Optimistische Mehrparteien-Vertragsunterzeichnung*

N. Asokan† Birgit Baum-Waidner§
Matthias Schunter‡ Michael Waidner¶

Zusammenfassung

Ein Vertrag ist ein unwiderruflicher Nachweis einer Einigung auf einen Vertragstext. Mit einem Vertrag können die Unterzeichner diese Einigung gegenüber beliebigen Instanzen, wie z.B. einem Gericht, nachweisen.

Ein Vertragsunterzeichnungsprotokoll (engl. *contract signing*) dient dazu, solch einen Vertrag *fair* zu erzeugen, so daß garantiert ist, daß entweder alle oder keiner der Unterzeichner einen gültigen Vertrag erhält, selbst wenn $n-1$ von n Unterzeichnern betrügen.

Ein sicheres Vertragsunterzeichnungsprotokoll ist *optimistisch*, falls eine als korrekt vorausgesetzte Dritte Partei nur bei Betrugsversuchen eingeschaltet wird. Angesichts der Tatsache, daß keine praktikablen Protokolle ganz ohne Dritte Partei existieren, scheint dies der praktikabelste Ansatz zu sein.

In diesem Beitrag beschreiben wir ein optimistisches Mehrparteien-Vertragsunterzeichnungsprotokoll (kurz *MPVU*). Dieses ist nur um den Faktor 2-3 schlechter als das triviale nicht-optimistische Protokoll.

Desweiteren zeigen wir, wie Vertragsunterzeichnung als Baustein zur Lösung anderer Atomizitätsprobleme wie *Certified Mail Fairer Austausch von Unterschriften* sowie *Fairer Austausch von Gütern* genutzt werden kann.

*Diese Arbeit wurde teilweise vom ACTS Projekt AC026, *SEMPER* unterstützt. Für den Inhalt sind jedoch die Autoren verantwortlich.

†Nokia Research, Helsinki, <asokan@research.nokia.com>.

‡Universität des Saarlandes, Saarbrücken, <schunter@acm.org>.

§Entrust Technologies Europe, Zürich, <birgit.baum@entrust.com>.

¶IBM Zurich Research Laboratory, Rüschlikon, <wmi@zurich.ibm.com>.

1 Einleitung

1.1 Vertragsunterzeichnung und zertifiziertes Commit

Um nachweisbare *Atomizität* zu garantieren, müssen sich die Teilnehmer über den geplanten Verlauf einer Geschäftstransaktion einigen:

1. Alle Teilnehmer müssen sich einigen, ob die Transaktion fortgeführt oder abgebrochen werden soll.
2. Falls die Transaktion nicht abgebrochen wird, muß die Entscheidung über den geplanten Verlauf später gegenüber Dritten (z.B. Notar, Gericht) nachweisbar sein.

Die Teilnehmer an einer Geschäftstransaktion werden sich in der Regel nicht vertrauen. Daher sollte die zweite Anforderung auch garantiert sein, falls $n - 1$ von n der Teilnehmer sich gegen einen ehrlichen Teilnehmer verbünden.

Ein Beispiel hierfür sind elektronische Zahlungen: Vor einer Zahlung müssen sich Händler und Kunde nachweisbar einigen, ob die Zahlung erfolgen soll, oder ob das Geschäft abgebrochen werden soll.

Ein weiteres Beispiel ist Vertragsunterzeichnung: Nach einer informellen Einigung auf einen potentiellen Vertragstext müssen alle potentiellen Unterzeichner für sich entscheiden, ob sie den Vertrag unterzeichnen (Entscheidung signed) oder ablehnen (Entscheidung rejected) wollen. Nach einer Unterzeichnung muß jeder in der Lage sein, seinen Vertrag bei einem beliebigen Teilnehmer, genannt „*Gericht*" (engl. *verifier*), vorzuzeigen.

Hierbei scheint die Vertragsunterzeichnung der wichtigste Baustein zur Lösung des allgemeinen Problems zu sein: Wie die üblichen commit-Protokolle [15] garantiert Vertragsunterzeichnung eine Einigung mit einer sicheren Rückfallposition rejected ($\cong$ abort) und garantiert zusätzlich auch die spätere Nachweisbarkeit einer positiven Entscheidung signed ($\cong$ commit).

Ein triviales Vertragsunterzeichnungsprotokoll [18] nutzt einen Teilnehmer namens Dritte Partei[1], auch wenn alle Unterzeichner korrekt sind: Diese wartet auf den Eingang aller digitalen Signaturen unter dem Vertragstext und leitet diese nach Erhalt an alle Unterzeichner weiter. Falls nicht alle Signaturen rechtzeitig eintreffen, wird der komplette Vertrag verworfen. Dieses Protokoll benötigt $2n$ Nachrichten und hat den

[1] Es wird angenommen, daß diese Dritte Partei und das Gericht korrekt sind.

Nachteil, daß durch die Teilnahme der Dritten Partei an allen Vertragsunterzeichnungen die Geheimhaltung der Verträge eingeschränkt wird und die Dritte Partei als Engpaß die Effizienz beschränkt.

Optimistische Protokolle vermeiden diesen Engpaß: Wie in der trivialen Lösung basiert die Sicherheit auf einer Dritten Partei. Diese wird jedoch nur bei Betrugsversuchen in Anspruch genommen. Da dies (hoffentlich) die Ausnahme ist, steigert dies die Effizienz und Geheimhaltung der resultierenden Protokolle, da die Dritte Partei in der Regel nicht am Protokoll teilnimmt.

In Abschnitt 2 definieren wir den Begriff (optimistische) Mehrparteien-Vertragsunterzeichnung (kurz *MPVU*). In Abschnitt 3 stellen wir ein neues[2] optimistisches MPVU vor, welches mit maximal $6n - 4$ Nachrichten in 6 Runden oder $4n - 4$ Nachrichten in 4 Runden bei ehrlichen Unterzeichnern nur maximal zwei bis drei mal weniger effizient als die triviale Lösung ist.

Das Protokoll ist nur für synchrone Netzwerke geeignet. Ein in [6] beschriebenes asynchrones Protokoll ist mit $O(n^3)$ Nachrichten in $O(n)$ Runden oder $O(n^2)$ Nachrichten in $O(n^2)$ Runden weit weniger effizient.

In Abschnitt 4 beschreiben wir Lösungen für andere Atomizitätsprobleme, welche auf optimistischer Vertragsunterzeichnung basieren und skizzieren ein generisches Protokoll für fairen Austausch von beliebigen Gütern. Dies erweitert die in [2, 4, 5] vorgeschlagenen Zweiparteienprotokolle für diese Probleme.

1.2 Zweiparteien-Vertragsunterzeichnung

Zweiparteien-Vertragsunterzeichnung ist bei hoher Nachrichtenkomplexität auch ohne Dritte Partei möglich, falls ein nicht-vernachlässigbarer Fehler (linear in der Anzahl der Runden) toleriert wird [7] oder angenommen wird, daß alle Teilnehmer äquivalente Rechenleistung besitzen [11] und die Rechenzeit des Protokolls nicht polynomiell beschränkt ist.

Das erste optimistische Zweiparteienprotokoll mit hoher Nachrichtenkomplexität wurde in [7] vorgeschlagen. Die Anzahl der Nachrichten wurde in [2, 17] optimiert. Ein nachrichtenoptimales Protokoll für asynchrone Netzwerke wurde in [3] vorgestellt.

Bis jetzt wurden Vertragsunterzeichnungsprotokolle noch nicht zum zertifizierten Commit eingesetzt. *Generische* optimistische Zweiparteienprotokolle im Sinne von

[2] Eine Vorversion von Protokoll 3.1 wurde bereits in [1] beschrieben.

Abschnitt 4.4 wurden erstmals in [2, 3] vorgeschlagen. Optimistische Zweiparteienprotokolle für den fairen Austausch von Signaturen und für Certified Mail wurden erstmals in [4, 5, 16] beschrieben.

Keines der existierenden Zweiparteienprotokolle kann trivial auf den Mehrparteienfall erweitert werden. Einige nichtoptimistische Protokolle für MPVU wurden in [12, 14] beschrieben. Die erste optimistische Variante wurde in einer Vorversion dieses Beitrages skizziert [1]. Das erste asynchrone optimistische MPVU wurde in [6] vorgestellt.

1.3 Annahmen und Bezeichnungen

Wir bezeichnen die Unterzeichner mit $\mathsf{P}_1, \ldots, \mathsf{P}_n$, die Dritte Partei mit T und das Gericht mit V. Jeder Teilnehmer ist in der Lage, digitale Unterschriften zu erzeugen und die Signaturen aller zu prüfen [9, 13]. Eine digitale Signatur von P_x unter Nachricht m wird mit $\mathrm{sign}_X(m)$ bezeichnet. Die Teilnehmer eines Protokolls werden in eckigen Klammern angegeben.

Das Protokoll gliedert sich in synchrone Runden[3]. Jeder Teilnehmer kann in jeder Runde Nachrichten empfangen, bearbeiten und versenden. Wir nehmen an, daß die Übertragung einer Nachricht zwischen je zwei korrekten Teilnehmern zwischen zwei Runden garantiert wird.[4]

Wir nehmen einen festen Angreifer an, der vorweg wählt, welche Unterzeichner korrekt und welche inkorrekt sein sollen. Der Angreifer kann alle Nachrichten zwischen Unterzeichnern lesen und fälschen, sowie die Signaturen der inkorrekten Teilnehmer erzeugen. Nachrichten von und zur Dritten Partei und zum Gericht können nur gelesen und geschrieben, nicht aber überschrieben werden.

Für die meisten Sicherheitsanforderungen werden wir annehmen, daß bis zu $n-1$ Unterzeichner inkorrekt sind. Das Gericht V wird immer als korrekt angenommen, nimmt aber nicht an der Unterzeichnung teil. Fairness kann nur bei korrekter Dritter Partei T garantiert werden, jedoch können selbst bei betrügerischer Dritter Partei keine Verträge für Unbeteiligte gefälscht werden.

Man beachte, daß die beschriebenen Protokolle in vielerlei Hinsicht idealisiert sind: Für erhöhte Lesbarkeit verzichten wir auf die notwendige Typung, auf Zertifikate und Schlüsselaustausch, auf Zeitstempel von Nachrichten und time-outs, sowie auf wichtige Parameter wie die Identität der zu verwendenden Dritten Partei T.

[3] Dies ist das Standardmodell für synchrone Netzwerke [15].

[4] Diese Zuverlässigkeit wird nur benötigt, damit T bei n korrekten Teilnehmern nicht unnötigerweise teilnehmen muß. Ansonsten genügt die Zuverlässigkeit der Verbindung mit T.

2 Definitionen

Definition 2.1 (Mehrparteien-Vertragsunterzeichnung)
Ein System mit mindestens $n+1$ Teilnehmern $\mathsf{P}_1, \ldots, \mathsf{P}_n$ und V, welches die folgenden Anforderungen erfüllt, heißt Mehrparteien-Vertragsunterzeichnungsystem *(kurz* MPVU*):*

Ein MPVU besteht aus den zwei Protokollen $\mathrm{sign}[P_1, \ldots, P_n]$ *und* $\mathrm{verify}[P_i, V]$. *Falls* $\mathrm{sign}[]$ *einen weiteren Teilnehmer namens* T *einschließt, so bezeichnen wir das resultierende System als* MPVU mit Dritter Partei.[5] *Der Teilnehmer* V *besitzt einen festen Zustand, in dem das* $\mathrm{verify}[]$*-Protokoll gestartet wird, d.h., aufeinanderfolgende Verifikationen dürfen nur von einer Initialisierung, nicht aber voneinander abhängen. Dies modelliert die universelle Gültigkeit eines Vertrages bei beliebigen Gerichten zu jeder Zeit, da* V *außer Initialisierungsdaten wie Verifikationsschlüsseln kein Vorwissen über den Vertrag benötigt.*

Zum Starten des $\mathrm{sign}[]$*-Protokolls wird*

$$\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs})$$

bei Teilnehmer $\mathsf{P_i}$ *eingegeben. Hierbei ist P_i der eigene Namen eines Teilnehmers,* $\mathsf{id\_set} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ *ist der Vektor mit den Namen aller gewünschten Unterzeichner,* tid *ist eine Transaktionsnummer welche so gewählt wird, daß das Tupel* $(\mathsf{id\_set}, \mathsf{tid})$ *eindeutig für alle Ausführungen von* $\mathrm{sign}[]$ *ist,* contr *ist der zu unterzeichnende Vertragstext, und* $\mathsf{decs} \in \{\mathsf{sign}, \mathsf{reject}\}$ *ist die Entscheidung, ob dieser Vertrag unterzeichnet werden soll oder nicht. Nach Erhalt einer Eingabe prüft das System, ob Signaturen für den Namen P_i erzeugt werden können und ob* tid *noch nie für* $\mathsf{id\_set}$ *benutzt wurde. Schlagen diese Tests fehl, so wird das Protokoll beendet.*

Nach Beendigung gibt das $\mathrm{sign}[]$*-Protokoll*

$$(\mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, d_i)$$

mit P_i, und $d_i \in \{\mathsf{signed}, \mathsf{rejected}\}$ aus. Dies wird als „P_i entscheidet d_i für tid*" bezeichnet. Am Teilnehmer* T *erfolgen weder Ein- noch Ausgaben, da dessen Verhalten komplett durch das Protokoll bestimmt ist.*

Zum Starten des Protokolls $\mathrm{verify}[]$ *mit einem Gericht* V *wird bei einem Unterzeichner* $\mathsf{P_i}$

$$\mathsf{show}(\mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, V)$$

[5] T nimmt nie an $\mathrm{verify}[]$ teil, d.h., eine Teilnahme beschränkt sich immer auf $\mathrm{sign}[]$.

eingegeben. Das Gericht V *startet das* verify[]*-Protokoll mit folgender Eingabe unter Verwendung identischer Parameter:*

$$\mathsf{verify}(V, P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr})$$

Nach Beendigung gibt das Protokoll $(\mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, d_V)$ *mit* $d_V \in \{\mathsf{signed}, \mathsf{rejected}\}$ *an* V *aus. Dies wird als „*V *entscheidet* d_i *für* tid*" bezeichnet.* $\mathsf{P_i}$ *erzeugt keine Ausgaben.*

Ein MPVU muß folgende Sicherheitsanforderungen erfüllen:

1. Korrekte Ausführung. *Falls alle Teilnehmer* $\mathsf{P_i}$ *korrekt sind und jeweils die Eingabe* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs} = \mathsf{sign})$ *verarbeitet haben, dann entscheiden sie auf* signed.

2. Fälschungssicherheit. *Falls ein korrekter Unterzeichner* $\mathsf{P_i}$ *nie eine Eingabe* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs})$ *mit* $\mathsf{decs} = \mathsf{sign}$ *erhalten hat, dann wird jedes korrekte Gericht nach Eingabe von* $\mathsf{verify}(V, P_k, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr})$ *auf* rejected *für* tid *entscheiden, selbst falls* T *inkorrekt ist.*

3. Vorzeigbarkeit. *Falls ein korrekter Unterzeichner* $\mathsf{P_i}$ *nach einer Eingabe* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs})$ *auf* signed *entscheidet und später* $\mathsf{show}(\mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, V)$ *eingibt, dann entscheidet ein korrektes Gericht* V *nach einer Eingabe* $\mathsf{verify}(V, P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr})$ *auf* signed.

4. Keine Überraschung bei abgelehntem Vertrag. *Falls* T *korrekt ist und* $\mathsf{P_i}$ *nach einer Eingabe* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs})$ *mit* $\mathsf{decs} = \mathsf{sign}$ *auf* rejected *entschieden hat, dann wird ein Gericht* V *nach einer Eingabe* $\mathsf{verify}(V, P_k, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}_V)$ *nie auf* signed *entscheiden.*

5. Terminierung von sign[]. *Falls* T *korrekt ist, wird das Protokoll* sign[] *bei jedem korrektem Unterzeichner* $\mathsf{P_i}$ *nach einer begrenzten Rundenanzahl ein korrektes Ergebnis ausgeben.*

6. Terminierung von verify[]. *Ein Gericht wird nach einer Eingabe von* $\mathsf{verify}(V, P_k, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}_V)$ *nach einer begrenzten Rundenanzahl eine korrekte Entscheidung ausgeben.*

□

Definition 2.2 (Optimistisches Protokoll)
Ein Protokoll $\mathrm{sign}[\mathsf{P}_1, \ldots, \mathsf{P}_n, \mathsf{T}]$ *heißt* optimistisch, *falls es bei korrekten Unterzeichnern terminiert, ohne daß* T *Nachrichten sendet oder empfängt.*

□

Man beachte, daß sich optimistische und herkömmliche Protokolle nur intern, jedoch nicht in den Benutzerein- und ausgaben unterscheiden.

3 Optimistische Mehrparteien-Vertragsunterzeichnung

Wir stellen nun ein System zur optimistischen Mehrparteien-Vertragsunterzeichnung auf synchronen Netzwerken vor. Falls alle Unterzeichner korrekt sind, benötigt dieses Protokoll nur 2 Kommunikationsrunden: In der ersten Runde sendet jeder Teilnehmer, welcher einen Vertrag unterzeichnen will, ein „Unterschriftsversprechen" (= Nachricht $m_{1,i}$). In der zweiten Runde sendet jeder Teilnehmer, der n solche Versprechen erhalten hat, seine Unterschrift unter dem Vertrag (= Nachricht $m_{2,i}$). Dies führt zur Terminierung nach 2 Runden, falls alle Teilnehmer korrekt sind und den Vertrag unterzeichnen wollen.

Falls sich jedoch einzelne Teilnehmer inkorrekt verhalten, können einzelne Teilnehmer nach Runde 2 keinen vollständigen Vertrag erhalten haben. Diese Inkonsistenz wird in diesem Fall durch weitere 2 Runden behoben, in denen jeder Teilnehmer mit n Versprechen diese durch T in einen korrekten Vertrag wandeln lassen kann. Falls T solch eine Erklärung ausstellt, wird diese in Runde 4 an alle Teilnehmer verteilt. Dies garantiert, daß alle Teilnehmer (auch die, die bisher noch keine n Versprechen erhalten haben) einen gültigen Vertrag erhalten. Falls ein Teilnehmer sein Versprechen versandt hat und bis Runde 4 keinen Vertrag erhält, so entscheidet dieser auf $\mathsf{rejected}$.

Protokoll 3.1 (Synchrones Optimistisches MPVU)
Ein synchrones und optimistisches MPVU besitzt die folgenden Protokolle:

Unterzeichnung *mit* $c := (\mathsf{id\_set}, \mathsf{tid}, \mathsf{contr})$.

1. *(a) Falls für* $\mathsf{P_i}$ *die Eingabe* $\mathsf{decs} = \mathsf{reject}$ *ist, so gibt es* $\mathsf{rejected}$ *aus und hält an.*

   (b) Ansonsten sendet $\mathsf{P_i}$ *die Nachricht* $m_{1,i} := \mathrm{sign}_i(1, c)$ *an alle anderen Unterzeichner.*

2. *Jeder Unterzeichner* $\mathsf{P_i}$ *sammelt* $M_1 := (m_{1,1}, \ldots, m_{1,n})$.

   (a) *Falls dies gelingt und jedes* $m_{1,j}$ *korrekt ist, so sendet* $\mathsf{P_i}$ *die Nachricht* $m_{2,i} := \mathrm{sign}_i(2, c)$ *an alle anderen Unterzeichner.*

   (b) *Ansonsten wartet* $\mathsf{P_i}$ *auf eine Nachricht von* T *in Runde 4.*

3. *Jeder Unterzeichner* $\mathsf{P_i}$ *sammelt* $M_2 := (m_{2,1}, \ldots, m_{2,n})$.

   (a) *Gelingt dies und alle* $m_{2,j}$ *sind korrekt, so entscheidet* $\mathsf{P_i}$ *auf* signed *und hält an.*

   (b) *Falls ansonsten* $\mathsf{P_i}$ *die Nachricht* $m_{2,i}$ *geschickt hat, sendet sie eine Anfrage auf Vervollständigung* $m_{3,i} := \mathrm{sign}_i(3, M_1)$ *an* T.

4. *Falls* T *mindestens eine vollständige und korrekte Nachricht* $m_{3,i}$ *in Runde 3 erhält, so schickt* T *den Vertragsersatz* $m_T := \mathrm{sign}_T(M_1)$ *an alle Unterzeichner, welche nach deren Empfang auf* signed *entscheiden und halten.*

   Jeder Unterzeichner, welcher in Runde 4 auf Nachrichten von T *wartet und keine empfängt, entscheidet auf* rejected *und hält an.*

Vorzeigen eines Vertrages: *Ein Gericht* V *entscheidet auf* signed*, falls es eine komplette konsistente Menge* M_2 *oder eine Nachricht* m_T *mit einer vollständigen und konsistenten Menge* M_1 *vorgelegt bekommt.*

□

Falls alle Unterzeichner korrekt sind, verteilt jeder Unterzeichner in jeder Runde höchstens 2 Nachrichten. Für 2 Teilnehmer ist das Protokoll identisch mit dem nachrichtenoptimalen Protokoll aus [17]. Analog zu [17] folgt auch, daß das vorgestellte Protokoll rundenoptimal ist.

Selbst bei inkorrekten Unterzeichnern benötigt das Protokoll nicht mehr als 4 Runden. Falls alle Unterzeichner korrekt sind, werden $2n$ Nachrichten an $n-1$ Teilnehmer verteilt. Falls einzelne Teilnehmer betrügen, können schlimmstenfalls bis zu n Nachrichten vom Typ $m_{3,i}$ und die Verteilung der Antwort von T hinzukommen.

Bei Implementierung der Nachrichtenverteilung durch Senden von n Kopien ergibt dies $2n^2 - 2n$ Nachrichten im Normalfall und maximal $2n^2$ Nachrichten im „worst case". Die Anzahl der Nachrichten kann auf $4n - 4$ reduziert werden, falls alle Unterzeichner korrekt sind. In diesem Fall verringert sich der „worst case" Aufwand auf $6n - 4$ Nachrichten: Grundidee dieser Effizienzverbesserung ist es, alle Nachrichten über einen beliebigen Teilnehmer (hier z.B. $\mathsf{P_1}$) zu verteilen:

Protokoll 3.2 (Nachrichtenoptimierung von Protokoll 3.1)
Das nachrichtenoptimierte Protokoll unterscheidet sich von Protokoll 3.1 nur durch die geänderte Nachrichtenübermittlung: Runde k ($k = 1, 2$) von Protokoll 3.1 wird jeweils durch die folgenden zwei Runden ersetzt:

1. $\mathsf{P_i}$ *schickt* $m_{k,i}$ *nur an* $\mathsf{P_1}$.
2. $\mathsf{P_1}$ *sammelt* M_k.
   (a) *Falls dies gelingt, schickt* $\mathsf{P_1}$ *die gesammelten Mengen* M_k *an alle anderen Unterzeichner.*
   (b) *Ansonsten hält* $\mathsf{P_1}$ *an.*

□

Satz 3.1 *Die Protokolle 3.1 und 3.2 sind optimistische MPVU für synchrone Netzwerke.*

□

Beweis. Aus Sicherheitssicht sind die Protokolle 3.1 und 3.2 und deren Sicherheitsbeweise identisch. Der verwendete Rundenbegriff bezieht sich jeweils auf Protokoll 3.1.

- *Korrekte Ausführung*, *Vorzeigbarkeit* und *Terminierung* sind offensichtlich.
- *Fälschungssicherheit* wird dadurch garantiert, daß ein korrekter Vertrag Signaturen aller Unterzeichner enthalten muß.
- *Keine Überraschungen.* Angenommen ein korrektes Gericht V entscheidet signed für c. Falls dies aufgrund von m_T geschieht, so haben alle Unterzeichner m_T erhalten und auf signed entschieden, da T korrekt ist. Falls dies aufgrund von M_2 geschieht, so gilt für jeden korrekten Unterzeichner $\mathsf{P_i}$, daß M_2 dessen Signatur aus Runde 2 enthält. Daher hat $\mathsf{P_i}$ die Menge M_1 in Runde 1 erhalten. Falls es darüberhinaus auch M_2 erhalten hat, so hat es auf signed entschieden. Ansonsten schickt es $m_{3,i}$ an ein korrektes T, welches mit m_T antwortet, so daß dieser Teilnehmer signed ausgibt. Somit ist klar, daß alle Teilnehmer signed ausgegeben haben.

- *Optimistisch:* Falls alle Unterzeichner korrekt sind und mit $\mathsf{decs} = \mathsf{sign}$ und konsistenten Verträgen starten, so wird T nicht einbezogen und alle entscheiden auf signed in Runde 2. Wenn mindestens ein Unterzeichner $\mathsf{P_i}$ mit $\mathsf{decs}_i = \mathsf{reject}$ startet, dann fehlt mindestens ein $m_{1,i}$ und M_1 bleibt somit unvollständig. Daher wird kein Teilnehmer $m_{2,j}$ schicken und kein Teilnehmer wird bei T anfragen. Nach Runde 4 werden alle Unterzeichner dann auf rejected entscheiden.

■

4 Anwendungen

4.1 Grundsätzliches Vorgehen

In den folgenden Abschnitten werden wir drei Beispiele für optimistische Protokolle zur Lösung von bekannten Mehrparteienproblemen vorstellen, welche auf MPVU basieren. Diese sind intern jeweils in 3 Schritte strukturiert:

1. Alle Teilnehmer bereiten die Transaktion vor, ohne unwiderrufliche Veränderungen vorzunehmen. Hierbei wird keine Dritte Partei benötigt.

2. Ein optimistisches MPVU wird zur Unterzeichnung der Transaktionsdaten verwendet. Ein Teilnehmer unterzeichnet nur, falls Schritt 1 erfolgreich war.

3. (a) Falls das MPVU signed ausgegeben hat, wird die Transaktion abgeschlossen. Falls einzelne Teilnehmer inkorrekt sind, wird gegebenenfalls T einbezogen, um die Transaktion erfolgreich zu beenden. Hierbei wird die Gültigkeit der Transaktion durch den in Schritt 2 unterschriebenen Vertrag nachgewiesen.

   (b) Falls das MPVU rejected ausgegeben hat, wird die Transaktion abgebrochen. Dies erfordert die Rücknahme eventueller Änderungen aus Schritt 1.

Bei Ausführung von Schritt 3b muß T sicherstellen, daß kein Vertrag unterzeichnet wurde. Dies wird dadurch garantiert, daß T die Teilnehmer fragt, ob jemand einen gültigen Vertrag vorlegen kann.

4.2 Certified Mail

Beim herkömmlichen Certified Mail verschickt ein Sender P_1 eine Nachricht m an einen Empfänger P_2 im Tausch gegen eine Quittung. Hierbei ist sicherzustellen, daß P_2 die Quittung unabhängig vom Inhalt der Nachricht ausstellt. Mögliche Erweiterungen für mehr Teilnehmer sind:

- **1-to-n**: P_1 verschickt eine Nachricht m an $\mathsf{P}_2, \ldots, \mathsf{P}_n$ und erhält entweder von allen Quittungen, oder kein einziger Empfänger erhält Information über die Nachricht.
- **n-to-1**: Jeder Teilnehmer $\mathsf{P}_2, \ldots, \mathsf{P}_n$ schickt je eine Nachricht an P_1 und erwartet eine Quittung. Hierbei wird sichergestellt, daß entweder alle oder keine der Nachrichten ankommt.
- **n-to-n** Maximal kann jeder Teilnehmer Nachrichten an alle anderen Teilnehmer schicken. In diesem Fall, werden entweder alle $n(n-1)$ Quittungen erzeugt oder keine einzige Nachricht übertragen.

Jedes dieser Probleme kann man gemäß dem in Kapitel 4.1 vorgestellten Muster lösen. Als Beispiel beschreiben wir hier die Lösung für 1-to-n Certified Mail. Die vorgestellte Lösung basiert auf einem Public-Key Verschlüsselungssystem, welches sicher gegen adaptive chosen-ciphertext Angriffe ist [8, 10]. Einzig T benötigt ein Schlüsselpaar. Die leere Nachricht bezeichnen wir mit ε.

Protokoll 4.1 (Optimistisches 1-to-n Certified Mail)

1. P_1 *verschlüsselt* $(\mathsf{id\_set}, \mathsf{tid}, m)$ *mit dem öffentlichen Schlüssel von* T *und schickt den Schlüsseltext* $\mathsf{cipher} = E_T(\mathsf{id\_set}, \mathsf{tid}, m)$ *an alle.*

   Jede Partei P_i*, die* cipher *erhalten hat und eine Quittung ausstellen will, setzt* $\mathsf{decs} = \mathsf{sign}$ *und* $\mathsf{decs} = \mathsf{reject}$ *sonst. Danach wird das optimistische MPVU mit* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher}, \mathsf{decs})$ *gestartet.*

2. *Teilnehmer* P_1*:*

   (a) *Falls die Entscheidung von* P_1 *in Phase 2* signed *war, gibt sie Protokoll* accepted *aus. Der unterzeichnete Vertrag gilt als Quittung.*

   Teilnehmer P_1 *schickt* m *an alle Empfänger und beweist, daß der Klartext von* cipher *wirklich* $(\mathsf{id\_set}, \mathsf{tid}, m)$ *war (z.B. durch Mitteilung der Zufallswerte, so daß die Empfänger die Verschlüsselung wiederholen können).*

(b) Ansonsten, d.h., falls Teilnehmer $\mathsf{P_i}$ rejected *entschieden hat, gibt der Teilnehmer ebenso* rejected *aus und hält an.*

Teilnehmer $\mathsf{P_i}, i \neq 1$*:*

(a) Falls $\mathsf{P_i}$ *mit* $i > 1$ *ein korrektes Tripel* $(\mathsf{id\_set}, \mathsf{tid}, m)$ *erhalten hat, gibt es* m *aus und hält an.*

(b) Ansonsten zeigt $\mathsf{P_i}$ *den Vertrag* $(\mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher}, \mathsf{decs})$ *durch Eingabe von* $\mathsf{show}(\mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher}, T)$ *bei* T *vor.*

3. *Falls* T *einen korrekten Vertrag* $(\mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher})$ *von einem* $\mathsf{P_i} \in \mathsf{id\_set}$ *erhalten hat, versucht es* cipher *zu entschlüsseln.*

   (a) Falls die Entschlüsselung gelingt und $(\mathsf{id\_set}, \mathsf{tid}, m)$ *für das gegebene* $(\mathsf{id\_set}, \mathsf{tid})$ *ergibt, dann wird* $m_T := \mathrm{sign}_T(\mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher}, m)$ *gesetzt.*

   (b) Ansonsten gilt $m_T := \mathrm{sign}_T(\mathsf{id\_set}, \mathsf{tid}, \mathsf{cipher}, \varepsilon)$.

   T *schickt* m_T *an* $\mathsf{P_i}$.

□

Satz 4.1 (Sicherheit von Protokoll 4.1)

Protokoll 4.1 ist ein sicheres und optimistisches Protokoll für 1-to-n Certified Mail.

□

Beweis. (Skizze)

Um dies zu zeigen, müssen wir zeigen, daß ohne Ausstellung einer Quittung keine Information über die Nachricht bekannt wird und daß keine Quittung ausgestellt wird, falls die quittierte Nachricht nicht verschickt wurde.

- Information über cipher kann nur von $\mathsf{P_1}$ oder T stammen, da sonst Protokoll 4.1 zum Brechen des Verschlüsselungssystems nutzbar wäre.

  $\mathsf{P_1}$ verschickt m nur nach Erhalt eines gültigen Vertrags, d.h., nach Erhalt der Quittung.

  T entschlüsselt und verschickt m nur, falls die Bedingung in Schritt 3a erfüllt wurde. Dies bedeutet, daß der Vertrag und der verschlüsselte Text cipher in $(\mathsf{id\_set}, \mathsf{tid})$ übereinstimmen und somit zu einer Protokollausführung gehören. Somit war $\mathsf{P_1}$ in id_set wirklich der Absender von cipher und war willens, m zu schicken. Somit darf T cipher entschlüsseln.

- Da das MPVU sicher ist, kann eine Quittung nur durch Unterzeichnung eines Vertrages in Schritt 1 erzeugt werden. Auf der Grundlage dieses Vertrages kann dann jeder Teilnehmer $\mathsf{P_i}$ in Schritt 2 oder Schritt 3 die Nachricht erhalten.

■

4.3 Fairer Austausch von digitalen Signaturen

Beim fairen Austausch von digitalen Signaturen sendet jeder Teilnehmer $\mathsf{P_i}$ eine Signatur s_i unter einer Nachricht m_i, welche allen Teilnehmern bekannt ist. Nach erfolgreichem Abschluß des Protokolls erfährt ein Unterzeichner $\mathsf{P_i}$ entweder alle Signaturen $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$ der anderen Unterzeichner oder keine einzige.

Zweiparteienprotokolle hierfür wurden bereits in [4, 5] beschrieben. Die in [4] beschriebenen Verfahren basieren auf einer Variante von „überprüfbarer Verschlüsselung" von Signaturen (engl. *verifiable encryption*) [19]. Im Prinzip handelt es sich hierbei um Public-Key Verschlüsselung mit der zusätzlichen Eigenschaft, daß bewiesen werden kann, daß ein für T verschlüsselter Schlüsseltext $c = E_T(s, (i, w))$ wirklich einen vorgegebenen Wert w sowie eine korrekte Signatur s unter einem bekannten Text enthält, ohne daß der Empfänger Informationen über s erhält.

Das darauf aufbauende MPVU hat den Vorteil, daß sich ein damit erzeugter Vertrag nicht von einem herkömmlichen Vertrag unterscheidet, d.h., auch im Fehlerfall besteht ein Vertrag aus dem Vertragstext mit den Signaturen aller Unterzeichner.[6]

Das MPVU basierend auf überprüfbarer Verschlüsselung von digitalen Signaturen funktioniert wie folgt:

Protokoll 4.2 (Optimistischer Austausch von digitalen Signaturen)
Die Parameter des aktuellen Austausches umfassen $\mathsf{id\_set}$, tid, *alle Nachrichten* m_i *für* $P_i \in \mathsf{id\_set}$, *sowie die öffentlichen Schlüssel der Teilnehmer. Diese werden als Vertragstext* w *bezeichnet und sind allen Teilnehmern bekannt.*

Die Signatur von $\mathsf{P_i}$ *unter* m_i *bezeichnen wir als* s_i. *Anfangs ist* s_i *nur* $\mathsf{P_i}$ *bekannt.*

1. *Jeder Teilnehmer* $\mathsf{P_i}$ *schickt eine überprüfbare Verschlüsselung* $c_i = E_T(s_i, (i, w))$ *an jeden anderen Teilnehmer* $\mathsf{P_j}$ *und beweist diesem, daß der*

[6] In Abschnitt 3 ist dies nicht der Fall, da ein gültiger Vertrag entweder die Form M_2 oder m_T hat.

Schlüsseltext die gewünschte Signatur und das bekannte w enthält.[7] *Bei erfolgreichem Abschluß dieser Beweise setzen die Teilnehmer* $\mathsf{decs} = \mathsf{sign}$. *Bei Fehlern setzen sie* $\mathsf{decs} = \mathsf{reject}$.

2. *Jeder Teilnehmer* $\mathsf{P_i}$ *startet ein MPVU mit der Eingabe* $\mathsf{sign}(P_i, \mathsf{id\_set}, \mathsf{tid}, \mathsf{contr}, \mathsf{decs})$ *und dem Vertragstext* $\mathsf{contr} = w$.

3. *Falls Teilnehmer* $\mathsf{P_i}$ *in Schritt 2 auf* signed *entscheidet, schickt er seine Signatur s_i an alle und wartet auf deren Signaturen. Falls eine dieser Signaturen nicht geschickt wird, so zeigt* $\mathsf{P_i}$ *den Vertrag mit dem Vertragstext w sowie den Schlüsseltext c_j bei* T *vor.*

   T *entschlüsselt c_j und erhält $(s_x, (x, w_x))$. Falls $w_x = w$ und s_x eine gültige Signatur auf m_x ist, wobei m_x die in w enthaltene Nachricht ist, so schickt* T *die Signatur s_x an* $\mathsf{P_i}$.[8].

□

Der Sicherheitsbeweis erfolgt analog zu Satz 4.1. Hierbei ist zu beachten, daß jeder Schlüsseltext den Vertrag w enthält, der sicherstellt, daß T den Schlüsseltext c_i nur bei Vorlage des korrekten Vertrages w entschlüsselt.

4.4 Fairer Austausch von Gütern

Abschließend beschreiben wir ein generisches Protokoll zum Austausch beliebiger digitaler Güter, welche eine der folgenden Eigenschaften erfüllen [2, 3]:

- *Annullierbare Güter* (engl. *revocable items*) garantieren, daß die Dritte Partei diese bei Fehlern annullieren kann. Dies ist bei Zahlungssystemen üblich.

- *Ersetzbare Güter* (engl. *generateable items*) können im Fehlerfall unter einer vom Sender festgelegten Bedingung von der Dritten Partei ersetzt werden. Diese Bedingung wird in einem vorweg auszuführenden Protokoll festgelegt, indem der Empfänger auch die später für eine eventuelle Ersetzung benötigte Information sammelt.

  Ein Beispiel ist die überprüfbare Verschlüsselung, die Ersetzbarkeit von Signaturen garantiert. Die Bedingung ist, daß ein gültiger Vertrag w vorgelegt wurde.

[7] Falls das Protokoll zur überprüfbaren Verschlüsselung keinen Mehrfachbeweis für einen Schlüsseltext zuläßt, so muß für jeden Empfänger ein neuer Schlüsseltext berechnet werden.

[8] Je nach Anwendung muß $\mathsf{P_i}$ vorher beweisen, daß er in $\mathsf{id\_set}$ aufgeführt ist.

- *Weiterleitbare Güter* (engl. *forwardable items*) können sowohl direkt als auch via die Dritte Partei vom Sender S an den Empfänger R versendet werden, wobei T die Korrektheit der Übertragung überprüfen kann. Hierbei nehmen wir an, daß ein Gut zuerst direkt und später via T übertragen werden kann, ohne daß der Empfänger mehr als ein Gut erhält.

Güter mit diesen Eigenschaften können durch das im folgenden beschriebene generische Mehrparteien-Austauschprotokoll ausgetauscht werden. Einfachheitshalber nehmen wir an, daß jeder Teilnehmer nur ein Gut versendet.

Protokoll 4.3 (Optimistischer Austausch von Gütern)
Wir bezeichnen die Parteien, welche annullierbare, ersetzbare oder weiterleitbare Güter versenden wollen, jeweils mit P_R, P_G, P_F. Die Anzahl der Teilnehmer sei $|P_R| + |P_G| + |P_F| = n$ mit $|P_F| \leq 1$.

1. *Jeder Teilnehmer $P_r \in P_R$ schickt seine Güter an alle Empfänger. Jeder Teilnehmer $P_g \in P_G$ schickt die für den Ersatz durch* T *nötige Information an alle Empfänger. Jede Partei prüft die empfangenen Daten und setzt* decs_i *dementsprechend.*

2. *Jeder Teilnehmer* $\mathsf{P_i}$ *führt das optimistische MPVU mit den Parametern* $\mathsf{sign}(P_i, \mathsf{id\_set}_i, \mathsf{tid}_i, \mathsf{contr}_i, \mathsf{decs}_i)$ *aus, wobei* contr_i *die Güter der einzelnen Teilnehmer sowie die in Schritt 1 erhaltenen Daten eindeutig fixiert.*

3. *Falls Schritt 2 einen Vertrag mit der erwarteten Menge $P_F = \{P_f\} \neq \emptyset$ ergeben hat, dann versendet P_f sein Gut an den Empfänger. Falls dieser das Gut von P_f nicht erhält, zeigt er den Vertrag bei* T *vor.* T *fordert P_f auf, den Versand via* T *zu wiederholen. Falls die Wiederholung nicht korrekt ist, schickt* T *eine Abbruchnachricht an alle Teilnehmer.*

4. (a) *Falls Schritt 2* signed *ausgegeben hat und* T *keine Abbruchnachricht versendet hat, dann versenden alle $P_g \in P_G$ ihre Güter. Falls eine Partei das erwartete Gut nicht erhält, zeigt es den Vertrag bei* T *vor, welcher das Gut ersetzt. Dieser Vertrag muß hinreichend genau sein, um die Ersatzbedingung des Senders prüfen zu können.*

   (b) *Ansonsten fordern alle Parteien $P_r \in P_R$ die Dritte Partei* T *auf, die irrtümlich gesendeten Güter zu annullieren. Dies darf nur geschehen, falls* kein *Vertrag existiert. Um dies zu prüfen, muß* T *alle beteiligten Teilnehmer auffordern, einen eventuell erhaltenen Vertrag vorzuzeigen.*

□

Da wir die drei Typen von Gütern nicht definiert haben, können wir die Sicherheit dieses Protokolls nur skizzieren:

- In Schritt 1 werden keine irreversiblen Änderungen vorgenommen. Die Güter von $P_r \in P_R$ können später von T annulliert werden. Die Empfänger erhalten keine Information über die Güter durch die Vorbereitungen der Teilnehmer $P_g \in P_G$.

- In Schritt 2 einigen sich entweder alle Teilnehmer auf Fortsetzung oder Beendigung des Austausches.

  Falls sie sich für einen Abbruch entscheiden, so erhalten die Empfänger keine Information über die Güter, da T keinen Ersatz erzeugen wird und alle bereits gesendeten Güter annulliert werden.

  Falls sich die Teilnehmer für eine Beendigung entschieden haben, kann T im Fehlerfall alle fehlenden Güter von $P_g \in P_G$ ersetzen und verhindern, daß die Güter von $P_r \in P_R$ annulliert werden. Das einzige Gut, worauf T keinen Einfluß hat, ist das von $P_f \in P_F$, welches somit den Ausgang des kompletten Protokolls bestimmt: Falls es versendet wird, erzwingt T eine Beendigung. Falls nicht, führt dies zum Abbruch.

- Schritt 4 beginnt nur, falls alle Güter von $P_i \in P_R \cup P_F$ bereits versendet wurden und ein Vertrag unterzeichnet wurde.

  Daher wird kein Gut annulliert, da T auf dem synchronen Netz feststellen kann, ob ein Vertrag unterzeichnet wurde oder nicht. Desweiteren wird jeder Teilnehmer $\mathsf{P_i}$ die erwarteten Güter von $P_g \in P_G$ erhalten, da $\mathsf{P_i}$ einen unterzeichneten Vertrag vorzeigen kann, welcher die für den Ersatz nötige Information enthält.

5 Unsere Ergebnisse

In diesem Artikel wurde das erste sichere optimistische Mehrparteien-Vertragsunterzeichnungsprotokoll beschrieben.

Im „worst case" benötigt das nachrichtenoptimierte Protokoll $6n - 4$ Nachrichten in 6 Runden. Falls alle Teilnehmer korrekt sind, benötigt es nur $4n - 4$ Nachrichten in 4 Runden. Die triviale nicht-optimistische Lösung benötigt $2n$ Nachrichten in 2 Runden, setzt jedoch die Teilnahme von T bei allen Vertragsunterzeichnungen voraus.

Die optimistische Lösung ist vorzuziehen, falls der Betrugsfall unwahrscheinlich ist oder falls die Kosten einer Beteiligung von T relativ hoch sind.

Ein Hauptanwendungsgebiet von Mehrparteien-Vertragsunterzeichnungsprotokollen scheint die Atomizität von sicheren Transaktionen zu sein. Es spielt hierbei eine ähnliche Rolle wie ein verteiltes commit-Protokoll, erweitert diese jedoch um betrügerische Teilnehmer (engl. *byzantine failure*). Um diese Einsatzmöglichkeiten als Atomizitätsmechanismus zu verdeutlichen, wurden Atomizitätsprotokolle für Certified Mail, fairen Austausch von Signaturen und fairer Austausch von beliebigen Gütern beschrieben.

6 Danksagung

Wir danken *Birgit Pfitzmann*, *Michael Steiner*, und *Victor Shoup* für hilfreiche Diskussionen.

Literatur

[1] N. Asokan, Matthias Schunter, Michael Waidner: Optimistic Protocols for Multi-Party Fair Exchange; IBM Research Report RZ 2892, IBM Zurich Research Laboratory, Zürich, November 1996.

[2] N. Asokan, Matthias Schunter, Michael Waidner: Optimistic Protocols for Fair Exchange; 4th ACM Conference on Computer and Communications Security, Zürich, April 1997, 6–17.

[3] N. Asokan, Victor Shoup, Michael Waidner: Asynchronous Protocols for Optimistic Fair Exchange; 1998 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1998, 86–99.

[4] N. Asokan. Victor Shoup, Michael Waidner: Optimistic Fair Exchange of Digital Signatures; Eurocrypt ’98, LNCS 1403, Springer-Verlag, Berlin 1998, 591–606.

[5] Feng Bao, Robert Deng, Wenbo Mao: Efficient and Practical Fair Exchange Protocols with Off-Line TTP; 1998 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1998, 77–85.

[6] Birgit Baum-Waidner, Michael Waidner: Optimistic Asynchronous Multi-Party Contract Signing; IBM Research Report RZ 3078, IBM Zurich Research Laboratory, Zürich, 1999.

[7] Michael Ben-Or, Oded Goldreich, Silvio Micali, Ronald L. Rivest: A Fair Protocol for Signing Contracts; IEEE Transactions on Information Theory 36/1 (1990) 40–46.

[8] Ronald Cramer, Victor Shoup: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack; Crypto ’98, LNCS 1462, Springer-Verlag, Berlin 1998, 13–25.

[9] Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644–654.

[10] Danny Dolev, Cynthia Dwork, Moni Naor: Non-Malleable Cryptography; 23rd Symposium on Theory of Computing (STOC) 1991, ACM, New York 1991, 542–552.

[11] Shimon Even, Oded Goldreich, Abraham Lempel: A Randomized Protocol for Signing Contracts; Communications of the ACM 28/6 (1985) 637–647.

[12] Matt Franklin, Gene Tsudik: Secure Group Barter: Multi-party Fair Exchange with Semi-Trusted Neutral Parties; 2nd International Conference on Financial Cryptography (FC ’98), LNCS 1465, Springer-Verlag, Berlin 1998, 90-102.

[13] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM Journal on Computing 17/2 (1988) 281–308.

[14] Steven P. Ketchpel, Hector Garcia-Molina: Making Trust Explicit in Distributed Commerce Transactions; 16th International Conference on Distributed Computing Systems,1996, IEEE Computer Society, 1996, 270–281.

[15] Nancy A. Lynch: Distributed Algorithms; Morgan Kaufmann, San Francisco 1996.

[16] Silvio Micali: Certified E-Mail with Invisible Post Offices; presented at 1997 RSA Conference.

[17] Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Optimal Efficiency of Optimistic Contract Signing; ACM Principles of Distributed Computing (PODC), Puerto Vallarta, June 1998, 113–122.

[18] Michael O. Rabin: Transaction Protection by Beacons; Journal of Computer and System Sciences 27/ (1983) 256–267.

[19] Markus Stadler: Publicly verifiable secret sharing; Eurocrypt ’96, LNCS 1070, Springer-Verlag, Berlin 1996, 190–199.