mathcal{F}$ genannt, wenn keine Umgebung $\mathcal{Z}$ sie unterscheiden kann. Dabei wird das Protokoll π von einem Angreifer $\mathcal{A}$ angegriffen, den ein Simulator $\mathcal{S}$ zu imitieren versucht.

lator, der zwar die von korrumpierten Parteien an die Funktionalität gesandten Eingaben wählen und die von der Funktionalität an die korrumpierten Parteien gesandten Ausgaben lesen kann, der aber keine darüber hinausgehenden Angriffsmöglichkeiten hat. (Vgl. auch Abbildung 1.)

Soweit unterscheidet sich das Modell nur dahingehend von den zuvor beschriebenen Ansätzen, daß wir nicht nur Funktionsauswertungen, sondern beliebige interaktive Applikationen modellieren können. Die wesentliche Neuerung liegt aber in der Art und Weise, in der wir das reale und das ideale Spiel miteinander vergleichen. Hierzu wird nämlich eine weitere Entitität eingeführt, die sogenannte Umgebung. Hierbei handelt es sich um eine Maschine, welche entweder mit dem realen Protokoll und dem Angreifer, oder mit der idealen Funktionalität und dem Simulator ausgeführt wird, und welche die Aufgabe hat zu erraten, in welcher der beiden Situationen sie sich befindet; sie muß also das reale und das ideale Spiel unterscheiden. Dabei darf die Umgebung in beliebiger Weise mit dem Angreifer oder Simulator kommunizieren, sowie Protokolleingaben geben (im Namen der unkorrumpierten Parteien) und die dazugehörigen Protokollausgaben lesen. Gelingt es keiner Umgebung, die Spiele zu unterscheiden, so können wir sagen, daß alles „schlimme“, was im realen passiert, auch im idealen geschieht, und damit daß das reale Protokoll so sicher wie die ideale Funktionalität ist.

In diesem Szenario hat also der Simulator die doppelte Aufgabe, einerseits in der Kommunikation mit der Umgebung den Angreifer überzeugend zu imitieren, und andererseits die Funktionalität durch geschickt gewählte Eingaben dazu zu bewegen, sich aus Sicht der Umgebung so zu verhalten wie das reale Protokoll. Dies wird dadurch erschwert, daß dem Simulator dabei die Möglichkeit des Angreifers, in die protokollinterne Kommunikation einzugreifen, nicht offensteht.

Fassen wir unsere Überlegungen zusammen, so können wir den Sicherheitsbegriff also in etwa wie folgt formulieren (wobei wir uns hier natürlich aus Platzgründen auf eine sehr ungenaue Formulierung beschränken müssen):

Definition 1 (Allgemeine Sicherheit) *Ein Protokoll π ist* so sicher wie *eine ideale Funktionalität $\mathcal{F}$, wenn folgendes gilt: Für jeden Angreifer $\mathcal{A}$ existiert ein Simulator $\mathcal{S}$, so daß für jede Umgebung $\mathcal{Z}$ gilt: Bei einer Ausführung des aus π, $\mathcal{A}$ und $\mathcal{Z}$ bestehenden Netzwerkes hat die Ausgabe von $\mathcal{Z}$ ungefähr die gleiche Verteilung wie bei einer Ausführung des aus $\mathcal{F}$, $\mathcal{S}$ und $\mathcal{Z}$ bestehenden Netzwerkes.*

Wir nennen den Sicherheitsbegriff in dieser Formulierung *allgemeine Sicherheit*. Bei genauerer Überlegung stellt sich aber heraus, daß eine andere Formulierung der obigen

Überlegungen genauso natürlich ist, welche wir die *spezielle Sicherheit* nennen. Der einzige Unterschied besteht darin, daß nun der Simulator nach der Umgebung gewählt wird, dessen Simulationsstrategie also von der Umgebung abhängen darf.

Definition 2 (Spezielle Sicherheit) *Ein Protokoll π ist* so sicher wie *eine ideale Funktionalität $\mathcal{F}$, wenn folgendes gilt: Für jeden Angreifer $\mathcal{A}$ und jede Umgebung $\mathcal{Z}$ existiert ein Simulator $\mathcal{S}$, so daß gilt: Bei einer Ausführung des aus π, $\mathcal{A}$ und $\mathcal{Z}$ bestehenden Netzwerkes hat die Ausgabe von $\mathcal{Z}$ ungefähr die gleiche Verteilung wie bei einer Ausführung des aus $\mathcal{F}$, $\mathcal{S}$ und $\mathcal{Z}$ bestehenden Netzwerkes.*

Der Unterschied zwischen den beiden Varianten der Sicherheit scheint klein, und tatsächlich fällt es auch sehr schwer zu erklären, worin der intuitive Unterschied zwischen den beiden Varianten liegt, und welche denn in natürlicherer Weise die intuitiven Sicherheitsanforderungen wiederspiegelt. Nichtsdestotrotz werden wir sehen, daß dieser Unterschied in der Quantorenreihenfolge große Auswirkungen hat. Doch bevor hierauf näher eingehen, wollen wir zunächst näher erläutern, warum diese Sicherheitsbegriffe mit Umgebung so nützliche Kompositionseigenschaften haben.

Hierzu betrachten wir die Umgebung aus einem anderen Blickwinkel. Da die Umgebung eine beliebige Maschine ist, kann sie insbesondere beliebige weitere Protokolle simulieren. Da wir die Sicherheit des realen Protokolls π für jede Umgebung fordern, folgt damit, daß diese Sicherheit auch noch gilt, wenn π zusammen mit beliebigen (hier von der Umgebung simulierten) anderen Protokollen läuft.

Diese Idee wurde unabhängig von [Can01] und von [PW01] genauer gefaßt. Wenn ein Protokoll σ ein anderes Protokoll π als Unterprotokoll verwendet, so schreiben wir σ^π für das resultierende (komponierte) Protokoll. Ersetzen wir π durch eine ideale Funktionalität $\mathcal{F}$, so schreiben wir $\sigma^\mathcal{F}$. In [Can01] wurde dann das folgende Theorem gezeigt:

Theorem 1 (Allgemeine Komposition) *Angenommen, σ^π verwendet eine beliebige Anzahl von Instanzen des Subprotokolls π, und π ist so sicher wie $\mathcal{F}$ bezüglich allgemeiner Sicherheit. Dann ist σ^π so sicher wie $\sigma^\mathcal{F}$ bezüglich allgemeiner Sicherheit.*

In anderen Worten, in jedem Protokollkontext können wir eine Funktionalität durch eine sichere Implementierung derselben ersetzen, was den modularen Aufbau von Protokollen erlaubt. In [PW01] wurde eine andere Variante des Kompositionstheorems gezeigt:

Theorem 2 (Einfache Komposition) *Angenommen, σ^π verwendet* nur eine *Instanz des Subprotokolls π, und π ist so sicher wie $\mathcal{F}$ bezüglich allgemeiner oder spezieller Sicherheit. Dann ist σ^π so sicher wie $\sigma^\mathcal{F}$ bezüglich allgemeiner bzw. spezieller Sicherheit.*

Offensichtlich erlaubt das einfache Kompositionstheorem nur eine weniger mächtige Form der Komposition. Viele Konstruktionen verwenden eine Vielzahl von Instanzen des Subprotokolls, so daß diese nur mit dem allgemeinen Kompositionstheorem gezeigt werden können. Allerdings gilt das einfache Kompositionstheorem auch für den Fall der speziellen Sicherheit, wohingegen die allgemeine Komposition nur bei der allgemeinen Sicherheit möglich ist. Es war lange Zeit eine offene Frage, ob allgemeine Komposition auch bei spezieller Sicherheit möglich ist.

Seit ihrer Einführung waren die Sicherheitsmodelle mit Umgebung Objekt intensiver Forschungen. Da diese Sicherheitsmodelle sehr hohe Anforderungen an die Protokolle stellen,

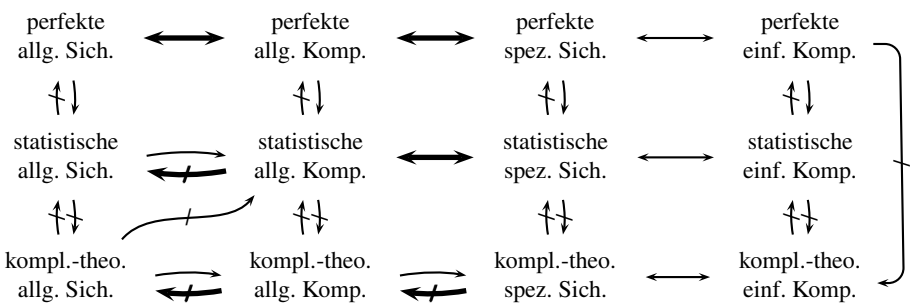
stellte sich die Frage, ob eine derart strenge Sicherheitsdefinition überhaupt notwendig ist, oder ob nicht eine schwächere existiert, die trotzdem noch Komposition ermöglicht. Diese Frage wurde von Lindell [Lin03] beantwortet, der zeigte, daß *spezielle* Sicherheit nicht nur hinreichend (wie in [PW01] gezeigt), sondern auch *notwendig* für die *einfache* Komposition ist. Jedoch konnte nicht beantwortet werden, welches die minimalen Anforderungen für *allgemeine* Komponierbarkeit sind. Es blieben also die folgenden offenen Fragen: Welcher Begriff ist notwendig für die allgemeine Komponierbarkeit, ist es die allgemeine oder die spezielle Sicherheit oder ein dazwischenliegender Begriff? Oder sind allgemeine Sicherheit und spezielle Sicherheit gar äquivalent, und somit beide sowohl notwendig als auch hinreichend für spezielle wie allgemeine Komponierbarkeit? Kurz, welches sind die Implikationen und Trennungen zwischen den folgenden vier Begriffen: spezielle Sicherheit, allgemeine Sicherheit, einfache Komponierbarkeit und allgemeine Komponierbarkeit? Diese Fragen werden in dieser Arbeit beantwortet.

3 Unsere Ergebnisse

Es stellt sich heraus, daß die Frage nach den Implikationen und Trennungen zwischen den Begriffen der speziellen Sicherheit, allgemeinen Sicherheit, einfachen Komponierbarkeit und allgemeinen Komponierbarkeit von anderen Details der Sicherheitsdefinition abhängt. So unterscheiden wir perfekte, statistische und komplexitätstheoretische Sicherheit: Bei der *perfekten Sicherheit* muß die Erfolgswahrscheinlichkeit eines Angriffs exakt Null sein, bei der *statistischen Sicherheit* darf eine vernachlässigbare Wahrscheinlichkeit eines erfolgreichen Angriffs bestehen, und bei der *komplexitätstheoretischen Sicherheit* beschränken wir uns zusätzlich auf polynomiell-beschränkte Angreifer, Simulatoren und Umgebungen.

Wir erarbeiten für jede dieser Varianten alle Implikationen und Trennungen zwischen allgemeiner und spezieller Sicherheit und einfacher und allgemeiner Komponierbarkeit.

Wir erhalten dann die in der folgenden Tabelle dargestellten Beziehungen. Die Ergebnisse dieser Arbeit sind dabei durch Fettdruck hervorgehoben. Man beachte, daß die Darstellung in dem Sinne vollständig ist, daß die Beziehung zwischen je zwei der Begriffe aus den angegebenen Pfeilen folgt.



Die Untersuchung dieser Beziehungen stellt das Hauptziel der Dissertation dar. Auf dem Weg zu dieser vollständigen Klassifizierung erarbeiten wir jedoch zwei Hilfsmittel, die

auch unabhängig von der vorliegenden Aufgabenstellung interessant sind.

Zum einen untersuchen wir sogenannte *Time-Lock Puzzles*. Es handelt sich dabei um eine erstmals in [May93] vorgestellte und von [RSW96] ausgearbeitete Idee. Ein Time-Lock Puzzle ist ein (interaktives) Problem, bei dem die Schwierigkeit des Problems von der das Problem stellenden Maschine relativ genau eingestellt werden kann. Wir geben die erste formale Definition eines solchen Time-Lock Puzzles an und untersuchen Bedingungen für deren Existenz. Wir verwenden Time-Lock Puzzles zur Konstruktion von trennenden Gegenbeispielen zwischen den oben genannten Sicherheitbegriffen im komplexitätstheoretischen Fall. Darüberhinaus stellt sich heraus, daß unsere Methodik auch für andere als die in der Dissertation betrachteten Szenarien anwendbar ist, vergleiche z. B. [HUMQ07].

Ein weiteres Hilfsmittel bei der Analyse der Beziehungen sind die *universellen Umgebungen und Simulatoren*. Wir übertragen hier das Konzept des Nash-Equilibriums aus der Spieltheorie auf die behandelten Sicherheitsmodelle. Wir untersuchen die Existenz und die Komplexität universeller Umgebungen und Simulatoren. Eine Folgerung aus ihrer Existenz ist die Tatsache, daß die Reihenfolge der Quantoren in der Sicherheitsdefinition irrelevant wird; wir folgern daraus die Äquivalenz zwischen statistischer allgemeiner und spezieller Sicherheit für bestimmte Protokollklassen und erhalten nebenbei neue Einsichten in die Komplexität von Angreifern im Falle der statistischen Sicherheit. Aus Platzgründen gehen wir in dieser Zusammenfassung nicht näher auf die universellen Umgebungen und Simulatoren ein.

4 Time-Lock Puzzles und ihre Verwendung

In diesem Abschnitt stellen wir exemplarisch eine der in der Dissertation erarbeiteten Techniken vor, die Time-Lock Puzzles. Unter einem Puzzle verstehen wir ganz allgemein ein Protokoll zwischen zwei Maschinen, dem Prover und dem Verifier, bei dem der Prover nach der Interaktion ein Bit ausgibt. Ist dieses Bit 1, so sagen wir, der Prover habe akzeptiert. Die Aufgabe des Verifiers ist es, den Prover zum Akzeptieren zu bringen. Gelingt dies, so sagen wir, der Verifier habe das Puzzle gelöst. Ein Time-Lock Puzzle ist nun ein spezieller Typ eines Puzzles, bei dem Prover und Verifier einen Schwierigkeitsgrad s als Eingabe erhalten, und das die folgenden zwei (stark vereinfachten) Eigenschaften hat:

- *Einfachheit:* Der Verifier kann das Puzzle innerhalb von in s polynomieller Zeit lösen. Der Aufwand des Provers darf mit s nicht (wesentlich) wachsen.
- *Schwierigkeit:* Zu jedem Verifier, der in polynomieller Zeit läuft, kann man einen polynomiell-beschränkten Schwierigkeitsgrad s wählen, so daß der Verifier das Puzzle nicht mehr lösen kann.

Die erste Frage, die sich hier stellt, ist, ob Time-Lock Puzzles überhaupt existieren. Insbesondere die Anforderung, daß der Aufwand des Provers von s im wesentlichen unabhängig sein muß, erschwert die Konstruktion von Time-Lock Puzzles. In der Dissertation werden verschiedene Konstruktionen von Time-Lock Puzzles vorgestellt und gezeigt, daß unter realistischen Komplexitätsannahmen Time-Lock Puzzles tatsächlich existieren.

4.1 Allgemeine und spezielle Sicherheit

Wie kann man nun Time-Lock Puzzles verwenden, um zu zeigen, daß allgemeine und spezielle Sicherheit im komplexitätstheoretischen Fall tatsächlich verschiedene Begriffe sind, das heißt, daß es Protokolle gibt, die bezüglich spezieller Sicherheit sicher und bezüglich allgemeiner Sicherheit unsicher sind?

Die Grundidee ist relativ einfach. Man konstruiert zunächst eine ideale Funktionalität $\mathcal{F}$ wie folgt: Die Funktionalität $\mathcal{F}$ erwartet, daß der Simulator $\mathcal{S}$ und die Umgebung $\mathcal{Z}$ jeweils einen Schwierigkeitsgrad $s_{\mathcal{S}}$ bzw. $s_{\mathcal{Z}}$ wählen und dann ein Time-Lock Puzzle des gewählten Schwierigkeitsgrads lösen. Löst die Umgebung das schwierigere Puzzle, so sendet die Funktionalität eine spezielle Nachricht `ideal` an die Umgebung. Löst der Simulator das schwierigere Puzzle, so wird keine solche Nachricht gesandt. Das reale Protokoll π verhält sich aus Sicht der Umgebung wie die Funktionalität $\mathcal{F}$, nur wird nie eine Nachricht `ideal` gesandt.

Unter welchen Umständen kann nun die Umgebung zwischen dem realem Protokoll π und der Funktionalität $\mathcal{F}$ unterscheiden? Dies gelingt genau dann, wenn $\mathcal{F}$ die Nachricht `ideal` schickt, was widerum genau dann geschieht, wenn die Umgebung das schwierigere Time-Lock Puzzle löst.

Nun betrachten wir die zwei Varianten der komplexitätstheoretischen Sicherheit. Bei der allgemeinen Sicherheit wird die Umgebung nach dem Simulator gewählt. Es sei also ein Simulator $\mathcal{S}$ gegeben. Nach der Schwierigkeitsbedingung der Time-Lock Puzzles und weil der Simulator in polynomieller Zeit läuft, existiert nun ein Polynom p, so daß $\mathcal{S}$ Puzzles der Schwierigkeit $s_{\mathcal{S}} \geq p$ nicht mehr lösen kann. Dann konstruieren wir eine Umgebung, die immer $s_{\mathcal{Z}} := p$ setzt und das Puzzle löst. Diese Umgebung läuft nach der Einfachheitsbedingung der Time-Lock Puzzles ebenfalls in polynomieller Zeit und löst Puzzle von größerer Schwierigkeit als der Simulator. Somit kann wird die Nachricht `ideal` gesandt und die Umgebung kann unterscheiden. Damit ist π *nicht* so sicher wie $\mathcal{F}$ bezüglich allgemeiner Sicherheit.

Bei der speziellen Sicherheit wird der Simulator in Abhängigkeit von der Umgebung gewählt. Analog zum vorigen Fall können wir also nun einen Simulator konstruieren, der Time-Lock Puzzle löst, die die Umgebung nicht lösen kann. Dann wird die Funktionalität $\mathcal{F}$ nie die Nachricht `ideal` senden, und die Umgebung kann nicht unterscheiden. Somit ist π so sicher wie $\mathcal{F}$ bezüglich spezieller Sicherheit, und es ergibt sich das folgende Ergebnis:

Theorem 3 *Wenn Time-Lock Puzzles existieren, dann sind spezielle und allgemeine Sicherheit nicht äquivalent.*

4.2 Allgemeine Komposition

Das Konzept der Time-Lock Puzzles kann ebenfalls verwendet werden, um zu zeigen, daß spezielle Sicherheit nicht einmal hinreichend für allgemeine Komponierbarkeit ist, daß also nicht nur formal ein Unterschied zwischen allgemeiner und spezieller Sicherheit

besteht, sondern der speziellen Sicherheit tatsächlich eine wichtige Eigenschaft gegenüber der allgemeinen fehlt.

Da diese Konstruktion jedoch wesentlich komplizierter ist als die im vorangegangenen Abschnitt vorgestellte, müssen wir uns aus Platzgründen darauf beschränken, nur die Grundidee vorzustellen. Um zu zeigen, daß spezielle Sicherheit nicht hinreichend für allgemeine Komponierbarkeit ist, genügt es, ein Protokoll π und eine Funktionalität $\mathcal{F}$ zu finden, so daß π so sicher wie $\mathcal{F}$ ist bezüglich spezieller Sicherheit, aber π^* nicht so sicher wie $\mathcal{F}^*$, wobei π^* das mehreren Kopien von π bestehende Protokoll ist und $\mathcal{F}^*$ analog. Ähnlich wie im vorangegangenen Abschnitt konstruieren wir nun eine Funktionalität $\mathcal{F}$, die eine spezielle Nachricht `ideal` ausgibt, wenn der Simulator keine hinreichend schwierigen Time-Lock Puzzles löst. Anders als im vorangegangenen Abschnitt jedoch muß die Umgebung keine Time-Lock Puzzles mehr lösen, sondern die Schwierigkeit der Puzzles wird wie folgt bestimmt: Es seien n Instanzen von $\mathcal{F}$ vorhanden. (Wenn wir direkt die Sicherheit von π im Verhältnis zu $\mathcal{F}$ betrachten, ist $n = 1$. Wenn wir aber π^* und $\mathcal{F}^*$ betrachten, kann n größere Werte annehmen.) Zunächst bestimmen die Instanzen von $\mathcal{F}$ die Zahl n. Dann erwarten diese Instanzen von $\mathcal{F}$, daß der Simulator ein Time-Lock Puzzle der Schwierigkeit 2^n löst.

Ist nun $n = 1$, so kann der Simulator ein solches Puzzle leicht lösen, und die Funktionalität $\mathcal{F}$ wird nie die Nachricht `ideal` an die Umgebung senden. Also ist π so sicher wie $\mathcal{F}$. Darf aber n größer sein, so ist 2^n exponentiell, und der Simulator kann die Puzzle nicht mehr lösen, und $\mathcal{F}$ schickt `ideal` an die Umgebung. Diese unterscheidet, und somit ist π^* nicht so sicher wie $\mathcal{F}^*$.

Leider läßt sich der beschriebene Ansatz so nicht umsetzen, denn die Instanzen der Funktionalität $\mathcal{F}$ kennen n nicht. Die Funktionalitäten keine direkte Verbindung haben, können sie sich auch nicht direkt durchzählen. Die einzige Möglichkeit, mit der die Funktionalitäten einander zählen können, ist Nachrichten über die Umgebung zu senden (über den Simulator zu senden hilft nicht, da dieser ein Interesse daran hat, die Nachrichten nicht weiterzuleiten und somit jede Funktionalität glauben zu machen, sie sei allein). Da die Umgebung aber wird, wenn sie kann, die Zählung verfälschen, um dem Simulator selbst im Falle $n = 1$ dem Simulator die Aufgabe zu erschweren. Durch Einsatz verschiedener kryptographischer Techniken (auf die wir hier aus Platzgründen nicht eingehen) kann aber auch dieses Problem gelöst werden, und wir erhalten schließlich:

Theorem 4 *Wenn Time-Lock Puzzles und sogenannte Enhanced Trapdoor Permutations*[1] *existieren, dann impliziert die spezielle Sicherheit nicht die allgemeine Komponierbarkeit im Sinne von Theorem 1.*

5 Schlußbemerkungen

Wir haben die verschiedenen Varianten des verbreiteten Begriffs der simulationsbasierten Sicherheit mit Umgebung untersucht und dabei erstmalig eine vollständige Klassifikation

[1] Hierbei handelt es sich um eine verbreitete kryptographische Komplexitätsannahme.

inklusive der zwischen den Begriffen bestehenden Relationen erarbeitet. Dabei haben wir nicht nur Ordnung in die Vielfalt der Begriffe gebracht, sondern die verwendeten Beweistechniken haben auch Einblicke in die genauere Wirkungsweise dieser Sicherheitsbegriffe und die Bedeutung der verschiedenen Designentscheidungen in ihrer Modellierung gezeigt. Die hierzu entwickelten Werkzeuge scheinen nicht nur bei der Aufgabenstellung dieser Arbeit Anwendung finden zu können, sondern auch bei der Analyse anderer Begriffe sehr hilfreich zu sein.

Literatur

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of FOCS 2001*, Seiten 136–145. IEEE, 2001.

[HUMQ07] Dennis Hofheinz, Dominique Unruh und Jörn Müller-Quade. A Simple and Versatile Notion of Polynomial Time for Cryptographic Protocols. Manuskript, 2007.

[Lin03] Yehuda Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *Proceedings of FOCS 2003*, Seiten 394–403. IEEE, 2003.

[May93] Timothy C. May. Timed-Release Crypto. Cypherpunks Mailing List, Februar 1993. `http://cypherpunks.venona.com/date/1993/02/msg00129.html`.

[PW01] Birgit Pfitzmann und Michael Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *Proceedings of SSP'01*, Seiten 184–200. IEEE, 2001.

[RSW96] Ronald Rivest, Adi Shamir und David Wagner. Time-lock puzzles and timed-release Crypto. Bericht MIT/LCS/TR-684, MIT, Februar 1996.

[Unr07] Dominique Unruh. *Protokollkomposition und Komplexität*. Logos, Berlin, 2007. Dissertation, Universität Karlsruhe (TH).

Dominique Unruh wurde am 17. März 1979 in Hannover geboren. Von 1993 bis 1997 besuchte er den Hochbegabtenzweig des Christophorus-Internatsgymnasiums in Braunschweig. Ab 1998 studierte er an der Universität Karlsruhe Informatik und erhielt 2003 sein Diplom mit Auszeichnung. Parallel dazu erhielt er außerdem ein Vordiplom der Mathematik. Danach arbeitete er als wissenschaftlicher Mitarbeiter am Institut für Algorithmen und Kognitive Systeme an der Universität Karlsruhe, bis er 2006 mit Auszeichnung zum Thema „Protokollkomposition und Komplexität“ promovierte. Er erhielt für seine Dissertation den Dissertationspreis des Fördervereins des Forschungszentrums Informatik, Karlsruhe und wurde für den Dissertationspreis der Gesellschaft für Informatik nominiert. Seit 2006 arbeitet er als wissenschaftlicher Mitarbeiter an der Information Security and Cryptography Group an der Universität des Saarlandes.