Kryptographische Protokolle und Null-Information*

Jörg Rothe
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
40225 Düsseldorf, Germany
rothe@cs.uni-duesseldorf.de

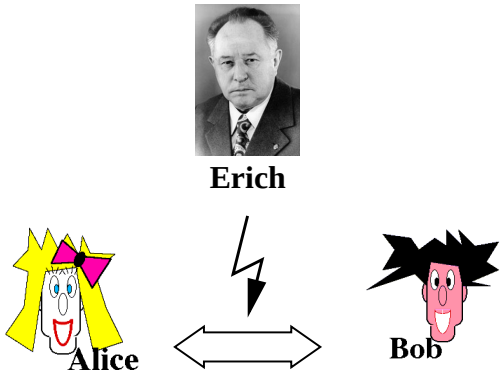
Zusammenfassung

Ein Ziel der Komplexitätstheorie ist der Nachweis, dass Probleme nicht effizient gelöst werden können. Solche Ergebnisse wirken oft „negativ“ und nicht wünschenswert. Dieser Beitrag beschäftigt sich mit ihrem „positiven“ Aspekt, den *Anwendungen der Ineffizienz*. Während man normalerweise möglichst effiziente Algorithmen zur Lösung von Problemen entwickeln will, ist man in der Kryptographie gerade im Gegenteil daran interessiert, die Ineffizienz bestimmter Probleme nachzuweisen und anzuwenden: Ein Beweis der Ineffizienz gewisser Probleme bedeutet hier einen Zuwachs an Sicherheit in der Übertragung verschlüsselter Nachrichten. In diesem Beitrag werden Protokolle für den Schlüsseltausch und digitale Signaturen vorgestellt sowie die kryptographischen Grundbausteine solcher Protokolle, wie z.B. „Einwegfunktionen“, beschrieben. Wir gehen auch auf das ganz aktuelle Thema der Zero-Knowledge-Protokolle ein. Das sind kryptographische Protokolle, die es erlauben, dass keinerlei Information ungewollt preisgegeben wird: Mit einem Zero-Knowledge-Protokoll kann Alice Bob davon überzeugen, dass sie ein gewisses Geheimnis kennt, ohne auch nur den geringsten Teil dieses Geheimnisses zu verraten.

Schlüsselwörter: *Public-Key-Kryptosysteme, Protokolle für Schlüsseltausch und digitale Signaturen, Einwegfunktionen, Zero-Knowledge-Protokolle, interaktive Beweissysteme*.

1 Protokolle für den Schlüsseltausch und Digitale Signaturen

Stellen wir uns die folgende Situation vor: Alice und Bob möchten Botschaften über einen unsicheren Kanal wie z.B. eine öffentliche Telefonleitung austauschen, wobei sie von Erich belauscht werden:



Deshalb wollen Alice und Bob ihre Botschaften verschlüsseln. Damit die Verschlüsselung effizient ist, entscheiden sie sich für ein symmetrisches Kryptosystem, in dem also der Schlüssel für die Verschlüsselung derselbe wie der für die Entschlüsselung ist. Aber wie können sie sich dann auf einen gemeinsamen geheimen Schlüssel einigen, wenn sie nur über einen unsicheren Kanal kommunizieren können? Würden

*Erschienen in *Informatik Spektrum*, Springer-Verlag, vol. 25, no. 2, pp. 120–131, April 2002. Dieser Text basiert auf Auszügen des Vorlesungsskripts [22] meiner Vorlesungsreihe im Rahmen der *11th Jyväskylä Summer School*, die im August 2001 an der University of Jyväskylä in Finnland stattfand. Diese Arbeit wurde unterstützt durch den grant NSF-INT-9815095/DAAD-315-PPP-gü-ab.

sie eine verschlüsselte Botschaft verschicken, die den künftig zu benutzenden Schlüssel enthält, ergibt sich die Frage: Mit welchem Schlüssel soll *diese* Botschaft verschlüsselt werden?

Diese Situation, die ein wenig an das bekannte Paradoxon von dem Ei und der Henne erinnert, ist als das *Problem des Schlüsseltauschs* (engl.: *secret-key agreement problem*) bekannt, das über Jahrtausende – seit Beginn der Kryptographie – als nicht lösbar galt. Es war eine ziemliche Überraschung, als 1976 Whitfield Diffie und Martin Hellman [5] eine Lösung dieses lange offenen, paradox wirkenden Problems vorschlugen: das allererste *secret-key agreement* Protokoll. Ihr Protokoll wird in Abbildung 1 dargestellt. Interessanterweise, und das ist das eigentlich Paradoxe am Diffie-Hellman-Protokoll, inspirierte dieses die Erfindung des RSA-Systems durch Rivest, Shamir und Adleman [21]. RSA ist das historisch erste Public-Key-Kryptosystem; auch heute noch ist es weit verbreitet. Das heißt, mit der Lösung des für die symmetrische Kryptographie wichtigen Schlüsseltauschproblems öffneten Diffie und Hellman die Tür zur *asymmetrischen* (oder *public-key*) Kryptographie, in der keine geheimen Schlüssel mehr über unsichere Kanäle verschickt werden müssen.

Eine andere Aufgabe in der Kryptographie ist die Erstellung digitaler Signaturen. Diese werden zur Authentifizierung von Dokumenten wie z.B. emails benötigt. Das Ziel ist, sicherzustellen, dass weder Erich noch Bob die digitale Unterschrift von Alice fälschen kann. Alice möchte ihre verschlüsselten Nachrichten an Bob so signieren, dass (a) Bob verifizieren kann, dass tatsächlich sie die Absenderin war, und (b) auch Dritte, die womöglich kein Vertrauen in Bob haben, sich von der Echtheit ihrer Unterschrift überzeugen können. Die Eigenschaft (a) leisten bereits die symmetrischen Authentikationscodes; die spezifische Asymmetrie der digitalen Signaturen kommt erst in der Eigenschaft (b) zum Ausdruck. Diese Eigenschaft (b) ist es, die die digitalen Signaturen beispielsweise für den sicheren E-Commerce nützlich und notwendig macht, denn dort sind Interessenskonflikte zwischen Alice und Bob zu erwarten.

1.1 Diffie-Hellman-Protokoll für den Schlüsseltausch

			
1	Alice und Bob einigen sich auf eine große Primzahl p und eine primitive Wurzel g von p; p und g sind öffentlich		
2	wählt zufällig eine große Zahl a, berechnet $\alpha = g^a \mod p$		wählt zufällig eine große Zahl b, berechnet $\beta = g^b \mod p$
3		$\overset{\alpha}{\Rightarrow}$ $\overset{\beta}{\Leftarrow}$	
4	berechnet ihren Schlüssel $k_A = \beta^a \mod p$		berechnet seinen Schlüssel $k_B = \alpha^b \mod p$

Abbildung 1: Diffie-Hellman-Protokoll für den Schlüsseltausch.

Abbildung 1 zeigt das Diffie-Hellman-Protokoll. Es beruht auf der modularen Exponentialfunktion mit Basis g und Modulus p, wobei p eine Primzahl und g eine primitive Wurzel von p in $\mathbb{Z}_p^*$ ist, der zyklischen Gruppe der primen Reste modulo p. Etwas formaler: $\mathbb{Z}_p^*$ ist die Menge aller natürlichen Zahlen a mit $1 \leq a \leq p-1$, die teilerfremd zu p sind. $\mathbb{Z}_p^*$ hat die Ordnung $\Phi(p) = p-1$, wobei Φ die Eulersche Funktion ist. Der Satz von Euler sagt, dass für alle $n \in \mathbb{N}$ und für alle $a \in \mathbb{Z}_n^*$ gilt: $a^{\Phi(n)} \equiv 1 \mod n$. Der Spezialfall, bei dem n eine Primzahl p ist, die a nicht teilt, ist bekannt als der „Kleine Fermat“: Ist p eine Primzahl und $a \in \mathbb{Z}_p^*$, so gilt $a^{p-1} \equiv 1 \mod p$. Eine *primitive Wurzel von* p ist jedes Element $a \in \mathbb{Z}_p^*$, so dass für jedes d mit $1 \leq d < \Phi(p)$ gilt: $a^d \not\equiv 1 \mod p$. Die Funktion $\alpha_{(g,p)} : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$, die definiert ist

durch

$$\alpha_{(g,p)}(a) = g^a \mod p,$$

heißt *modulare Exponentialfunktion mit Basis g und Modulus p*. Ihre Umkehrfunktion, die für fixiertes p und g den Wert $\alpha_{(g,p)}(a)$ auf $a = \log_g \alpha \mod p$ abbildet, heißt der *diskrete Logarithmus*. Das Protokoll ist korrekt, da in der Arithmetik modulo p gilt:

$$k_A = \beta^a = g^{ba} = g^{ab} = \alpha^b = k_B.$$

Die Sicherheit des Diffie-Hellman-Protokolls beruht auf der (unbewiesenen, aber plausiblen) Annahme, dass die Berechnung diskreter Logarithmen eine sehr schwere Aufgabe ist. Im Gegensatz dazu kann die modulare Exponentialfunktion mit der „*square-and-multiply*"-Methode effizient berechnet werden. Deshalb wird diese Funktion als eine *Einwegfunktion* betrachtet, eine Funktion, die zwar leicht zu berechnen, aber nur schwer zu invertieren ist. Einwegfunktionen spielen eine wichtige Rolle in der Kryptographie.

Wenn Erich die Kommunikation von Alice und Bob aufmerksam belauscht, dann kennt er p, g, α und β. Er möchte ihren gemeinsamen geheimen Schlüssel $k_A = k_B$ berechnen. Könnte er das Problem des diskreten Logarithmus effizient lösen, dann könnte er leicht $a = \log_g \alpha \mod p$ und $b = \log_g \beta \mod p$ und somit $k_A = \beta^a \mod p$ und $k_B = \alpha^b \mod p$ berechnen. Glücklicherweise ist dieses Problem, wie oben erwähnt, schwer, und folglich ist diese Attacke keine wirkliche Bedrohung. Allerdings gibt es für das Diffie-Hellman-Protokoll bis heute keinen „Beweis der Sicherheit". Das unmittelbare Berechnen des Schlüssels $k_A = k_B$ aus α und β ist nicht die einzige mögliche Attacke auf das Diffie-Hellman-Protokoll. Zum Beispiel ist es verwundbar durch die „*Man-in-the-middle*"-Attacke, bei welcher sich Erich gegenüber Alice als Bob und Bob gegenüber als Alice ausgibt. Diese Attacke wirft das Problem der Authentifizierung von *Personen* – im Gegensatz zu *Dokumenten* – auf, wobei wir unter einer „Person" im weitesten Sinn eine Instanz verstehen wollen, die ihr Wissen oder beliebige andere Informationen aktiv an andere Instanzen weitergeben kann. Eine „Person" in unserem Sinne könnte also auch ein Computer sein, der so programmiert ist, dass er einem anderen Computer seine Daten übermittelt. Insofern unterscheidet sich unser Begriff der Authentifizierung von Personen etwa von der biometrischen Authentifizierung natürlicher Personen. Wir kommen auf die *Man-in-the-middle*-Attacke und das aus ihr resultierende Authentikationsproblem für Personen später zurück.

ElGamal [6] modifizierte das Protokoll von Diffie und Hellman, um sowohl ein Public-Key-Kryptosystem als auch ein Protokoll für digitale Signaturen zu entwickeln. Eine besonders effiziente Variante dieses Protokolls, die auf eine Idee von Schnorr [24] zurückgeht, ist heute der *Digital Signature Standard* [19] der USA.

1.2 Rivest-Sherman-Rabi-Protokolle für Schlüsseltausch und Digitale Signaturen

Ron Rivest, Alan Sherman und Muhammad Rabi entwickelten andere Protokolle für den Schlüsseltausch und für digitale Signaturen. Das Schlüsseltausch-Protokoll aus Abbildung 2 geht auf Rivest und Sherman zurück; siehe [20]. Rabi und Sherman [20] modifizierten es zu einem Protokoll für digitale Signaturen, welches in Abbildung 3 dargestellt ist.

Der entscheidende Grundbaustein beider Protokolle ist eine stark nichtinvertierbare, assoziative Einwegfunktion. Eine totale (d.h. überall definierte) zweistellige Funktion σ heißt *assoziativ*, falls für alle x, y und z gilt: $(x\sigma y)\sigma z = x\sigma(y\sigma z)$.[1] Die Assoziativität von σ sichert, dass Alice und Bob denselben Schlüssel berechnen: $k_A = k_B$. Wie oben erwähnt, lassen sich Einwegfunktionen leicht berechnen, aber nur schwer invertieren. Eine zweistellige Einwegfunktion heißt *stark nichtinvertierbar*, falls sie selbst dann schwer invertierbar ist, wenn zusätzlich zum Funktionswert eines ihrer Argumente gegeben ist. Im traditionellen Modell der *worst-case* Komplexität versteht man unter „schwer invertierbar", dass kein Polynomialzeit-Algorithmus in der Lage ist, für jedes Element z aus dem Bild der Funktion eines der Urbilder von z zu berechnen. Die starke Nichtinvertierbarkeit von σ sichert, dass Erich, der ja in den Protokollen sowohl einen Funktionswert als auch eines der zugehörigen Argumente erlauschen kann, das jeweils andere zugehörige Argument nicht leicht ermitteln kann.

Der Begriff der komplexitätstheoretischen (d.h. *worst-case*) Einwegfunktion geht auf Grollmann und Selman [15] zurück, die die Existenz verschiedener Typen von *worst-case* Einwegfunktionen durch Separationen geeigneter Komplexitätsklassen charakterisierten; siehe auch [17, 23]. Beispielsweise ist bekannt,

[1] Für zweistellige Funktionen verwenden wir die Infixnotation $x\sigma y$ statt der Präfixnotation $\sigma(x, y)$.

	 Alice		 Bob
1	wählt zufällig große Zahlen x und y, hält x geheim und berechnet $x\sigma y$		
2		$y, x\sigma y \Rightarrow$	
3			wählt zufällig eine große Zahl z, hält z geheim und berechnet $y\sigma z$
4		$y\sigma z \Leftarrow$	
5	berechnet ihren Schlüssel $k_A = x\sigma(y\sigma z)$		berechnet seinen Schlüssel $k_B = (x\sigma y)\sigma z$

Abbildung 2: Rivest-Sherman-Protokoll für den Schlüsseltausch, basierend auf einer stark nichtinvertierbaren, assoziativen Einwegfunktion σ.

	 Alice		 Bob
1	wählt zufällig große Zahlen x_A and y_A, hält x_A geheim und berechnet $x_A\sigma y_A$		
2		$y_A, x_A\sigma y_A \Rightarrow$	
3	berechnet ihre Signatur $\text{sig}_A(m) = m\sigma x_A$ für die Nachricht m		
4		$m, \text{sig}_A(m) \Rightarrow$	
5			verifiziert Alice' Signatur durch Überprüfen der Gleichheit $m\sigma(x_A\sigma y_A) = (m\sigma x_A)\sigma y_A$

Abbildung 3: Rabi-Sherman-Protokoll für digitale Signaturen, basierend auf einer stark nichtinvertierbaren, assoziativen Einwegfunktion σ.

dass solche Einwegfunktionen genau dann existieren, wenn P $\neq$ NP, wobei P die Klasse der in Polynomialzeit lösbaren Probleme und NP die Klasse der in nichtdeterministischer Polynomialzeit lösbaren Probleme bezeichnet. Folglich kann man an die Existenz von *worst-case* Einwegfunktionen mit derselben Gewissheit glauben, mit der man an die Gültigkeit der Hypothese P $\neq$ NP glaubt. Mit Euler könnte man, in leichter Abwandlung seines Ausspruchs,[2] sagen: „P $\neq$ NP, also existieren Gott und Einwegfunktionen!“

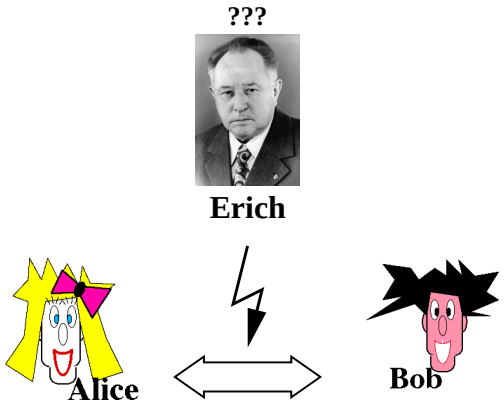
Doch kehren wir zurück zu den Protokollen. Die entscheidende Frage hier ist: *Gibt es denn Einwegfunktionen mit all diesen schönen Eigenschaften?* Die Antwort auf diese Frage scheint heute und in der nahen Zukunft noch nicht möglich zu sein; schließlich impliziert die Existenz von *beliebigen* Einwegfunktionen (also solchen ohne diese Eigenschaften) die Ungleichheit von P und NP, und ein Beweis der Vermutung „P $\neq$ NP“ konnte in über dreißig Jahren intensiver Forschung nicht erbracht werden.

Etwas bescheidener fragen wir also: *Kann man Einwegfunktionen mit diesen schönen Eigenschaften wenigstens aus beliebigen Einwegfunktionen (äquivalent: aus der Annahme* P $\neq$ NP*) konstruieren?* Obwohl Rabi und Sherman [20] bewiesen, dass assoziative Einwegfunktionen im *worst-case* Modell genau dann existieren, wenn P $\neq$ NP, ließen sie die Frage offen, ob irgendeine natürliche komplexitätstheoretische Bedingung hinreichend für die Existenz starker, totaler, kommutativer und assoziativer Einwegfunktionen ist.[3] Hemaspaandra und Rothe [16] beantworten (ebenfalls im *worst-case* Modell) diese Frage: Stark nichtinvertierbare, totale, kommutative und assoziative Einwegfunktionen existieren genau dann, wenn P $\neq$ NP.

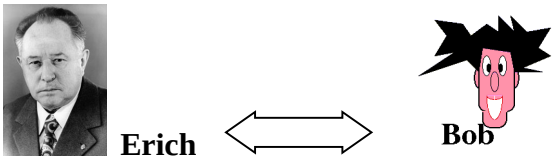
Für die meisten kryptographischen Anwendungen genügt das *worst-case* Modell nicht. Statt dessen wird das Modell der *average-case* Komplexität zugrundegelegt und gefordert, dass nicht einmal randomisierte Algorithmen in der Lage seien, Einwegfunktionen mit vernünftig beschränkter Fehlerwahrscheinlichkeit zu invertieren. Dies für stark nichtinvertierbare, assoziative Einwegfunktionen zu leisten, ist eine interessante Aufgabe künftiger Forschung. Und es besteht Hoffnung: Ajtai [1] zeigte die Äquivalenz der *worst-case* und der *average-case* Komplexität für das NP-harte *Shortest Lattice Vector Problem*.

2 Interaktive Beweissysteme und Zero-Knowledge-Protokolle

Im Zusammenhang mit dem Diffie-Hellman-Protokoll wurde die *Man-in-the-middle*-Attacke erwähnt. Stellen wir uns vor, dass Bob sich gerade mit seiner Partnerin über eine öffentliche Telefonleitung auf einen gemeinsamen geheimen Schlüssel geeinigt hat. Bob war so clever, das Diffie-Hellman-Protokoll zu verwenden, und so glaubt er, dass Erich über den von ihnen gewählten geheimen Schlüssel keine Ahnung hat:



Aber Erich war noch schlauer. In Wirklichkeit ist dies passiert:



[2] Wörtlich hatte Euler, im Streitgespräch mit Diderot, gesagt (siehe [27]): „$(a+b^n)/n = x$, also existiert Gott, antworten Sie!“ Diderot blieb die Antwort schuldig und reiste aus St. Petersburg ab.

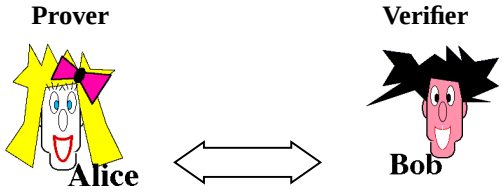
[3] Die Kommutativität ist nötig, um die Protokolle auf mehr als zwei Parteien zu erweitern.

Diese Situation wirft die Frage der Authentifizierung von Personen auf: Wie kann sich Bob sicher sein, dass es tatsächlich Alice war, mit der er kommunizierte, und nicht Erich, der sich als Alice ausgab? Anders gesagt, wie kann Alice zweifelsfrei ihre Identität beweisen? Eine Möglichkeit, dieses Ziel zu erreichen, ist es, mit Alice eine geheime Information zu verknüpfen. Das könnte z.B. ihre PIN ("*P*ersonal *I*dentification *N*umber") sein oder irgendein anderes privates Geheimnis, das niemandem außer ihr bekannt ist. Diese Information nennen wir Alice' *Geheimnis*.

Aber hier ist noch ein Haken. Alice möchte gern Bob von ihrer Identität überzeugen, indem sie beweist, dass sie ihr Geheimnis kennt. Ideal wäre es, wenn sie dabei dieses Geheimnis nicht enthüllen müsste, denn sonst wäre es ja kein Geheimnis mehr: Wüßte Bob Alice' Geheimnis, so könnte er in der Kommunikation mit einem Dritten so tun, als wäre er selbst Alice. Die Frage ist also: *Wie kann man die Kenntnis eines Geheimnisses mitteilen, ohne es zu verraten?* Und das ist genau, worum es bei den Zero-Knowledge-Protokollen geht.

2.1 Interaktive Beweissysteme

Zero-Knowledge-Protokolle sind spezielle interaktive Beweissysteme; letztere werden zunächst beschrieben. Sie wurden von Shafi Goldwasser, Silvio Micali und Charles Rackoff [13] eingeführt. Wie in den bisherigen Protokollen betrachten wir die Kommunikation zwischen zwei Parteien, „*Prover*" Alice und „*Verifier*" Bob:



Vorerst interessieren uns nicht die Sicherheitsaspekte, die hierbei auftreten können, sondern wir sind nur an dem folgenden Kommunikationsproblem interessiert: Alice und Bob wollen gemeinsam ein gegebenes Problem L lösen, d.h., sie wollen entscheiden, ob ein gegebenes Eingabewort zu L gehört. Konkret betrachten wir das Graphisomorphie-Problem: Entscheide, ob ein gegebenes Paar von Graphen isomorph ist. Ein Isomorphismus zwischen zwei ungerichteten Graphen ist eine kantenerhaltende Bijektion zwischen den Knotenmengen der Graphen. `GI` bezeichne die Menge aller Paare von isomorphen Graphen. Dieses Problem liegt in NP, aber es ist nicht bekannt, ob es NP-vollständig ist, d.h., ob es zu den schwersten NP-Problemen gehört. Andererseits ist kein effizienter Algorithmus bekannt, der es lösen würde, und es gibt eine Reihe von Indizien, die für die Härte dieses Problems sprechen.

Das Kommunikationsproblem von Alice und Bob besteht nun darin, gemeinsam zu entscheiden, ob ein gegebenes Paar (G, H) von Graphen isomorph ist. Alice versucht, ihre Isomorphie zu *beweisen*, durch Angabe eines Isomorphismus zwischen G und H. Sie versucht, Bob von der Isomorphie der Graphen zu überzeugen, *egal ob diese Graphen isomorph sind oder nicht*. Aber Bob ist misstrauisch. Um die Eingabe zu akzeptieren, will er mit überwältigender Wahrscheinlichkeit überzeugt werden. Noch schlimmer: Bob ist erst dann überzeugt, wenn *jede denkbare Prover-Strategie* von Alice eine überwältigende Erfolgswahrscheinlichkeit hat. Kann Alice dies leisten, so akzeptiert er, andernfalls lehnt er ab.

Um diese Intuition zu formalisieren, stellen wir uns Alice und Bob als zwei Turingmaschinen vor. Alice ist eine „allmächtige" Turingmaschine A von unbeschränkter Berechnungskraft. Bob ist eine randomisierte Turingmaschine B, die in Polynomialzeit arbeitet. Ein *interaktives Beweissystem* (oder kurz *IP-Protokoll*) (A, B) ist ein Protokoll zwischen Alice und Bob. Beide greifen auf dasselbe Eingabeband und auf ein gemeinsames Kommunikationsband zu, und sie haben private Arbeitsbänder für interne Berechnungen. Alice sieht nicht Bobs zufällige Münzwürfe, die seine randomisierte Berechnung beeinflussen. $\Pr((A, B)(x) = 1)$ bezeichne – gemäß der zufälligen Wahlen in Bobs und, gegebenenfalls, in Alice' Berechnung – die Wahrscheinlichkeit, dass Bob die Eingabe x akzeptiert; d.h., für eine spezielle Folge von Zufallsbits ist „$(A, B)(x) = 1$" das Ereignis, dass Bob von Alice' Beweis für x überzeugt ist und akzeptiert.

Ein *interaktives Beweissystem* (A, B) *akzeptiert ein Problem* L genau dann, wenn für jedes x gilt:

$$x \in L \quad \text{impliziert, dass es ein } A \text{ gibt mit} \quad \Pr((A,B)(x) = 1) \geq \frac{3}{4}; \tag{2.1}$$

$$x \notin L \quad \text{impliziert, dass für alle } \widehat{A} \text{ gilt} \quad \Pr((\widehat{A},B)(x) = 1) \leq \frac{1}{4}, \tag{2.2}$$

wobei $\widehat{A}$ *irgendein Prover* (d.h. irgendeine Turingmaschine von unbeschränkter Berechnungskraft) ist, die Alice ersetzen kann. Das heißt, im Falle der Akzeptierung genügt es, dass Alice *eine* Strategie findet, die Bob überzeugt. Im Falle der Ablehnung ist es formal zweckmäßiger, über alle potenziellen Prover zu quantifizieren, anstatt alle potenziellen Strategien von Alice zu betrachten. Die Bedingung (2.1) wird als *Vollständigkeit* (engl.: *completeness*) und die Bedingung (2.2) als *Verlässlichkeit* (engl.: *soundness*) bezeichnet.

IP bezeichne die Klasse aller Probleme, die von interaktiven Beweissystemen akzeptiert werden können. Die Akzeptierungswahrscheinlichkeiten von wenigstens $\frac{3}{4}$ für $x \in L$ bzw. von höchstens $\frac{1}{4}$ für $x \notin L$ sind dabei willkürlich gewählt. Mit den üblichen Techniken zur Wahrscheinlichkeitsamplifikation kann man statt dessen beliebige Konstanten $\frac{1}{2} + \epsilon$ bzw. $\frac{1}{2} - \epsilon$ wählen, wobei $\epsilon > 0$. Man kann sogar eine Akzeptierungswahrscheinlichkeit von genau 1 in der Vollständigkeitsbedingung (2.1) fordern, ohne damit die Klasse IP zu verändern [11]. In der Literatur findet man für „*Prover*“ und „*Verifier*“ auch die Bezeichnungen *Merlin* und *Arthur*: Der Begriff der Arthur-Merlin-Spiele [2] ist äquivalent zu dem der interaktiven Beweissysteme.

Was, wenn Bob die Münzen ausgehen? Das heißt, was passiert, wenn er sich deterministisch verhält beim Verifizieren von Alice' Beweis für „$x \in L$“? Wegen ihrer unbeschränkten Berechnungskraft kann Alice Beweise unbeschränkter Länge liefern. Da Bob jedoch eine polynomialzeit-beschränkte Turingmaschine ist, kann er nur Beweise einer in der Größe von x polynomiell beschränkten Länge überprüfen. Folglich ist IP, eingeschränkt auf deterministische Polynomialzeit-Verifizierer, nur eine etwas umständliche Art, die Klasse NP zu definieren. Da GI zu NP gehört, muss dieses Problem also auch zur (nicht eingeschränkten) Klasse IP gehören.

Aber was ist mit $\overline{\text{GI}}$, dem Komplement von GI? Gibt es ein interaktives Beweissystem, das entscheidet, ob zwei gegebene Graphen *nicht* isomorph sind? Selbst wenn Alice über unbeschränkte Berechnungskraft verfügt, kann sie dann in Schwierigkeiten kommen, wenn sie die Nicht-Isomorphie zweier Graphen zu beweisen versucht. Betrachten wir z.B. zwei nicht isomorphe Graphen mit jeweils 1000 Knoten. Ein Beweis für ihre Nicht-Isomorphie verlangt von Alice anscheinend zu zeigen, dass keine der 1000! möglichen Permutationen ein Isomorphismus zwischen den Graphen ist. Nicht nur wäre es für Bob unmöglich, einen so langen Beweis zu überprüfen, auch für Alice wäre es wohl unmöglich, diesen auch nur hinzuschreiben. Schließlich ist 1000! ungefähr $4 \cdot 10^{2567}$. Diese Zahl übersteigt die Anzahl der Atome im gesamten sichtbaren Universum,[4] die derzeit auf etwa 10^{77} geschätzt wird, um einen wahrhaft astronomischen Faktor.

Deshalb war es eine Überraschung, als Goldreich, Micali und Wigderson [12] zeigten: $\overline{\text{GI}}$ ist in IP. Somit enthält IP nicht nur ganz NP, sondern auch ein Problem aus coNP, der Klasse der Komplemente von NP-Problemen. $\overline{\text{GI}}$ liegt vermutlich nicht in NP. Das wirft die Frage auf, wie groß die Klasse IP eigentlich ist. Ein berühmtes Resultat von Adi Shamir [26] beantwortet diese Frage: IP ist gleich PSPACE, der Klasse der Probleme, die in polynomialem Raum gelöst werden können.

2.2 Zero-Knowledge-Protokolle

Nun sind wir soweit, dass wir den Begriff der Zero-Knowledge-Protokolle definieren können, und wollen uns dann der Frage zuwenden, wie sich diese für Authentifizierungszwecke verwenden lassen. Wie oben erwähnt, liegt GI in IP. Um zu beweisen, dass zwei gegebene Graphen isomorph sind, schickt Alice einfach einen Isomorphismus π an Bob, den dieser deterministisch in Polynomialzeit verifiziert. Nehmen wir jedoch an, dass Alice π geheim halten möchte. Einerseits möchte sie ihr Geheimnis nicht enthüllen; andererseits möchte sie Bob davon überzeugen, dass sie es kennt. Was sie braucht, ist ein ganz spezielles IP-Protokoll, das nichts über ihren geheimen Isomorphismus verrät und dennoch beweist, dass die Graphen isomorph sind. Ein solches Protokoll für GI wird im nächsten Abschnitt angegeben.

Aber was ist ein Zero-Knowledge-Protokoll und wie kann man diesen Begriff formal fassen? Die Intuition ist dies. Stellen wir uns vor, dass Alice eine Zwillingsschwester hat, die Malice heißt und genau

[4] Ohne dunkle Masse.

wie sie aussieht. Malice kennt jedoch Alice' Geheimnis nicht. Außerdem verfügt Malice nicht über Alice' unbeschränkte Berechnungskraft; statt dessen arbeitet sie wie eine randomisierte Turingmaschine, genau wie der Verifizierer Bob. Sie versucht, Alice' Kommunikation mit Bob zu simulieren. Ein IP-Protokoll hat die *Zero-Knowledge-Eigenschaft*, falls die Information, die in Malice' simuliertem Protokoll kommuniziert wird, nicht von der Information zu unterscheiden ist, die in Alice' Originalprotokoll kommuniziert wird. Malice, die das Geheimnis nicht kennt, kann natürlich keinerlei Information über das Geheimnis in ihr simuliertes Protokoll stecken. Dennoch ist sie in der Lage, einen Klon des Originalprotokolls zu erzeugen, der von einem unabhängigen Beobachter nicht vom Original zu unterscheiden ist. Es folgt, dass der Verifizierer Bob (oder irgendeine andere Partei wie z.B. der betrügerische Erich) keinerlei Information aus dem Original herausziehen kann. Kurz gesagt: Wo nichts drin ist, kann man auch nichts herausholen.

Diese Intuition kann nun so formalisiert werden: Ein interaktives Beweissystem (A, B) für ein Problem L ist genau dann ein *Zero-Knowledge-Protokoll für* L, wenn es einen Simulator Malice gibt, so dass gilt:

- Malice arbeitet wie eine randomisierte Polynomialzeit-Turingmaschine M und simuliert Alice' Protokoll mit Bob, wobei ein simuliertes Protokoll (M, B) entsteht;
- für jedes $x \in L$ sind die Tupel $(a_1, a_2, \ldots, a_k)$ und $(m_1, m_2, \ldots, m_k)$, die die Kommunikation in (A, B) bzw. in (M, B) repräsentieren, identisch verteilt über die Münzwürfe von A und B im Protokoll (A, B) bzw. von M und B im Protokoll (M, B).

Diese Definition wird in der Literatur als „*honest-verifier perfect zero-knowledge*“ bezeichnet. Das heißt zum einen, dass in der obigen Definition von einem „ehrlichen“ Verifizierer ausgegangen wird; zum anderen, dass man eine „vollkommene“ Übereinstimmung (bzgl. der entsprechenden Wahrscheinlichkeitsverteilungen) der kommunizierten Information im simulierten und im Originalprotokoll fordert. In den meisten kryptographischen Anwendungen kann man jedoch nicht voraussetzen, dass Bob immer ehrlich ist. Deshalb modifiziert man die obige Definition durch die stärkere Forderung, dass es *für jeden* Verifizierer B^* einen Simulator M^* geben soll, der ein vom Originalprotokoll nicht zu unterscheidendes simuliertes Protokoll erzeugt. Andererseits gibt es auch andere, schwächere Begriffe für Zero-Knowledge, bei denen die Ununterscheidbarkeit der kommunizierten Information nicht „perfekt“ ist, wie z.B. „*statistical zero-knowledge*“ und „*computational zero-knowledge*“.

2.3 Zero-Knowledge-Protokoll für das Graphisomorphie-Problem

Oded Goldreich, Silvio Micali und Avi Wigderson [12] entwickelten das in Abbildung 4 dargestellte Zero-Knowledge-Protokoll für `GI`. Da bislang nur Zero-Knowledge-Protokolle für Probleme, die sowohl in NP als auch in coNP liegen, bekannt waren und da `GI` vermutlich nicht in coNP liegt, war ihr Resultat eine Überraschung.

Alice' Geheimnis im Protokoll ist der von ihr gewählte Isomorphismus π. Das Protokoll ist korrekt, weil Alice ihr Geheimnis π und die von ihr gewählte zufällige Permutation ρ kennt und somit leicht den Isomorphismus σ mit $\sigma(G_b) = H$ berechnen kann, um Bob ihre Identität zu beweisen. Dabei muss sie ihr Geheimnis π nicht verraten. G_1 und G_2 sind isomorph, also gibt es ein A mit $\Pr((A, B)(G_1, G_2) = 1) = 1$, und die Implikation (2.1) gilt. Andererseits hat Alice selbst die beiden Graphen isomorph gewählt, so dass der Fall $(G_1, G_2) \notin \texttt{GI}$ nicht eintreten kann und die Implikation (2.2) trivialerweise gilt. Somit ist das Protokoll ein interaktives Beweissystem.

Nehmen wir nun an, dass Erich oder Malice betrügen und sich als Alice ausgeben wollen. Sie kennen ihren geheimen Isomorphismus π nicht, aber sie kennen die öffentlichen Graphen G_1 und G_2. Sie würden Bob gern davon überzeugen, dass sie Alice' Geheimnis π kennen, das zum Paar (G_1, G_2) passen muss. Wenn Bobs Bit b zufällig mit ihrem zuvor gewählten Bit a übereinstimmt, dann gewinnen sie. Ist jedoch $b \neq a$, dann erfordert die Berechnung von $\sigma = \rho \circ \pi$ oder $\sigma = \rho \circ \pi^{-1}$ die Kenntnis von π. Ohne π zu kennen, ist die Berechnung von π nur aus den öffentlichen Graphen G_1 und G_2 so gut wie unmöglich, weil `GI` ein schweres Problem ist, zu schwer selbst für randomisierte Polynomialzeit-Maschinen. Sind die Graphen groß genug gewählt, werden Betrüger wie Malice oder Erich in diesem Falle scheitern.

Da ihre beste Chance ist, das Bit b zu raten, können sie höchstens mit Wahrscheinlichkeit $\frac{1}{2}$ betrügen. Natürlich können sie das Bit b immer raten, und somit ist ihre Chance, erfolgreich zu betrügen, genau $\frac{1}{2}$. Verlangt Bob, dass, sagen wir, k unabhängige Runden des Protokolls ausgeführt werden, dann wird die

			
	Erzeugung zweier isomorpher Graphen und eines geheimen Isomorphismus		
1	wählt einen großen Graphen G_1, eine zufällige Permutation π auf den Knoten von G_1 und den Graphen $G_2 = \pi(G_1)$; G_1, G_2 sind öffentlich, π ist privat		
	Protokoll		
2	wählt zufällige Permutation ρ der Knoten von G_1 und Bit $a \in \{1, 2\}$, berechnet $H = \rho(G_a)$		
3		$\overset{H}{\Rightarrow}$	
4			wählt zufälliges Bit $b \in \{1, 2\}$ und möchte einen Isomorphismus zwischen G_b und H sehen
5		$\overset{b}{\Leftarrow}$	
6	berechnet die Permutation $\sigma = \begin{cases} \rho & \text{falls } b = a \\ \rho \circ \pi & \text{falls } 1 = b \neq a = 2 \\ \rho \circ \pi^{-1} & \text{falls } 2 = b \neq a = 1, \end{cases}$ die $\sigma(G_b) = H$ erfüllt		
7		$\overset{\sigma}{\Rightarrow}$	
8			verifiziert, dass tatsächlich $\sigma(G_b) = H$ gilt, und akzeptiert entsprechend

Abbildung 4: Zero-Knowledge-Protokoll für GI von Goldreich, Micali und Wigderson.

Betrugswahrscheinlichkeit so klein wie 2^{-k}. Es ist also sehr wahrscheinlich, dass ein Betrüger erwischt wird. Nicht mehr als 20 Runden des Protokolls bewirken, dass die Chancen der betrügerischen Malice, unentdeckt davon zu kommen, schlechter als eins zu einer Million sind. Folglich ist das Protokoll korrekt.

Es bleibt zu zeigen, dass das Protokoll in Abbildung 4 zero-knowledge ist. Abbildung 5 zeigt ein simuliertes Protokoll mit Malice, die Alice ersetzt, ohne ihr Geheimnis π zu kennen. Die Information, die in einer Runde des Protokolls kommuniziert wird, ist durch ein Tripel der Form (H, b, σ) gegeben. Jedes Mal, wenn Malice ein Bit a mit $a = b$ wählt, sendet sie einfach $\sigma = \rho$ an Bob und gewinnt: Bob, oder irgendein anderer unabhängiger Beobachter, wird nicht entdecken, dass sie in Wirklichkeit Malice ist. Im anderen Falle, immer wenn $a \neq b$ gilt, hat Malice Pech. Das ist jedoch überhaupt kein Problem: Sie löscht einfach diese misslungene Runde aus dem simulierten Protokoll und wiederholt den Versuch. In dieser Art und Weise kann sie eine Folge von Tripeln der Form (H, b, σ) erzeugen, die von der entsprechenden Folge von Tripeln im Originalprotokoll zwischen Alice und Bob nicht zu unterscheiden ist. Es folgt, dass das Protokoll von Goldreich, Micali und Wigderson für GI die Zero-Knowledge-Eigenschaft hat.

	 Malice		 Bob
	Simulierte Erzeugung zweier isomorpher Graphen		
1	kennt das öffentliche Paar (G_1, G_2) isomorpher Graphen, kennt nicht Alice' Geheimnis π mit $\pi(G_1) = G_2$		
	Simuliertes Protokoll		
2	wählt zufällige Permutation ρ der Knoten von G_1 und Bit $a \in \{1, 2\}$, berechnet $H = \rho(G_a)$		
3		$\overset{H}{\Rightarrow}$	
4			wählt zufälliges Bit $b \in \{1, 2\}$ und möchte einen Isomorphismus zwischen G_b und H sehen
5		$\overset{b}{\Leftarrow}$	
6	falls $b \neq a$, dann löscht M alle in dieser Runde übertragene Information und wiederholt; falls $b = a$, dann schickt M $\sigma = \rho$		
7		$\overset{\sigma}{\Rightarrow}$	
8			$b = a$ impliziert $$\sigma(G_b) = H,$$ Bob akzeptiert Malice als Alice

Abbildung 5: Simulation des Zero-Knowledge-Protokolls für GI, ohne Kenntnis von π.

2.4 Zero-Knowledge-Protokoll von Feige, Fiat und Shamir

Amos Fiat und Adi Shamir [9] entwarfen ein Schema zur Authentikation und für digitale Signaturen, welches von Uriel Feige, Fiat und Shamir [8] zu einem Zero-Knowledge-Identifikationsprotokoll modifiziert wurde. Genauer gesagt entwickelten sie ein Zero-Knowledge-Protokoll für ein zahlentheoretisches Problem, das auf der Annahme beruht, dass das Wurzelziehen in $\mathbb{Z}_n^*$ für sehr große Zahlen n praktisch nicht machbar ist. Wegen seiner Eigenschaften ist das Protokoll von Feige, Fiat und Shamir besonders gut für die Authentifizierung von Nutzern in großen Computer-Netzwerken geeignet. Es ist ein Public-Key-Protokoll, es ist effizienter als die meisten anderen Public-Key-Protokolle wie z.B. RSA, es kann auf einer Chip-Karte implementiert werden, und es ist zero-knowledge. Diese Vorteile führten zu einem raschen Einsatz des Protokolls in praktischen Anwendungen. Das Feige-Fiat-Shamir-Protokoll ist in das „Videocrypt" Pay-TV-System [7] integriert.

Das Protokoll von Feige, Fiat und Shamir ist in Abbildung 6 dargestellt. Es ist korrekt, weil Alice das von ihr gewählte Geheimnis $s \in \mathbb{Z}_n^*$ kennt. Folglich kann sie $y = r \cdot s^b$ berechnen, wobei b das von Bob zufällig gewählte Bit ist. Bob akzeptiert Alice' Identität, weil in $\mathbb{Z}_n^*$ gilt:

$$y^2 \equiv (r \cdot s^b)^2 \equiv r^2 \cdot s^{2b} \equiv r^2 \cdot v^b \equiv x \cdot v^b \mod n.$$

Nehmen wir nun an, dass Erich oder Malice betrügen und sich als Alice ausgeben wollen. Sie kennen ihr Geheimnis s nicht und ebensowenig die Primzahlen p und q, aber sie kennen die öffentlichen Zahlen $n = pq$ und $v = s^2 \mod n$. Sie möchten Bob davon überzeugen, dass sie Alice' Geheimnis s kennen, die

			
	Schlüsselerzeugung		
1	wählt zwei große Primzahlen p und q und Geheimnis $s \in \mathbb{Z}_n^*$, $n = pq$, berechnet $v = s^2 \mod n$; p, q und s sind geheim, n und v sind öffentlich		
	Protokoll		
2	wählt zufälliges $r \in \mathbb{Z}_n^*$ und berechnet $x = r^2 \mod n$		
3		$\overset{x}{\Rightarrow}$	
4			wählt zufälliges Bit $b \in \{0,1\}$
5		$\overset{b}{\Leftarrow}$	
6	berechnet $y = r \cdot s^b \mod n$		
7		$\overset{y}{\Rightarrow}$	
8			verifiziert, dass tatsächlich $y^2 \equiv x \cdot v^b \mod n$ gilt, und akzeptiert entsprechend

Abbildung 6: Zero-Knowledge-Protokoll von Feige, Fiat und Shamir.

Quadratwurzel von v modulo n. Falls Bob zufällig das Bit $b = 0$ wählt, so gilt $y = r \cdot s^0 = r$, und sie gewinnen. Ist jedoch $b = 1$, dann erfordert die Berechnung eines y mit $y^2 \equiv x \cdot v \mod n$ die Kenntnis des Geheimnisses s, vorausgesetzt, dass das Berechnen von Quadratwurzeln modulo n für große n sehr schwer ist. Wären Malice oder Erich auch ohne Kenntnis von s dazu in der Lage, die korrekte Antwort sowohl für $b = 0$ als auch für $b = 1$ zu geben – sagen wir, y_b mit $y_b^2 \equiv x \cdot v^b \mod n$ –, dann könnten sie Quadratwurzeln modulo n wie folgt effizient berechnen: die Kongruenzen $y_0^2 \equiv x \mod n$ und $y_1^2 \equiv x \cdot v \mod n$ implizieren $(\frac{y_1}{y_0})^2 \equiv v \mod n$; folglich ist $\frac{y_1}{y_0}$ eine Quadratwurzel von v modulo n.

Somit können sie höchstens mit Wahrscheinlichkeit $\frac{1}{2}$ betrügen. Natürlich können sie stets das Bit b zuvor raten und ihre Antwort entsprechend präparieren. Die Wahl von $x = r^2 \cdot v^{-b} \mod n$ und $y = r$ impliziert, dass gilt:

$$y^2 \equiv r^2 \equiv r^2 \cdot v^{-b} \cdot v^b \equiv x \cdot v^b \mod n. \tag{2.3}$$

In diesem Fall wird Bob keinerlei Irregularität entdecken, und er akzeptiert. Die Betrugswahrscheinlichkeit ist damit exakt $\frac{1}{2}$ und kann wie üblich durch k unabhängige Runden des Protokolls beliebig klein gemacht werden.

Es bleibt zu zeigen, dass das Feige-Fiat-Shamir-Protokoll in Abbildung 6 die Zero-Knowledge-Eigenschaft hat. Abbildung 7 zeigt ein simuliertes Protokoll mit Malice, die Alice ersetzt, ohne ihr Geheimnis s zu kennen. Die Information, die in einer Runde des Protokolls kommuniziert wird, ist durch ein Tripel der Form (x, b, y) gegeben. Zusätzlich zum zufällig gewählten $r \in \mathbb{Z}_n^*$ rät Malice ein Bit $c \in \{0,1\}$, berechnet $x = r^2 \cdot v^{-c} \mod n$ und schickt dieses x an Bob. Stimmt c zufällig mit Bobs Bit b überein, so schickt Malice einfach $y = r$ und gewinnt. Ein zur Gleichung (2.3) analoges Argument zeigt dann, dass weder Bob noch irgendein anderer unabhängiger Beobachter den Betrug aufdecken kann:

$$y^2 \equiv r^2 \equiv r^2 \cdot v^{-c} \cdot v^b \equiv x \cdot v^b \mod n.$$

	 Malice		 Bob
	Simulierte Schlüsselerzeugung		
1	kennt das öffentliche $n = pq$ und $v = s^2 \mod n$; kennt nicht die privaten Primzahlen p und q und Alice' Geheimnis s		
	Simuliertes Protokoll		
2	wählt zufälliges $r \in \mathbb{Z}_n^*$ und ein Bit $c \in \{0,1\}$, berechnet $x = r^2 \cdot v^{-c} \mod n$		
3		$\overset{x}{\Rightarrow}$	
4			wählt zufälliges Bit $b \in \{0,1\}$
5		$\overset{b}{\Leftarrow}$	
6	falls $b \neq c$, dann löscht M alle in dieser Runde übertragene Information und wiederholt; falls $b = c$, so schickt M $y = r$		
7		$\overset{y}{\Rightarrow}$	
8			$b = c$ impliziert $y^2 = r^2 = r^2 v^{-c} v^b \equiv x \cdot v^b \mod n,$ Bob akzeptiert Malice als Alice

Abbildung 7: Simulation des Feige-Fiat-Shamir-Protokolls, ohne Kenntnis von s.

Im anderen Falle, immer wenn $c \neq b$ gilt, hat Malice Pech. Das ist jedoch überhaupt kein Problem: Sie löscht einfach diese misslungene Runde aus dem simulierten Protokoll und wiederholt den Versuch. In dieser Art und Weise kann sie eine Folge von Tripeln der Form (x, b, y) erzeugen, die von der entsprechenden Folge von Tripeln im Originalprotokoll zwischen Alice und Bob nicht zu unterscheiden ist. Es folgt, dass das Feige-Fiat-Shamir-Protokoll ein Zero-Knowledge-Protokoll ist.

3 Abschließende Bemerkungen und vergleichender Ausblick

In diesem Beitrag wurden verschiedene kryptographische Protokolle und Techniken vorgestellt. Diese sind jeweils auf spezifische Szenarien zugeschnitten, die bei der Kommunikation über unsichere Kanäle auftreten können. Speziell wurden Protokolle für den Schlüsseltausch bei symmetrischen Kryptosystemen sowie digitale Signaturen und Zero-Knowledge-Protokolle vorgestellt, und es wurden diesbezügliche Sicherheitsaspekte kurz diskutiert. Im Vordergrund stand dabei die Darstellung der Modelle und der mathematischen Grundlagen kryptographischer Techniken; Aspekte des *Security-Engineering* wie etwa der Entwurf von sicheren Public-Key-Infrastrukturen [4, Kapitel 14], die für die praktische Umsetzung der Modelle erforderlich sind, wurden nicht behandelt.

Natürlich bilden die hier vorgestellten kryptographischen Techniken nur einen kleinen, sehr selektiven Einblick in das weite Feld kryptographischer Anwendungen. So gäbe es etwa über digitale Signaturen, ihre Sicherheitseigenschaften (die auch formal gefasst werden müssen wie in der klassischen Definition in [14])

und ihre Bedeutung in großen Computer-Netzwerken und für sichere Internet-Technologien, noch viel mehr zu sagen. Schöne Übersichtsdarstellungen zu digitalen Signaturen mit Hinweisen zur weiterführenden Literatur findet man z.B. in den Büchern von Stinson [28, Kapitel 6] und von Buchmann [4, Kapitel 11]. Für ganz aktuelle Entwicklungen auf diesem sehr aktiven Forschungsgebiet sei beispielhaft auf eine Arbeit von Micali und Reyzin [18] verwiesen, in welcher eine „exaktere Sicherheit“ für dem Fiat-Shamir-Protokoll [9] ähnliche Verfahren zur Erstellung digitaler Signaturen untersucht wird.

Ebenso ist das Gebiet der Zero-Knowledge-Protokolle nur kurz angerissen worden. Für eine deutlich tiefer gehende Lektüre und eine mathematisch rigorose Behandlung der Zero-Knowledge-Protokolle sei auf das Kapitel 4 im Buch von Oded Goldreich [10] verwiesen. Eine schöne und lockere Darstellung findet man in den Büchern von Albrecht Beutelspacher et al. [3] und Uwe Schöning [25], von denen die Präsentation konkreter Zero-Knowledge-Protokolle in den Abschnitten 2.3 und 2.4 profitierte.

Ein wesentlicher Unterschied zwischen digitalen Signaturen und den Zero-Knowledge-Identifikationsprotokollen liegt darin, dass die einen dem Beweis der Echtheit von Unterschriften unter Dokumenten dienen, während die anderen für die Authentikation von Personen – von Instanzen, die ihr Wissen aktiv an andere weitergeben können – verwendet werden. Gewissermaßen verhalten sich digitale Signaturen und Zero-Knowledge-Identifikationsprotokolle dual zueinander. Digitale Signaturen leisten deutlich mehr hinsichtlich der *Authentizität*: (a) sie authentifizieren nicht nur den Urheber einer Nachricht, sondern beweisen auch die Echtheit der Nachricht selbst; (b) die unter (a) genannten Eigenschaften sind transferierbar, d.h., auch Dritte können sich von ihnen überzeugen. Demgegenüber verfolgen Zero-Knowledge-Protokolle ein entgegengesetztes Ziel: (c) sie leisten perfekt und beweisbar die *Geheimhaltung* des zur Authentikation notwendigen Geheimnisses; (d) das mittels eines Zero-Knowledge-Verfahrens erlangte Wissen bleibt in dem Sinne „geheim“, dass es nicht in beweisbarer Form an Dritte weitergegeben werden kann.

Danksagung. Ich danke Andreas Pfitzmann für seine interessanten und sehr hilfreichen Kommentare zu diesem Text und insbesondere zum Vergleich von digitalen Signaturen und Zero-Knowledge-Identifikationsprotokollen, welche die Darstellung deutlich verbessert haben.

Literatur

[1] M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, 1996.

[2] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[3] A. Beutelspacher, J. Schwenk, and K. Wolfenstetter. *Moderne Verfahren der Kryptographie*. Vieweg, 4th edition, 2001. In German.

[4] J. Buchmann. *Einführung in die Kryptographie*. Springer-Verlag, second edition, 2001. In German.

[5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[6] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[7] M. Cohen and J. Hashkes. A system for controlling access to broadcast transmissions. European Patent Application 0 428252 A2, May 1991.

[8] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO '86*, pages 186–194. Springer-Verlag *Lecture Notes in Computer Science #263*, 1986.

[10] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[11] O. Goldreich, Y. Mansour, and M. Sipser. Interactive proof systems: Provers that never fail and random selection. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 449–461. IEEE Computer Society Press, October 1987.

[12] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.

[13] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[14] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[15] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.

[16] L. Hemaspaandra and J. Rothe. Creating strong, total, commutative, associative one-way functions from any one-way function in complexity theory. *Journal of Computer and System Sciences*, 58(3):648–659, 1999.

[17] L. Hemaspaandra and J. Rothe. Characterizing the existence of one-way permutations. *Theoretical Computer Science*, 244(1–2):257–261, 2000.

[18] S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1):1–18, 2002.

[19] National Institute of Standards and Technology (NIST). Digital signature standard (DSS). *Federal Register*, 56(169), August 1991.

[20] M. Rabi and A. Sherman. An observation on associative one-way functions in complexity theory. *Information Processing Letters*, 64(5):239–244, 1997.

[21] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[22] J. Rothe. Some facets of complexity theory and cryptography: A five-lectures tutorial. Technical Report cs.CC/0111056, Computing Research Repository (CoRR), November 2001. 50 pages. Available on-line at http://xxx.lanl.gov/abs/cs.CC/0111056.

[23] J. Rothe and L. Hemaspaandra. On characterizing the existence of partial one-way permutations. *Information Processing Letters*, 82(3):165–171, May 2002.

[24] C. Schnorr. Efficient identification and signature schemes for smart cards. In *CRYPTO '89*, pages 239–251. Springer-Verlag *Lecture Notes in Computer Science #435*, February 1990.

[25] U. Schöning. *Perlen der Theoretischen Informatik*. BI Wissenschaftsverlag, Mannheim, 1995.

[26] A. Shamir. IP=PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[27] S. Singh. *Fermat's Last Theorem*. John Lynch, 1997.

[28] D. Stinson. *Cryptography Theory and Practice*. CRC Press, Boca Raton, 1995.