Langfristig sichere elektronische Geldsysteme mit Observern

Inauguraldissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

eingereicht beim
Fachbereich Mathematik und Informatik, Physik, Geographie
der Justus-Liebig-Universität Gießen

von

Thomas Schwarzpaul

geboren in Frankfurt am Main

Gießen, März 2009

Dekan:	Prof. Dr. Bernd Baumann
Gutachter:	Prof. Dr. Albrecht Beutelspacher (Gießen)
	Prof. Dr. Jörg Schwenk (Bochum)
Datum der Disputation:	17. April 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Auf Grund der rasanten technologischen Entwicklung der letzten Jahre erfreut sich das Internet, mit all seinen Facetten, immer größerer Beliebtheit. Neben dem Schreiben von E-Mails, der Suchmaschinenrecherche oder dem Lesen von Nachrichten zählen virtuelle Einkaufstouren dank Amazon und Co. mittlerweile zu den Lieblingsbeschäftigungen der Deutschen im Internet. Der Online-Handel mit Waren, aber auch das Abrufen kostenpflichtiger, digitaler Inhalte, wie beispielsweise Musikdownloads, erfahren stetig wachsende Popularität. Nach Angaben des Bundesverbands Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM) haben im Jahr 2007 bereits 41 Prozent der Deutschen im Internet Waren oder Dienstleistungen bestellt [Bun08b]. Im Vergleich dazu lag die Quote 2005 noch bei 32 Prozent. Mit der wachsenden Popularität des Online-Handels geht der über das Internet generierte Umsatz der Online-Händler einher. So hat sich der Umsatz im Jahr 2007 laut einer Studie der Gesellschaft für Konsumforschung (GfK) gegenüber dem Jahr 2002 fast verdreifacht [Ges08].
Allerdings dominieren immer noch traditionelle Zahlungsmethoden den Zahlungsverkehr des Online-Handels, die nicht speziell für das Internet konzipiert sind. Den Spitzenplatz belegt die Zahlung per Lastschrift. Laut einer BITKOM-Studie [Bun07] haben 38 Prozent der Deutschen beim Online-Shopping schon einmal einen Betrag auf diese Weise beglichen. Es folgen Rechnungsstellung (29 Prozent), Kreditkarte (20 Prozent) und Nachnahme mit 17 Prozent. Lediglich jeder Zehnte nutzt laut BITKOM inzwischen speziell für den Online-Handel geschaffene Bezahlsysteme. Vor allem Sicherheitsbedenken und Ungewohntheiten sind es, die die Nutzer davon abhalten neu geschaffene Bezahlsysteme zu verwenden.
Ein einfaches und sicheres Bezahlsystem verspricht nicht nur eine hohe Akzeptanz bei den Nutzern, sondern erreicht auch potenzielle Kunden, deren Skepsis gegenüber oben genannter Verfahren bisher zu groß war. Des Weiteren verheißt ein solches Bezahlsystem, auf Grund seiner hohen Akzeptanz sowie der hohen Wachstumsdynamik im Internet, weitere Umsatzsteigerungen des Online-Handels. Ein adäquates Zahlungsmittel hat sich jedoch im Bereich des E-Commerce bisher noch nicht etabliert.
Die derzeit favorisierten Zahlungsmethoden, wie Lastschrift, Rechnungsstellung oder Kreditkarte stellen dabei nur bedingt eine angemessene Lösung für den Online-Zah-

lungsverkehr dar. Auf der einen Seite stehen speziell beim Abruf kostenpflichtiger, digitaler Inhalte, die Transaktionskosten der genannten Bezahlsysteme oftmals in keinem Verhältnis zum Wert der erworbenen Inhalte. Auf der anderen Seite bieten sie nicht die Anonymität, die Kunden im Umgang mit herkömmlichem Bargeld gewohnt sind. Da zum einen Kunden gegenüber Online-Händlern sensible private Daten wie beispielsweise Bankverbindung oder Kreditkartennummer preisgeben müssen und zum anderen das für den Zahlungsverkehr zuständige Kreditinstitut alle Einkäufe nachvollziehen und detaillierte Nutzerprofile erstellen kann. Besonders der im August 2008 von der Verbraucherzentrale Schleswig-Holstein aufgedeckte, illegale Handel mit kundenbezogenen Daten [Ver08], [Una08] bestätigt die Skepsis vieler Nutzer gegenüber aktuell eingesetzten Bezahlsystemen und verdeutlicht den Bedarf eines anonymen Zahlungsmittels.
Eine Alternative ist der Einsatz *elektronischer Geldsysteme*, die anonyme und kostengünstige Transaktionen ermöglichen.

1.1 Elektronische Geldsysteme

In Anlehnung an herkömmliches Bargeld wurde 1988 von D. Chaum, A. Fiat und M. Naor [CFN89] ein erstes anonymes elektronisches Zahlungssystem, ein so genanntes *elektronisches Geldsystem*, entwickelt. Man spricht auch von *elektronischem Geld* oder *elektronischen Münzen*. Als digitales Äquivalent zu unserem gewöhnlichen Bargeld bietet elektronisches Geld Anonymität, Fälschungssicherheit und Verifizierbarkeit der Echtheit als wesentliche Sicherheitsmerkmale. Die Anonymität elektronischer Münzen garantiert, dass niemand, insbesondere nicht die Bank, Zahlungsvorgänge eines Kunden nachvollziehen kann. Gleichzeitig ist die Anonymität das auszeichnende Sicherheitsmerkmal elektronischer Geldsysteme, da Bezahlsysteme wie Lastschrift, Kreditkarte usw. diese nicht aufweisen.
Ermöglicht wurde die Konstruktion der ersten elektronischen Geldsysteme durch die Entwicklung *blinder Signaturverfahren*. Das Konzept der blinden Signatur, bei der die Signatur dem Signierer keinerlei Information über den Besitzer der unterzeichneten Nachricht liefert, stammt von D. Chaum [Cha83],[Cha84]. Anschaulich verbirgt sich hinter diesem Konzept die Idee, die zu unterzeichnende Nachricht mit einem Kohlepapier in einen Umschlag zu stecken und den Signierer auf dem Umschlag unterschreiben zu lassen. Da der Signierer die Nachricht nie gesehen hat, wird er sie auch später nicht wiedererkennen können. Elektronische Münzen können somit von einer Bank als blind signierte Nachrichten ausgegeben werden. Echtheit und Fälschungssicherheit der ausgegebenen Münzen werden durch die Signatur der Bank sichergestellt, Anonymität wird durch die zusätzliche *Blindheit* des Verfahrens gewährleistet.
In ihrer kryptografischen Umsetzung sind elektronische Münzen nichts weiter als digitale Datensätze, die ein Kunde auf seinem PC, Handheld oder Ähnlichem speichert. Für den Kunden bietet sich deshalb generell die Möglichkeit, Duplikate der Münzen zu erstellen (etwa durch ein Systembackup) und diese mehrfach auszugeben, was sich mit rein kryptografischen Methoden grundsätzlich nicht verhindern lässt und nur auf physikalischem Weg zu unterbinden ist. Prinzipiell sind unterschiedliche Wege denkbar, dem Problem

der Duplizierbarkeit bzw. Mehrfachausgabe elektronischer Münzen entgegenzutreten. Man unterscheidet dabei zwischen *elektronischen Online-* bzw. *Offline-Geldsystemen*.
In die Klasse der elektronischen Online-Geldsysteme (z.B. [Cha83]) fallen Verfahren, bei denen während des Bezahlvorgangs online verifiziert wird, ob die vorliegende Münze bereits ausgegeben wurde. Das heißt, die Bank verwaltet eine Liste aller bereits ausgegebenen Münzen. Bei jedem Bezahlvorgang wird vom Händler eine entsprechende Anfrage an die Bank gestellt, womit die Mehrfachausgabe elektronischer Münzen verhindert wird. Damit in einem Online-Geldsystem ein reibungsloser Betrieb möglich ist, sind auf Seiten der Bank Echtzeit-Verifikation der Bezahlvorgänge und hochverfügbare Systeme erforderlich. Aus Kostengründen ist der Einsatz von Online-Geldsystemen in der Praxis somit nur in Bereichen sinnvoll, in denen hauptsächlich Bezahlvorgänge mit hohem Transaktionsvolumen auftreten.
Im Gegensatz dazu vermeiden elektronische Offline-Geldsysteme die kostenintensive Online-Verifikation jedes einzelnen Bezahlvorgangs und arbeiten stattdessen mit der so genannten *Double-Spending-Detection*, einem Verfahren, bei dem die Mehrfachausgabe zwar nicht verhindert, aber im Nachhinein aufgedeckt werden kann. Das heißt, die Anonymität des Kunden bleibt beim erstmaligen Ausgeben einer Münze erhalten; erst wenn die Münze ein weiteres Mal ausgegeben wird, kann die Bank mit Hilfe der Double-Spending-Detection die *Identität* des betrügerischen Münzbesitzers, auch als *Double-Spender* bezeichnet, aufdecken.
Da sich diese Arbeit ausschließlich mit Offline-Systemen befasst, bezeichnet im Weiteren ein elektronisches Geldsystem stets ein elektronisches Offline-Geldsystem, sofern nicht explizit auf ein Online-System hingewiesen wird.
Eine weitere Möglichkeit das Problem der Mehrfachausgabe elektronischer Münzen in einem Offline-Geldsystem zu lösen, ist die Integration zusätzlicher, manipulationssicherer Hardware in das System. Um dem Kunden den direkten Zugriff auf die Münzen zu verwehren, werden bei diesem Ansatz Teile der Münzen auf der zusätzlichen Hardware, dem so genannten *Observer*, gespeichert. Da der Observer auch in das Bezahlen mit Münzen mit einbezogen ist, kann auf diese Weise die Mehrfachausgabe elektronischer Münzen verhindert werden. Geeignete Speichermedien, die für den Einsatz als Observer infrage kommen, sind beispielsweise Chipkarten oder USB-Token. Offline-Systeme, die zum Schutz vor Double-Spendern neben der Double-Spending-Detection spezielle, manipulationssichere Hardware einsetzen, werden auch als *elektronische Offline-Geldsysteme mit Observern* oder *elektronische Geldbörsen* bezeichnet. Elektronische Geldbörsen besitzen also die gleichen Sicherheitsmerkmale wie Systeme ohne Observer und schützen die Bank zusätzlich vor Double-Spendern.
Zur Konstruktion elektronischer Geldsysteme findet man in der Literatur im Wesentlichen zwei Ansätze: Elektronische Geldsysteme, die auf blinden Signaturverfahren aufbauen (z.B. [CFN89], [OO91], [Bra94], [Fer93], [GT03]) und solche, die ohne den Baustein der blinden Signatur auskommen (z.B. [STS99], [STSY00], [CHL05], [Wei05]). Der Unterschied beider Ansätze liegt in der Realisierung der Anonymität. Während die Anonymität bei ersteren beim Abheben der Münzen geschaffen wird, wird bei letzteren erst beim Bezahlen mit einer Münze für die Anonymität gesorgt. Der Fokus dieser Arbeit liegt ausschließlich auf den Geldsystemen, die mit blinden Signaturverfahren arbeiten.

Bei diesen Systemen ist besonders das Geldsystem von S. Brands [Bra94] zu erwähnen, da es eins der effizientesten elektronischen Geldsysteme ist. Im Vergleich zu vorherigen Systemen wie [CFN89], [OO91], [FY93] verwendet es zur Realisierung der Double-Spending-Detection *Geheimnisteilungsverfahren* [Sha79]. Zudem kommt es ohne die sehr ineffiziente *Cut-And-Choose-Technik* aus und bildet die Grundlage zahlreicher anderer elektronischer Geldsysteme (z.B. [FTY96], [FTY98], [FN01], [NS03]). Gleichzeitig kann [Bra94] zu einer elektronischen Geldbörse erweitert werden. Weitere elektronische Geldbörsen sind beispielsweise [CP93], [CP94], [ST98], [NS03]. Aus praktischer Sicht hat [Bra94] zusätzlich den Vorteil, dass die Arithmetik des Systems auf jeder Gruppe basieren kann, in der das Diskreter-Logarithmus-Problem schwierig ist. Für eine Implementierung kann man somit auf elliptische Kurven (siehe z.B. [BNS05],[Wer02]) zurückgreifen. Auf diese Weise erhält man bei gleicher Sicherheit verhältnismäßig kurze Schlüssel (ca. 224 Bit statt 2048 Bit) und infolgedessen wesentlich kürzere Laufzeiten.
Anonymität ist die auszeichnende Eigenschaft elektronischer Geldsysteme und ein Großteil der in der Literatur vorgeschlagenen Systeme bietet *perfekte Anonymität*. Das heißt, selbst ein rechnerisch unbeschränkter Angreifer wird nicht in der Lage sein, Münzen ihren Besitzern zuzuordnen. Für kleinere Beträge ist dies unproblematisch. Für größere Summen ist aber selbst bei herkömmlichem Bargeld keine hundertprozentige Anonymität gewährleistet, denn Geldscheine können anhand ihrer eindeutigen Seriennummer verfolgt werden. Dies kann insbesondere bei einem Bankraub, einer Erpressung oder Geldwäsche nützlich sein. In [SN92] findet man eine Aufstellung krimineller Möglichkeiten, die Kunden auf Grund der perfekten Anonymität elektronischer Geldsysteme besitzen. Deshalb sollte es auch in elektronischen Geldsystemen die Möglichkeit geben, die Anonymität im Bedarfsfall aufzuheben. Man spricht dann auch von *aufhebbarer Anonymität*.
Je nach Szenario werden zwei unterschiedliche Deanonymisierungsmechanismen benötigt. Zum einen sollte es möglich sein, den Besitzer vorliegender Münzen aufzudecken und zum anderen den Ausgabeort einzelner Münzen zu finden, um diese ggf. sperren zu können. Um die Anonymität ehrlicher Kunden nicht zu gefährden, darf die Bank diese Mechanismen nicht alleine ausführen können. Man braucht dazu eine weitere, vertrauenswürdige Partei, eine so genannte *Trusted-Third-Party* (TTP), die nur in begründeten Verdachtsmomenten die Anonymität einzelner Kunden aufhebt. Elektronische Geldsysteme, die über diese beiden Deanonymisierungsmechanismen verfügen, bezeichnet man auch als *fair*. Faire elektronische Geldsysteme sind beispielsweise [BGK95], [CPS95], [FTY96],[FTY98], [GT03].

1.2 Kompromittierung elektronischer Geldsysteme

Mit dem immer stärker werdenden Einsatz moderner Informationstechnik wächst auch das Risiko möglicher Schäden durch Sicherheitslücken in Software oder Betriebssystemen und Schadsoftware wie beispielsweise Viren, Würmer und Trojaner. Besonders Banken sehen sich einem erhöhten Risiko ausgesetzt und sind immer wieder Ziel massiver Hackerangriffe.
In einem elektronischen Geldsystem sind Fälschungssicherheit und Verifizierbarkeit der

Echtheit wesentliche Sicherheitsmerkmale und werden durch eine digitale Signatur der Bank garantiert. Beide Eigenschaften sind aber nur gewährleistet, solange der Signaturschlüssel der Bank nicht kompromittiert wird. Das heißt, die Sicherheit eines elektronischen Geldsystems ist nur dann garantiert, wenn die Bank eine *sichere Aufbewahrung* des Signaturschlüssels garantieren kann. Selbst wenn die Bank ihre Systeme, die für die Sicherung des Signaturschlüssels verantwortlich sind, stets auf dem aktuellen Sicherheitsstand hält, sind Sicherheitslücken und somit auch die Kompromittierung des Signaturschlüssels nicht hundertprozentig auszuschließen. Eine Kompromittierung des Signaturschlüssels impliziert, dass oben genannte Sicherheitsmerkmale nicht mehr gegeben sind. Dann können einerseits Angreifer, die über den Signaturschlüssel verfügen, Münzen fälschen, andererseits ist es aber auch nicht mehr möglich, die Echtheit der Münzen zu verifizieren, da sich echte und gefälschte Münzen nicht unterscheiden. Folgerichtig können die noch im Umlauf befindlichen Münzen nicht mehr als Zahlungsmittel verwendet werden und verlieren somit ihren Wert.

1.3 Ziele und Aufbau der Arbeit

Die im vorherigen Abschnitt angesprochenen Risiken einer nicht auszuschließenden Kompromittierung des Signaturschlüssels der Bank machen deutlich, dass im Bereich der elektronischen Geldsysteme über geeignete Schutzmaßnahmen nachgedacht werden muss, um die Folgen im Falle einer Kompromittierung möglichst gering zu halten. In den letzten Jahren wurde zum Beispiel speziell für digitale Signaturverfahren eine Reihe kryptografischer Methoden vorgeschlagen, die sich mit dieser Problematik beschäftigen, unter anderem *Forward-Security* [And97], [BM99], *Key-Insulation* [DKXY03] und *Intrusion-Resilience* [Itk01]. Auf dem Gebiet der elektronischen Geldsysteme sind nach bestem Wissen keine Veröffentlichungen bekannt, die sich konkret mit dieser Fragestellung beschäftigen.
Ein Ziel dieser Arbeit ist es daher zu untersuchen, welches der bereits bestehenden kryptografischen Konzepte sich auf elektronische Geldsysteme übertragen lässt, die als Baustein blinde Signaturen nutzen. Im Vordergrund stehen dabei die auf [Bra94] basierenden elektronischen Geldsysteme, hauptsächlich das faire Geldsystem [FTY98]. Mit den geeigneten Methoden soll, aufbauend auf [FTY98], eine neue faire elektronische Geldbörse entwickelt werden, die somit als erstes elektronisches Geldsystem überhaupt spezielle Schutzmaßnahmen im Falle einer Kompromittierung der Bank bietet.
Grundlegende kryptografische Verfahren bzw. Protokolle, die als Bausteine der im weiteren Verlauf dieser Arbeit konstruierten elektronischen Geldsysteme benötigt werden, werden in Kapitel 2 beschrieben und analysiert. Kapitel 3 gibt einen detaillierten Überblick über Aufbau und Funktionalität elektronischer Geldsysteme, insbesondere über die Designprinzipien elektronischer Geldbörsen und die Zusatzfunktionen fairer Geldsysteme. Des Weiteren werden in Kapitel 3 Sicherheitseigenschaften wie *Unfälschbarkeit* und *Anonymität* formalisiert, denen ein sicheres elektronisches Geldsystem genügen muss.
Die in Abschnitt 1.2 angesprochenen Risiken werden in Kapitel 4 noch einmal aufgegriffen und ausführlich diskutiert. Wobei hier nicht nur die Bank als potenzielles Ziel

von Angreifern in Erwägung gezogen wird, sondern auch die Trusted-Third-Party, die in einem fairen System für die Deanonymisierung der Kunden verantwortlich ist. Ferner werden in Abschnitt 4.2 mögliche Lösungsansätze diskutiert und die bereits erwähnten Konzepte Forward-Security, Key-Insulation und Intrusion-Resilience dargestellt und hinsichtlich ihrer Integrierbarkeit in das faire Geldsystem von Y. Frankel, Y. Tsiounis und M. Yung [FTY98] geprüft. Es wird sich zeigen, dass das Konzept der Key-Insulation aufgrund der in [DKXY03] vorgeschlagenen Techniken hervorragend geeignet ist.
Ziel von Kapitel 5 ist es, den im Key-Insulation-Modell verfolgten Ansatz auf elektronische Geldsysteme zu übertragen. Da sich diese Arbeit ausschließlich auf elektronische Geldsysteme konzentriert, die auf blinden Signaturen basieren, wird in Abschnitt 5.2 zunächst untersucht, mit welchen Auswirkungen sich dieser Ansatz auf blinde Signaturen übertragen lässt; das Modell der *langfristig sicheren* blinden Signaturen wird eingeführt und formal definiert. Darauf aufbauend wird in den Abschnitten 5.3 und 5.4 das in Kapitel 3 dargestellte formale Sicherheitsmodell elektronischer bzw. fairer elektronischer Geldsysteme zum Modell der *langfristig sicheren* elektronischen bzw. *langfristig sicheren fairen* elektronischen Geldsysteme erweitert.
Die Entwicklung eines neuen langfristig sicheren fairen elektronischen Geldsystems ist Gegenstand von Kapitel 6. Da das System von Y. Frankel *et al.* [FTY98] die Grundlage für das neu entwickelte langfristig sichere System bildet, wird in Abschnitt 6.1 zunächst [FTY98] vorgestellt. Im darauffolgenden Abschnitt werden die Protokolle des neuen, langfristig sicheren fairen Geldsystems beschrieben. Abschließend wird die Sicherheit des Systems analysiert. Damit Double-Spending von der Bank nicht nur im Nachhinein aufgedeckt, sondern bereits im Voraus verhindert werden kann, zeigt Abschnitt 6.3, wie man in das neu entworfene System einen Observer integrieren kann, ohne dass die Anonymität der Kunden davon betroffen ist.
In elektronischen Geldsystemen händigt stets die Bank die Observer an ihre Kunden aus. Es ist aber durchaus denkbar, dass die Observer in einem fairen Geldsystem von der TTP verteilt werden. Daher wird in Kapitel 7 als weiteres Ziel dieser Arbeit analysiert, ob sich dadurch effizientere Protokolle entwerfen lassen und welche Anforderungen in einem solchen System an den Observer zu stellen sind. Zunächst wird in Abschnitt 7.1 ein neues generisches faires Geldsystem konstruiert, das auf einem beliebigen sicheren blinden Signaturverfahren aufgebaut werden kann. Anschließend werden veschiedene Ansätze diskutiert, wie man in diesem System mit Hilfe des Observers die aufhebbare Anonymität fairer Geldsysteme erreichen kann. Dabei ist einer der Ansätze mit den in [KPW97] vorgeschlagenen *Gruppensignaturen mit Observer* vergleichbar. Gruppensignaturen wurden im Jahre 1991 von D. Chaum und E. van Heyst [CH91] vorgeschlagen und bieten den Mitgliedern einer Gruppe die Möglichkeit, im Namen dieser Gruppe anonym Signaturen zu erstellen, wobei die Anonymität der Mitglieder bei Bedarf durch eine Trusted-Third-Party aufgehoben werden kann. In Abschnitt 7.3 wird ein neues Gruppensignaturverfahren mit Observer vorgestellt und analysiert. Das Verfahren eignet sich als Baustein für das in Abschnitt 7.1 entwickelte generische System.
In Kapitel 8 werden die Ergebnisse abschließend analysiert und offene Arbeitsfelder im Bereich elektronischer Geldsysteme aufgezeigt, die sich aus dieser Arbeit ergeben haben.

Die in dieser Arbeit untersuchten Fragestellungen sind überwiegend durch das von der Deutschen Forschungsgemeinschaft geförderte Projekt „*Kryptografische Protokolle mit Observerfunktion*“, das in den Jahren 2006 und 2007 am Lehrstuhl von Herrn Prof. Dr. Albrecht Beutelspacher durchgeführt wurde, motiviert. Teile dieser Arbeit sind im Rahmen dieses Projektes entstanden.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Albrecht Beutelspacher für die Möglichkeit, diese Arbeit zu verfassen und für die ausgezeichnete Betreuung bedanken. Mein Dank gilt ebenso Herrn Prof. Dr. Jörg Schwenk für die Bereitschaft diese Arbeit zu begutachten. Des Weiteren möchte ich mich bei meinen Kolleginnen und Kollegen Björn Fay, Dr. Jörn Schweisgut, Dr. Heike Neumann und Dr. Christian Tobias für das Korrekturlesen und wertvolle Anregungen bedanken.

Mein ganz besonderer Dank gilt Manuela Holighaus, meinen Eltern Elfi und Edmund, sowie meiner Schwester Sarah für ihre Aufmunterung und Unterstützung während der letzten Jahre.

Kapitel 2

Kryptografische Grundlagen

Elektronische Geldsysteme haben vielschichtige Sicherheitsziele, die durch den Einsatz verschiedener kryptografischer Bausteine realisiert werden. So wird beispielsweise die Fälschungssicherheit elektronischer Münzen mit Hilfe digitaler Signaturen garantiert, während die Anonymität eines Kunden durch den Einsatz einer blinden Signatur erreicht werden kann. In diesem Kapitel werden die kryptografischen Grundlagen und Bausteine vorgestellt, auf denen die in den Kapiteln 5, 6 und 7 entwickelten Systeme basieren.

2.1 Notationen und Konventionen

Im weiteren Verlauf dieser Arbeit werden die im Folgenden aufgeführten Notationen verwendet:

- *Zufällige Wahl von Elementen einer Menge*: Ist G eine endliche Menge, so bezeichnet $g_1, \ldots, g_n \in_{\mathcal{R}} G$ die zufällige und unabhängige Wahl von Elementen $g_1, \ldots, g_n$ gemäß der Gleichverteilung auf G.
- *Zeichenfolgen über einem Alphabet*: Ist Σ ein Alphabet, so bezeichnet Σ^n die Menge aller Zeichenfolgen der Länge n und Σ^* die Menge aller endlichen Zeichenfolgen.
- *Protokollansicht*: Es bezeichnet $\mathsf{view}_A^B(P)$ die Ansicht von Teilnehmer A bei Durchführung des Protokolls P mit Teilnehmer B.

Des Weiteren wird ein deterministischer oder probabilistischer Algorithmus mit polynomieller Laufzeit auch als *effizienter* Algorithmus bezeichnet.

2.2 Kryptografische Probleme und Annahmen

Die Sicherheit asymmetrischer Verfahren basiert auf der Schwierigkeit der den Verfahren zugrunde liegenden kryptografischen Problemen. In diesem Abschnitt werden u.a. das Diskreter-Logarithmus-Problem sowie das Computational- und Decisional-Diffie-Hellman-Problem vorgestellt.

Definition 2.2.1 (**Vernachlässigbare Funktionen**)**.** Eine Funktion $\nu : \mathbb{N} \to \mathbb{R}$ heißt *vernachlässigbar*, falls es für jedes $c \in \mathbb{N}$ ein $k_0 \in \mathbb{N}$ gibt, so dass für alle $k \geq k_0$ gilt:

$$| \nu(k) | \leq k^{-c}.$$

2.2.1 Das Diskreter-Logarithmus-Problem

Das Problem des diskreten Logarithmus wurde erstmals von W. Diffie und M. Hellman zum Entwurf der *Diffie-Hellman-Schlüsselvereinbarung* [DH76] benutzt.

Definition 2.2.2 (**Diskreter Logarithmus**)**.** Gegeben seien eine zyklische Gruppe G der Ordnung $n \in \mathbb{N}$ und ein Generator $g \in G$. Die Abbildung $\exp_g : \mathbb{N} \to G$ mit $a \mapsto g^a$ heißt *diskrete Exponentialfunktion* zur Basis g. Ist $h \in G$, so ist der *diskrete Logarithmus* $\log_g h$ von h zur Basis g die eindeutig bestimmte natürliche Zahl $x \in \mathbb{N}$ mit $0 \leq x \leq n-1$ für die $h = g^x$ gilt. Damit kann man die *diskrete Logarithmusfunktion*, die Umkehrfunktion der diskreten Exponentialfunktion, definieren.

Definition 2.2.3 (**Diskreter-Logarithmus-Problem**)**.** Gegeben seien eine zyklische Gruppe G der Ordnung $n \in \mathbb{N}$, ein Generator $g \in G$ und ein Element $h \in G$. Finde die eindeutig bestimmte natürliche Zahl $x \in \mathbb{N}$ mit $0 \leq x \leq n-1$ für die $h = g^x$ gilt.

Eine typische Wahl für G ist die multiplikative Gruppe $\mathbb{Z}_p^*$ des endlichen Körpers $\mathbb{Z}_p$ mit Primzahl p. Häufig wird für G auch die eindeutig bestimmte zyklische Untergruppe G_q der Primzahlordnung q von $\mathbb{Z}_p^*$ verwendet, wobei $q \mid (p-1)$.
Während die diskrete Exponentialfunktion mit dem *Square-And-Multiply-Algorithmus* (siehe z.B. [BSW06], [BNS05]) effizient berechnet werden kann, gibt es bis heute keinen effizienten Algorithmus zur Berechnung diskreter Logarithmen. Man vermutet daher, dass es sich bei der diskreten Exponentialfunktion um eine Einwegfunktion handelt. Dies ist die so genannte *Diskreter-Logarithmus-Annahme*.
Da im weiteren Verlauf dieser Arbeit für G stets die eindeutig bestimmte zyklische Untergruppe G_q der Primzahlordnung q von $\mathbb{Z}_p^*$ betrachtet wird, werden die Diskreter-Logarithmus-Annahme und auch alle weiteren Annahmen speziell für diese Gruppe formuliert.

Annahme 2.1 (**Diskreter-Logarithmus-Annahme**)**.** Gegeben seien der Sicherheitsparameter $k \in \mathbb{N}$, eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$, eine Primzahl q mit $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$, die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q, ein zufällig gewählter Generator $g \in_{\mathcal{R}} G_q$ und ein zufällig gewähltes Element $h \in_{\mathcal{R}} G_q$. Dann ist für jeden effizienten Algorithmus $\mathcal{M}$ die Wahrscheinlichkeit, dass $\mathcal{M}$ bei Eingabe von p, q, g, h den diskreten Logarithmus $\log_g h$ von h zur Basis g berechnet, vernachlässigbar in k. Das heißt, für jeden effizienten Algorithmus $\mathcal{M}$ gibt es eine vernachlässigbare Funktion ν und ein $k_0 \in \mathbb{N}$, so dass für alle $k \geq k_0$ gilt:

$$\mathbf{P}\left[\mathcal{M}(p, q, g, h) = \log_g h\right] \leq \nu(k).$$

Für kryptografische Verfahren, deren Sicherheit auf dem Diskreter-Logarithmus-Problem oder den Diffie-Hellman-Problemen (vgl. Abschnitt 2.2.2) beruht, gelten derzeit Primzahlen der Größenordnung 2048 Bit für die Wahl von p und Primzahlen der Größe 224 Bit für die Wahl von q als untere Grenze (siehe z.B. [Bun08a]).

2.2.2 Die Diffie-Hellman-Probleme

Auf die Diffie-Hellman-Schlüsselvereinbarung gehen die beiden *Diffie-Hellman-Probleme* zurück, die in enger Beziehung zum Diskreter-Logarithmus-Problem stehen. Zunächst wird das *Computational-Diffie-Hellman-Problem* vorgestellt.

Definition 2.2.4 (**Computational-Diffie-Hellman-Problem**)**.** Gegeben seien eine zyklische Gruppe G der Ordnung $n \in \mathbb{N}$, ein Generator $g \in G$ und Elemente g^a, g^b mit $0 < a, b < n$. Finde g^{ab}.

In Gruppen, in denen das Diskreter-Logarithmus-Problem effizient gelöst werden kann, kann auch das Computational-Diffie-Hellman-Problem effizient gelöst werden. Offen ist hingegen, ob die beiden Probleme äquivalent sind.

Das zweite der Diffie-Hellman-Probleme ist das *Decisional-Diffie-Hellman-Problem*.

Definition 2.2.5 (**Decisional-Diffie-Hellman-Problem**)**.** Gegeben seien eine zyklische Gruppe G der Ordnung $n \in \mathbb{N}$, ein Generator $g \in G$ und Elemente $g^a, g^b, g^c \in G$ mit $0 < a, b, c < n$. Entscheide, ob $c = ab$ gilt.

In Gruppen, in denen das Computational-Diffie-Hellman-Problem effizient gelöst werden kann, kann auch das Decisional-Diffie-Hellman-Problem effizient gelöst werden.
Ähnlich zum Diskreter-Logarithmus-Problem formuliert man auch im Fall der beiden Diffie-Hellman-Probleme entsprechende Annahmen. Die Vermutung, dass das Computational-Diffie-Hellman-Problem auch ohne die Berechnung diskreter Logarithmen nicht effizient gelöst werden kann, ist formal durch die *Computational-Diffie-Hellman-Annahme* gegeben.

Annahme 2.2 (**Computational-Diffie-Hellman-Annahme**)**.** Gegeben seien der Sicherheitsparameter $k \in \mathbb{N}$, eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$, eine Primzahl q mit $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$, die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q, ein zufällig gewählter Generator $g \in_{\mathcal{R}} G_q$ und zwei zufällig gewählte Elemente $g^a, g^b \in_{\mathcal{R}} G_q$. Dann ist für jeden effizienten Algorithmus $\mathcal{M}$ die Wahrscheinlichkeit, dass $\mathcal{M}$ bei Eingabe von p, q, g, g^a und g^b das Element g^{ab} berechnet, vernachlässigbar in k. Das heißt, für jeden Algorithmus $\mathcal{M}$ gibt es eine vernachlässigbare Funktion ν und ein $k_0 \in \mathbb{N}$, so dass für alle $k \geq k_0$ gilt:

$$\mathbf{P}\left[\mathcal{M}(p, q, g, g^a, g^b) = g^{ab}\right] \leq \nu(k).$$

Für Untergruppen G_q von $\mathbb{Z}_p^*$ mit Primzahlordnung q vermutet man außerdem, dass auch das Decisional-Diffie-Hellman-Problem nicht effizient gelöst werden kann. Dies ist die so genannte *Decisional-Diffie-Hellman-Annahme*.

Annahme 2.3 (**Decisional-Diffie-Hellman-Annahme**)**.** Gegeben seien der Sicherheitsparameter $k \in \mathbb{N}$, eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebnes $\delta \in \mathbb{N}$, eine Primzahl q mit $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$, die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q, ein zufällig gewählter Generator $g \in_{\mathcal{R}} G_q$ und drei zufällig gewählte Elemente $g^a, g^b, g^c \in_{\mathcal{R}} G_q$. Dann gibt es keinen effizienten Algorithmus $\mathcal{M}$, der bei Eingabe von p, q, g, g^a, g^b und g^c entscheiden kann, ob $g^{ab} = g^c$ gilt. Das heißt, für jeden Algorithmus $\mathcal{M}$ gibt es eine vernachlässigbare Funktion ν und ein $k_0 \in \mathbb{N}$, so dass für alle $k \geq k_0$ gilt:

$$\mathbf{P}\left[\mathcal{M}(p, q, g, g^a, g^b, g^c) = 1 \mid a, b \in_{\mathcal{R}} \mathbb{Z}_q, g^c = g^{ab}\right] \leq 1/2 + \nu(k).$$

Obwohl man annimmt, dass auch in $\mathbb{Z}_p^*$ das Computational-Diffie-Hellman-Problem nicht effizient gelöst werden kann, gilt die Decisional-Diffie-Hellman-Annahme für $\mathbb{Z}_p^*$ nicht, da man in $\mathbb{Z}_p^*$ anhand des Legendre-Symbols effizient entscheiden kann, ob ein Element ein quadratischer Rest oder Nichtrest ist.
Im Jahre 2003 wurde von A. Joux und K. Nguyen in [JN03] gezeigt, dass das Decisional-Diffie-Hellman-Problem für eine bestimmte Klasse von elliptischen Kurven, den *supersingulären* elliptischen Kurven (siehe z.B. [Wer02]), mit Hilfe des Weil-Pairing effizient gelöst werden kann.

2.2.3 Das Darstellungsproblem

Eng verwandt mit dem Diskreter-Logarithmus-Problem ist das von S. Brands im Jahre 1993 vorgestellte *Darstellungsproblem in Gruppen mit Primzahlordnung* [Bra93], welches bereits in [CEG87] erstmals erwähnt wurde. Es bildet die Grundlage zahlreicher elektronischer Geldsysteme (z.B. [Bra94], [FTY96], [FTY98]) und der in den Abschnitten 6.2 und 6.3 entwickelten Systeme.

Definition 2.2.6. Gegeben seien Primzahlen p und q mit $q \mid (p-1)$, $n \in \mathbb{N}, n \geq 2$ sowie die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q. Ein *Generatortupel* der Länge n ist ein n-Tupel $(g_1, \ldots, g_n)$ mit $g_i \in G_q \backslash \{1\}$ und $g_i \neq g_j$ für $i \neq j$. Für $h \in G_q$ ist eine *Darstellung* von h bzgl. eines Generatortupels $(g_1, \ldots, g_n)$ ein Tupel $(a_1, \ldots, a_n)$ mit $a_i \in \mathbb{Z}_q$ für $i = 1, \ldots, n$, so dass $h = \prod_{i=1}^{n} g_i^{a_i}$.

Ist $(g_1, \ldots, g_n)$ ein Generatortupel von G_q, so bezeichnet man für $h = 1$ die Darstellung $(0, \ldots, 0)$ von h bzgl. $(g_1, \ldots, g_n)$ als die *triviale Darstellung*.

Definition 2.2.7 (**Darstellungsproblem in Gruppen von Primzahlordnung**)**.** Gegeben seien Primzahlen p und q mit $q \mid (p-1)$, $n \in \mathbb{N}, n \geq 2$, die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q und ein Generatortupel $(g_1, \ldots, g_n)$ von G_q. Finde eine Darstellung $(a_1, \ldots, a_n)$ von h bzgl. $(g_1, \ldots, g_k)$.

Für Gruppen mit Primzahlordnung wurde von S. Brands die Äquivalenz von Diskreter-Logarithmus-Problem und Darstellungsproblem gezeigt [Bra93]. Ist $h \in G_q$, so unterscheiden sich zwei unterschiedliche Darstellung $(a_1, \ldots, a_n)$ und $(b_1, \ldots, b_n)$ von h bzgl. eines Generatortupels $(g_1, \ldots, g_n)$ in genau einer nicht trivialen Darstellung der Eins.

Aus der Äquivalenz von Diskreter-Logarithmus-Problem und Darstellungsproblem folgt somit das folgende Korollar:

Korollar 2.2.1. Gegeben seien der Sicherheitsparameter $k \in \mathbb{N}$, eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$, eine Primzahl q mit $q = \gamma\,(p-1)$ für ein vorgegebenes $\gamma \in \mathbb{N}$, $n \in \mathbb{N}, n \geq 2$ sowie die eindeutig bestimmte zyklische Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q. Dann existiert unter Annahme 2.1 kein effizienter Algorithmus $\mathcal{M}$, der bei zufällig gewähltem Generatortupel $(g_1, \ldots, g_n)$ ein $h \in G_q$ und zwei verschiedene Darstellungen $(a_1, \ldots, a_n)$ und $(b_1, \ldots, b_n)$ von h berechnet.

Beweis. siehe [Bra93]. □

2.3 Public-Key-Verschlüsselungsverfahren

Das Konzept der *Public-Key-Kryptografie*, auch als *asymmetrische Kryptografie* bezeichnet, wurde von W. Diffie und M. Hellman im Jahre 1976 vorgeschlagen [DH76]. Im Gegensatz zu den symmetrischen Verschlüsselungsverfahren, bei denen Sender und Empfänger zum Ver- bzw. Entschlüsseln einer Nachricht den gleichen Schlüssel verwenden, ermöglichen es *Public-Key-Verschlüsselungsverfahren* zum Ver- bzw. Entschlüsseln einer Nachricht unterschiedliche Schlüssel zu verwenden.
Jeder Teilnehmer eines Public-Key-Verschlüsselungsverfahrens besitzt ein Schlüsselpaar, bestehend aus einem *privaten* und einem *öffentlichen Schlüssel*, wobei der öffentliche Schlüssel allen anderen Teilnehmern bekannt gegeben wird. Zum Verschlüsseln einer Nachricht wird der öffentliche Schlüssel des Empfängers eingesetzt. Den daraus resultierenden Geheimtext kann der Empfänger anhand seines privaten Schlüssels entschlüsseln. Dabei darf es nicht effizient möglich sein, aus dem öffentlichen Schlüssel des Empfängers dessen privaten Schlüssel zu berechnen. Dieses Kriterium wird auch als *Public-Key-Eigenschaft* (siehe z.B. [BSW06], [BNS05], [Sti06]) bezeichnet. Zu den bekanntesten Public-Key-Verschlüsselungsverfahren zählen das RSA-Verschlüsselungsverfahren [RSA78] und das ElGamal-Verschlüsselungsverfahren [ElG85], welches in Abschnitt 2.3.2 vorgestellt wird.
Auch in fairen elektronischen Geldsystemen spielen Public-Key-Verschlüsselungsverfahren eine entscheidende Rolle und werden dazu eingesetzt, die Anonymität eines Kunden bei Bedarf aufzuheben.

Formal ist ein Public-Key-Verschlüsselungsverfahren wie folgt definiert:

Definition 2.3.1 (**Public-Key-Verschlüsselungsverfahren**)**.** Gegeben seien $k \in \mathbb{N}$, M die Menge aller Klartexte, C die Menge aller Geheimtexte und die folgenden drei polynomiellen Algorithmen:

1. Der *Schlüsselgenerierungsalgorithmus* Gen ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k einen *öffentlichen Schlüssel* pk und einen *privaten Schlüssel* sk erzeugt. Die Ausgabe von $\mathsf{Gen}(1^k)$ ist das Paar $(\mathsf{pk}, \mathsf{sk})$.

2. Der *Verschlüsselungsalgorithmus* Enc ist ein probabilistischer Algorithmus, der bei Eingabe des öffentlichen Schlüssels pk und eines Klartextes $m \in M$ einen Geheimtext $c \in \mathcal{C}$ berechnet. Die Ausgabe ist $c = \mathsf{Enc}(\mathsf{pk}, m)$.

3. Der *Entschlüsselungsalgorithmus* Dec ist ein deterministischer Algorithmus, der bei Eingabe des Schlüsselpaars $(\mathsf{pk}, \mathsf{sk})$ und eines Geheimtextes $c \in \mathcal{C}$ einen Klartext $m = \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, c)$ berechnet.

Das Tripel $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ heißt *Public-Key-Verschlüsselungsverfahren*, falls folgende Bedingung erfüllt ist: Für jedes Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ und alle Klartexte $m \in M$ gilt:

$$\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m.$$

2.3.1 Polynomielle Ununterscheidbarkeit

Grundsätzlich gibt es zwei Konzepte, mit denen man die Sicherheit eines Public-Key-Verschlüsselungsverfahrens definieren kann. Zum einen gibt es die *polynomielle Ununterscheidbarkeit* und zum anderen die *semantische Sicherheit*. Beide Sicherheitsbegriffe wurden 1984 von S. Goldwasser und S. Micali [GM84] entwickelt. Im Jahre 1988 konnte die Äquivalenz beider Sicherheitsbegriffe von S. Micali, C.Rackoff und B. Sloan [MRS88] gezeigt werden. Aus diesem Grund werden wir uns im Folgenden auf die Darstellung der polynomiellen Ununterscheidbarkeit beschränken. Die polynomielle Ununterscheidbarkeit greift folgendes Konzept auf: Ein effizienter Angreifer wählt zwei Klartexte, erhält zu einem der beiden Klartexte den Geheimtext und soll anhand des Geheimtextes bestimmen, welcher der beiden Klartexte verschlüsselt wurde.
Um die Sicherheit eines Public-Key-Verschlüsselungsverfahren mit Hilfe der polynomiellen Ununterscheidbarkeit (semantischen Sicherheit) zu beschreiben, muss auch berücksichtigt werden, unter welchen möglichen Angriffen das Verfahren polynomielle Ununterscheidbarkeit gewährleistet. Dabei wird grundsätzlich zwischen den folgenden Angriffsarten unterschieden:

- *Angriff mit gewählten Klartexten*: Der Angreifer $\mathcal{A}$ kann sich Klartexte seiner Wahl verschlüsseln lassen.

- *Angriff mit gewählten Geheimtexten*: Der Angreifer $\mathcal{A}$ kann sich einen oder mehrere Geheimtexte entschlüsseln lassen.

Unter einem Angriff mit gewählten Klartexten wird die polynomielle Ununterscheidbarkeit für ein Public-Key-Verschlüsselungsverfahren $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ mit Sicherheitsparameter $k \in \mathbb{N}$ durch das folgende Spiel modelliert, dass zwischen einem effizienten Angreifer $\mathcal{A}$ und einem Orakel O durchgeführt wird.

Spiel 2.3.1 (**Polynomielle Ununterscheidbarkeit**)**.**

Schritt 1: Das Orakel O führt bei Eingabe des Sicherheitsparameters 1^k den Algorithmus $\mathsf{Gen}(1^k)$ aus. Die Ausgabe von $\mathsf{Gen}(1^k)$ ist ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$.

Schritt 2: Der Angreifer $\mathcal{A}$ wählt zwei Klartexte $m_0, m_1 \in M$ und sendet diese an das Orakel O.

Schritt 3: Das Orakel O wählt zufällig ein Bit $b \in_{\mathcal{R}} \{0,1\}$ und führt folgende Verschlüsselung durch:

$$c = \begin{cases} \mathsf{Enc}(\mathsf{pk}, m_0) & \text{falls } b = 0, \\ \mathsf{Enc}(\mathsf{pk}, m_1) & \text{falls } b = 1. \end{cases}$$

Das Orakel O sendet den Geheimtext $c \in C$ an den Angreifer $\mathcal{A}$.

Schritt 4: Der Angreifer gibt ein Bit $b' \in \{0,1\}$ aus.

Der Angreifer $\mathcal{A}$ gewinnt Spiel 2.3.1, falls $\mathcal{A}$ nach Ablauf des Spiels entscheiden kann, welchen Klartext das Orakel verschlüsselt hat. Das heißt $\mathcal{A}$ gewinnt, falls $b = b'$.

Formal heißt ein Public-Key-Verschlüsselungsverfahren $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ polynomiell ununterscheidbar unter einem Angriff mit gewählten Klartexten, falls der folgenden Definition genügt:

Definition 2.3.2 (**Polynomielle Ununterscheidbarkeit**)**.** Ein Public-Key-Verschlüsselungsverfahren $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ mit Sicherheitsparameter $k \in \mathbb{N}$ heißt *polynomiell ununterscheidbar* unter einem Angriff mit gewählten Klartexten (semantisch sicher), falls es für jeden effizienten Angreifer $\mathcal{A}$, der Spiel 2.3.1 durchführt, eine vernachlässigbare Funktion ν gibt, so dass folgendes gilt:

$$\mathbf{P}\left[\mathcal{A}\left(1^k, \mathsf{pk}, m_0, m_1, c\right) = 0 \mid c = \mathsf{Enc}\left(\mathsf{pk}, m_0\right)\right] \leq 1/2 + \nu\left(k\right).$$

Das heißt bei einem, unter einem Angriff mit gewählten Klartexten, polynomiell ununterscheidbaren Public-Key-Verschlüsselungsverfahren ist die Wahrscheinlichkeit des Angreifers, bei gegebenem Geheimtext zu entscheiden welcher der beiden Klartexte verschlüsselt wurde, nur unwesentlich besser als zu raten. Da der Angreifer die beiden Klartexte stets probeverschlüsseln kann, sind *deterministische* Public-Key-Verschlüsselungsverfahren, wie zum Beispiel das RSA-Verschlüsselungsverfahren [RSA78], nicht sicher im Sinne der polynomiellen Ununterscheidbarkeit.

Bemerkung 2.3.1. Der Begriff der polynomiellen Ununterscheidbarkeit unter einem Angriff mit gewählten Klartexten ist für die Sicherheitsanalyse der in den Kapiteln 5, 6 und 7 neu entwickelten Verfahren ausreichend.
Polynomielle Ununterscheidbarkeit unter einem (adaptiven) Angriff mit gewählten Geheimtexten kann man durch ein ähnliches Spiel (Spiel 2.3.1) modellieren. Im Vergleich zu Spiel 2.3.1 kann sich der Angreifer nach der „Initialisierungsphase“ (vgl. Schritt 1 aus Spiel 2.3.1) zusätzlich Geheimtexte seiner Wahl vom Orakel entschlüsseln lassen. Anschließend werden analog zu Spiel 2.3.1 die Schritte 2 bis 4 aus Spiel 2.3.1 durchgeführt, wobei der Angreifer die beiden Nachrichten in Abhängigkeit der vom Orakel entschlüsselten Geheimtexte wählen darf. Wird der Angriff adaptiv durchgeführt, darf sich der Angreifer auch nach Schritt 3 aus Spiel 2.3.1 Geheimtexte seiner Wahl entschlüsseln lassen, wobei der vom Orakel berechnet Geheimtext ausgenommen ist.

2.3.2 Das ElGamal-Verschlüsselungsverfahren

Im Jahre 1984 wurde von T. ElGamal ein Public-Key-Verschlüsselungsverfahren [ElG85] vorgestellt, das sich leicht aus der Diffie-Hellman-Schlüsselvereinbarung [DH76] konstruieren lässt und dessen Sicherheit auf der Decisional-Diffie-Hellman-Annahme beruht. Das von T. ElGamal ursprünglich vorgestellte Verschlüsselungsverfahren arbeitet, genauso wie die Diffie-Hellman-Schlüsselvereinbarung, auf der zyklischen Gruppe $\mathbb{Z}_p^*$; kann aber auf jeder anderen zyklischen Gruppe, auf der die Decisional-Diffie-Hellman-Annahme gilt, eingesetzt werden. Die im Folgenden beschriebene Variante der ElGamal-Verschlüsselung arbeitet auf der eindeutig bestimmten Untergruppe G_q von $\mathbb{Z}_p^*$ der Primzahlordnung q und wurde unter anderem von Y. Tsiounis und M. Yung in [TY98] vorgestellt.

Schlüsselgenerierung: Um ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ zu generieren, werden bei Eingabe des Sicherheitsparameters 1^k vom Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ folgende Schritte durchgeführt:

Schritt 1: Wähle zufällig eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$ und bestimme eine Primzahl q, so dass $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$. Wähle zufällig einen Generator g der eindeutigen Untergruppe G_q von $\mathbb{Z}_p^*$ mit $|G_q| = q$.

Schritt 2: Wähle zufällig eine Zahl $x \in_{\mathcal{R}} \mathbb{Z}_q$ und berechne $y = g^x$.

Die Ausgabe des Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ ist das Schlüsselpaar:

$$(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, y), x).$$

Verschlüsseln: Um einen Klartext $m \in G_q$ unter dem öffentlichen Schlüssel pk zu verschlüsseln, werden bei Eingabe von m und pk die folgenden Schritte vom Verschlüsselungsalgorithmus $\mathsf{Enc}(\mathsf{pk}, m)$ durchgeführt:

Schritt 1: Wähle zufällig eine Zahl $r \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: Berechne $c_1 = m \cdot y^r$ und $c_2 = g^r$.

Die Ausgabe des Verschlüsselungsalgorithmus $\mathsf{Enc}(\mathsf{pk}, m)$ ist der Geheimtext (c_1, c_2).

Entschlüsseln: Um einen Geheimtext (c_1, c_2) mit dem privaten Schlüssel sk zu entschlüsseln, wird der Klartext m bei Eingabe von sk und (c_1, c_2) vom Entschlüsselungsalgorithmus $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, (c_1, c_2))$ wie folgt berechnet:

$$m = c_1 \cdot c_2^{-x}.$$

Die Ausgabe des Entschlüsselungsalgorithmus $\mathsf{Dec}\,(\mathsf{pk}, \mathsf{sk}, (c_1, c_2))$ ist der Klartext m.

Bemerkung 2.3.2. Alternativ kann der Klartext m unter pk auch verschlüsselt werden, indem in Schritt 2 des Verschlüsselungsalgorithmus $\mathsf{Enc}(\mathsf{pk}, m)$ der Geheimtext (c_1, c_2) durch

$$c_1 = m \cdot g^r \quad \text{und} \quad c_2 = y^r$$

berechnet wird. Entsprechend wird der Entschlüsselungsalgorithmus $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, (c_1, c_2))$ den Geheimtext (c_1, c_2) zu

$$m = c_1 \cdot c_2^{-x^{-1}}$$

entschlüsseln. Diese Variante der ElGamal-Verschlüsselung wird für das in Abschnitt 6.1 vorgestellte elektronische Geldsystem von Y.Frankel, Y. Tsiounis und M. Yung [FTY98] benötigt und auch als *inverse ElGamal-Verschlüsselung* bezeichnet.

In der ursprünglichen, auf $\mathbb{Z}_p^*$ arbeitenden ElGamal-Verschlüsselung kann die quadratische Rest- bzw. Nichtresteigenschaft eines Klartextes mit der des entsprechenden Klartext in Beziehung gebracht werden (siehe z.B. [Mao04]). Da man diese Eigenschaft effizient nachprüfen kann, ist die ElGamal-Verschlüsselung auf $\mathbb{Z}_p^*$ nicht sicher im Sinne der polynomiellen Ununterscheidbarkeit. Im Gegensatz dazu wurde von Y. Tsiounis und M. Yung in [TY98] gezeigt, dass die hier vorgestellte Variante der ElGamal-Verschlüsselung sicher ist im Sinne der polynomiellen Ununterscheidbarkeit.

Satz 2.3.1. Unter der Decisional-Diffie-Hellman-Annahme ist die hier vorgestellte Variante des ElGamal-Verschlüsselungsverfahrens unter einem Angriff mit gewählten Klartexten sicher im Sinne der polynomiellen Ununterscheidbarkeit.

2.3.3 Das modifizierte ElGamal-Verschlüsselungsverfahren

Das in diesem Abschnitt vorgestellte Public-Key-Verschlüsselungsverfahren ist eine Erweiterung der in Abschnitt 2.3.2 vorgestellten ElGamal-Verschlüsselung und bildet eine der Grundlagen für die in den Abschnitten 6.2 und 6.3 entwickelten elektronischen Geldsysteme. Im Vergleich zur ElGamal-Verschlüsselung arbeitet das Verfahren mit zwei Generatoren und wird daher in Anlehnung an die ElGamal-Verschlüsselung auch als *modifiziertes ElGamal-Verschlüsselungsverfahren* bezeichnet. Ebenso wie die ElGamal-Verschlüsselung ist das modifizierte ElGamal-Verschlüsselungsverfahren unter der Decisional-Diffie-Hellman-Annahme unter einem Angriff mit gewählten Klartexten semantisch sicher.

Schlüsselgenerierung: Um ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ zu generieren, werden bei Eingabe des Sicherheitsparameters 1^k vom Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ folgende Schritte durchgeführt:

Schritt 1: Wähle zufällig eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$ und bestimme eine Primzahl q, so dass $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$. Wähle zufällig zwei Generator g, h der eindeutigen Untergruppe G_q von $\mathbb{Z}_p^*$ mit $|G_q| = q$.

Schritt 2: Wähle zufällig Zahlen $x, y \in_{\mathcal{R}} \mathbb{Z}_q$ und berechne $z = g^x h^y$.

Die Ausgabe von $\mathsf{Gen}(1^k)$ ist das Schlüsselpaar:

$$(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, h, z), (x, y)).$$

Verschlüsseln: Um einen Klartext $m \in G_q$ unter dem öffentlichen Schlüssel pk zu verschlüsseln, werden bei Eingabe von m und pk die folgenden Schritte vom Verschlüsselungsalgorithmus $\mathsf{Enc}(\mathsf{pk}, m)$ durchgeführt:

Schritt 1: Wähle zufällig eine Zahl $r \in_{\mathcal{R}} \mathbb{Z}_q$

Schritt 2: Berechne den Geheimtext (c_1, c_2, c_3) wie folgt:

$$c_1 = g^r,\ c_2 = h^r,\ c_3 = z^r m.$$

Die Ausgabe des Verschlüsselungsalgorithmus $\mathsf{Enc}(\mathsf{pk}, m)$ ist das Tupel (c_1, c_2, c_3).

Entschlüsseln: Um einen Geheimtext (c_1, c_2, c_3) mit dem privaten Schlüssel sk zu entschlüsseln, wird der Klartext m bei Eingabe von sk und (c_1, c_2, c_3) durch den Entschlüsselungsalgorithmus Dec wie folgt berechnet:

$$m = c_3 c_1^{-x} c_2^{-y}.$$

Die Ausgabe des Entschlüsselungsalgorithmus $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, (c_1, c_2, c_3))$ ist der Klartext m.

Für die Sicherheitsanalyse, der in den Abschnitten 6.2 und 6.3 entwickelten elektronischen Geldsysteme, spielt folgende Beziehung zwischen der ElGamal-Verschlüsselung aus Abschnitt 2.3.2 und der modifizierten ElGamal-Verschlüsselung eine entscheidende Rolle.

Satz 2.3.2. Sei $(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, y), x)$ ein Schlüsselpaar des ElGamal-Verschlüsselungsverfahrens, g_1 ein weiterer Generator von G_q sowie $m = g_1^u$ ein Klartext mit $u \in \mathbb{Z}_q$. Ist $(c_1, c_2) = (my^k, g^k)$ ein Geheimtext von m unter pk, so gibt es einen effizienten Algorithmus $\mathcal{M}$, der bei Eingabe von $(\mathsf{pk}, (c_1, c_2))$ ein Tupel $(\mathsf{pk}', (d_1, d_2, d_3))$ ausgibt, so dass (d_1, d_2, d_3) ein Geheimtext von m unter dem öffentlichen Schlüssel pk' des modifizierten ElGamal-Verschlüsselungsverfahrens ist.

Beweis. Bei Eingabe des öffentlichen Schlüssels $\mathsf{pk} = (p, q, g, y)$ und des Geheimtextes $(c_1, c_2) = (my^k, g^k)$ geht der Algorithmus $\mathcal{M}(\mathsf{pk}, (c_1, c_2))$ wie folgt vor:

Schritt 1: Wähle zufällig Zahlen $r_1, r_2, r_3 \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: Berechne $g_2 = g^{r_1}, g_3 = g_2^{r_2}$ und $y' = y g_3^{r_3}$.

Schritt 3: Berechne $d_1 = c_2^{r_1}, d_2 = d_1^{r_2}$ und $d_3 = c_1 d_2^{r_3}$

Die Ausgabe von $\mathcal{M}(\mathsf{pk}, (c_1, c_2))$ ist der öffentliche Schlüssel $\mathsf{pk}' = (p, q, g_2, g_3, y')$ und der Geheimtext (d_1, d_2, d_3).
Der öffentliche Schlüssel pk' ist ein öffentlicher Schlüssel des modifizierten ElGamal-Verfahrens. Das Tupel (d_1, d_2, d_3) ist ein Geheimtext des Klartextes m, denn

$$\begin{aligned} d_1 &= c_2^{r_1} = (g^k)^{r_1} = g_2^k \\ d_2 &= d_1^{r_2} = (g_2^k)^{r_2} = (g_2^{r_2})^k = g_3^k \\ d_3 &= c_1 d_2^{r_3} = m y^k d_2^{x_3} = m y^k (g_3^k)^{x_3} = m(y g_3^{x_3})^k = m y'^k \end{aligned}$$

□

Diese Beziehung gilt natürlich nicht nur zwischen der ElGamal- und der modifizierten ElGamal-Verschlüsselung, sondern auch zwischen der inversen ElGamal- und der modifizierten ElGamal-Verschlüsselung, wie der folgende Satz zeigt.

Satz 2.3.3. Sei $(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, y), x)$ ein Schlüsselpaar des inversen ElGamal-Verschlüsselungsverfahrens, g_1 ein weiterer Generator von G_q und $m = g_1^u$ ein Klartext mit $u \in \mathbb{Z}_q$. Ist $(c_1, c_2) = (mg^k, y^k)$ ein Geheimtext von m unter pk, so gibt es einen effizienten Algorithmus $\mathcal{M}$, der bei Eingabe von $(\mathsf{pk}, (c_1, c_2))$ ein Tupel $(\mathsf{pk}', (d_1, d_2, d_3))$ ausgibt, so dass (d_1, d_2, d_3) ein Geheimtext der Nachricht m unter dem öffentlichen Schlüssel pk' des modifizierten ElGamal-Verschlüsselungsverfahrens ist.

Beweis. Bei Eingabe des öffentlichen Schlüssels $\mathsf{pk} = (p, q, g, y)$ und des Geheimtextes $(c_1, c_2) = (mg^k, y^k)$ geht der Algorithmus $\mathcal{M}(\mathsf{pk}, (c_1, c_2))$ wie folgt vor:

Schritt 1: Wähle zufällig Werte $r_1, r_2, r_3 \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: Berechne $g_2 = y^{r_1}, g_3 = g_2^{r_2}$ und $y' = g g_3^{r_3}$.

Schritt 3: Berechne $d_1 = (c_2)^{r_1}, d_2 = d_1^{r_2}$ und $d_3 = c_1 d_2^{r_3}$

Die Ausgabe von $\mathcal{M}(\mathsf{pk}, (c_1, c_2))$ ist der öffentliche Schlüssel $\mathsf{pk}' = (p, q, g_2, g_3, y')$ und der Geheimtext (d_1, d_2, d_3).
Der öffentliche Schlüssel pk' ist ein öffentlicher Schlüssel des modifizierten ElGamal-Verfahrens. Das Tupel (d_1, d_2, d_3) ist ein Geheimtext der Nachricht m, denn

$$\begin{aligned} d_1 &= c_2^{r_1} = (y^k)^{r_1} = (y^{r_1})^k = g_2^k \\ d_2 &= d_1^{r_2} = (g_2^k)^{r_2} = (g_2^{r_2})^k = g_3^k \\ d_3 &= c_1 d_2^{x_3} = m g^k d_2^{r_3} = m g^k (g_3^k)^{r_3} = m(g g_3^{r_3})^k = m y'^k \end{aligned}$$

□

2.4 Hashfunktionen

Eine *Hashfunktion* bildet eine Zeichenkette beliebiger Länge auf einen so genannten *Hashwert*, eine Zeichenkette fester Länge, ab. Formal ist eine Hashfunktion durch die folgende Definition gegeben:

Definition 2.4.1 (**Hashfunktionen**)**.** Es sei $\Sigma = \{0, 1\}$. Eine Funktion $\mathrm{H} : \Sigma^* \to \Sigma^n$ heißt *Hashfunktion*, falls es einen polynomiellen Algorithmus gibt, der für jedes Urbild $m \in \Sigma^*$ das Bild $h(m)$ berechnet. Eine Hashfunktion H heißt *Einweg-Hashfunktion*, falls es nicht effizient möglich ist, zu einem zufällig gewählten $y \in \Sigma^n$ ein $x \in \Sigma^*$ zu finden, so dass $h(x) = y$. Eine Hashfunktion h heißt

- *schwach kollisionsresistent*, falls es keinen effizienten Algorithmus gibt, der bei Eingabe von $m \in \Sigma^*$ ein $m' \in \Sigma^*$ findet, so dass $\mathrm{H}(m) = \mathrm{H}(m')$.
- *stark kollisionsresistent*, falls es keinen effizienten Algorithmus gibt, der $m, m' \in \Sigma^*$ mit $m \neq m'$ findet, so dass $\mathrm{H}(m) = \mathrm{H}(m')$.

Eines der wichtigsten Einsatzgebiete von Hashfunktionen sind die digitalen Signaturverfahren (vgl. auch Abschnitt 2.5). Zum einen werden Hashfunktionen innerhalb der so genannten *Hash-and-Sign-Verfahren* (siehe z.B. [BSW06], [BNS05]) eingesetzt, zum anderen werden sie dazu verwendet *Honest-Verifier-Zero-Knowledge-Beweise* (vgl. Abschnitt 2.7) mittels der *Fiat-Shamir-Heuristik* (vgl. Abschnitt 2.8) in digitale Signaturverfahren zu transformieren.

2.5 Digitale Signaturen

Das von W. Diffie und M. Hellman im Jahre 1976 vorgeschlagene Prinzip der Public-Key-Kryptografie ermöglicht es nicht nur Public-Key-Verschlüsselungsverfahren, sondern auch *digitale Signaturverfahren* zu entwerfen. Ziel einer *digitalen Signatur* ist es, die wesentlichen Eigenschaften einer handschriftlichen Unterschrift in digitaler Form zu verwirklichen. Genauso wie in einem Public-Key-Verschlüsselungsverfahren verfügt jeder Teilnehmer über einen privaten und einen öffentlichen Schlüssel. Zum Signieren einer Nachricht verwendet der Signierer seinen privaten Schlüssel. Mit dem öffentlichen Schlüssel des Signierers kann jeder Teilnehmer dessen Signaturen verifizieren.

Formal ist ein digitales Signaturverfahren wie folgt definiert:

Definition 2.5.1 (**Digitales Signaturverfahren**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter und M die Menge aller Nachrichten. Gegeben seien die folgenden drei polynomiellen Algorithmen:

1. Der *Schlüsselgenerierungsalgorithmus* Gen ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k einen *öffentlichen Schlüssel* pk und einen *privaten Schlüssel* sk erzeugt. Die Ausgabe von $\mathsf{Gen}(1^k)$ ist das Paar $(\mathsf{pk}, \mathsf{sk})$.

2. Der *Signaturalgorithmus* Sign ist ein probabilistischer Algorithmus und erhält als Eingabe eine Nachricht $m \in M$, den privaten Schlüssel sk und den öffentlichen Schlüssel pk. Die Ausgabe von $\mathsf{Sign}(m, \mathsf{sk}, \mathsf{pk})$ ist eine Signatur s.

3. Der *Verifikationsalgorithmus* Verify ist ein deterministischer Algorithmus und erhält als Eingabe eine Nachricht $m \in M$, den öffentlichen Schlüssel pk, sowie eine Signatur s. Die Ausgabe von $\mathsf{Verify}(m, \mathsf{pk}, s)$ ist der Wert 1 (= true) oder 0 (= false).

Das Tripel $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ heißt *digitales Signaturverfahren*, falls folgende Bedingung erfüllt ist: Für jedes Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ und für alle $m \in M$ gilt:

$$\mathsf{Verify}(m, \mathsf{pk}, \mathsf{Sign}(m, \mathsf{sk}, \mathsf{pk})) = 1.$$

Angriffsarten

Analog zu den Public-Key-Verschlüsselungsverfahren müssen zur Analyse der Sicherheit digitaler Signaturverfahren die möglichen Angriffe auf das Verfahren berücksichtigt werden. Man unterscheidet dabei zwischen den folgenden vier Angriffsarten:

- *Angriff ohne bekannte Signaturen*: Der Angreifer $\mathcal{A}$ kennt nur den öffentlichen Schlüssel des Signierers.
- *Angriff mit bekannten Signaturen*: Der Angreifer $\mathcal{A}$ kennt den öffentlichen Schlüssel des Signierers und mehrere gültige vom Signierer erzeugte Nachrichten-Signaturpaare. Auf die Wahl der Nachrichten hat $\mathcal{A}$ keinen Einfluss.
- *Angriff mit gewählten Nachrichten*: Der Angreifer $\mathcal{A}$ kennt den öffentlichen Schlüssel und kann sich mehrere Nachrichten wählen, zu denen der Signierer gültige Signaturen erzeugt.
- *Adaptiver Angriff mit gewählten Nachrichten*: Der Angreifer $\mathcal{A}$ kennt den öffentlichen Schlüssel und kann sich, wie beim Angriff mit gewählten Nachrichten, mehrere Nachrichten wählen, zu denen der Signierer gültige Signaturen erzeugt. Dabei kann $\mathcal{A}$ seine Nachrichten aber in Abhängigkeit vorangegangener Ergebnisse wählen.

Mögliche Erfolge

Die Sicherheit eines digitalen Signaturverfahrens wird anhand des möglichen Erfolgs eines Angriffs auf das Verfahren gemessen. Man unterscheidet dabei zwischen den folgenden vier Erfolgsstufen:

- *Existentielle Fälschbarkeit*: Der Angreifer $\mathcal{A}$ kann zu einer Nachricht, die nicht notwendig von ihm ausgesucht wurde, eine gültige Signatur erstellen.
- *Selektive Fälschbarkeit*: Der Angreifer $\mathcal{A}$ kann zu einer oder mehreren Nachrichten seiner Wahl gültige Signaturen erstellen.

- *Universelle Fälschbarkeit*: Der Angreifer $\mathcal{A}$ kann gültige Signaturen zu beliebigen Nachrichten erstellen, ist aber nicht im Besitz des privaten Signaturschlüssels.

- *Kompromittierung des Signaturschlüssels*: Der Angreifer $\mathcal{A}$ kann den privaten Signaturschlüssel berechnen.

Ein digitales Signaturverfahren erfüllt das höchste Maß an Sicherheit, wenn es für einen effizienten Angreifer unter einem adaptiven Angriff mit gewählten Nachrichten nicht existentiell fälschbar ist.

2.5.1 Die Schnorr-Signatur

Im Jahre 1989 wurde von C.-P. Schnorr ein digitales Signaturverfahren vorgestellt, das eine Variante des ElGamal-Signaturverfahrens [ElG85] darstellt und auch als *Schnorr-Signatur* [Sch91] bezeichnet wird. Die Sicherheit der Schnorr-Signatur basiert auf der Diskreter-Logarithmus-Annahme.

Schlüsselgenerierung: Um ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ zu generieren, werden bei Eingabe des Sicherheitsparameters 1^k vom Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ folgende Schritte durchgeführt:

Schritt 1: Wähle zufällig eine Primzahl p mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$ und bestimme eine Primzahl q, so dass $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$. Wähle zufällig einen Generator g der eindeutigen Untergruppe G_q von $\mathbb{Z}_p^*$ mit $|G_q| = q$.

Schritt 2: Wähle zufällig eine Zahl $x \in_{\mathcal{R}} \mathbb{Z}_q$ und berechne $y = g^x$.

Schritt 3: Wähle eine kollisionsresistente Hashfunktion $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Die Ausgabe des Algorithmus $\mathsf{Gen}(1^k)$ ist das Paar $(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, y, \mathrm{H}), x)$.

Signieren einer Nachricht: Um eine Signatur der Nachricht $m \in \{0,1\}^*$ mit dem privaten Schlüssel $\mathsf{sk} = x$ zu erstellen, werden bei Eingabe von m, pk und sk vom Signaturalgorithmus $\mathsf{Sign}(m, \mathsf{sk}, \mathsf{pk})$ folgende Schritte durchgeführt:

Schritt 1: Wähle zufällig eine Zahl $w \in_{\mathcal{R}} \mathbb{Z}_q$ und berechne $a = g^w$.

Schritt 2: Berechne $c = \mathrm{H}(m, a) \in \mathbb{Z}_q$ und $r = cx + w \bmod q$.

Die Ausgabe des Signaturalgorithmus $\mathsf{Sign}(m, \mathsf{sk}, \mathsf{pk})$ ist die Signatur $s = (c, r)$.

Verifikation einer Signatur: Um die Signatur $s = (c, r)$ einer Nachricht m zu verifizieren, werden bei Eingabe von m, s und pk vom Verifikationsalgorithmus $\mathsf{Verify}(m, \mathsf{pk}, s)$ folgende Zahlen berechnet:

$$a' = g^r y^{-c} \text{ und } c' = \mathrm{H}(m, a').$$

Die Ausgabe des Verifikationsalgorithmus $\mathsf{Verify}(m, \mathsf{pk}, s)$ ist 1, falls $c' = c$ und 0 sonst.

Im *Random-Oracle-Modell* (vgl. Abschnitt 2.9) konnten D. Pointcheval und J. Stern [PS00] mit Hilfe des *Forking-Lemma* (siehe z.B. [PS00]) den folgenden Satz beweisen.

Satz 2.5.1. Im Random-Oracle-Modell ist die Schnorr-Signatur unter der Diskreter-Logarithmus-Annahme nicht existentiell fälschbar unter einem adaptiven Angriff mit gewählten Nachrichten.

2.6 Blinde digitale Signaturen

Das Konzept der *blinden Signaturen* geht auf D. Chaum [Cha83] zurück und wurde von ihm im Jahre 1982 vorgeschlagen. In einem *blinden digitalen Signaturverfahren* wird die Signatur einer Nachricht interaktiv zwischen dem Signierer der Nachricht und dem Empfänger der Signatur erstellt. Dabei erhält der Empfänger eine gültige Signatur auf eine von ihm gewählte Nachricht, ohne dass der Signierer nachvollziehen kann welche Nachricht von ihm signiert wurde. Insbesondere kann der Signierer, falls ihm zu einem späteren Zeitpunkt die Nachricht zusammen mit der Signatur vorliegt, nicht feststellen, für wen er die Nachricht signiert hat. Eines der wichtigsten Anwendungsgebiete blinder Signaturen sind die elektronischen Geldsysteme. Dort werden blinde Signaturen häufig zur Gewährleistung der Anonymität eingesetzt (z. B. in [CFN89], [Bra94], [Fer93], [FTY98], [GT03], [NS06]).

Eine vollständige formale Beschreibung blinder digitaler Signaturverfahren wurde erstmals von A. Jules, M. Luby und R. Ostrovsky [JLO97] gegeben. Gemäß [JLO97] ist ein blindes digitales Signaturverfahren wie folgt definiert:

Definition 2.6.1 (**Blindes digitales Signaturverfahren**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter und M die Menge aller Nachrichten. Gegeben seien die beiden polynomiellen Algorithmen Gen, Verify aus Definition 2.5.1.
Weiter seien $\mathcal{S}$ und $\mathcal{R}$ zwei probabilistische, polynomielle interaktive Turing-Maschinen, die das interaktive *Signaturprotokoll* Sign ausführen, dessen Rundenzahl polynomiell in k ist. Für ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ sei $\mathsf{pk}, m \in M$ die Eingabe des *Empfängers* $\mathcal{R}$ und $(\mathsf{pk}, \mathsf{sk})$ die Eingabe des *Signierers* $\mathcal{S}$. Die Ausgabe von $\mathcal{R}(\mathsf{pk}, m)$ ist entweder die Signatur s oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{S}(\mathsf{pk}, \mathsf{sk})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$. Das Tupel $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ heißt *blindes digitales Signaturverfahren*, falls die folgende Bedingung für jedes Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$ und für alle $m \in M$ erfüllt ist:

Folgen $\mathcal{R}$ und $\mathcal{S}$ dem Signatur-Protokoll, so erhält $\mathcal{R}$ als Ausgabe s und es gilt

$$\mathsf{Verify}(m, \mathsf{pk}, s) = 1.$$

Sicherheit blinder digitaler Signaturen

Die Sicherheitsanalyse blinder Signaturverfahren umfasst zwei Aspekte. Zum einen die bereits erwähnte *Blindheit* der Signatur, die im Interesse des Empfängers $\mathcal{R}$ liegt und zum anderen die *Unfälschbarkeit* der Signatur, die vom Signierer $\mathcal{S}$ gefordert wird. Wobei Unfälschbarkeit in diesem Kontext bedeutet, dass $\mathcal{R}$ nach der Durchführung von ℓ Signaturprotokollen mit $\mathcal{S}$ nicht in der Lage sein darf $\ell + 1$ gültige Signaturen zu erstellen.

Während der Blindheitsbegriff von D. Chaum [Cha83] bereits 1982 eingeführt wurde und die meisten blinden Signaturverfahren in Bezug auf ihre Blindheit gut untersucht sind, wurde die Unfälschbarkeit der Verfahren mehr oder weniger implizit vorausgesetzt und erstmals 1996 von D. Pointcheval und J. Stern [PS96], [PS00] formalisiert und genauer analysiert.

Analog zu den digitalen Signaturverfahren müssen zur Analyse der Unfälschbarkeit blinder digitaler Signaturverfahren die möglichen Angriffsszenarien berücksichtigt werden. In [PS00] wird dabei zwischen den beiden folgenden Angriffen unterschieden:

- *Sequentieller Angriff*: Der Angreifer $\mathcal{A}$ interagiert sequentiell mit dem Signierer $\mathcal{S}$. Das heißt, der Angreifer $\mathcal{A}$ lässt sich nacheinander Signaturen von $\mathcal{S}$ erstellen.
- *Paralleler Angriff*: Der Angreifer $\mathcal{A}$ interagiert parallel mit dem Signierer $\mathcal{S}$. Das heißt, der Angreifer $\mathcal{A}$ lässt sich gleichzeitig und möglicherweise abhängig voneinander Signaturen von $\mathcal{S}$ erstellen. Dabei kann $\mathcal{A}$ zu einem beliebigen Zeitpunkt ein neues Signaturprotokoll starten und die Nachrichten in Abhängigkeit von zuvor erhaltenen Nachrichten-Signaturpaaren wählen.

Wie bereits erwähnt bedeutet Unfälschbarkeit, dass ein Angreifer nach ℓ Interaktionen mit dem Signierer nicht in der Lage sein darf $\ell + 1$ Signaturen zu erstellen. Da hierbei die Größe von ℓ von der Rechen- und Speicherkapazität des Angreifers abhängt, wird in [PS00] zwischen den folgenden Erfolgsstufen unterschieden:

- $(\ell, \ell + 1)$-*Fälschung*: Der Angreifer $\mathcal{A}$ erstellt $\ell + 1$ gültige Signaturen nach ℓ Interaktionen mit dem Signierer $\mathcal{S}$.
- *One-More-Fälschung*: Der Angreifer $\mathcal{A}$ erstellt $\ell + 1$ gültige Signaturen nach ℓ Interaktionen mit dem Signierer $\mathcal{S}$, wobei ℓ polynomiell im Sicherheitsparameter k beschränkt ist.
- *Starke One-More-Fälschung*: Der Angreifer $\mathcal{A}$ erstellt $\ell + 1$ gültige Signaturen nach ℓ Interaktionen mit dem Signierer $\mathcal{S}$, wobei es eine Konstante c gibt, so dass $\ell \leq (\log k)^c$ gilt. Wobei k der Sicherheitsparameter ist.

Weiter formalisiert wurde der Begriff der Unfälschbarkeit in [JLO97]. Aufbauend auf der One-More-Fälschung wird die Unfälschbarkeit in [JLO97] durch das im Folgenden beschriebene Spiel modelliert.

Spiel 2.6.1 (**Unfälschbarkeit**).

Schritt 1: Der Signierer $\mathcal{S}$ führt bei Eingabe des Sicherheitsparameters 1^k den Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ aus. Die Ausgabe von $\mathsf{Gen}(1^k)$ ist ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$.

Schritt 2: Der Angreifer $\mathcal{A}$ führt polynomiell in k viele parallele und nicht notwendig unabhängige Protokolle mit identischen Kopien von $\mathcal{S}$ aus, wobei $\mathcal{A}$ adaptiv entscheidet, wann der Angriff aufhört. Sei ℓ die Anzahl der durchgeführten Protokolle, bei denen $\mathcal{S}$ am Ende $\mathsf{completed}$ ausgibt.

Schritt 3: Der Angreifer $\mathcal{A}$ gibt j gültige Nachrichten-Signatur-Paare aus

$$(m_1, s_1), \ldots, (m_j, s_j).$$

Der Angreifer $\mathcal{A}$ gewinnt Spiel 2.6.1, falls die Anzahl j der in Schritt 3 ausgegebenen gültigen Nachrichten-Signatur-Paare größer ist als die Anzahl ℓ der durchgeführten Protokolle. Das heißt, der Angreifer $\mathcal{A}$ gewinnt, falls $j > \ell$.

Auch die Blindheit des Signaturverfahrens wird in [JLO97], ähnlich zu Spiel 2.6.1, durch ein Spiel modelliert, welches im Folgenden dargestellt wird.

Spiel 2.6.2 (**Blindheit**). Sei $b \in_{\mathcal{R}} \{0, 1\}$, so dass der Angreifer $\mathcal{A}$ das zufällig gewählte Bit b nicht kennt.

Schritt 1: Der Angreifer $\mathcal{A}$ führt bei Eingabe des Sicherheitsparameters 1^k den Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ aus. Die Ausgabe von $\mathsf{Gen}(1^k)$ ist ein Schlüsselpaar $(\mathsf{pk}, \mathsf{sk})$.

Schritt 2: Der Angreifer $\mathcal{A}$ wählt zwei Nachrichten $m_0, m_1 \in M$, deren Länge polynomiell in k ist und die von pk und sk abhängen können.

Schritt 3: Der Angreifer $\mathcal{A}$ führt zwei parallele und nicht notwendig unabhängige Protokolle mit zwei Empfängern $\mathcal{R}_0$ und $\mathcal{R}_1$ aus. Beide Empfänger erhalten als Eingabe den öffentlichen Schlüssel pk und eine der beiden Nachrichten: $\mathcal{R}_0(\mathsf{pk}, m_b)$, $\mathcal{R}_1(\mathsf{pk}, m_{1-b})$.

Schritt 4: Geben $\mathcal{R}_0$ und $\mathcal{R}_1$ am Ende der durchgeführten Protokolle Signaturen s_b, bzw. s_{1-b} aus, so erhält $\mathcal{A}$ als zusätzliche Information die Nachrichten-Signatur-Paare $((m_0, s_0), (m_1, s_1))$.

Schritt 5: Der Angreifer $\mathcal{A}$ gibt ein $b' \in \{0, 1\}$ aus.

Der Angreifer $\mathcal{A}$ gewinnt Spiel 2.6.2, falls $\mathcal{A}$ nach Ablauf des Spiels entscheiden kann, welche Nachricht $\mathcal{A}$ für welchen Empfänger signiert hat. Das heißt, der Angreifer $\mathcal{A}$ gewinnt, falls $b = b'$. Diese beiden Spiele bilden die Grundlage des in [JLO97] vorgeschlagenen Sicherheitsmodells blinder Signaturverfahren. Dabei werden Unfälschbarkeit

und Blindheit des Verfahrens gleichzeitig definiert, d.h. die Erfolgswahrscheinlichkeit eines Angreifers $\mathcal{A}$ in den beiden Spielen wird durch die gleiche vernachlässigbare Funktion beschrieben.

Definition 2.6.2 (**Sicherheit blinder Signaturverfahren**)**.** Ein blindes digitales Signaturverfahren $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ mit Sicherheitsparameter $k \in \mathbb{N}$ heißt *sicher*, falls für alle effizienten Algorithmen $\mathcal{A}$ eine vernachlässigbare Funktion ν existiert, so dass

1. *Unfälschbarkeit*: Für die Erfolgswahrscheinlichkeit $\epsilon(k)$ von $\mathcal{A}$ in Spiel 2.6.1 gilt:

$$\epsilon(k) \leq \nu(k).$$

2. *Blindheit*: Für die Erfolgswahrscheinlichkeit $\epsilon(k)$ von $\mathcal{A}$ in Spiel 2.6.2 gilt:

$$\epsilon(k) \leq 1/2 + \nu(k).$$

Bemerkung 2.6.1. Der in [JLO97] definierte Blindheitsbegriff wird auch als *rechnerische Blindheit* bezeichnet. Rechnerisch blinde Signaturen findet man überwiegend im Bereich der *fairen* blinden Signaturen (siehe z.B. [CPS95]), die genauso wie faire elektronische Geldsysteme (vgl. auch Abschnitt 3.3) eine Möglichkeit der Deanonymisierung bieten. Dahingegen sind z.B. [Cha83] oder die in Abschnitt 2.6.1 vorgestellte blinde Schnorr-Signatur *perfekt blind*, d.h. sie erfüllen die folgende Definition der *perfekten Blindheit*, wie sie zum Beispiel in [CPS94] vorgeschlagen wurde.

Definition 2.6.3 (**Perfekte Blindheit**)**.** Ein blindes Signaturverfahren heißt *perfekt blind*, falls die Protokollansicht $\mathsf{view}_{\mathcal{S}}^{\mathcal{R}}(P)$ des Signierers $\mathcal{S}$ bei Durchführung des Signaturprotokolls P mit dem Empfänger $\mathcal{R}$ stochastisch unabhängig ist vom Nachrichten-Signatur-Paar (m, s).

2.6.1 Die blinde Schnorr-Signatur

Es gibt mehrere blinde Varianten der Schnorr-Signatur. Die Idee zur Blendung der Schnorr-Signatur stammt von T. Okamoto [Oka92]. Die im Folgenden beschriebene Variante geht auf P. Horster, M. Michels und H. Petersen [HMP94] zurück und bildet die Grundlage des elektronischen Geldsystems von S. Brands [Bra94], auf dem die in den Abschnitten 6.2 und 6.3 entwickelten elektronischen Geldsysteme basieren.

Schlüsselgenerierung: Die Schlüsselgenerierung entspricht der aus Abschnitt 2.5.1. Die Ausgabe des Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k)$ ist das Schlüsselpaar

$$(\mathsf{pk}, \mathsf{sk}) = ((p, q, g, y, \mathrm{H}), x).$$

Signaturprotokoll: Um eine Signatur der Nachricht $m \in \{0,1\}^*$ zu erstellen, wird zwischen dem Empfänger $\mathcal{R}(\mathsf{pk}, m)$ und dem Signierer $\mathcal{S}(\mathsf{pk}, \mathsf{sk})$ das in Abbildung 2.1 dargestellte Signaturprotokoll durchgeführt. Die Ausgabe von $\mathcal{R}(\mathsf{pk}, m)$ ist die Signatur $s = (c', r')$ der Nachricht m.

$\mathcal{R}(\mathsf{pk}, m)$		$\mathcal{S}(\mathsf{pk}, \mathsf{sk})$
		$w \in_{\mathcal{R}} \mathbb{Z}_q$
		$a = g^w$
	$\overleftarrow{a}$	
$u, v \in_{\mathcal{R}} \mathbb{Z}_q$		
$a' = a^u g^v$		
$c' = \mathrm{H}(m, a')$		
$c = c' u^{-1} \bmod q$		
	$\overrightarrow{c}$	
	$\overleftarrow{r}$	$r = cx + w \bmod q$
$r' = ru + v \bmod q$		

Abbildung 2.1: Die blinde Schnorr-Signatur

Verifikation einer Signatur: Erfolgt gemäß Abschnitt 2.5.1.

Bei der blinden Schnorr-Signatur handelt es sich um eine *perfekt blinde* Signatur. Allerdings existiert für die blinde Schnorr-Signatur kein vollständiger Sicherheitsbeweis. Insbesondere konnte die Unfälschbarkeit der blinden Schnorr-Signatur im Sinne der One-More-Fälschung bis heute noch nicht gezeigt werden.

2.6.2 Restriktiv blinde Signaturverfahren

Eine spezielle Klasse blinder Signaturverfahren sind die von S. Brands vorgestellten *restriktiv blinden Signaturen* [Bra93], [Bra94]. In einem restriktiv blinden Signaturverfahren wird der Empfänger im Blenden seiner Nachricht eingeschränkt, so dass in der geblendeten Nachricht, für die der Empfänger eine Signatur des Signierers erhält, die „Struktur“ der ungeblendeten Nachricht erhalten bleibt. Auf Grund dieser Eigenschaft spielen restriktiv blinde Signaturen insbesondere als Baustein elektronischer Geldsysteme eine bedeutende Rolle (z.B. [Bra94], [FTY96], [FTY98], [GT03]).
Da wir im weiteren Verlauf dieser Arbeit auf der eindeutigen zyklischen Untergruppe G_q der Primzahlordnung q von $\mathbb{Z}_p^*$ arbeiten, definieren wir die *Restriktivität* nur für blinde Signaturverfahren, die auf G_q operieren. Ein auf G_q arbeitendes blindes Signaturverfahren $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ heißt restriktiv blind, falls es folgender Definition genügt:

Definition 2.6.4. Gegeben sei eine Nachricht $m \in G_q$. Zu Beginn des blinden Signaturprotokolls Sign kennt der Empfänger $\mathcal{R}$ die Darstellung $(a_1, \ldots, a_n)$ von m bzgl. des Generatortupels $(g_1, \ldots, g_n)$. Nachdem das Protokoll Sign beendet wurde kennt $\mathcal{R}$ die Darstellung $(b_1, \ldots, b_n)$ der geblendeten Nachricht m'. Das blinde Signaturverfahren $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ heißt *restriktiv* blindes Signaturverfahren, falls für alle Nachrichten $m \in G_q$ und alle möglichen Blendungen, die $\mathcal{R}$ während des Protokolls Sign durchführt, *blendungsinvariante* Funktionen I_1 und I_2 existieren, so dass

$$I_1(a_1, \ldots, a_n) = I_2(b_1, \ldots, b_n).$$

2.7 Zero-Knowledge-Beweise

Neben den blinden Signaturen (vgl. Abschnitt 2.6) sind die *Zero-Knowlege-Beweise* einer der wichtigsten kryptografischen Bausteine elektronischer Geldsysteme. Beispielsweise werden Zero-Knowledge-Beweise dazu eingesetzt, die erfolgreiche Durchführung von Kunden- bzw. Münztracing in fairen Geldsystemen zu garantieren.
Die Idee der Zero-Knowledge-Beweise geht auf S. Goldwasser, S. Micali und C. Rackoff [GMR89] zurück und beruht auf den von den Autoren vorgeschlagenen *interaktiven Beweisen*. In einem interaktiven Beweis zwischen einem *Prover* $\mathcal{P}$ (dem Beweisführenden) und einem *Verifier* $\mathcal{V}$ (dem Verifizierenden), versucht der Prover $\mathcal{P}$ den Verifier $\mathcal{V}$ von der Gültigkeit einer Behauptung zu überzeugen. Dabei findet die Kommunikation zwischen $\mathcal{P}$ und $\mathcal{V}$ interaktiv statt, d.h. beide führen gemeinsam ein *Challenge-and-Response-Protokoll* durch, in dem $\mathcal{P}$ auf die *Challenge* (Frage) von $\mathcal{V}$ mit einer *Response* (Antwort) antwortet. Anhand der Response kann $\mathcal{V}$ den Beweis prüfen, d.h. $\mathcal{V}$ akzeptiert den Beweis oder lehnt ihn ab.
Im Weiteren wird ein *betrügerischen Prover*, der versucht $\mathcal{V}$ von einer falschen Behauptung zu überzeugen, mit $\mathcal{P}^*$ und ein *betrügerischer Verifier*, der sich nicht notwendig an das Protokoll hält, mit $\mathcal{V}^*$ bezeichnet. Formal ist ein *interaktives Beweissystem* wie folgt definiert:

Definition 2.7.1 (**Interaktives Beweissystem**)**.** Gegeben sei ein Alphabet Σ und eine Sprache L über Σ^*. Ein Paar $(\mathcal{P}, \mathcal{V})$ von interaktiven Turing-Maschinen heißt ein *interaktives Beweissystem* für die Behauptung $x \in L$, falls die folgenden beiden Eigenschaften erfüllt sind:

1. *Durchführbarkeit* (Completeness): Es gibt ein $k_0 \in \mathbb{N}$, so dass für alle $x \in L$ mit $|x| = k \geq k_0$ gilt:
$$\mathbf{P}\left[(\mathcal{P}, \mathcal{V})(x) = \mathsf{true}\right] \geq 1 - \nu(k),$$
mit einer in k vernachlässigbaren Funktion ν.

2. *Korrektheit* (Soundness): Es gibt ein $k_0 \in \mathbb{N}$, so dass für alle $x \notin L$ mit $|x| = k \geq k_0$ gilt:
$$\mathbf{P}\left[(\mathcal{P}^*, \mathcal{V})(x) = \mathsf{false}\right] \geq 1 - \mu(k),$$
mit einer in k vernachlässigbaren Funktion ν.

Das heißt, die Durchführbarkeit eines interaktiven Beweises garantiert einem *ehrlichen* Prover $\mathcal{P}$, dass er den Verifier $\mathcal{V}$ mit großer Wahrscheinlichkeit von der Gültigkeit seiner Behauptung überzeugen kann. Dahingegen garantiert die Korrektheit des Beweissystems, dass ein *betrügerischer Prover* $\mathcal{P}^*$ den Verifier $\mathcal{V}$ nur mit vernachlässigbarer Wahrscheinlichkeit von der Gültigkeit einer falschen Behauptung überzeugen kann.

Definition 2.7.2. Gegeben sei eine Sprache L und und ein interaktives Beweissystem $(\mathcal{P}, \mathcal{V})$ für die Behauptung $x \in L$. Mit $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}(x)$ bezeichnen wir die *Protokollansicht* von $\mathcal{V}$ während der Interaktion mit $\mathcal{P}$.

Hat ein solcher interaktiver Beweis zusätzlich die Eigenschaft, dass der Verifier $\mathcal{V}$ während der Durchführung des interaktiven Beweises keine weiteren Informationen erhält, außer dass die Behauptung des Provers $\mathcal{P}$ richtig ist, so wird der Beweis *Zero-Knowledge-Beweis* genannt (siehe z.B. [BSW06], [Mao04]). Die *Zero-Knowledge-Eigenschaft* des interaktiven Beweises besagt, dass sich der Beweis von einem Simulator $\mathcal{S}$, der die Behauptung selbst nicht beweisen kann, effizient simulieren lässt und die Simulation nicht von einem echten Beweis zu unterscheiden ist. Je nach Güte der Simulation und Verhalten des Verifiers unterscheidet man zwischen einem *perfekten*, *rechnerischen* oder *Honest-Verifier*-Zero-Knowledge-Beweis. Formal ist die Zero-Knowledge-Eigenschaft in diesen Fällen wie folgt definiert:

Definition 2.7.3. Gegeben sei eine Sprache L und ein interaktives Beweissystem $(\mathcal{P}, \mathcal{V})$ für die Behauptung $x \in L$. Das interaktive Beweissystem $(\mathcal{P}, \mathcal{V})$ heißt

- *perfekter Zero-Knowledge-Beweis*, falls es für jede probabilistische polynomielle interaktive Turing-Maschine $\mathcal{V}^*$ einen probabilistischen polynomiellen Algorithmus $\mathcal{S}$ gibt, so dass für alle $x \in L$ die Ausgabe von $\mathcal{S}(x)$ und die Protokollansicht $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ identisch verteilt sind.

- *rechnerischer Zero-Knowledge-Beweis*, falls es für jede probabilistische polynomielle interaktive Turing-Maschine $\mathcal{V}^*$ einen probabilistischen polynomiellen Algorithmus $\mathcal{S}$ gibt, so dass für alle $x \in L$ die Ausgabe von $\mathcal{S}(x)$ und die Protokollansicht $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ rechnerisch ununterscheidbar sind.

- *Honest-Verifier-Zero-Knowledge-Beweis*, falls es einen probabilistischen polynomiellen Algorithmus $\mathcal{S}$ gibt, so dass für alle $x \in L$ die Ausgabe von $\mathcal{S}(x)$ und Protokollansicht $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x)$ rechnerisch ununterscheidbar sind.

Die Ausgabe von $\mathcal{S}(x)$ und $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x)$ heißen *rechnerisch ununterscheidbar*, falls für alle $x \in L$ mit $|x| = k$ und für alle probabilistischen polynomiellen Algorithmen $\mathcal{D}$ gilt:

$$|\mathbf{P}\left[\mathcal{D}(x, \mathcal{S}(x)) = 1\right] - \mathbf{P}\left[\mathcal{D}(x, \mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x)) = 1\right]| \leq \nu(k),$$

wobei ν eine in k vernachlässigbare Funktion ist. Der Algorithmus $\mathcal{D}$ ist ein Unterscheidungsalgorithmus, der versucht die beiden Verteilungen zu unterscheiden.

Während in einem perfekten bzw. rechnerischen Zero-Knowledge-Beweis die Simulierbarkeit der Protokollansichten eines betrügerischen Verifiers $\mathcal{V}^*$, der sich nicht notwendigerweise gemäß des Protokolls verhält, gefordert wird, wird in einem Honest-Verifier-Zero-Knowledge-Beweis lediglich die Simulierbarkeit der Protokollansichten eines ehrlichen Verifiers $\mathcal{V}$ gefordert. Honest-Verifier-Zero-Knowledge (HVZK) ist somit der schwächste dieser drei Zero-Knowledge-Begriffe. Für reale Anwendungen und Sicherheitsbeweise im Random-Oracle-Modell (vgl. Abschnitt 2.9) sind Honest-Verifier-Zero-Knowledge-Beweise dennoch von großer Bedeutung, da sie mit Hilfe der *Fiat-Shamir-Heuristik* (vgl. Abschnitt 2.8) in digitale Signaturverfahren und im Random-Oracle-Modell in *nichtinteraktive* Zero-Knowledge-Beweise transformiert werden können.

2.7.1 Zero-Knowledge-Beweise bestimmter Darstellungen

Um den korrekten Protokollablauf der in Kapitel 6 entwickelten neuen elektronischen Geldsysteme zu garantieren, werden in den dort vorgestellten Protokollen verschiedene Honest-Verifier-Zero-Knowledge-Beweise benötigt, die im folgenden Abschnitt vorgestellt werden. Da die später vorgestellten Systeme stets auf der eindeutig bestimmten zyklischen Untergruppe G_q von $\mathbb{Z}_p^*$ mit Primzahlordnung q arbeiten, sei diese auch im Folgenden gegeben (für die genau Wahl der Primzahlen p und q siehe Abschnitt 2.5.1). Zusätzlich seien a, b, c sowie $g_1, \ldots g_5$ zufällig gewählte, öffentlich bekannte Generatoren von G_q, von denen der Prover $\mathcal{P}$ die diskreten Logarithmen der einzelnen Generatoren untereinander nicht kennt.

2.7.1.1 Der interaktive Beweis HVZK $\left[(\alpha, \beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$

Der Prover $\mathcal{P}$ kennt Werte $A, B, C \in G_q$, die auch dem Verifier $\mathcal{V}$ bekannt sind. Mit dem interaktiven Beweis

$$\text{HVZK}\left[(\alpha, \beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$$

möchte $\mathcal{P}$ den Verifier $\mathcal{V}$ davon überzeugen, dass $\mathcal{P}$ zwei Werte α, β kennt, so dass (α, β) eine Darstellung von A bzgl. (g_1, g_2) und β eine Darstellung von B und C bzgl. g_3 bzw. g_4 ist und zusätzlich die diskreten Logarithmen $\log_{g_3} B$ und $\log_{g_4} C$ mit dem g_2-Anteil der Darstellung von A übereinstimmen. Um $\mathcal{V}$ von dieser Behauptung zu überzeugen, wird zwischen $\mathcal{P}$ und $\mathcal{V}$ das folgende Protokoll ausgeführt (vgl. auch Abbildung 2.2):

Schritt 1: Der Prover $\mathcal{P}$ wählt zufällig Werte $x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$, berechnet $A' = g_1^{x_1} g_2^{x_2}$, $B' = g_3^{x_2}, C' = g_4^{x_2}$ und sendet A', B', C' an $\mathcal{V}$.

Schritt 2: Der Verifier $\mathcal{V}$ wählt zufällig einen Wert $c \in_{\mathcal{R}} \mathbb{Z}_q$ und sendet c an $\mathcal{P}$.

Schritt 3: Der Prover $\mathcal{P}$ berechnet $r_1 = c\alpha + x_1 \bmod q$ sowie $r_2 = c\beta + x_2 \bmod q$ und sendet r_1, r_2 an $\mathcal{V}$.

Schritt 4: Der Verifier $\mathcal{V}$ überprüft, ob die folgenden Gleichungen erfüllt sind:

$$\begin{aligned} g_1^{r_1} g_2^{r_2} &= A^c A' \\ g_3^{r_2} &= B^c B' \\ g_4^{r_2} &= C^c C'. \end{aligned}$$

Bei einem solchen interaktiven Beweis spricht man auch von einem *3-Runden Beweis*. Die von $\mathcal{P}$ in Schritt 1 gewählten Werte A', B', C' werden *Commitment* genannt, der von $\mathcal{V}$ im zweiten Schritt gewählte Wert c *Challenge* und die von $\mathcal{P}$ in Schritt 3 berechneten Werte r_1, r_2 *Response*. Den gesamten Protokollablauf bezeichnet man mit (commit, challenge, response).

$\mathcal{P}(\alpha, \beta, g_1, \ldots, g_4, A, B, C)$		$\mathcal{V}(g_1, \ldots, g_4, A, B, C)$
$\mathcal{P}$ zeigt: $A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta$		
$x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$ $A' := g_1^{x_1} g_2^{x_2}$ $B' := g_3^{x_2}$ $C' := g_4^{x_2}$	$\xrightarrow{A', B', C'}$	$c \in_{\mathcal{R}} \mathbb{Z}_q$
	$\xleftarrow{c}$	
$r_1 := c\alpha + x_1 \bmod q$ $r_2 := c\beta + x_2 \bmod q$	$\xrightarrow{r_1, r_2}$	$g_1^{r_1} g_2^{r_2} \stackrel{?}{=} A^c A'$ $g_3^{r_2} \stackrel{?}{=} B^c B'$ $g_4^{r_2} \stackrel{?}{=} C^c C'$

Abbildung 2.2: HVZK $\left[(\alpha, \beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$

Bemerkung 2.7.1. Für einen öffentlichen Schlüssel $\mathsf{pk} = (p, q, g, h, z) = (p, q, g_3, g_4, g_2)$ des modifizierten ElGamal-Verschlüsselungsverfahrens (vgl. Abschnitt 2.3.3) kann $\mathcal{P}$ mit diesem interaktiven Beweis insbesondere nachweisen, dass es sich bei dem Tupel (A, B, C) um eine ihm bekannte, modifizierte ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel pk handelt. Für die in den Abschnitten 6.2 und 6.3 entwickelten Geldsysteme wird der Beweis vom Kunden dazu verwendet, die Bank von der korrekten Verschlüsselung eines Münz-Identifikators zu überzeugen, der in einem fairen elektronischen Geldsystem dazu benötigt wird ggf. den Ausgabeort einer elektronischen Münze zu bestimmen (vgl. Abschnitt 1.1 bzw. 3.3).

Satz 2.7.1. Der interaktive Beweis HVZK $\left[(\alpha, \beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$ ist ein Honest-Verifier-Zero-Knowledge-Beweis.

Beweis. Wir zeigen, dass das Protokoll gemäß Definition 2.7.1 ein interaktiver Beweis zwischen $\mathcal{P}$ und $\mathcal{V}$ ist und die Honest-Verifier-Eigenschaft aus Definition 2.7.3 erfüllt.

Durchführbarkeit: Verhalten sich $\mathcal{P}$ und $\mathcal{V}$ gemäß dem Protokoll und kennt $\mathcal{P}$ die korrekten Darstellungen von A, B und C, so kann $\mathcal{P}$ auf jede Challenge $c \in \mathbb{Z}_q$ von $\mathcal{V}$ korrekt antworten:

$$\begin{aligned} g_1^{r_1} g_2^{r_2} &= g_1^{c\alpha + x_1} g_2^{c\beta + x_2} = (g_1^\alpha)^c g_1^{x_1} (g_2^\beta)^c g_2^{x_2} = A^c A' \\ g_3^{r_2} &= g_3^{c\beta + x_2} = (g_3^\beta)^c g_3^{x_2} = B^c B' \\ g_4^{r_2} &= g_4^{c\beta + x_2} = (g_4^\beta)^c g_4^{x_2} = C^c C'. \end{aligned}$$

Korrektheit: Es genügt zu zeigen, dass der Prover $\mathcal{P}$ die erforderlichen Darstellungen

$$A = g_1^\alpha g_2^\beta,\ B = g_3^\beta \text{ und } C = g_4^\beta$$

kennt, falls $\mathcal{P}$ auf zwei verschiedene Challenges des Verifiers $\mathcal{V}$ antworten kann. Denn unter der Diskreter-Logarithmus-Annahme kann $\mathcal{P}$, der den diskreten Logarithmus $\log_{g_2} g_1$ nicht kennt, keine weiteren Darstellungen von A bzgl. (g_1, g_2) und B, C bzgl. g_3, g_4 kennen (vgl. Abschnitt 2.2.3). Für $\mathcal{P}$ gibt es daher genau zwei Möglichkeiten: Entweder $\mathcal{P}$ kann auf genau eine Challenge von $\mathcal{V}$ antworten oder $\mathcal{P}$ kennt die erforderlichen Darstellungen von A, B und C.
Kennt $\mathcal{P}$ die erforderlichen Darstellungen nicht, so kann $\mathcal{P}$ versuchen die Challenge c von $\mathcal{V}$ zu erraten und mit zufällig gewählten Werten $r_1, r_2 \in_{\mathcal{R}} \mathbb{Z}_q$ die Commitments $A' = g_1^{r_1} g_2^{r_2} A^{-c}$, $B' = g_3^{r_2} B^{-c}$ und $C' = g_4^{r_2} C^{-c}$ in Abhängigkeit von c zu berechnen. Da die Challenge c von $\mathcal{V}$ aber zufällig gewählt wird, hat $\mathcal{P}$, falls er die erforderlichen Darstellungen nicht kennt, in jedem Fall nur eine vernachlässigbare Erfolgswahrscheinlichkeit von $\frac{1}{q}$.
Kann $\mathcal{P}$ auf zwei verschiedene Challenges $c, c' \in \mathbb{Z}_q$ korrekt antworten, so existieren die beiden Tupel (c, r_1, r_2) und (c', r_1', r_2') und es gilt:

$$A' = g_1^{r_1} g_2^{r_2} A^{-c} = g_1^{r_1'} g_2^{r_2'} A^{-c'}$$
$$B' = g_3^{r_2} B^{-c} = g_3^{r_2'} B^{-c'}$$
$$C' = g_4^{r_2} C^{-c} = g_4^{r_2'} C^{-c'}.$$

Mit diesen Gleichungen ergibt sich:

$$A = g_1^{\frac{r_1 - r_1'}{c - c'}} g_2^{\frac{r_2 - r_2'}{c - c'}}$$
$$B = g_3^{\frac{r_2 - r_2'}{c - c'}}$$
$$C = g_4^{\frac{r_2 - r_2'}{c - c'}}.$$

Das heißt, der Prover $\mathcal{P}$ kennt die erforderlichen Darstellungen, falls $\mathcal{P}$ auf zwei verschiedene Challenges $c, c' \in \mathbb{Z}_q$ antworten kann.

Honest-Verifier-Zero-Knowledge: Unter der Voraussetzung, dass sich der Verifizierer $\mathcal{V}$ die Challenge c gemäß dem Protokoll wählt, kann der Beweis von einem Simulator $\mathcal{S}$ effizient simuliert werden. Der Simulator $\mathcal{S}$ geht wie folgt vor:

Schritt 1: Der Simulator $\mathcal{S}$ wählt $c, r_1, r_2 \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: Der Simulator $\mathcal{S}$ berechnet Commitments $A' = g_1^{r_1} g_2^{r_2} A^{-c}, B' = g_3^{r_2} B^{-c}$ und $C' = g_4^{r_2} C^{-c}$.

Bleibt zu zeigen, dass die Ausgabe von $\mathcal{S}$ und die Protokollansicht $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ von $\mathcal{V}$ identisch verteilt sind. Aus Schritt 1 der Simulation folgt, dass $c, r_1, r_2 \in \mathbb{Z}_q$ unabhängig und gleichverteilt sind. Es ist $x_1 = r_1 - c\alpha \bmod q$ und $x_2 = r_2 - c\beta \bmod q$. Auf Grund der Unabhängigkeit und der Gleichverteilung von c, r_1 und r_2 sind somit auch x_1, x_2 und c unabhängig und gleichverteilt. Woraus die identische Verteilung der Ausgabe von $\mathcal{S}$ und $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ folgt.

□

2.7.1.2 Der interaktive Beweis

$$\text{HVZK}\left[(\alpha,\beta,\gamma) : A = g_1^{\alpha} g_2^{\beta} \wedge B = g_1^{\alpha} g_3^{\gamma} \wedge C = g_4^{\gamma} \wedge D = g_5^{\gamma}\right]$$

Ähnlich zu Abschnitt 2.7.1.1 kennen Prover $\mathcal{P}$ und Verifier $\mathcal{V}$ die Werte $A, B, C, D \in G_q$. Mit dem interaktiven Beweis

$$\text{HVZK}\left[(\alpha,\beta,\gamma) : A = g_1^{\alpha} g_2^{\beta} \wedge B = g_1^{\alpha} g_3^{\gamma} \wedge C = g_4^{\gamma} \wedge D = g_5^{\gamma}\right]$$

möchte $\mathcal{P}$ den Verifier $\mathcal{V}$ davon überzeugen, dass $\mathcal{P}$ Werte α, β, γ kennt, so dass (α, β) eine Darstellung von A bzgl. (g_1, g_2), (α, γ) eine Darstellung von B bzgl. (g_1, g_3) sowie γ eine Darstellung von C und D bzgl. g_4 bzw. g_5 ist. Zusätzlich möchte $\mathcal{P}$ den Verifier $\mathcal{V}$ mit diesem interaktiven Beweis davon überzeugen, dass zum einen die g_1-Anteile der Darstellungen von A und B übereinstimmen und zum anderen auch der g_3-Anteil der Darstellung von B mit den beiden diskreten Logarithmen $\log_{g_4} C$ und $\log_{g_5} D$ übereinstimmt. Dazu wird zwischen $\mathcal{P}$ und $\mathcal{V}$ das folgende Protokoll durchgeführt (vgl. auch Abbildung 2.3):

Schritt 1: Der Prover $\mathcal{P}$ wählt zufällig die Werte $x_1, x_2, x_3 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die Commitments $A' = g_1^{x_1} g_2^{x_2}, B' = g_1^{x_1} g_3^{x_3}, C' = g_4^{x_3}, D' = g_5^{x_3}$, die $\mathcal{P}$ anschließend an $\mathcal{V}$ sendet.

Schritt 2: Der Verifier $\mathcal{V}$ wählt zufällig eine Challenge $c \in_{\mathcal{R}} \mathbb{Z}_q$ und sendet c an $\mathcal{P}$.

Schritt 3: Der Prover $\mathcal{P}$ berechnet die Responses $r_1 = c\alpha + x_1 \bmod q, r_2 = c\beta + x_2 \bmod q$, $r_3 = c\gamma + x_3 \bmod q$ und sendet r_1, r_2, r_3 an $\mathcal{V}$.

Schritt 4: Der Verifier $\mathcal{V}$ überprüft, ob die folgenden Gleichungen erfüllt sind:

$$\begin{aligned} g_1^{r_1} g_2^{r_2} &= A^c A' \\ g_1^{r_1} g_3^{r_3} &= B^c B' \\ g_4^{r_3} &= C^c C' \\ g_5^{r_3} &= D^c D'. \end{aligned}$$

Bemerkung 2.7.2. Analog zu Abschnitt 2.7.1.1 kann $\mathcal{P}$ mit diesem interaktiven Beweis für einen öffentlichen Schlüssel $\mathsf{pk} = (p, q, g, h, z) = (p, q, g_4, g_5, g_3)$ des modifizierten ElGamal-Verschlüsselungsverfahrens (vgl. Abschnitt 2.3.3) nachweisen, dass es sich bei dem Tupel (B, C, D) um eine modifizierte ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel pk handelt. Zusätzlich beweist $\mathcal{P}$ aber auch, dass (B, C, D) eine Verschlüsselung des g_1-Anteil von A ist. In den Abschnitten 6.2 und 6.3 wird dieser Beweis sicherstellen, dass ein Kunde beim Bezahlen seine Identität verschlüsselt, damit später Kundentracing durchgeführt werden kann.

$\mathcal{P}(\alpha, \beta, \gamma, g_1, \ldots, g_5, A, B, C, D)$		$\mathcal{V}(g_1, \ldots, g_5, A, B, C, D)$
$\mathcal{P}$ zeigt: $A = g_1^\alpha g_2^\beta \wedge B = g_1^\alpha g_3^\gamma \wedge C = g_4^\gamma \wedge D = g_5^\gamma$		
$x_1, x_2, x_3 \in_\mathcal{R} \mathbb{Z}_q$ $A' = g_1^{x_1} g_2^{x_2}$ $B' = g_1^{x_1} g_3^{x_3}$ $C' = g_4^{x_3}$ $D' = g_5^{x_3}$	$\xrightarrow{A', B', C', D'}$	$c \in_\mathcal{R} \mathbb{Z}_q$
	$\xleftarrow{c}$	
$r_1 = c\alpha + x_1 \bmod q$ $r_2 = c\beta + x_2 \bmod q$ $r_3 = c\gamma + x_3 \bmod q$	$\xrightarrow{r_1, r_2, r_3}$	$g_1^{r_1} g_2^{r_2} \stackrel{?}{=} A^c A'$ $g_1^{r_1} g_3^{r_3} \stackrel{?}{=} B^c B'$ $g_4^{r_3} \stackrel{?}{=} C^c C'$ $g_5^{r_3} \stackrel{?}{=} D^c D'$

Abbildung 2.3: HVZK $\left[(\alpha, \beta, \gamma) : A = g_1^\alpha g_2^\beta \wedge B = g_1^\alpha g_3^\gamma \wedge C = g_4^\gamma \wedge D = g_5^\gamma\right]$

Satz 2.7.2. Das in Abbildung 2.3 dargestellte Protokoll ist ein Honest-Verifier-Zero-Knowledge-Beweis.

Beweis. Wir zeigen, dass das Protokoll gemäß Definition 2.7.1 ein interaktiver Beweis zwischen $\mathcal{P}$ und $\mathcal{V}$ ist und die Honest-Verifier-Eigenschaft aus Definition 2.7.3 erfüllt.

Durchführbarkeit: Verhalten sich $\mathcal{P}$ und $\mathcal{V}$ gemäß dem Protokoll und kennt $\mathcal{P}$ die korrekten Darstellungen von A, B, C und D, so kann $\mathcal{P}$ auf jede Challenge $c \in \mathbb{Z}_q$ von $\mathcal{V}$ korrekt antworten:

$$\begin{aligned}
g_1^{r_1} g_2^{r_2} &= g_1^{c\alpha + x_1} g_2^{c\beta + x_2} = (g_1^\alpha)^c g_1^{x_1} (g_2^\beta)^c g_2^{x_2} = A^c A' \\
g_1^{r_1} g_3^{r_3} &= g_1^{c\alpha + x_1} g_3^{c\gamma + x_3} = (g_1^\alpha)^c g_1^{x_1} (g_3)^\gamma g_3^{x_3} = B^c B' \\
g_4^{r_3} &= (g_4^\gamma)^c g_4^{x_3} = C^c C' \\
g_5^{r_3} &= (g_5^\gamma)^c g_5^{x_3} = D^c D'.
\end{aligned}$$

Korrektheit: Analog zu Abschnitt 2.7.1.1 genügt es auch hier zu zeigen, dass der Prover $\mathcal{P}$ die erforderlichen Darstellungen

$$A = g_1^\alpha g_2^\beta,\ B = g_1^\alpha g_3^\gamma,\ C = g_4^\gamma \text{ und } D = g_5^\gamma$$

kennt, falls $\mathcal{P}$ auf zwei verschiedene Challenges des Verifiers $\mathcal{V}$ antworten kann. Denn unter der Diskreter-Logarithmus-Annahme kann $\mathcal{P}$, der die diskreten Logarithmen $\log_{g_2} g_1$ und $\log_{g_3} g_1$ nicht kennt, keine weiteren Darstellungen kennen. Das heißt, der Prover $\mathcal{P}$

kann entweder auf genau eine Challenge von $\mathcal{V}$ antworten oder $\mathcal{P}$ kennt die erforderlichen Darstellungen von A, B, C und D.
Kennt $\mathcal{P}$ die erforderlichen Darstellungen nicht, so kann $\mathcal{P}$ versuchen die Challenge c von $\mathcal{V}$ zu erraten und mit zufällig gewählten Werten $r_1, r_2, r_3 \in_{\mathcal{R}} \mathbb{Z}_q$ die Commitments $A' = g_1^{r_1} g_2^{r_2} A^{-c}$, $B' = g_1^{r_1} g_3^{r_3} B^{-c}$, $C' = g_4^{r_3} C^{-c}$ und $D' = g_5^{r_3} D^{-c}$ in Abhängigkeit von c zu berechnen. In jedem Fall hat $\mathcal{P}$ nur eine vernachlässigbare Erfolgswahrscheinlichkeit von $\frac{1}{q}$.
Kann $\mathcal{P}$ auf zwei verschiedene Challenges $c, c' \in \mathbb{Z}_q$ korrekt antworten, so existieren die beiden Tupel (c, r_1, r_2, r_3) und (c', r_1', r_2', r_3') und es gilt:

$$\begin{aligned} A' &= g_1^{r_1} g_2^{r_2} A^{-c} = g_1^{r_1'} g_2^{r_2'} A^{-c'} \\ B' &= g_1^{r_1} g_3^{r_3} B^{-c} = g_1^{r_1'} g_3^{r_3'} B^{-c'} \\ C' &= g_4^{r_3} C^{-c} = g_4^{r_3'} C^{-c'} \\ D' &= g_5^{r_3} D^{-c} = g_5^{r_3'} D^{-c'}. \end{aligned}$$

Mit diesen Gleichungen ergibt sich:

$$\begin{aligned} A &= g_1^{\frac{r_1 - r_1'}{c - c'}} g_2^{\frac{r_2 - r_2'}{c - c'}} \\ B &= g_1^{\frac{r_1 - r_1'}{c - c'}} g_3^{\frac{r_3 - r_3'}{c - c'}} \\ C &= g_4^{\frac{r_3 - r_3'}{c - c'}} \\ D &= g_5^{\frac{r_3 - r_3'}{c - c'}}. \end{aligned}$$

Das heißt, der Prover $\mathcal{P}$ kennt die erforderlichen Darstellungen, falls $\mathcal{P}$ auf zwei verschiedene Challenges $c, c' \in \mathbb{Z}_q$ antworten kann.

Honest-Verifier-Zero-Knowledge: Unter der Voraussetzung, dass sich der Verifizierer $\mathcal{V}$ die Challenge c gemäß dem Protokoll wählt, kann der Beweis von einem Simulator $\mathcal{S}$ effizient simuliert werden. Der Simulator $\mathcal{S}$ geht wie folgt vor:

Schritt 1: Der Simulator $\mathcal{S}$ wählt $c, r_1, r_2, r_3 \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: Der Simulator $\mathcal{S}$ berechnet die folgenden Commitments: $A' = g_1^{r_1} g_2^{r_2} A^{-c}$, $B' = g_1^{r_1} g_3^{r_3} B^{-c}$, $C' = g_4^{r_3} C^{-c}$ und $D' = g_5^{r_3} D^{-c}$.

Bleibt zu zeigen, dass die Ausgabe von $\mathcal{S}$ und die Protokollansicht $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ von $\mathcal{V}$ identisch verteilt sind. Aus Schritt 1 der Simulation folgt, dass $c, r_1, r_2, r_3 \in \mathbb{Z}_q$ unabhängig und gleichverteilt sind. Es ist $x_1 = r_1 - c\alpha \bmod q$, $x_2 = r_2 - c\beta \bmod q$ und $x_3 = r_3 - c\gamma \bmod q$. Auf Grund der Unabhängigkeit und der Gleichverteilung von c, r_1 und r_2 sind somit auch x_1, x_2, x_3 und c unabhängig und gleichverteilt. Woraus die identische Verteilung der Ausgabe von $\mathcal{S}$ und $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ folgt. □

2.7.1.3 Der interaktive Beweis

$$\text{HVZK}\left[(\alpha, \beta_1, \ldots, \beta_6) : A = a^{\alpha} g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^{\alpha} g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^{\alpha} g_3^{\beta_5} g_4^{\beta_6}\right]$$

In diesem interaktiven Beweis kennen Prover $\mathcal{P}$ und Verifier $\mathcal{V}$ Werte $A, B, C \in G_q$. Der Prover $\mathcal{P}$ möchte den Verifier $\mathcal{V}$ zum einen davon überzeugen, dass $\mathcal{P}$ Werte $\alpha, \beta_1, \ldots, \beta_6$ kennt, so dass $(\alpha, \beta_1, \beta_2)$ eine Darstellungen von A bzgl. (a, g_1, g_2), $(\alpha, \beta_3, \beta_4)$ eine Darstellung von B bzgl. (b, g_2, g_3) und $(\alpha, \beta_5, \beta_6)$ eine Darstellung von C bzgl. (c, g_3, g_4) ist. Zum anderen beweist $\mathcal{P}$ zusätzlich, dass der g_1-Anteil der Darstellung von A mit dem g_2-Anteil der Darstellung von B übereinstimmt und dieser auch mit dem g_3-Anteil der Darstellung von C identisch ist. Dazu wird zwischen $\mathcal{P}$ und $\mathcal{V}$ das folgende Protokoll durchgeführt (vgl. auch Abbildung 2.4):

Schritt 1: Der Prover $\mathcal{P}$ wählt zufällig die Werte $x, x_1, \ldots, x_7 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die Commitments $A' = a^x g_1^{x_1} g_2^{x_2}, B' = b^x g_2^{x_3} g_3^{x_4}, C' = c^x g_3^{x_5} g_4^{x_6}$, die $\mathcal{P}$ anschließend an $\mathcal{V}$ sendet.

Schritt 2: Der Verifier $\mathcal{V}$ wählt zufällig eine Challenge $c \in_{\mathcal{R}} \mathbb{Z}_q$ und sendet d an $\mathcal{P}$.

Schritt 3: Der Prover $\mathcal{P}$ berechnet Responses $r = d\alpha + x \bmod q$, $r_1 = d\beta_1 + x_1 \bmod q$, $\hat{r}_1 = d\beta_2 + x_2 \bmod q$, $r_2 = d\beta_3 + x_3 \bmod q$, $\hat{r}_2 = d\beta_4 + x_4 \bmod q$, $r_3 = d\beta_5 + x_5 \bmod q$, $\hat{r}_3 = d\beta_6 + x_6 \bmod q$ und sendet $r, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3, \hat{r}_3$ an $\mathcal{V}$.

Schritt 4: Der Verifier $\mathcal{V}$ überprüft, ob die folgenden Gleichungen erfüllt sind:

$$\begin{aligned} a^r g_1^{r_1} g_2^{\hat{r}_1} &\stackrel{?}{=} A^d A' \\ b^r g_2^{r_2} g_3^{\hat{r}_2} &\stackrel{?}{=} B^d B' \\ c^r g_3^{r_3} g_4^{\hat{r}_3} &\stackrel{?}{=} C^d C'. \end{aligned}$$

Bemerkung 2.7.3. In den Abschnitten 6.2 und 6.3 wird der Kunde mit diesem Beweis die Bank davon überzeugen, dass er den korrekten Münz-Identifikator verschlüsselt hat. Anhand des verschlüsselten Münz-Identifikators kann die Bank im Bedarfsfall mit Hilfe des Deanonymisierers ggf. den Ausgabeort der entsprechenden elektronischen Münze ausfindig machen (vgl. Abschnitt 1.1 bzw. 3.3).

Satz 2.7.3. Das in Abbildung 2.4 dargestellte Protokoll ist ein Honest-Verifier-Zero-Knowledge-Beweis.

Beweis. Wir zeigen, dass das Protokoll gemäß Definition 2.7.1 ein interaktiver Beweis zwischen $\mathcal{P}$ und $\mathcal{V}$ ist und die Honest-Verifier-Eigenschaft aus Definition 2.7.3 erfüllt.

$\mathcal{P}(\alpha, \beta_1, \ldots, \beta_6, \ldots, A, B, C)$		$\mathcal{V}(a, b, c, \ldots, A, B, C)$
$\mathcal{P}$ zeigt: $A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}$		
$x, x_1, \ldots, x_7 \in_{\mathcal{R}} \mathbb{Z}_q$ $A' = a^x g_1^{x_1} g_2^{x_2}$ $B' = b^x g_2^{x_3} g_3^{x_4}$ $C' = c^x g_3^{x_5} g_4^{x_6}$	$\xrightarrow{A', B', C'}$	
		$d \in_{\mathcal{R}} \mathbb{Z}_q$
	$\xleftarrow{d}$	
$r = d\alpha + x \bmod q$ $r_1 = d\beta_1 + x_1 \bmod q$ $\hat{r}_1 = d\beta_2 + x_2 \bmod q$ $r_2 = d\beta_3 + x_3 \bmod q$ $\hat{r}_2 = d\beta_4 + x_4 \bmod q$ $r_3 = d\beta_5 + x_5 \bmod q$ $\hat{r}_3 = d\beta_6 + x_6 \bmod q$	$\xrightarrow{r, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3, \hat{r}_3}$	$a^r g_1^{r_1} g_2^{\hat{r}_1} \stackrel{?}{=} A^d A'$ $b^r g_2^{r_2} g_3^{\hat{r}_2} \stackrel{?}{=} B^d B'$ $c^r g_3^{r_3} g_4^{\hat{r}_3} \stackrel{?}{=} C^d C'$

Abbildung 2.4: HVZK $\left[(\alpha, \beta_1, \ldots, \beta_6) : A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}\right]$

Durchführbarkeit: Verhalten sich $\mathcal{P}$ und $\mathcal{V}$ gemäß dem Protokoll und kennt $\mathcal{P}$ die korrekten Darstellungen von A, B und C, so kann $\mathcal{P}$ auf jede Challenge $d \in \mathbb{Z}_q$ von $\mathcal{V}$ korrekt antworten:

$$\begin{aligned}
a^r g_1^{r_1} g_2^{\hat{r}_1} &= a^{d\alpha+x} g_1^{d\beta_1+x_1} g_2^{d\beta_2+x_2} = (a^\alpha)^d a^x (g_1^{\beta_1})^d g_1^{x_1} (g_2^{\beta_2})^d g_2^{x_2} = A^d A' \\
b^r g_2^{r_2} g_3^{\hat{r}_2} &= b^{d\alpha+x} g_2^{d\beta_3+x_3} g_3^{d\beta_4+x_4} = (b^\alpha)^d b^x (g_2^{\beta_3})^d g_2^{x_3} (g_3^{\beta_4})^d g_3^{x_4} = B^d B' \\
c^r g_2^{r_3} g_3^{\hat{r}_3} &= c^{d\alpha+x} g_3^{d\beta_5+x_5} g_3^{d\beta_6+x_6} = (c^\alpha)^d b^x (g_3^{\beta_5})^d g_3^{x_5} (g_4^{\beta_6})^d g_4^{x_6} = C^d C'.
\end{aligned}$$

Korrektheit: Auch hier genügt es zu zeigen, dass der Prover $\mathcal{P}$ die erforderlichen Darstellungen

$$A = a^\alpha g_1^{\beta_1} g_2^{\beta_2},\ B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \text{ und } C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}$$

kennt, falls $\mathcal{P}$ auf zwei verschiedene Challenges des Verifiers $\mathcal{V}$ antworten kann. Denn unter der Diskreter-Logarithmus-Annahme kann $\mathcal{P}$, der die diskreten Logarithmen $\log_{g_2} g_1$, $\log_{g_3} g_2$ und $\log_{g_3} g_4$ nicht kennt, keine weiteren Darstellungen kennen. Das heißt, der Prover $\mathcal{P}$ kann entweder auf genau eine Challenge von $\mathcal{V}$ antworten oder $\mathcal{P}$ kennt die erforderlichen Darstellungen von A, B und C.
Kennt $\mathcal{P}$ die erforderlichen Darstellungen nicht, so kann $\mathcal{P}$ versuchen die Challenge d von $\mathcal{V}$ zu erraten und mit zufällig gewählten Werten $r, r_1, \hat{r}_1, \ldots, r_3, \hat{r}_3$ die Commitments A', B' und C' in Abhängigkeit von d zu berechnen. In jedem Fall hat $\mathcal{P}$ nur eine vernachlässigbare Erfolgswahrscheinlichkeit von $\frac{1}{q}$.

Kann $\mathcal{P}$ auf zwei verschiedene Challenges $d, d' \in \mathbb{Z}_q$ antworten, so existieren die beiden Tupel $(d, r, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3, \hat{r}_3)$ und $(d', r', r'_1, \hat{r}'_1, r'_2, \hat{r}'_2, r'_3, \hat{r}'_3)$ und es gilt:

$$\begin{aligned} A' &= a^r g_1^{r_1} g_2^{\hat{r}_1} A^{-d} = a^{r'} g_1^{r'_1} g_2^{\hat{r}'_1} A^{-d'} \\ B' &= b^r g_2^{r_2} g_3^{\hat{r}_2} B^{-d} = b^{r'} g_2^{r'_2} g_3^{\hat{r}'_2} B^{-d'} \\ C' &= c^r g_3^{r_3} g_4^{\hat{r}_3} C^{-d} = c^{r'} g_3^{r'_3} g_4^{\hat{r}'_3} C^{-d'}. \end{aligned}$$

Mit diesen Gleichungen ergibt sich:

$$\begin{aligned} A &= a^{\frac{r-r'}{d-d'}} g_1^{\frac{r_1-r'_1}{d-d'}} g_2^{\frac{\hat{r}_1-\hat{r}'_1}{d-d'}} \\ B &= b^{\frac{r-r'}{d-d'}} g_2^{\frac{r_2-r'_2}{d-d'}} g_3^{\frac{\hat{r}_2-\hat{r}'_2}{d-d'}} \\ C &= c^{\frac{r-r'}{d-d'}} g_3^{\frac{r_3-r'_3}{d-d'}} g_4^{\frac{\hat{r}_3-\hat{r}'_3}{d-d'}}. \end{aligned}$$

Das heißt, der Prover $\mathcal{P}$ kennt die erforderlichen Darstellungen, falls $\mathcal{P}$ auf zwei verschiedene Challenges $d, d' \in \mathbb{Z}_q$ antworten kann.

Honest-Verifier-Zero-Knowledge: Unter der Voraussetzung, dass sich der Verifizierer $\mathcal{V}$ die Challenge d gemäß dem Protokoll wählt, kann der Beweis von einem Simulator $\mathcal{S}$ effizient simuliert werden. Der Simulator $\mathcal{S}$ geht wie folgt vor:

Schritt 1: $\mathcal{S}$ wählt $d, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3, \hat{r}_3 \in_{\mathcal{R}} \mathbb{Z}_q$.

Schritt 2: $\mathcal{S}$ berechnet $A' = a^r g_1^{r_1} g_2^{\hat{r}_1} A^{-d}$, $B' = b^r g_2^{r_2} g_3^{\hat{r}_2} B^{-d}$, $C' = c^r g_3^{r_3} g_4^{\hat{r}_3} C^{-d}$.

Bleibt zu zeigen, dass die Ausgabe von $\mathcal{S}$ und die Protokollansicht $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ von $\mathcal{V}$ identisch verteilt sind. Aus Schritt 1 der Simulation folgt, dass $d, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3, \hat{r}_3 \in \mathbb{Z}_q$ unabhängig und gleichverteilt sind. Es ist $x = r - d\alpha \bmod q$ und

$$x_1 = r_1 - d\beta_1 \bmod q,\ x_2 = \hat{r}_1 - d\beta_2 \bmod q,\ x_3 = r_2 - d\beta_3 \bmod q$$

sowie

$$x_4 = \hat{r}_2 - d\beta_4,\ x_5 = r_3 - d\beta_5 \bmod q,\ x_6 = \hat{r}_3 - d\beta_6 \bmod q.$$

Auf Grund der Unabhängigkeit und der Gleichverteilung von $d, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3$ und $\hat{r}_3$ sind somit auch $x, x_1, \ldots, x_6$ und d unabhängig und gleichverteilt. Woraus die identische Verteilung der Ausgabe von $\mathcal{S}$ und $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}$ folgt. □

2.8 Die Fiat-Shamir-Heuristik

Im Jahre 1986 wurde von A. Fiat und A. Shamir [FS87] ein allgemeines Verfahren vorgeschlagen, mit dem jeder Honest-Verifier-Zero-Knowledge-Beweis im Random-Oracle-Modell in ein digitales Signaturverfahren transformiert werden kann. Bezeichnet $(\mathsf{commit}, \mathsf{challenge}, \mathsf{response})$ den Protokollablauf eines interaktiven 3-Runden Honest-

Verifier-Zero-Knowledge-Beweises zwischen einem Prover $\mathcal{P}$ und einem Verifier $\mathcal{V}$ (vgl. Abschnitt 2.7.1.1), so kann das Protokoll in ein *nichtinteraktives* Protokoll transformiert werden, indem die von $\mathcal{V}$ zufällig gewählte Challenge $\mathsf{challenge}$ durch eine geeignete Hashfunktion H erzeugt wird:

$$\mathsf{challenge} = \mathrm{H}(\mathsf{commit}).$$

Diese Vorgehensweise, bei der die zufällige Wahl der Challenge durch eine Hashfunktion simuliert wird, bezeichnet man auch als *Fiat-Shamir-Heuristik*. Der Fiat-Shamir-Heuristik liegt der Gedanke zugrunde, dass eine Hashfunktion idealerweise von einer echten Zufallsfunktion ununterscheidbar ist.
Ist $m \in \{0,1\}^*$ eine Nachricht und ersetzt man die Challenge $\mathsf{challenge}$ durch den Wert $\mathrm{H}(m, \mathsf{commit})$, so kann jeder Honest-Verifier-Zero-Knowledge-Beweis durch Anwenden der Fiat-Shamir-Heuristik im Random-Oracle-Modell in ein digitales Signaturverfahren transformiert werden. Eines der bekanntesten Beispiele ist die Schnorr-Signatur (vgl. Abschnitt 2.5.1), die aus dem Schnorr-Identifikationsprotokoll [Sch91] hervorgeht.

2.9 Das Random-Oracle-Modell

Das *Random-Oracle-Modell* wurde im Jahre 1993 von M. Bellare und P. Rogaway [BR93] vorgeschlagen und stellt eine geeignete Methode zur Verfügung, die Sicherheit kryptografischer Bausteine zu analysieren, in denen Hashfunktionen zum Einsatz kommen. Insbesondere wird auch die Sicherheit der in den Kapiteln 6 und 7 entwickelten Verfahren im Random-Oracle-Modell analysiert. Das Modell formalisiert den der Fiat-Shamir-Heuristik zugrunde liegenden Gedanken einer *idealen* Hashfunktion, dem so genannten *Random-Oracle*.

2.9.1 Nichtinteraktive Zero-Knowledge-Beweise

Die Fiat-Shamir-Heuristik (vgl. Abschnitt 2.8) bietet den Vorteil, jeden 3-Runden Honest-Verifier-Zero-Knowledge-Beweis in ein nichtinteraktives Protokoll zu transformieren. Durch das Anwenden der Fiat-Shamir-Heuristik geht jedoch die Zero-Knowledge-Eigenschaft verloren (siehe z.B. [Mao04]).
Allerdings kann man zeigen, dass es sich bei den so transformierten Protokollen um *nichtinteraktive Zero-Knowledge-Beweise* im Random-Oracle-Modell handelt. Die zur Transformation verwendete Hashfunktion H wird im Random-Oracle-Modell durch ein Random-Oracle $O : \{0,1\}^* \to \{0,1\}^\infty$ ersetzt, dessen Ausgabe effizient berechenbar, deterministisch und gleichverteilt ist. Sowohl der Prover $\mathcal{P}$ als auch der Verifier $\mathcal{V}$ haben zur Erstellung bzw. Verifikation des nichtinteraktiven Protokolls π Zugriff auf das Random-Oracle O. Analog zu den interaktiven Beweissystemen, spricht man von einem *nichtinteraktiven Beweissytem* im Random-Oracle-Modell, falls das Paar $(\mathcal{P}, \mathcal{V})$ die Eigenschaften Durchführbarkeit und Korrektheit erfüllt (vgl. Definition 2.9.1).
Hat der nichtinteraktive Beweis π, der vom Prover $\mathcal{P}$ ausgegeben wird, zusätzlich die Eigenschaft, dass sich der Beweis π und das Random-Oracle O von einem Simulator

$\mathcal{S}$, der die Behauptung selbst nicht beweisen kann, effizient simulieren lassen und die Simulation (O', π') nicht von (O, π) zu unterscheiden ist, spricht man von einem *nichtinteraktiven Zero-Knowledge-Beweis* (NIZK) im Random-Oracle-Modell. Formal ist das nichtinteraktive Beweissystem im Random-Oracle-Modell wie folgt definiert:

Definition 2.9.1 (**Nichtinteraktives Beweissystem**)**.** Gegeben sei ein Alphabet Σ, eine Sprache L über Σ^* sowie ein probabilistischer, polynomieller Algorithmus $\mathcal{P}$ und ein deterministischer polynomieller Algorithmus $\mathcal{V}$. Das Paar $(\mathcal{P}, \mathcal{V})$ heißt *nichtinteraktives Beweissystem* im Random-Oracle-Modell für die Behauptung $x \in L$, falls die beiden folgenden Eigenschaften erfüllt sind:

1. *Durchführbarkeit* (Completeness): Es gibt ein $k_0 \in \mathbb{N}$, so dass für alle $x \in L$ mit $|x| = k > k_0$ gilt

$$\mathbf{P}\left[\mathcal{V}\left(O, x, \pi\right) = \mathsf{true} \mid \pi = \mathcal{P}\left(O, x\right)\right] \geq 1 - \nu\left(k\right)$$

   mit einer in k vernachlässigbaren Funktion ν.

2. *Korrektheit* (Soundness): Es gibt ein $k_0 \in \mathbb{N}$, so dass für alle $x \notin L$ mit $|x| = k > k_0$ gilt

$$\mathbf{P}\left[\mathcal{V}\left(O, x, \pi\right) = \mathsf{false} \mid \left(x, \pi\right) = \mathcal{P}^*\left(O\right)\right] \geq 1 - \nu\left(k\right)$$

   mit einer in k vernachlässigbaren Funktion ν.

Das nichtinteraktive Beweissystem $(\mathcal{P}, \mathcal{V})$ heißt *nichtinteraktiver Zero-Knowledge-Beweis* im Random-Oracle-Modell, falls die folgende Bedingung erfüllt ist:

- *Zero-Knowledge*: Es gibt einen probabilistischen, polynomiellen Algorithmus $\mathcal{S}$, so dass für alle $x \in L$ mit $|x| = k$ und alle probabilistischen polynomiellen Algorithmen $\mathcal{D}$ gilt:

$$\left|\mathbf{P}\left[\mathcal{D}\left(x, \mathcal{S}\left(x\right)\right) = 1\right] - \mathbf{P}\left[\mathcal{D}\left(x, \mathsf{view}_{\mathcal{V}}^{\mathcal{P}}\left(x\right)\right) = 1\right]\right| \leq \nu\left(k\right)$$

  mit einer in k vernachlässigbaren Funktion ν.

2.9.1.1 Der nichtinteraktive Beweis NIZK $\left[(\alpha, \beta) : A = g_1^{\alpha} g_2^{\beta} \wedge B = g_3^{\beta} \wedge C = g_4^{\beta}\right]$

Mit Hilfe der Fiat-Shamir-Heuristik ist es möglich, die in Abschnitt 2.7.1 vorgestellten Honest-Verifier-Zero-Knowledge-Beweise in nichtinteraktive Beweise zu transformieren, deren Zero-Knowledge-Eigenschaft im Random-Oracle-Modell erhalten bleibt. Exemplarisch soll dies für das in Abschnitt 2.7.1.1 vorgestellte interaktive Protokoll gezeigt werden. Der nichtinteraktive Beweis

$$\mathrm{NIZK}\left[(\alpha, \beta) : A = g_1^{\alpha} g_2^{\beta} \wedge B = g_3^{\beta} \wedge C = g_4^{\beta}\right]$$

ist in Abbildung 2.5 dargestellt. Zusätzlich zu den in Abschnitt 2.7.1 gewählten Parametern sei $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$ eine kollisionsresistente Hashfunktion.

$\mathcal{P}(\alpha, \beta, g_1, \ldots, g_4, A, B, C)$		$\mathcal{V}(g_1, \ldots, g_4, A, B, C)$
$\mathcal{P}$ zeigt: $A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta$		
$x_1, x_2 \in_\mathcal{R} \mathbb{Z}_q$ $A' = g_1^{x_1} g_2^{x_2}$ $B' = g_3^{x_2}$ $C' = g_4^{x_2}$ $c = \mathrm{H}(A', B', C')$ $r_1 = c\alpha + x_1 \bmod q$ $r_2 = c\beta + x_2 \bmod q$	$\xrightarrow{c, r_1, r_2}$	$\tilde{A} := g_1^{r_1} g_2^{r_2} A^{-c}$ $\tilde{B} = g_3^{r_2} B^{-c}$ $\tilde{C} = g_4^{r_2} C^{-c}$ $c \stackrel{?}{=} \mathrm{H}(\tilde{A}, \tilde{B}, \tilde{C})$

Abbildung 2.5: NIZK $\left[(\alpha, \beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$

Korollar 2.9.1. Das in Abbildung 2.5 dargestellte Protokoll ist im Random-Oracle-Modell ein nichtinteraktiver Zero-Knowledge-Beweis.

Beweis. Im Random-Oracle-Modell wird die im Protokoll verwendete Hashfunktion H (vgl. Abbildung 2.5) durch ein Random-Oracle O ersetzt. Wir zeigen nun, dass das Protokoll gemäß Definition 2.9.1 ein nichtinteraktiver Beweis ist und im Radom-Oracle-Modell die Zero-Knowledge-Eigenschaft erfüllt.

Durchführbarkeit, Korrektheit: Folgen direkt aus der Durchführbarkeit bzw. Korrektheit der interaktiven Variante aus Abschnitt 2.7.1.1 (vgl. Satz 2.7.1).

Zero-Knowledge: Der nichtinteraktive Beweis kann im Random-Oracle-Modell von einem Simulator $\mathcal{S}$ effizient simuliert werden. Der Simulator $\mathcal{S}$ erstellt die Simulation π' des nichtinteraktiven Beweises wie folgt:

Schritt 1: Der Simulator $\mathcal{S}$ wählt $c, r_1, r_2 \in_\mathcal{R} \mathbb{Z}_q$.

Schritt 2: Der Simulator $\mathcal{S}$ berechnet Commitments $A' = g_1^{r_1} g_2^{r_2} A^{-c}$, $B' = g_3^{r_2} B^{-c}$ und $C' = g_4^{r_2} C^{-c}$.

Aus der Honest-Verifier-Zero-Knowledge-Eigenschaft der interaktiven Variante (vgl. Satz 2.7.1) folgt die Ununterscheidbarkeit zwischen der Simulation π' und einem echten nichtinteraktiven Beweis π.
Das Random-Oracle O' simuliert $\mathcal{S}$ indem er $O'(A', B', C') := c$ setzt und die restliche Ausgabe von O' zufällig und gleichverteilt simuliert. Somit sind das simulierte Random-Oracle O' und das Random-Oracle O rechnerisch ununterscheidbar. □

Bemerkung 2.9.1. Durch Anwenden der Fiat-Shamir-Heuristik können die in Abschnitt 2.7.1 vorgestellten Honest-Verifier-Zero-Knowledge-Beweise im Random-Oracle-Modell nicht nur in nichtinteraktive Zero-Knowledge-Beweise, sondern auch in digitale Signaturverfahren transformiert werden (vgl. Abschnitt 2.8). Für eine Nachricht $m \in \{0,1\}^*$ kann beispielsweise der in Abbildung 2.5 dargestellte nichtinteraktive Zero-Knowledge-Beweis NIZK $\left[(\alpha,\beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right]$ in ein digitales Signaturverfahren umgewandelt werden, indem man die Challenge $c = \mathrm{H}(A', B', C')$ durch $c = \mathrm{H}(m, A', B', C')$ ersetzt. Im weiteren Verlauf der Arbeit wird diese Vorgehensweise durch folgende Notation angedeutet:

$$\mathrm{NIZK}\left[(\alpha,\beta) : A = g_1^\alpha g_2^\beta \wedge B = g_3^\beta \wedge C = g_4^\beta\right](m)\,.$$

2.9.1.2 Der nichtinteraktive Beweis

$$\mathrm{NIZK}\left[(\alpha,\beta_1,\ldots,\beta_6) : A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}\right]$$

in Kooperation mit einem weiteren Prover

Wie bereits erwähnt kann auch der in Abschnitt 2.7.1.3 vorgestellte Honest-Verifier-Zero-Knowledge-Beweis im Random-Oracle-Modell mittels der Fiat-Shamir-Heuristik in einen nichtinteraktiven Zero-Knowledge-Beweis transformiert werden. Kennt der Prover $\mathcal{P}$ eine der geforderten Darstellungen nicht, kann $\mathcal{P}$ den nichtinteraktiven Beweis NIZK $\left[(\alpha,\beta_1,\ldots,\beta_6) : A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}\right]$ nicht erstellen.
Im Folgenden wird gezeigt, wie dieser nichtinteraktive Beweis gemeinsam von zwei Provern $\mathcal{P}_1$ und $\mathcal{P}_2$ erstellt werden kann, wenn der g_1-Anteil der Darstellung von A bzgl. (a, g_1, g_2) zwischen den beiden Prover $\mathcal{P}_1$ und $\mathcal{P}_2$ aufgeteilt wird. Dabei wird der nichtinteraktive Beweis so konstruiert, dass $\mathcal{P}_2$ zur Erstellung des nichtinteraktiven Beweises ausschließlich mit $\mathcal{P}_1$ kommuniziert. Insbesondere kann $\mathcal{P}_2$ nicht auf direktem Wege mit dem Verifier $\mathcal{V}$ interagieren. In Kapitel 6 werden Kunde und Observer die Rolle von $\mathcal{P}_1$ und $\mathcal{P}_2$ übernehmen, um auf diese Weise die Mehrfachausgabe elektronischer Münzen zu verhindern (vgl. Abschnitt 1.1).
Zusätzlich zu den aus Abschnitt 2.7.1 bekannten Parametern wird das Geheimnis $\beta_1 \in \mathbb{Z}_q$ wie folgt zwischen $\mathcal{P}_1$ und $\mathcal{P}_2$ aufgeteilt: Es ist $\beta_1 = \beta_{\mathcal{P}_1} + \beta_{\mathcal{P}_2} \bmod q$, wobei $\beta_{\mathcal{P}_1} \in \mathbb{Z}_q$ dem Prover $\mathcal{P}_1$ und $\beta_{\mathcal{P}_2} \in \mathbb{Z}_q$ dem Prover $\mathcal{P}_2$ bekannt ist. Außerdem kennt $\mathcal{P}_1$ den Wert $D = g_1^{\beta_{\mathcal{P}_2}}$.
Mit der Fiat-Shamir-Heuristik und der in [Bra93] beschriebenen Methode kann der aus Abschnitt 2.7.1.3 bekannte interaktive Beweis so transformiert werden, dass $\mathcal{P}_1$ den nichtinteraktiven Beweis nur gemeinsam mit $\mathcal{P}_2$ erstellen kann. Dazu wird folgendes Protokoll zwischen $\mathcal{P}_1$ und $\mathcal{P}_2$ durchgeführt (vgl. auch Abbildung 2.6):

Schritt 1: Der Prover $\mathcal{P}_2$ wählt zufällig den Wert $x_{\mathcal{P}_2}$, berechnet $D' = g_1^{x_{\mathcal{P}_2}}$ und sendet D' an $\mathcal{P}_1$.

Schritt 2: Der Prover $\mathcal{P}_1$ wählt zufällig die Werte $x, x_1, \ldots, x_7, x_{\mathcal{P}_1} \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $A' = a^x g_1^{x_1} D^{x_{\mathcal{P}_1}} D' g_2^{x_2}$, $B' = b^x g_2^{x_3} g_3^{x_4}, C' = c^x g_3^{x_5} g_4^{x_6}$. Weiter berechnet $\mathcal{P}_1$ die

$\mathcal{P}_2(\beta_{\mathcal{P}_2})$		$\mathcal{P}_1(\alpha, \beta_{\mathcal{P}_1}, \beta_2, \ldots, \beta_6, \ldots, A, B, C)$	$\mathcal{V}(a, b, c, \ldots, A, B, C)$
$\mathcal{P}_1$ und $\mathcal{P}_2$ zeigen gemeinsam: $A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}$			
$x_{\mathcal{P}_2} \in_{\mathcal{R}} \mathbb{Z}_q$			
$D' = g_1^{x_{\mathcal{P}_2}}$	$\xrightarrow{D'}$	$x, x_1, \ldots, x_7, x_{\mathcal{P}_1} \in_{\mathcal{R}} \mathbb{Z}_q$	
		$A' = a^x g_1^{x_1} D^{x_{\mathcal{P}_1}} D' g_2^{x_2}$	
		$B' = b^x g_2^{x_3} g_3^{x_4}$	
		$C' = c^x g_3^{x_5} g_4^{x_6}$	
		$d = \mathrm{H}(A', B', C')$	
	$\xleftarrow{d'}$	$d' = c + x_{\mathcal{P}_1} \bmod q$	
$r_1' = d'\beta_{\mathcal{P}_2} + x_{\mathcal{P}_2} \bmod q$	$\xrightarrow{r_1'}$	$g_1^{r_1'} \stackrel{?}{=} D^{d'} D'$	
		$r = d\alpha + x \bmod q$	
		$r_1 = r_1' + d\beta_{\mathcal{P}_1} + x_1 \bmod q$	
		$\hat{r}_1 = d\beta_2 + x_2 \bmod q$	
		$r_2 = d\beta_3 + x_3 \bmod q$	
		$\hat{r}_2 = d\beta_4 + x_4 \bmod q$	
		$r_3 = d\beta_5 + x_5 \bmod q$	
		$\hat{r}_3 = d\beta_6 + x_6 \bmod q$	$\xrightarrow{d, r, r_1, \hat{r}_1, \ldots, r_3, \hat{r}_3}$
			$\tilde{A} = a^r g_1^{r_1} g_2^{\hat{r}_1} A^{-d}$
			$\tilde{B} = b^r g_2^{r_2} g_3^{\hat{r}_2} B^{-d}$
			$\tilde{C} = c^r g_3^{r_3} g_4^{\hat{r}_3} C^{-d}$
			$d \stackrel{?}{=} \mathrm{H}(\tilde{A}, \tilde{B}, \tilde{C})$

Abbildung 2.6: NIZK $\left[(\alpha, \beta_1, \ldots, \beta_6) : A = a^\alpha g_1^{\beta_1} g_2^{\beta_2} \wedge B = b^\alpha g_2^{\beta_3} g_3^{\beta_4} \wedge C = c^\alpha g_3^{\beta_5} g_4^{\beta_6}\right]$

Challenge $d = \mathrm{H}(A', B', C')$ und blendet d zu $d' = d + x_{\mathcal{P}_1} \bmod q$. Die geblendete Challenge d sendet $\mathcal{P}_1$ an $\mathcal{P}_2$.

Schritt 3: Der Prover $\mathcal{P}_2$ berechnet die Response $r_1' = d'\beta_{\mathcal{P}_2} + x_{\mathcal{P}_2} \bmod q$ und sendet r_1' an $\mathcal{P}_1$.

Schritt 4: Der Prover $\mathcal{P}_1$ verifiziert $g_1^{r_1'} \stackrel{?}{=} D^{d'} D$ und berechnet ggf. $r = d\alpha + x \bmod q$, $r_1 = d\beta_1 + x_1 \bmod q$, $\hat{r}_1 = d\beta_2 + x_2 \bmod q$, $r_2 = d\beta_3 + x_3 \bmod q$, $\hat{r}_2 = d\beta_4 + x_4 \bmod q$, $r_3 = d\beta_5 + x_5 \bmod q$ sowie $\hat{r}_3 = d\beta_6 + x_6 \bmod q$. Abschließend sendet $\mathcal{P}_1$ die Werte $d, r, r_1, \hat{r}_1, r_2, \hat{r}_2, r_3$ und $\hat{r}_3$ an $\mathcal{V}$.

Schritt 5: Der Verifizierer $\mathcal{V}$ verifiziert den nichtinteraktiven Beweis, indem $\mathcal{V}$ die Werte $\tilde{A} = a^r g_1^{r_1} g_2^{\hat{r}_1} A^{-d}$, $\tilde{B} = b^r g_2^{r_2} g_3^{\hat{r}_2} B^{-d}$ sowie $\tilde{C} = c^r g_3^{r_3} g_4^{\hat{r}_3} C^{-d}$ berechnet und überprüft, ob $d = \mathrm{H}(\tilde{A}, \tilde{B}, \tilde{C})$ gilt.

Während der Erstellung dieses nichtinteraktiven Beweises zeigt $\mathcal{P}_2$ gegenüber $\mathcal{P}_1$, dass $\mathcal{P}_2$ den diskreten Logarithmus $\log_{g_1} D = \beta_{\mathcal{P}_2}$ von D zur Basis g_1 kennt. Betrachtet man

dieses Unterprotokoll für sich, so ist dies ein Honest-Verifier-Zero-Knowledge-Beweis für die Kenntnis des diskreten Logarithmus $\log_{g_1} D$. Das heißt, der Prover $\mathcal{P}_1$ erhält während der Erstellung des nichtinteraktiven Beweises keine Information über das Geheimnis $\beta_{\mathcal{P}_2}$ von $\mathcal{P}_2$.

Bemerkung 2.9.2. Wie bereits erwähnt, übernehmen in der in Abschnitt 6.3 entwickelten elektronische Geldbörse Kunde und Observer die Rolle der beiden Prover, um durch die Aufteilung des Geheimnisses β_1 die Mehrfachausgabe elektronischer Münzen zu verhindern. Anhand dieses nichtinteraktiven Beweises überzeugt der Kunde mit Hilfe seines Observers die Bank davon, dass er den korrekten Münz-Identifikator, der für das Münztracing verwendet wird, verschlüsselt hat. Da der Kunde in Abschnitt 6.3 die Darstellungen seiner elektronischen Münzen alleine nicht kennt, kann er die Bank nur mit Hilfe des Observers von der richtigen Konstruktion des Münz-Identifikators überzeugen.

Kapitel 3

Überblick elektronischer Geldsysteme und ihrer Sicherheitsziele

Während die Internetverbreitung und damit auch die Nutzung des Online-Handels in Deutschland immer weiter fortschreitet (siehe z. B. [Ges08], [ARD08]), dominieren noch immer traditionelle Zahlungsverfahren, die nicht speziell für das Internet konzipiert sind, den Online-Zahlungsverkehr. Darunter fallen zum Beispiel Vorkasse, Nachnahme, Lastschriftverfahren und Kreditkartenabrechnung, mit all ihren Risiken, wie sie bereits in Kapitel 1.1 diskutiert wurden.
Dieses Kapitel gibt einen Überblick über die verschiedenen Eigenschaften und Sicherheitsziele elektronischer Geldsysteme, die vor allem auf Grund ihrer *Anonymität* als elektronisches Pendant zu gewöhnlichem Bargeld bezeichnet werden. Zunächst werden in Abschnitt 3.1 das grundlegende Modell elektronischer Geldsysteme sowie die wichtigsten Sicherheitsziele wie *Unfälschbarkeit*, *Nichterweiterbarkeit* und *Anonymität* erläutert und formalisiert. Im darauffolgenden Abschnitt 3.2 wird das Konzept der elektronischen Geldbörse vorgestellt, mit dem die Mehrfachausgabe elektronischer Münzen verhindert werden soll. Abschließend werden in Abschnitt 3.3 Eigenschaften und Sicherheitsziele *fairer* elektronischer Geldsysteme dargestellt, die einen Schutz vor Erpressung und Geldwäsche bieten.

3.1 Elektronische Geldsysteme

Nicht nur auf Grund ihrer Sicherheitsmerkmale werden elektronische Geldsysteme als elektronisches Äquivalent zu gewöhnlichem Bargeld angesehen. Auch der Umgang mit elektronischen Münzen ähnelt aus Sicht der Kunden, bis auf eine Einschränkung, stark dem herkömmlichen Bargeld (vgl. Abbildung 3.1). In einem elektronischen Geldsystem werden Münzen in einem *Abhebe-Protokoll* bei der Bank abgehoben, um sie anschließend in einem *Bezahl-Protokoll* als Zahlungsmittel bei einem Händler zu verwenden. Während ein Händler Bargeld weiter als Wechselgeld bzw. Zahlungsmittel nutzen kann, muss er die elektronischen Münzen in einem *Einlöse-Protokoll* wieder bei der Bank einzahlen,

damit diese mit dem *Identifizierungsalgorithmus* eine mögliche Mehrfachausgabe aufdecken kann. Formal lässt sich ein elektronisches Geldsystem wie folgt beschreiben, wobei sich die Definition an [Tsi97] bzw. [FY93] orientiert.

Definition 3.1.1 (**Elektronisches Geldsystem**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter. Ein *elektronisches (Offline-) Geldsystem* besteht aus einer probabilistischen, polynomiellen interaktiven Turing-Maschine $\mathcal{B}$, der Bank, zwei Mengen probabilistischer, polynomieller interaktiver Turing-Maschinen K und H, den Kunden bzw. Händlern und den folgenden polynomiellen Algorithmen bzw. Protokollen:

1. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{BKeyGen}$ der Bank $\mathcal{B}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k einen öffentlichen Schlüssel $\mathsf{pk}_\mathcal{B}$ und einen privaten Schlüssel $\mathsf{sk}_\mathcal{B}$ erzeugt. Die Ausgabe von $\mathsf{BKeyGen}(1^k)$ ist das Paar $(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B})$.

2. Das *Kontoeröffnungs-Protokoll* Reg ist ein interaktives Protokoll zwischen einem Kunden $\mathcal{K} \in K$ und der Bank $\mathcal{B}$, bei dem $\mathsf{pk}_\mathcal{B}$ sowohl die Eingabe des Kunden $\mathcal{K}$ als auch der Bank $\mathcal{B}$ ist. Die Ausgabe des Kunden $\mathcal{K}(\mathsf{pk}_\mathcal{B})$ ist das Schlüsselpaar $(\mathsf{pk}_\mathcal{K}, \mathsf{sk}_\mathcal{K})$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{B}(\mathsf{pk}_\mathcal{B})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

3. Das *Abhebe-Protokoll* $\mathsf{Withdrawal}$ ist ein interaktives Protokoll zwischen einem Kunden $\mathcal{K} \in K$ und der Bank $\mathcal{B}$, bei dem $\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{K}$ die Eingabe des Kunden $\mathcal{K}$ und $(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B})$ die Eingabe der Bank $\mathcal{B}$ ist. Die Ausgabe von $\mathcal{K}(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{K})$ ist eine elektronische Münze $\mathsf{Coin} = (m, s)$, bestehend aus einer Nachricht $m \in M$ und einer Signatur s. Entsprechend ist die Ausgabe von $\mathcal{B}(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

4. Das *Bezahl-Protokoll* $\mathsf{Payment}$ ist ein interaktives Protokoll zwischen einem Kunden $\mathcal{K} \in K$ und einem Händler $\mathcal{H} \in H$. Es ist $\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{K}, \mathsf{Coin}$ die Eingabe des Kunden $\mathcal{K}$ und $\mathsf{pk}_\mathcal{H}, \mathsf{pk}_\mathcal{B}$ die Eingabe von $\mathcal{H}$. Die Ausgabe des Händlers $\mathcal{H}(\mathsf{pk}_\mathcal{H}, \mathsf{pk}_\mathcal{B})$ ist der Wert 1 (= $\mathsf{accepted}$) oder 0 (= $\mathsf{not\ accepted}$). Entsprechend ist die Ausgabe von $\mathcal{K}(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{K}, \mathsf{Coin})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

5. Das *Einlöse-Protokoll* $\mathsf{Deposit}$ ist ein interaktives Protokoll zwischen einem Händler $\mathcal{H}$ und der Bank $\mathcal{B}$. Es ist $\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{H}$ sowie $\mathsf{view}_\mathcal{H}^\mathcal{K}(P)$ die Eingabe des Händlers $\mathcal{H}$ und $\mathsf{pk}_\mathcal{B}$ die Eingabe der Bank $\mathcal{B}$. Die Ausgabe der Bank $\mathcal{B}(\mathsf{pk}_\mathcal{B})$ ist der Wert 1 (= $\mathsf{accepted}$) oder 0 (= $\mathsf{not\ accepted}$). Entsprechend ist die Ausgabe von $\mathcal{H}\left(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{H}, \mathsf{view}_\mathcal{H}^\mathcal{K}(P)\right)$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

6. Der *Identifizierungsalgorithmus* $\mathsf{Identify}$ der Bank $\mathcal{B}$ ist ein deterministischer Algorithmus. Bei Eingabe der Protokollansichten $\mathsf{view}_\mathcal{B}^{\mathcal{H}_1}(D_1), \mathsf{view}_\mathcal{B}^{\mathcal{H}_2}(D_2)$ zweier Einlöse-Protokolle D_1 und D_2 gibt $\mathsf{Identify}\left(\mathsf{view}_\mathcal{B}^{\mathcal{H}_1}(D_1), \mathsf{view}_\mathcal{B}^{\mathcal{H}_2}(D_2)\right)$ die Identität id des betrügenden Kunden bzw. Händlers aus, falls in D_1 und D_2 die selbe Münze eingelöst wurde und $\perp$ sonst.

Im weiteren Verlauf dieser Arbeit bezeichnen wir mit

- $\mathsf{view}_{\mathcal{K}}^{\mathcal{B}}(W)$ die Protokollansicht eines Kunden $\mathcal{K}$ bei Durchführung des Abhebe-Protokolls W mit der Bank $\mathcal{B}$; entsprechend ist $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ die Ansicht von $\mathcal{B}$.

- $\mathsf{view}_{\mathcal{K}}^{\mathcal{H}}(P)$ die Protokollansicht von $\mathcal{K}$ bei Durchführung des Bezahl-Protokolls P mit einem Händler $\mathcal{H}$ bzw. $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ die Protokollansicht von $\mathcal{H}$.

- $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ die Ansicht von $\mathcal{B}$ bei Durchführung des Einlöse-Protokolls D mit $\mathcal{H}$.

Für $i \in \mathbb{N}$ sei $\mathbf{W}_i := \{\mathsf{view}_{\mathcal{K}_1}^{\mathcal{B}}(W_1), \ldots, \mathsf{view}_{\mathcal{K}_i}^{\mathcal{B}}(W_i)\}$ die Menge der Protokollansichten von i Abhebe-Protokollen und $\mathbf{P}_i := \{\mathsf{view}_{\mathcal{H}_1}^{\mathcal{K}_1}(P_1), \ldots, \mathsf{view}_{\mathcal{H}_N}^{\mathcal{K}_i}(P_i)\}$ die Menge der Protokollansichten der entsprechenden i Bezahl-Protokolle.

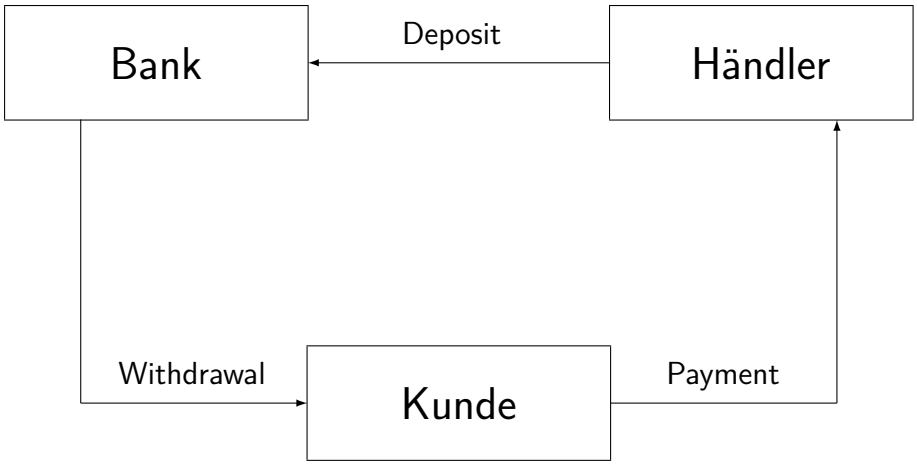

Abbildung 3.1: Das Modell elektronischer Geldsysteme

Wie bereits in Abschnitt 1.1 erwähnt, haben sich in der Literatur vor allem zwei Ansätze zur Entwicklung elektronischer Geldsysteme etabliert: blinde Signaturen (vgl. Abschnitt 2.6) und Systeme die auf Zero-Knowledge basieren (vgl. Abschnitt 2.7).
Der Unterschied beider Ansätze ist der Zeitpunkt, an dem in den Geldsystemen die Anonymität der Kunden erreicht wird. Während elektronische Geldsysteme, die auf blinden Signaturen aufbauen (z. B. [CFN89], [CP93], [Fer93], [Bra94], [GT03]), die Anonymität bereits durch das Abhebe-Protokoll realisieren, wird die Anonymität in elektronischen Geldsystemen, die auf Zero-Knowledge basieren (z. B. [STS99], [STSY00], [CHL05]) erst mit dem Bezahl-Protokoll erzielt.
Bei den auf blinden Signaturverfahren basierenden elektronischen Geldsystemen zählen [Bra94] und darauf aufbauende Verfahren wie [FTY96],[FTY98] zu den effizientesten Systemen, da diese zur Identifizierung eines Double-Spenders *Secret-Sharing-Verfahren* [Sha79] einsetzen, im Gegensatz zu Verfahren wie [CFN89], [OO91], [FY93], [XY03], die zur Identifizierung eines Double-Spenders mit der, auf Grund ihres enormen Kommunikations- und Rechenaufwands, deutlich ineffizienteren *Cut-And-Choose-Technik* arbeiten.

3.1.1 Zusatzeigenschaften elektronischer Geldsysteme

Gegenüber gewöhnlichem Bargeld haben elektronische Geldsysteme, wie sie in diesem Abschnitt vorgestellt wurden, einen entscheidenden Nachteil. Während Bargeld von Kunde zu Kunde weitergegeben bzw. vom Händler auch als Wechselgeld verwendet werden kann, werden diese Möglichkeiten von einem elektronischen Geldsystem in seiner grundlegenden Form nicht unterstützt, da elektronische Münzen auf Grund der Double-Spending-Detection immer den Zyklus Abheben-Bezahlen-Einlösen (vgl. Abbildung 3.1) durchlaufen.
Aber auch diese Eigenschaft des Bargeldes, die man als *Übertragbarkeit* bezeichnet, lässt sich in elektronische Geldsysteme integrieren [CP94]. Aus der Übertragbarkeit elektronischer Münzen folgt aber unmittelbar, dass eine *übertragbare Münze* von jedem Kunden, der diese Münze einmal erhalten hat, mehrfach ausgegeben werden kann. Folglich muss auch die Identität jedes einzelnen Kunden in die übertragbare Münze integriert werden, damit die Bank in einem System mit Übertragbarkeit Double-Spender erfolgreich identifizieren kann. Mit anderen Worten, die Größe einer übertragbaren Münze wächst mit jeder Weitergabe, wie von D. Chaum und T. Pedersen [CP94] gezeigt wurde. Eine Eigenschaft, die übertragbare Systeme aus praktischen Gesichtspunkten sicherlich uninteressant macht.
Die fehlende Möglichkeit in einem, in der Praxis einsetzbaren, elektronischen Geldsystem Wechselgeld ausstellen zu können, zwingt die Kunden dazu beim Bezahlen immer den exakten, zu bezahlenden Betrag parat zu haben. Dementsprechend müssen die Kunden jederzeit einige Münzen verschiedener Denominierungen zur Verfügung haben. Dies motivierte im Jahre 1991 T. Okamoto und K. Ohta [OO91] zum Entwurf der so genannten *teilbaren* elektronischen Münzen. Teilbare Münzen können von den Kunden in mehrere Münzen verschiedener, kleinerer Denominierungen aufgeteilt werden und ermöglichen somit exakte Bezahlungen. Übersteigen die beim Bezahlen verwendeten Münzen den Nennwert der ursprünglich abgehobenen Münze, so kann die Bank, genauso wie im Fall der Mehrfachausgabe elektronischer Münzen, die Identität des betrügerischen Kunden aufdecken.
Elektronische Geldsysteme mit *Teilbarkeit* sind beispielsweise [OO91], [EO94], [Oka95], [Tsi97]. Innerhalb der teilbaren Geldsysteme wird zwischen Systemen mit *verknüpfbaren, teilbaren Münzen* (z.B. [OO91], [EO94], [Oka95]) und Systemen, deren teilbare Münzen nicht verknüpfbar sind (z.B. [Tsi97]), unterschieden.

3.1.2 Sicherheitsziele elektronischer Geldsysteme

Über die Akzeptanz elektronischer Bezahlsysteme entscheidet in erster Linie deren Sicherheit. Die Sicherheitsanalyse elektronischer Geldsysteme ist vielschichtig und umfasst mehrere Aspekte. Bereits in Abschnitt 1.1 wurden wesentliche Sicherheitsmerkmale elektronischer Geldsysteme angesprochen. Ein vorrangiges Sicherheitsziel aller involvierten Parteien ist die Echtheit der Münzen. Aus Sicht der Bank sind Unfälschbarkeit und die Möglichkeit Double-Spender zu identifizieren unabdingbare Eigenschaften, während für die Kunden der Bank die Anonymität des Zahlungsverkehrs im Vordergrund steht.

Im Jahre 1997 wurde von Y. Tsiounis [Tsi97] ein formales Sicherheitsmodell vorgestellt, das auf der Arbeit von M. Franklin und M. Yung [FY93] aufbaut. Ein elektronisches Geldsystem heißt sicher, falls es folgender Definition genügt:

Definition 3.1.2 (**Sicherheit elektronischer Geldsysteme**)**.** Ein elektronisches Geldsystem mit Sicherheitsparameter k heißt *sicher*, falls es die folgenden Bedingungen erfüllt:

1. *Identitäts-Nachweis*: Es sei Coin eine von $\mathcal{K}$ bei $\mathcal{B}$ abgehobene Münze. Falls Coin zweimal bei $\mathcal{B}$ eingelöst wird, kann $\mathcal{B}$ anhand der Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_0}(D_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}(D_1)$ der beiden Einlöse-Protokolle D_0, D_1 den Betrug aufdecken und mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers effizient berechnen.

2. *Nichterweiterbarkeit*: Es gibt keinen effizienten Angreifer $\mathcal{A}$, der bei Eingabe von $\mathbf{W}_N$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $N+1$ gültige Münzen ausgibt, die erfolgreich ausgegeben werden können.

3. *Unfälschbarkeit*: Falls ein effizienter Angreifer $\mathcal{A}$ existiert, der bei Eingabe von $\mathbf{W}_N$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit eine Münze Coin ausgibt, die zweimal erfolgreich ausgegeben wird, so kann $\mathcal{B}$ den Betrug aufdecken und mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers effizient berechnen.

4. *Anonymität*: Es gibt keinen effizienten Angreifer $\mathcal{A}$, der bei Eingabe zweier Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ $(b, \bar{b} \in \{0,1\}, b \neq \bar{b})$, den Protokollansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ und den $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ entsprechenden Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört. Dabei ist ν eine nicht vernachlässigbare Funktion.

Einige Bemerkungen zur Anonymität.

1. Anonymität gemäß Definition 3.1.2 wird auch als *rechnerische Anonymität* bezeichnet, im Gegensatz zur *perfekten* Anonymität (vgl. auch Abschnitt 2.6). Perfekte Anonymität bieten beispielsweise elektronische Geldsysteme wie [CFN89], [Bra94], [Bra93], während faire elektronische Geldsysteme (vgl. Abschnitt 3.3) lediglich rechnerische Anonymität erreichen können (vgl. Satz 3.3.1).

2. Für die Anonymität elektronischer Geldsysteme spielt auch die *Nichtverknüpfbarkeit* elektronischer Münzen eine wichtige Rolle. Zwei eingelöste Münzen werden als *nicht verknüpfbar* bezeichnet, falls die Bank anhand der beiden eingelösten Münzen nicht effizient entscheiden kann, ob die beiden Münzen von ein und demselben Kunden abgehoben wurden. Für den Fall, dass die Münzen eines Kunden verknüpfbar wären, könnte die Bank die Anonymität des Kunden möglicherweise anhand zusätzlicher Informationen aufheben. Dass dies in einem gemäß Definition 3.1.2 sicheren System nicht möglich ist, wurde in [Tsi97] gezeigt, denn Anonymität, wie sie

in Definition 3.1.2 definiert ist, impliziert die Nichtverknüpfbarkeit elektronischer Münzen.

3. Völlige Anonymität können selbst elektronische Geldsysteme nicht gewährleisten. Sichere elektronische Geldsysteme bieten zwar Anonymität im Sinne von Definition 3.1.2, die Anonymität der Kunden kann jedoch möglicherweise auch auf anderen Ebenen aufgehoben werden. Denn sowohl die Bank als auch die Händler können während der Kommunikation mit ihren Kunden deren IP-Adressen protokollieren. Falls die Händler mit der Bank kooperieren, kann die Bank anhand der protokollierten IP-Adressen eingelöste Münzen ihren ursprünglichen Besitzern zuzuordnen; vorausgesetzt die Kunden verfügen über feste IP-Adressen. Vor einem solchen Szenario können sich Kunden durch den Einsatz von *Anonymisierungsdiensten*, die auf *MIX-Kaskaden* (siehe z.B. [BSW06], [BNS05]) basieren, schützen.

3.2 Elektronische Geldbörsen

Gegenüber ihrem physikalischen Counterpart haben elektronische Münzen einen grundlegenden Nachteil; sie können leicht kopiert und mehrfach ausgegeben werden (vgl. Abschnitt 1.1). Um die Mehrfachausgabe elektronischer Münzen nicht nur im Nachhinein aufzudecken, sondern sie bereits im Vorfeld zu verhindern, wurde im Jahre 1992 von D. Chaum und T. Pedersen [CP93] das Konzept der *elektronischen Geldbörse*, auch als *digitale Geldbörse* bezeichnet, vorgeschlagen und von R. Cramer und T. Pedersen [CP94] weiterentwickelt.
Bei diesem Konzept erhält jeder Kunde von der Bank ein manipulationssicheres Speichermedium, einen so genannten *Observer*. Der Observer kann nur direkt mit dem Kunden kommunizieren; insbesondere kann er keine direkte Verbindung zur Bank oder anderen Kunden bzw. Händlern aufbauen.
Während in einem elektronischen Geldsystem gemäß Definition 3.1.1 Münzen allein vom Kunden abgehoben und ausgegeben werden, sind die Protokolle einer elektronischen Geldbörse so entworfen, dass der Kunde Münzen nur noch zusammen mit seinem Observer abheben und ausgeben kann (vgl. Abbildung 3.2). Um dies zu gewährleisten, wird ein Teil der Information, die der Kunde zum Bezahlen mit einer Münze benötigt, auf dessen Observer gespeichert. Beim erneuten Versuch des Kunden, eine Münze auszugeben, wird der Observer nicht mit dem Kunden kooperieren und ihn somit an der Mehrfachausgabe seiner Münzen hindern. Aus Sicht der Bank und Händler ändert sich im Vergleich zu einem System ohne Observer nichts, da sie die Protokolle nach wie vor nur mit dem Kunden durchführen.
Damit die Anonymität der Kunden auch beim Einsatz von der Bank ausgegebener Observer gewährleistet bleibt, müssen Abhebe- und Bezahl-Protokoll einer elektronischen Geldbörse so konzipiert werden, dass selbst die indirekte Kommunikation zwischen Observer und Bank keinerlei Möglichkeit bietet, Information (zwischen Observer und Bank) auszutauschen, die es der Bank ermöglichen würden die Anonymität der Kunden aufzuheben. Dabei ist auch zu berücksichtigen, dass die auf den Observer gespeicherten Daten

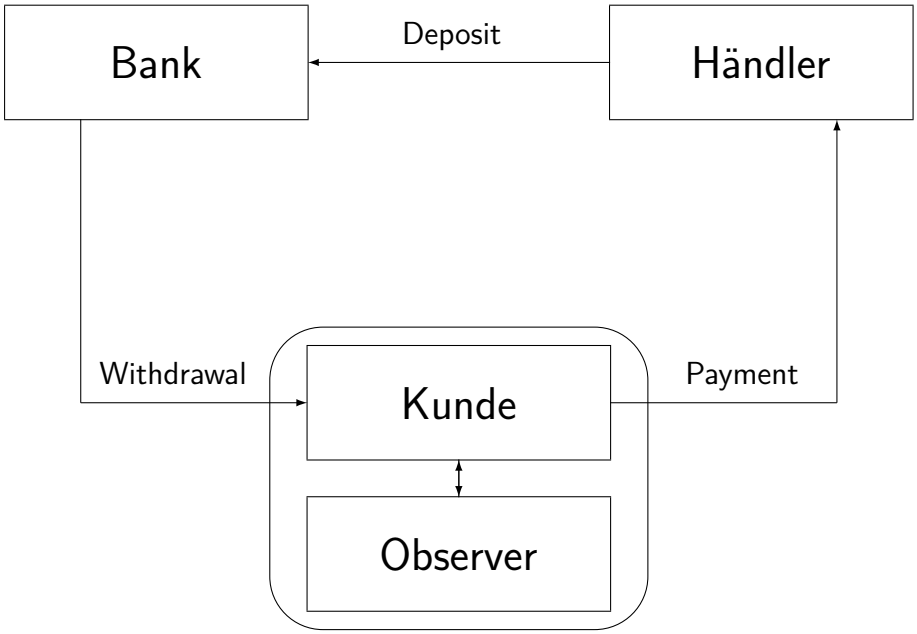

Abbildung 3.2: Das Konzept der elektronischen Geldbörse

möglicherweise von der Bank ausgelesen werden können, zum Beispiel bei einem Defekt des Observers oder im Kündigungsfall. Es wird zwischen *eingehender* und *ausgehender* Information unterschieden (siehe [CP93]).

- *Eingehende Information*: Verhält sich der Kunde dem Abhebe- bzw. Bezahl-Protokoll entsprechend, so kann die Bank dem Observer keine zusätzliche Information zukommen lassen. Selbst wenn Bank und Observer von Abhebe- bzw. Bezahl-Protokoll abweichen.
- *Ausgehende Information*: Verhält sich der Kunde dem Abhebe- bzw. Bezahl-Protokoll entsprechend, so kann der Observer der Bank keine zusätzliche Information zukommen lassen. Selbst wenn Bank und Observer von Abhebe- bzw. Bezahl-Protokoll abweichen.

Elektronische Geldbörsen müssen demnach sowohl eingehende als auch ausgehende Information zwischen Bank und Observern verhindern. Systeme, die diese Forderungen erfüllen, sind beispielsweise [Bra94],[Bra93],[ST98],[NS03].

Einige Bemerkungen zu elektronischen Geldbörsen

1. Elektronische Geldbörsen werden in den meisten Fällen mit Hilfe von Chipkarten, deren Speicher- und Rechenkapazitäten stark beschränkt sind, realisiert. Aus Effizienzgründen sollten die Protokolle einer elektronischen Geldbörse daher so entworfen werden, dass auf dem Observer möglichst wenige Rechenoperationen durchgeführt und pro gespeicherter Münze möglichst wenig Speicherplatz verbraucht werden.

2. Auch bei einer elektronischen Geldbörse kann vollkommene Anonymität nicht gewährleistet werden (vgl. Abschnitt 3.1). Um die Anonymität der Kunden zusätzlich zu schützen, dürfen auf den Observern gespeicherte Daten keinerlei zeitliche Information enthalten, da die Bank andernfalls bei Rückgabe der Observer ihre Daten mit den, auf den Observern gespeicherten Daten, zeitlich abgleichen kann.

3. Wurde das Konzept der elektronischen Geldbörse in den 90er Jahren des vergangenen Jahrhunderts ursprünglich dazu entworfen die Interessen der Bank zu schützen, bietet der Einsatz zusätzlicher, manipulationssicherer Hardware den Kunden heutzutage gleichzeitig Schutz vor Angriffen durch Trojaner, Viren, usw. Zum Beispiel schützen Observer die Kunden vor Viren, die versuchen deren elektronische Münzen zu löschen oder zu entwenden bzw. die versuchen sich der geheimen Information zu bemächtigen, die nötig ist, Münzen auszugeben.

3.3 Faire elektronische Geldsysteme

Im Vergleich zu herkömmlichem Bargeld bieten elektronische Münzen eine stärkere Form der Anonymität, da die Spur gewöhnlicher Geldscheine anhand ihrer Seriennummern zurückverfolgt werden kann. Diese stärkere Form der Anonymität kann, wie bereits in Abschnitt 1.1 angesprochen, zur Geldwäsche oder Erpressung missbraucht werden (siehe z.B. [SN92], [BGK95], [CPS95]).
Um kriminelle Handlungen dieser Art verfolgen zu können, bedarf es in elektronischen Geldsystemen weiterer Mechanismen. Liegt zum Beispiel der Verdacht der Geldwäsche nahe, so ist für die Ermittlungsbehörden der Besitzer der Münzen von großer Bedeutung. Ebenso ist es bei einer Erpressung nützlich die erpressten Münzen beim Ausgeben zu erkennen und gegebenenfalls zu sperren.
Diese beiden Szenarien machen deutlich, dass zwei verschiedene Mechanismen zur Aufhebung der Anonymität benötigt werden. Zum einen das so genannte *Münztracing*, bei dem es möglich sein sollte, Münzen bestimmter Abhebevorgänge ausfindig zu machen und zum anderen das *Kundentracing*, bei dem der Besitzer der Münze aufgedeckt wird. Elektronische Geldsysteme, die über solche Tracing-Mechanismen verfügen, werden auch als *faire* elektronische Geldsysteme bezeichnet.
Da die Bank nicht in der Lage sein darf Kunden- und Münztracing durchzuführen, denn sonst wäre die Anonymität der Kunden nur noch auf das Vertrauen gegenüber der Bank gestützt, wird eine weitere Partei (vgl. [BGK95], [CPS95]) eingeführt. Diese Partei sollte unabhängig von der Bank handeln und das Vertrauen aller Kunden genießen und wird aus diesem Grund auch als *Trusted-Third-Party* bezeichnet. Im weiteren Verlauf dieser Arbeit wird eine Trusted-Third-Party, die für die Aufhebung der Anonymität zuständig ist, auch als *Deanonymisierer* bezeichnet. Mit Hilfe des Deanonymisierers kann die Bank Kunden- und Münztracing durchführen:

- *Kundentracing*: Anhand der Protokollansicht eines Einlöse-Protokolls kann die Bank mit Hilfe des Deanonymisierers die Identität des Kunden aufdecken, der die

vorliegende Münze abgehoben hat. Bei diesem Vorgang wird lediglich die Anonymität der vorliegenden Münze aufgehoben; die Anonymität aller anderen Münzen dieses Kunden ist nicht betroffen.

- *Münztracing*: Anhand der Protokollansicht eines Abhebe-Protokolls kann die Bank mit Hilfe des Deanonymisierers die in diesem Abhebe-Protokoll abgehobene Münze berechnen und somit das Ziel der entsprechenden Münze bestimmen bzw. die Münze beim Einlösen erkennen. Bei diesem Vorgang wird lediglich die Anonymität dieser einen Münze aufgehoben; die Anonymität aller anderen Münzen des betroffenen Kunden bleibt gewährleistet.

In gleicher Weise wie die Bank in einem (Offline-) elektronischen Geldsystem nicht in das Bezahl-Protokoll involviert ist, sollte auch der Denanonymisierer aus Effizienz- und Sicherheitsgründen lediglich an den beiden Tracing-Protokollen beteiligt sein; die übrigen Protokolle sollten in einem fairen Geldsystem auch weiterhin ohne den Deanonymisierer durchführbar sein (vgl. Abbildung 3.3). Die Abwesenheit des Deanonymisierers während der übrigen Protokolle impliziert gleichzeitig, dass der Deanonymisierer nicht in der Lage sein sollte, elektronische Münzen zu fälschen bzw. diese unerkannt mehrfach auszugeben. Die ersten in der Literatur vorgeschlagenen, fairen elektronischen Geldsysteme, wie zum Beispiel [BGK95], [CPS95] konnten diese Forderung zunächst nicht erfüllen. In dem von E. Brickell, P. Gemmel und D. Kravitz vorgeschlagenen System [BGK95], das lediglich Münztracing unterstützt, muss der Deanonymisierer auch am Abhebe-Protokoll beteiligt werden, um erfolgreiches Münztracing zu gewährleisten. Im Gegensatz dazu müssen sich in dem von J. Camenisch, J. Piveteau und M. Stadler entwickelten System [CPS95] alle Kunden zunächst bei dem Deanonymisierer registrieren. Eine weitere in [CPS95] entwickelte Variante kommt zwar auch ohne die Registrierung der Kunden aus, verwendet aber die sehr ineffiziente Cut-And-Choose-Technik.

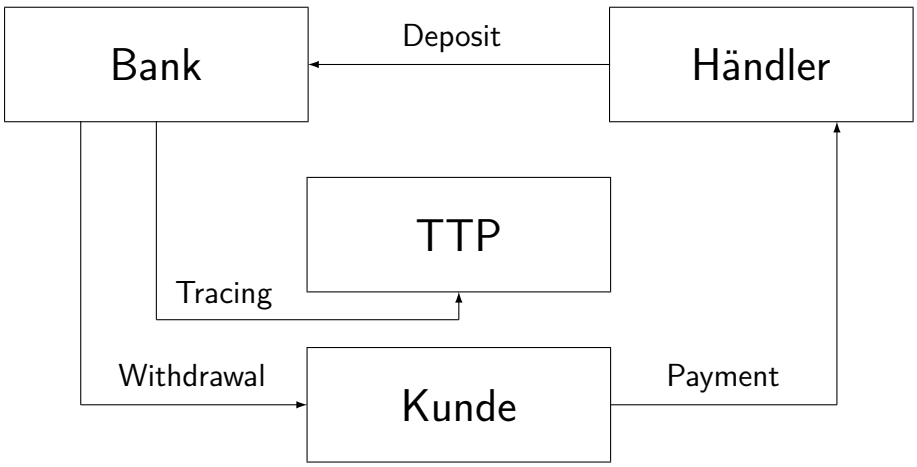

Abbildung 3.3: Das Modell fairer elektronischer Geldsysteme

Die ersten effizienten fairen elektronischen Geldsysteme, bei denen der Deanonymisierer nur an den beiden Tracing-Protokollen beteiligt ist, wurden von Y. Frankel, Y. Tsiounis

und M. Yung [FTY96] bzw. J. Camenisch, U. Maurer und M. Stadler [CMS97] vorgeschlagen. Wobei in [FTY96] das Konzept der *Indirect-Discourse-Proofs* entwickelt wurde, mit dem auf [Bra94] basierende Geldsysteme um Münz- bzw. Kundentracing erweitert werden können. Weitere effiziente faire Geldsysteme sind beispielsweise [FTY98], [ST98], [GT03], [NS03]. Formal ist ein faires elektronisches Geldsystem wie folgt definiert:

Definition 3.3.1 (**Faires elektronisches Geldsystem**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter. Ein *faires elektronisches (Offline-) Geldsystem* ist ein elektronisches Geldsystem (vgl. Definition 3.1.1) und besteht zusätzlich aus einer probabilistischen, polynomiellen interaktiven Turing-Maschine $\mathcal{D}$, dem Deanonymisierer, sowie den im Folgenden beschriebenen Algorithmen bzw. Protokollen. Wobei der Deanonymisierer ausschließlich an Kunden- und Münztracing partizipiert.

1. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{DKeyGen}$ des Deanonymisierers $\mathcal{D}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k und des öffentlichen Schlüssels $\mathsf{pk}_{\mathcal{B}}$ einen öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ und einen privaten Schlüssel $\mathsf{sk}_{\mathcal{D}}$ erzeugt. Die Ausgabe von $\mathsf{DKeyGen}(1^k, \mathsf{pk}_{\mathcal{B}})$ ist das Paar $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$.

2. Das *Kundentracing-Protokoll* KTrace ist ein interaktives Protokoll zwischen der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$. Es ist $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ die Eingabe der Bank $\mathcal{B}$ und $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ die Eingabe des Deanonymisierers $\mathcal{D}$. Die Ausgabe von $\mathcal{D}(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ ist $\mathsf{completed}$ oder $\mathsf{not\ completed}$. Die Ausgabe der Bank $\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)\right)$ ist die Identität $\mathrm{id}_{\mathcal{K}}$ eines Kunden $\mathcal{K}$ oder $\perp$.

3. Das *Münztracing-Protokoll* MTrace ist ein interaktives Protokoll zwischen der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$. Es ist $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ die Eingabe des Deanonymisierers $\mathcal{D}$ und $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ die Eingabe der Bank $\mathcal{B}$. Die Ausgabe von $\mathcal{D}(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ ist $\mathsf{completed}$ oder $\mathsf{not\ completed}$. Die Ausgabe der Bank $\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)\right)$ ist die in W abgehobene Münze Coin oder $\perp$.

Einige Bemerkungen

1. In Abschnitt 1.3 wurde schon auf den Zusammenhang zwischen fairen elektronischen Geldsystemen und Gruppensignaturverfahren hingewiesen. Auf Grund ihrer aufhebbaren Anonymität eignen sich Gruppensignaturen dazu, faire elektronische Geldsysteme zu konstruieren bzw. Kundentracing in bestehende elektronische Geldsysteme zu integrieren. Beispielsweise wurde 2001 von C. Boyd und G. Maitland [BM01] ein elektronisches Geldsystem mit Kundentracing vorgeschlagen, dass auf dem Gruppensignaturverfahren [ACJT00] aufbaut und später von C. Popescu [Pop04] zu einem fairen Geldsystem mit Münztracing erweitert wurde. Weitere Beispiele sind [Tra99], [CT03] und das von uns in [NS06] vorgeschlagene System, das in Kapitel 7 ausführlich vorgestellt wird.

2. Gegenüber dem Deanonymisierer ist die Anonymität der Kunden in einem fairen elektronischen Geldsystem nur gewährleistet, solange der Deanonymisierer seine

Befähigung nicht missbraucht und Münz- bzw. Kundentracing lediglich im dafür vorgesehenen rechtlichen Rahmen durchgeführt werden. Anonymität ist demnach mit einem vergleichsweise hohen Maß an Vertrauen der Kunden in den Deanonymisierer gleichzusetzen.

Im Jahr 2005 wurde von J. Camenisch, S. Hohenberger und A. Lysyanskaya [CHL05] ein elektronisches Geldsystem vorgestellt, das ein, dem Münztracing vergleichbares, Tracing-Verfahren ermöglicht, gleichzeitig aber ohne eine Trusted-Third-Party auskommt. In dem von J. Camenisch, S. Hohenberger und A. Lysyanskaya entwickelten System hebt ein Kunde während eines Abhebe-Protokolls n Münzen ab. Wird ein Kunde der Mehrfachausgabe einer dieser n Münzen überführt, so kann die Bank anhand der Identität des Kunden mit diesem Tracing-Verfahren alle n abgehobenen Münzen des Kunden berechnen.

3.3.1 Sicherheitsziele fairer elektronischer Geldsysteme

Ein faires elektronisches Geldsystem muss nicht nur die Sicherheitseigenschaften aus Definition 3.1.2 erfüllen. Gleichzeitig muss durch das System sichergestellt werden, dass die Bank bei begründetem Verdacht Münz- bzw. Kundentracing mit Hilfe des Deanonymisierers durchführen kann und dabei zweifelsfreie Ergebnisse erhält, d.h. Kunden bzw. Händler können nicht irrtümlich beschuldigt werden. Formal heißt ein faires elektronisches Geldsystem sicher, falls es folgender Definition genügt:

Definition 3.3.2 (**Sicherheit fairer elektronischer Geldsysteme**)**.** Ein faires elektronisches Geldsystem mit Sicherheitsparameter k heißt sicher, falls es ein gemäß Definition 3.1.2 sicheres elektronisches Geldsystem ist und zusätzlich folgende Eigenschaften erfüllt sind:

1. Ist $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ die Protokollansicht eines Einlöse-Protokolls D, bei dem $\mathcal{B}$ am Ende true ausgegeben hat, so gibt $\mathcal{B}$ bei Durchführung des Kundentracing-Protokolls KTrace mit einer in k überwältigenden Wahrscheinlichkeit am Ende die Identität $\mathrm{id}_{\mathcal{K}}$ des Kunden $\mathcal{K}$ aus, der die in $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ enthaltene Münze Coin bei $\mathcal{B}$ abgehoben hat.

2. Ist $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ die Protokollansicht eines Abhebe-Protokolls W, bei dem $\mathcal{B}$ am Ende $\mathsf{completed}$ ausgegeben hat, so gibt $\mathcal{B}$ bei Durchführung des Münztracing-Protokolls MTrace mit einer in k überwältigenden Wahrscheinlichkeit am Ende die in W abgehobene Münze Coin aus.

Wie bereits in Abschnitt 3.1.2 erwähnt, bieten faire elektronische Geldsysteme keine perfekte Anonymität, sondern können lediglich rechnerische Anonymität (vgl. Definition 3.1.2) erreichen. Dies beruht im Wesentlichen auf der Tatsache, dass der Deanonymisierer in einem fairen elektronischen Geldsystem nur an der Durchführung der beiden Tracing-Protokolle beteiligt ist und beim Abheben bzw. Bezahlen mit elektronischen Münzen offline ist. Der folgende Satz wurde im Jahre 1996 von Y. Frankel, Y. Tsiounis und M. Yung bewiesen.

Satz 3.3.1. Perfekte Anonymität des Kunden ist in einem fairen elektronischen Geldsystem nicht realisierbar. Auch wenn das System lediglich Münztracing oder Kundentracing unterstützt.

Beweis. siehe [FTY96]. □

3.3.2 Faire elektronische Geldbörsen

Grundsätzlich sind faire elektronische Geldsysteme auf die gleiche Weise mit dem Problem der Mehrfachausgabe elektronischer Münzen konfrontiert, wie die zu Beginn dieses Kapitels in Abschnitt 3.1 vorgestellten Systeme. Allerdings können auch faire Geldsysteme, mit dem in Abschnitt 3.2 beschriebenen Konzept der elektronischen Geldbörse, zu fairen elektronischen Geldbörsen erweitert werden, so dass die Mehrfachausgabe von Münzen nicht nur aufgedeckt, sondern auch im Voraus verhindert werden kann (vgl. Abbildung 3.4). In der Literatur sind solche Systeme nicht so häufig vertreten, Beispiele sind [ST98] und [NS03].

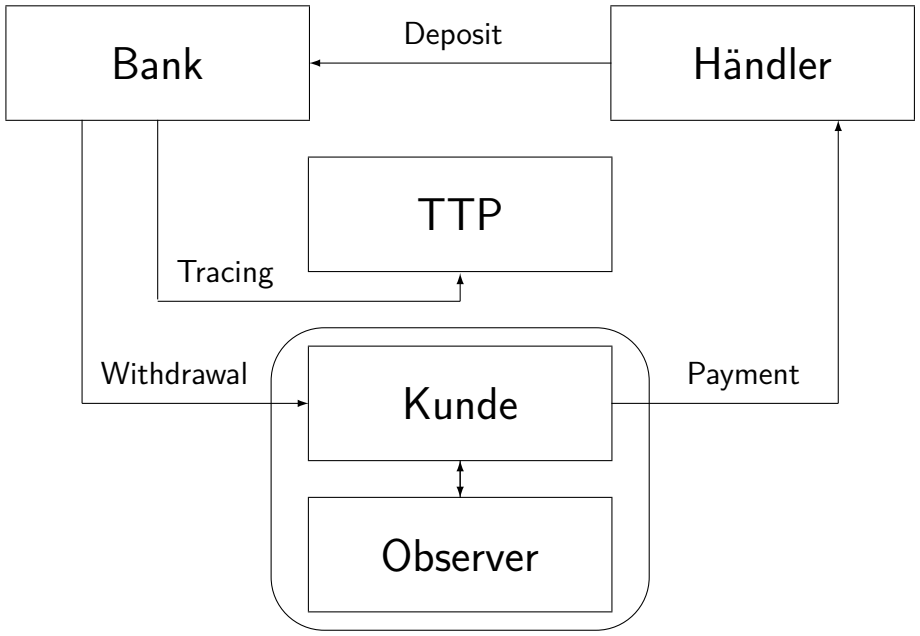

Abbildung 3.4: Das Konzept der fairen elektronischen Geldbörse

Kapitel 4

Kompromittierung elektronischer Geldsysteme

Bereits in Abschnitt 1.2 wurde das durch die steigende Nutzung moderner Kommunikations- und Informationstechnologien wachsende Risiko potenzieller Schäden durch Sicherheitslücken und Schadsoftware angesprochen.
Für die Sicherheit elektronischer bzw. fairer elektronischer Geldsysteme ist eine *sichere Aufbewahrung* des Signaturschlüssels der Bank respektive des privaten Schlüssels des Deanonymisierers (vgl. Abschnitt 3.3) entscheidend. Sicherheitslücken in den Systemen beider Institutionen, die für eine sichere Aufbewahrung der privaten Schlüssel sorgen, können zu einer *Kompromittierung* dieser führen. Im Zusammenhang mit elektronischen Geldsystemen, insbesondere auch den fairen Systemen, hat die potenzielle Gefahr einer solchen *Schlüsselkompromittierung* und die damit verbundenen Folgen in der Literatur bis heute wenig Beachtung gefunden.
Mit welchen möglichen Ansätzen man dieser Gefahr in fairen elektronischen Geldsystemen entgegen treten kann, soll in diesem Kapitel untersucht werden. Zunächst werden in Abschnitt 4.1 die weitreichenden Konsequenzen dargelegt, die eine Schlüsselkompromittierung in fairen elektronischen Geldsystemen mit sich bringt. Abschließend werden in Abschnitt 4.2 bereits bestehende Lösungsansätze für das Problem der Schlüsselkompromittierung aus dem Bereich der Public-Key-Kryptografie, wie zum Beispiel *Forward-Security*, *Key-Insulation* und *Intrusion-Resilience* kurz vorgestellt und unter dem Aspekt der Integrierbarkeit in elektronische Geldsysteme untersucht. Der Fokus liegt hier auf Systemen, die auf [Bra94] basieren, insbesondere [FTY98].

4.1 Die Folgen einer Schlüsselkompromittierung

Auch wenn Bank und Deanonymisierer ihre Systeme, die eine ungefährdete Aufbewahrung privater Schlüssel sicherstellen sollen, stets auf dem aktuellen Sicherheitsstand halten, sind Sicherheitslücken in ihren Systemen und somit auch eine Kompromittierung privater Schlüssel nicht gänzlich auszuschließen.

4.1.1 Kompromittierung der Bank

In einem elektronischen Geldsystem ist die Bank als geldausgebende Institution ein lohnendes Ziel möglicher Angreifer. Bei einer erfolgreichen Kompromittierung der Bank kann der Angreifer, der nun im Besitz des Signaturschlüssels der Bank ist, Münzen in Eigenregie erstellen bzw. fälschen. Infolgedessen sind die Sicherheitseigenschaften eines elektronischen Geldsystems (vgl. Definition 3.1.2) von diesem Zeitpunkt an nicht mehr gewährleistet. Auf Grund der Tatsache, dass ab diesem Augenblick weder die Bank noch die Händler die im Umlauf befindlichen *echten* Münzen von den *gefälschten* Münzen unterscheiden können, verlieren zwangsläufig alle elektronischen Münzen ihre Gültigkeit, die zum Zeitpunkt der Kompromittierung der Bank noch nicht wieder bei dieser eingelöst wurden. Der durch eine erfolgreiche Kompromittierung der Bank verursachte Schaden trifft also nicht nur die Bank, sondern vor allem Kunden und Händler.
Zusammengefasst hat die Kompromittierung der Bank für die einzelnen Parteien eines elektronischen Geldsystems daher folgende Konsequenzen:

- Die Bank kann nach der Kompromittierung ihres Signaturschlüssels keine weiteren Münzen ausstellen. Solange sie die Kompromittierung ihres Signaturschlüssels nicht bemerkt, wird sie auch von einem Angreifer gefälschte Münzen beim Einlösen akzeptieren und auf diese Weise Verluste erzielen.

- Kunden können ihre bereits abgehobenen, aber noch nicht wieder ausgegebenen Münzen nicht mehr als gültiges Zahlungsmittel verwenden. Denn weder die Bank, noch die Händler können die Münzen von gefälschten Münzen unterscheiden und werden sie somit nicht annehmen.

- Händler können sich den Wert der Münzen, die sie noch nicht bei der Bank eingelöst haben, nicht mehr durch die Bank gutschreiben lassen. Denn die Bank wird die möglicherweise gefälschten Münzen nicht annehmen.

4.1.2 Kompromittierung des Deanonymisierers

Anonymität ist sicherlich eines der charakteristischsten Sicherheitsmerkmale elektronischer Geldsysteme. Durch sie wird unter anderem die Analyse des Kaufverhaltens einzelner Kunden und die daraus resultierende Möglichkeit gezielter, kundenspezifischer Werbung erschwert. In einem fairen elektronischen Geldsystem kann die Anonymität jedoch durch einen Deanonymisierer aufgehoben werden. Das heißt, der Deanonymisierer ist in einem fairen Geldsystem neben der Bank ein weiteres lukratives Ziel potenzieller Angreifer.
Zwar ist die Kompromittierung des Deanonymisierers aus Sicht des Deanonymisierers, der Bank bzw. der Händler eher unbedeutend, da sich die Kompromittierung nicht auf für sie relevante Sicherheitseigenschaften, wie zum Beispiel Unfälschbarkeit, auswirkt. Aus Sicht der Kunden stellt sie allerdings eine große Gefahr für deren Privatsphäre dar. Da der Deanonymisierer in einem fairen Geldsystem für die Durchführung von Kunden- und Münztracing verantwortlich ist, kann die Anonymität der Kunden im

Falle einer erfolgreichen Kompromittierung des Deanonymisierers nicht mehr aufrecht erhalten werden. Für die Kunden kann dies folgende Konsequenzen haben:

- Kooperiert die Bank mit dem Angreifer, so kann die Bank:
  - zusammen mit dem Angreifer Kundentracing durchführen und alle ihr vorliegenden Münzen ihren Besitzern zuordnen. Die Bank bzw. der Angreifer sind somit in der Lage detaillierte Kundenprofile zu erstellen.
  - zusammen mit dem Angreifer Münztracing durchführen und kennt dann alle Münzen, die bei ihr abgehoben wurden. Dies ermöglicht der Bank bzw. dem Angreifer ebenfalls das Anfertigen umfangreicher Kundenprofile.

    Da Münztracing auch dazu gedacht ist, im Falle einer Erpressung die erpressten Münzen auf eine Sperrliste zu setzen, kann dies von der Bank bzw. dem Angreifer auch gezielt dafür genutzt werden, Münzen auf Sperrlisten zu setzen und den entsprechenden Kunden auf diese Weise nicht unerheblichen Schaden zuzufügen.
- Geht die Bank keine Kooperation mit dem Angreifer ein, so kann der Angreifer in jedem Fall Kundentracing durchführen und somit entscheiden, ob Münzen von ein und demselben Kunden abgehoben wurden. Durch Analyse des Kaufverhaltens kann er möglicherweise auf die Identitäten der Kunden schließen und daher ggf. personalisierte Analyse betreiben.

4.2 Mögliche Lösungsansätze

Der zweifelsohne am nächsten liegende Ansatz, die in Abschnitt 4.1 beschriebenen Konsequenzen einer Kompromittierung der Bank bzw. des Deanonymisierers zu minimieren und damit den Schaden der Kunden und Händler so gering wie möglich zu halten, bzw. die Privatsphäre der Kunden bestmöglich zu schützen, ist, die Schlüsselpaare der Bank und des Deanonymisierers in relativ kurzen und regelmäßigen Intervallen neu zu generieren und alle bereits abgelaufenen privaten Schlüssel zu löschen.
Durch diesen Ansatz verlieren bei einer Kompromittierung der Bank lediglich die mit dem von der Kompromittierung betroffenen Signaturschlüssel der Bank ausgestellten Münzen ihre Gültigkeit. Alle Münzen, die vor der Kompromittierung ausgestellt wurden, bleiben weiterhin gültig und können somit von den Kunden bzw. Händlern auch künftig ausgegeben und eingelöst werden. Aus dem gleichen Grund sind bei einer Kompromittierung des Deanonymisiers lediglich die von den kompromittierten privaten Schlüsseln betroffenen Münzen beeinträchtigt. Das heißt, die Anonymität der Kunden ist in den übrigen Intervallen nicht gefährdet.
Allerdings ist der Ansatz, in kurzen und regelmäßigen Intervallen neue Schlüsselpaare zu generieren, in der Praxis aus unterschiedlichen Gründen nicht rentabel:

- Durch die regelmäßige Generierung neuer Schlüsselpaare ist die Privatsphäre der Kunden zwar bestmöglich geschützt, in einem fairen elektronischen Geldsystem

bedeutet es aber gleichzeitig, dass der Deanonymisierer Kunden- und Münztracing nur für das aktuelle Intervall durchführen kann, da der Deanonymisierer die privaten Schlüssel bereits abgelaufener Intervalle bei Neugenerierung löscht.

- Die periodische Erzeugung neuer Schlüsselpaare ist sowohl für die Bank als auch für den Deanonymisierer mit Mehrkosten verbunden; denn beide Parteien müssen sich für alle neu generierten öffentlichen Schlüssel von einer Zertifizierungsinstanz neue Zertifikate ausstellen lassen und öffentliche Schlüssel sowie Zertifikate an alle Parteien des elektronischen Geldsystems verteilen.

- Zusätzlich müssen alle Parteien des Systems, um die Gültigkeit der ausgestellten Münzen überprüfen zu können, eine Liste aller, von der Bank und dem Deanonymisierer, ausgegebenen öffentlichen Schlüssel verwalten, die linear mit der Anzahl der Zeitintervalle wächst.

Um den Mehraufwand, der durch die Zertifizierung und Verteilung neuer öffentlicher Schlüssel entsteht, zu vermeiden, ist es wünschenswert, die öffentlichen Schlüssel der Bank und des Deanonymisierers während der Initialisierungsphase des Systems einmalig zu publizieren, von diesem Zeitpunkt an nicht mehr zu erneuern und lediglich die privaten Schlüssel der Bank und des Deanonymisierers in kurzen und regelmäßigen Intervallen neu zu generieren, damit auch weiterhin alle Parteien optimal vor den Risiken bzw. Schäden einer Schlüsselkompromittierung geschützt sind.

4.2.1 Forward-Security

Der im vorhergehenden Abschnitt angedeutete Ansatz, den privaten Schlüssel, bei einem einmalig herausgegebenen öffentlichen Schlüssel, in kurzen und regelmäßigen Intervallen zu erneuern, verbirgt sich hinter dem Konzept der so genannten *Forward-Security*, das von R. Anderson [And97], [And02] im Jahre 1997 für digitale Signaturverfahren vorgeschlagen und später durch M. Bellare und S. Miner [BM99] formalisiert wurde:

> Die Gültigkeit von Signaturen, die zu einem Zeitpunkt vor der Kompromittierung des Signaturschlüssels erstellt wurden, soll auch nach der Kompromittierung gewährleistet bleiben (vgl. Abbildung 4.1).

Forward-Security basiert auf folgender Idee: Jeder Teilnehmer eines digitalen Signaturverfahrens generiert sich einen öffentlichen Schlüssel pk und einen dazugehörigen ersten privaten Schlüssel sk_0. Die Laufzeit, für die der öffentliche Schlüssel pk ausgestellt wurde, wird in N Intervalle unterteilt. Während der öffentliche Schlüssel über seine gesamte Laufzeit unverändert bleibt, wird der private Schlüssel sk_i zu Beginn des i-ten Intervalls neu generiert. Der private Schlüssel sk_{i-1} des abgelaufenen Intervalls wird gelöscht. Damit durch eine Kompromittierung des Signaturschlüssels im i-ten Intervall die vorherigen Schlüssel $\mathsf{sk}_0, \ldots, \mathsf{sk}_{i-1}$ nicht kompromittiert werden, wird sk_i durch Anwenden einer Einwegfunktion auf sk_{i-1} berechnet. Im i-ten Intervall enthält die Signatur einer Nachricht zusätzlich den Index i des aktuellen Intervalls. Anhand des öffentlichen Schlüssels pk, der Nachricht und dem Index kann die Gültigkeit einer Signatur überprüft werden. In den

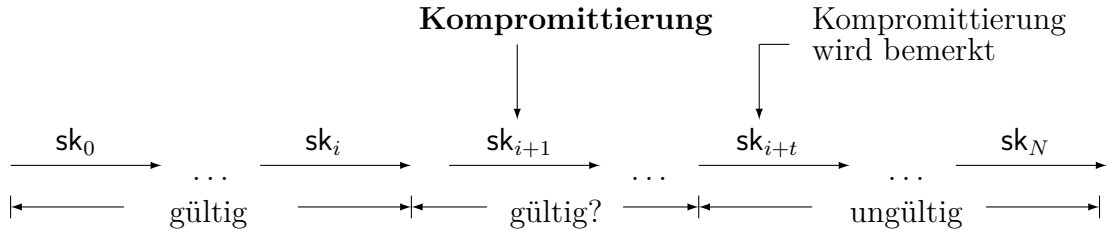

Abbildung 4.1: Das Konzept der Forward-Security

vergangenen Jahren wurden mehrere *digitale Signaturverfahren mit Forward-Security* vorgeschlagen, beispielsweise [BM99], [Kra00], [IR01], [KR01], [CK06]. Prinzipiell lassen sich digitale Signaturverfahren mit Forward-Security in zwei Klassen einteilen: Der ersten Klasse gehören Verfahren an, die das zugrunde liegende digitale Signaturverfahren als Blackbox verwenden (z.B. [BM99], [Kra00]), während die Verfahren der zweiten Klasse bestehende Signaturverfahren modifizieren (z.B. [BM99], [IR01], [CK06]).

Die Verfahren der ersten Klasse verwenden in der Regel einen öffentlichen Schlüssel pk um den im i-ten Intervall gültigen öffentlichen Schlüssel pk_i durch eine Zertifikatskette zu zertifizieren. Da alle Zertifikate und öffentliche Schlüssel gespeichert werden müssen und zur Verifikation einer Signatur die gesamte Zertifikatskette verifiziert werden muss, benötigen diese Verfahren einen höheren Rechenaufwand bzw. mehr Speicherplatz und sind somit weniger effizient als Verfahren der zweiten Klasse, die genauso effizient arbeiten können, wie das Signaturverfahren auf dem sie basieren (z.B. [IR01]). Verfahren, die das zugrunde liegende digitale Signaturverfahren als Blackbox verwenden, lassen sich prinzipiell auch auf blinde Signaturen und damit auch auf elektronische Geldsysteme, die auf blinden Signaturverfahren basieren, übertragen. Allerdings sind diese Verfahren für ein in der Praxis verwendetes elektronisches Geldsystem, in dem Münzen permanent auf ihre Echtheit überprüft werden müssen, auf Grund der bereits erwähnten Nachteile eher ungeeignet.

Wenig Beachtung innerhalb der zweiten Klasse fanden bisher die blinden Signaturverfahren, ein entscheidender Baustein elektronischer Geldsysteme. Das meines Wissens bisher einzige veröffentlichte *blinde Signaturverfahren mit Forward-Security* basiert auf [IR01], wurde im Jahre 2003 von D.N. Duc, J.H. Cheon und K. Kim veröffentlicht [DCK03], und eignet sich auf Grund der von F. Bao, R. Deng und S. Wang [BDW05] vorgestellten Angriffe nicht als Baustein eines sicheren elektronischen Geldsystems.

Zur Konstruktion digitaler Signaturverfahren mit Forward-Security der zweiten Klasse sind bis heute lediglich Methoden bekannt, deren Sicherheit auf dem Problem der Faktorisierung, wie beispielsweise der RSA- oder der starken RSA-Annahme (siehe z.B. [BNS05], [Mao04]), beruht. Signaturverfahren mit Forward-Security, die auf dem Diskreter-Logarithmus-Problem oder den Diffie-Hellman-Problemen basieren, sind nicht bekannt.

Folglich sind heute bekannte Techniken zur Entwicklung nicht generischer Signaturverfahren mit Forward-Security im Hinblick auf den Entwurf blinder Signaturverfahren bzw. elektronischer Geldsysteme mit Forward-Security, deren Sicherheit auf dem Diskreter-Logarithmus-Problem oder einem verwandten Problem beruht, nicht geeignet. Davon sind besonders die auf [Bra94] aufbauenden elektronischen Geldsysteme, wie [FTY98], betroffen, deren Sicherheit auf dem Darstellungsproblem in primen Untergruppen von $\mathbb{Z}_p^*$ beruht (vgl. Abschnitt 2.2.3).

4.2.2 Key-Insulation

Digitale Signaturverfahren mit Forward-Security bieten den Vorteil, dass Signaturen, die vor der Schlüsselkompromittierung erstellt wurden, ihre Gültigkeit auch nach einer Kompromittierung nicht verlieren. *Zukünftige Intervalle* sind aber durch das Konzept der Forward-Security nicht geschützt, d.h. die Sicherheit eines digitalen Signaturverfahrens mit Forward-Security ist für alle auf die Kompromittierung folgenden Intervalle nicht mehr gewährleistet.
Um auch zukünftige Intervalle vor einer Kompromittierung zu schützen, wurde von Y. Dodis, J. Katz, S. Xu und M. Yung im Jahre 2002 das Konzept der so genannten *Key-Insulation* [DKXY02] vorgeschlagen. Analog zu Verfahren mit Forward-Security wird auch hier die Laufzeit des öffentlichen Schlüssels pk in N Intervalle unterteilt und der private Schlüssel sk_i zu Beginn des i-ten Intervalls bei gleichbleibendem öffentlichen Schlüssel pk neu generiert. Im Unterschied zu den Verfahren mit Forward-Security können bis zu t der N Intervalle ($t < N$) von einem Angreifer kompromittiert werden (vgl. Abbildung 4.2). Die privaten Schlüssel der restlichen $n - t$ Zeitintervalle sind von der Kompromittierung nicht betroffen. Somit wirkt sich eine Kompromittierung insbesondere nicht auf die Sicherheit der privaten Schlüssel zukünftiger Intervalle aus.

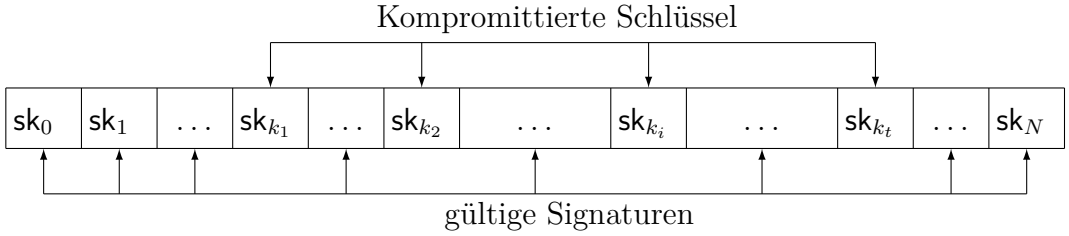

Abbildung 4.2: Das Konzept der Key-Insulation

Diese stärkere Form der Sicherheit wird dadurch erreicht, dass der für das i-te Intervall benötigte private Schlüssel sk_i mit sk_{i-1} und Hilfe eines, auf einem manipulationssicheren Medium gespeicherten, *Generalschlüssels* sk^* berechnet wird. Selbst wenn der Generalschlüssel sk^* kompromittiert wird, ist die Sicherheit für alle der n Intervalle garantiert, solange keiner der übrigen privaten Schlüssel kompromittiert wird.
Zunächst wurde Key-Insulation von Y. Dodis *et al.* [DKXY02] im Jahre 2002 für Public-Key-Verschlüsselungsverfahren vorgeschlagen. Neben einem allgemeinen Konstruktions-

prinzip wurde in [DKXY02] ein effizientes, auf dem modifizierten ElGamal-Verschlüsselungsverfahren (vgl. Abschnitt 2.3.3) basierendes, (t, N)-Key-Insulated-Public-Key-Verschlüsselungsverfahren (vgl. Abschnitt 5.1) vorgestellt. Von den gleichen Autoren wurde das Konzept der Key-Insulation im Jahre 2003 auf digitale Signaturverfahren übertragen [DKXY03] und ein allgemeines Konstruktionsprinzip sowie ein (t, N)-Key-Insulated-Signaturverfahren vorgeschlagen, dessen Sicherheit auf der Diskreter-Logarithmus-Annahme basiert.
Gegenüber den nicht generischen Signaturverfahren mit Forward-Security bietet die in [DKXY03] zur Konstruktion des (t, N)-Key-Insulated-Signaturverfahren genutzte Technik den Vorteil, dass sie sich prinzipiell auch auf Signaturverfahren bzw. Geldsysteme übertragen lässt, die auf dem Diskreter-Logarithmus-Problem basieren. Insbesondere auf alle elektronischen Geldsysteme, die auf [Bra94] aufbauen.
Um in einem fairen elektronischen Geldsystem nicht nur der Bank einen verbesserten Schutz gegen die in Abschnitt 4.1.1 dargestellten Konsequenzen einer Kompromittierung zu bieten, sondern auch die Kunden vor den in Abschnitt 4.1.2 aufgeführten Folgen einer Kompromittierung des Deanonymisierers zu schützen, müssen auch die privaten Schlüssel des Deanonymisierers entsprechend abgesichert werden. Auch hier ist das (t, N)-Key-Insulated-Public-Key-Verschlüsselungsverfahren [DKXY02] ein geeigneter Kandidat zur Integration in [FTY98], da es auf der modifizierten ElGamal-Verschlüsselung (vgl. Abschnitt 2.3.3) aufbaut, während [FTY98] die ElGamal-Verschlüsselung einsetzt. Gleichzeitig bietet [DKXY02] die Möglichkeit zu jedem Zeitpunkt die privaten Schlüssel beliebiger anderer Intervalle zu berechnen, was für eine erfolgreiche Durchführung des Kunden- bzw. Münztracings von großer Bedeutung ist.

4.2.3 Weitere Ansätze

Einen noch stärkeren Schutz vor einer Kompromittierung als Forward-Security bzw. Key-Insulation bieten die von G. Itkis und L. Reyzin [IR02], [Itk01] entwickelten Signaturen mit *Intrusion-Resilience*, die die Vorteile beider Konzepte miteinander vereinen. Im Gegensatz zur Key-Insulation wird der Generalschlüssel zu Beginn jedes Intervalls neu generiert und sowohl privater Schlüssel als auch Generalschlüssel können innerhalb eines Intervalls neu generiert werden. Die Kompromittierung beider Schlüssel wirkt sich nicht auf die Sicherheit des Verfahrens aus, solange nicht beide im gleichen Intervall kompromittiert werden. In diesem Fall ist weiterhin Forward-Security garantiert.
Während das in [Itk01] entworfene Signaturverfahren mit Intrusion-Resilience ein generisches Verfahren darstellt, basiert das in [IR02] vorgeschlagene Signaturverfahren auf dem Signaturverfahren mit Forward-Security [IR01]. Diese Verfahren sind aus den gleichen Gründen, die bereits in Abschnitt 4.2.1 diskutiert wurden, zur Integration in [FTY98] nicht verwendbar.
Im Vergleich zu Konzepten wie Forward-Security, Key-Insulation oder Intrusion-Resilience, die zur „Verbesserung der Sicherheit“ privater Schlüssel diese in kurzen zeitlichen Abständen erneuern, verfolgen *Schwellenverfahren* (engl. *threshold cryptography*) wie beispielsweise [Sha79], [DF89],[LHL94] einen grundlegend anderen Ansatz. Anstatt einen privaten Schlüssel regelmäßig zu ändern, wird dieser in einem (t, n)-Schwellenverfahren

unter n Parteien aufgeteilt, so dass t oder mehr der Parteien den privaten Schlüssel gemeinsam wieder rekonstruieren können.
Ein enormer Nachteil der (t, n)-Schwellenverfahren ist, dass an der Durchführung solcher Verfahren, um beispielsweise eine Signatur zu erstellen, mindestens t der n Parteien beteiligt sein müssen. Bedenkt man, dass in einem elektronischen Geldsystem von der Bank permanent elektronische Münzen ausgegeben werden, so ist es äußerst ineffizient, das Abhebe-Protokoll zwischen Kunde und Bank als (t, n)-Schwellenverfahren zu entwerfen, an dem mindestens t Rechner der Bank beteiligt sein müssen. Auf Grund ihres zu hohen Kommunikationsaufwands sind Schwellenverfahren daher für elektronische Geldsysteme weniger praktikabel. Des Weiteren erhöht sich durch das Aufteilen des privaten Schlüssels auf n Parteien zusätzlich das Risiko durch mögliche Sicherheitslücken in den Systemen der n Parteien doch von einer Kompromittierung des privaten Schlüssels betroffen zu sein.

Kapitel 5

Langfristig sichere elektronische Geldsysteme

Ziel dieses Kapitels ist es, ein formales Sicherheitsmodell für faire elektronische Geldsysteme zu entwerfen, das den bereits in Kapitel 4 dargelegten Risiken einer Schlüsselkompromittierung Rechnung trägt.
In Abschnitt 5.1 werden zunächst das zuvor in Abschnitt 4.2.2 angesprochene Modell der Key-Insulated Public-Key-Verschlüsselung [DKXY02], sowie das in [DKXY02] entworfene Key-Insulated Verschlüsselungsverfahren vorgestellt, das in Kapitel 6 zur Konstruktion eines neuen fairen elektronischen Geldsystems verwendet wird.
Da zahlreiche Sicherheitseigenschaften der in dieser Arbeitet betrachteten elektronischen Geldsysteme, wie beispielsweise Nichterweiterbarkeit, Unfälschbarkeit und Anonymität, auf der Sicherheit des zugrunde liegenden blinden Signaturverfahrens beruhen, wird in Abschnitt 5.2 zunächst auf Basis des Konzepts der Key-Insulation das Modell *langfristig sicherer* blinder Signaturverfahren entwickelt. Anschließend wird in den darauffolgenden Abschnitten 5.3 und 5.4 das Modell der *langfristig sicheren* bzw. *langfristig sicheren fairen* elektronischen Geldsysteme definiert. Neben der Formalisierung der neuen Sicherheitsmodelle werden in den Abschnitten 5.2 und 5.3 außerdem generische Verfahren zur Konstruktion langfristig sicherer blinder Signaturverfahren bzw. langfristig sicherer elektronischer Geldsysteme vorgestellt. Abschnitt 5.2 beinhaltet zusätzlich eine langfristig sichere Variante der blinden Schnorr-Signatur, die die Grundlage für die in Kapitel 6 neu entworfenen langfristig sicheren elektronischen Geldsysteme bildet.

5.1 Key-Insulated Public-Key-Verschlüsselung

Die grundlegende Idee des Modells der Key-Insulation ist es, den privaten Schlüssel mit Hilfe eines manipulationssicheren Speichermediums zu Beginn jedes neuen Intervalls zu aktualisieren (vgl. Abschnitt 4.2.2). Ein solches Verschlüsselungsverfahren bezeichnet man als *aktualisierbares Public-Key-Verschlüsselungsverfahren* (engl. key-updating public-key encryption scheme) [DKXY02], wobei sich die Aktualisierbarkeit in diesem Zusammenhang auf die Möglichkeit bezieht, den privaten Schlüssel zu aktualisieren. Formal ist ein aktualisierbares Public-Key-Verschlüsselungsverfahren wie folgt definiert:

Definition 5.1.1 (**Aktualisierbare Public-Key-Verschlüsselungsverfahren**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter und M die Menge aller Klartexte. Ein *aktualisierbares Public-Key-Verschlüsselungsverfahren* ist ein 5-Tupel polynomieller Algorithmen $(\mathsf{Gen}, \mathsf{KeyUpdate}^*, \mathsf{KeyUpdate}, \mathsf{Enc}, \mathsf{Dec})$ mit folgenden Eigenschaften:

1. Der *Schlüsselgenerierungsalgorithmus* Gen ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameter 1^k und der Anzahl der Zeitintervalle N einen öffentlichen Schlüssel pk, einen *Generalschlüssel* sk^* und einen ersten privaten Schlüssel sk_0 erzeugt. Die Ausgabe von $\mathsf{Gen}(1^k, N)$ ist das Tripel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0)$.

2. Der *Schlüsselaktualisierungsalgorithmus* $\mathsf{KeyUpdate}^*$ ist ein deterministischer Algorithmus, der bei Eingabe eines Index i $(1 \leq i \leq N)$ und des Generalschlüssels sk^* einen *temporären privaten Schlüssel* sk'_i für das Intervall i erzeugt. Die Ausgabe von $\mathsf{KeyUpdate}^*(i, \mathsf{sk}^*)$ ist sk'_i.

3. Der *Schlüsselaktualisierungsalgorithmus* $\mathsf{KeyUpdate}$ ist ein deterministischer Algorithmus, der bei Eingabe eines Index i, eines privaten Schlüssels sk_{i-1} und eines temporären privaten Schlüssels sk'_i den privaten Schlüssel sk_i für das i-te Intervall erzeugt und den privaten Schlüssel sk_{i-1} löscht. Die Ausgabe von $\mathsf{KeyUpdate}(i, \mathsf{sk}_{i-1}, \mathsf{sk}'_i)$ ist sk_i.

4. Der *Verschlüsselungsalgorithmus* Enc ist ein probabilistischer Algorithmus, der bei Eingabe eines Index i, eines öffentlichen Schlüssels pk und einer Nachricht $m \in M$ einen Geheimtext $\langle i, C \rangle$ erzeugt. Die Ausgabe von $\mathsf{Enc}(i, \mathsf{pk}, m)$ ist der Geheimtext $\langle i, C \rangle$.

5. Der *Entschlüsselungsalgorithmus* Dec ist ein deterministischer Algorithmus, der bei Eingabe eines Geheimtextes $\langle i, C \rangle$ und eines privaten Schlüssels sk_i eine Nachricht m oder $\perp$ ausgibt.

Dabei gilt für alle Nachrichten $m \in M$, dass $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(i, \mathsf{pk}, m)) = m$ ist.

Wie bereits erwähnt, wird der Generalschlüssel sk^* auf einem manipulationssicheren Medium gespeichert. Um den für das i-te Intervall gültigen privaten Schlüssel sk_i zu berechnen, lässt sich der Anwender zu Beginn des i-ten Intervalls den temporären privaten Schlüssel sk'_i durch den Algorithmus $\mathsf{KeyUpdate}^*(i, \mathsf{sk}^*)$ berechnen, der auf dem manipulationssicheren Medium ausgeführt wird. Anschließend kann er sich mit den beiden Schlüsseln sk_{i-1} und sk'_i den aktuellen privaten Schlüssel $\mathsf{sk}_i = \mathsf{KeyUpdate}(i, \mathsf{sk}_{i-1}, \mathsf{sk}'_i)$ berechnen, mit dem er im i-ten Intervall verschlüsselte Nachrichten wieder entschlüsseln kann.
Kann der Anwender mit den beiden Algorithmen $\mathsf{KeyUpdate}^*$ und $\mathsf{KeyUpdate}$ nicht nur die privaten Schlüssel zweier aufeinanderfolgender Intervalle aktualisieren, sondern die Aktualisierung für zwei beliebige Intervalle i, j durchführen, so spricht man von einer *direkten Schlüsselaktualisierung*. Die beiden Algorithmen erhalten dann als Eingabe das Paar (i, j), so dass $\mathsf{KeyUpdate}^*((i, j), \mathsf{sk}^*)$ den temporären privaten Schlüssel $\mathsf{sk}'_{i,j}$ und $\mathsf{KeyUpdate}((i, j), \mathsf{sk}_i, \mathsf{sk}'_{i,j})$ den privaten Schlüssel sk_j ausgibt.

Angriffe

Kompromittiert ein Angreifer einen Anwender, so ist der Angreifer nach seinem Angriff im Besitz dessen privaten Schlüssels sk_i. Im Weiteren wird dabei zwischen den folgenden drei Angriffen unterschieden:

- *Schlüssel-Angriff* (engl. *key exposure*): Der Angreifer kompromittiert den Anwender und verfügt über dessen privaten Schlüssel sk_i.
- *Temporärer Schlüssel-Angriff*: Der Angreifer kompromittiert den Anwender während der Schlüsselaktualisierung und verfügt über die beiden Schlüssel $\mathsf{sk}_{i-1}, \mathsf{sk}'_i$.
- *Generalschlüssel-Angriff*: Der Angreifer kompromittiert das manipulationssichere Speichermedium und verfügt über den Generalschlüssel sk^*.

Ein aktualisierbares Public-Key-Verschlüsselungsverfahren soll den Anwender vor genau diesen drei Angriffstypen schützen. Um diese Angriffe formal zu beschreiben, erhält der Angreifer Zugriff auf die im Folgenden beschriebenen zwei bzw. drei Orakel; je nachdem, ob der Angreifer einen Angriff mit gewählten Klartexten oder einen Angriff mit gewählten Geheimtexten durchführt (vgl. Abschnitt 2.3):

- Das *Schlüssel-Orakel* $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ modelliert die Kompromittierung eines privaten Schlüssels sk_i. Bei Eingabe eines Index i gibt $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}(i)$ den im i-ten Intervall gültigen privaten Schlüssel sk_i aus.
- Das *Links-Rechts-Orakel* $\mathsf{LR}_{\mathsf{pk},\mathbf{b}}$ (siehe z.B. [BDJR97]) modelliert die Verschlüsselungsanfragen des Angreifers. Dabei ist $\mathbf{b} = b_1 \dots b_N \in \{0,1\}^N$ und das Orakel ist definiert durch $\mathsf{LR}_{\mathsf{pk},\mathbf{b}}(i, m_0, m_1) := \mathsf{Enc}(i, \mathsf{pk}, m_{b_i})$.
- Das *Entschlüsselungs-Orakel* $\mathsf{Dec}^*_{\mathsf{sk}^*,\mathsf{sk}_0}$ modelliert die Entschlüsselungsanfragen eines Angreifers. Das Orakel ist definiert durch $\mathsf{Dec}^*_{\mathsf{sk}^*,\mathsf{sk}_0}(\langle i, C\rangle) := \mathsf{Dec}(\mathsf{sk}_i, \langle i, C\rangle)$.

Die Anfragen an die beiden Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ und $\mathsf{LR}_{\mathsf{pk},\mathbf{b}}$ darf der Angreifer miteinander verknüpfen, d.h. er kann insbesondere die Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ adaptiv und in beliebiger Reihenfolge stellen.

Sicherheit

Bei einem Schlüssel-Angriff wird die Sicherheit eines aktualisierbaren Public-Key-Verschlüsselungsverfahrens dadurch beschrieben, dass der Vektor $\mathbf{b}$ für das $\mathsf{LR}_{\mathsf{pk},\mathbf{b}}$-Orakel zufällig gewählt wird und der Angreifer erfolgreich ist, falls er das Bit b_i für ein von ihm nicht kompromittiertes Intervall korrekt bestimmen kann. Das Verfahren heißt sicher, falls die Erfolgswahrscheinlichkeit des Angreifers nicht wesentlich besser ist als zu raten.

Definition 5.1.2. Sei $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}^*, \mathsf{KeyUpdate}, \mathsf{Enc}, \mathsf{Dec})$ ein aktualisierbares Public-Key-Verschlüsselungsverfahren. Für einen Angreifer $\mathcal{A}$ sei

$$\begin{aligned}\mathsf{Succ}_{\mathcal{A},\Pi}(k) \;:=\; & \mathbf{P}[(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^k, N); \mathbf{b} \leftarrow \{0,1\}^N; \\ & (i, b') \leftarrow \mathcal{A}^{\mathsf{LR}_{\mathsf{pk},\mathbf{b}}(\cdot,\cdot,\cdot),\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}(\cdot),O(\cdot)}(\mathsf{pk}) : b' = b_i],\end{aligned}$$

wobei i niemals an Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}(\cdot)$ übergeben wurde. Für den Fall, dass $\mathcal{A}$ einen Angriff mit gewählten Klartexten durchführt, sei $O(\cdot) = \perp$. Falls $\mathcal{A}$ einen Angriff mit gewählten Geheimtexten durchführt, sei $O(\cdot) = \mathsf{Dec}^*_{\mathsf{sk}^*,\mathsf{sk}_0}(\cdot)$.
Das Verfahren Π heißt (t, N)-*Key-Insulated Public-Key-Verschlüsselungsverfahren*, falls $|\mathsf{Succ}_{\mathcal{A},\Pi}(k) - \frac{1}{2}|$ für jeden effizienten Angreifer $\mathcal{A}$, der höchstens t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellt, in k vernachlässigbar ist.

Ein möglicher temporärer Schlüssel-Angriff wird in Definition 5.1.2 nicht berücksichtigt. Wird ein solcher Angriff erfolgreich durchgeführt, kann der Angreifer anhand von sk_{i-1} und sk'_i den privaten Schlüssel sk_i berechnen. Das Verfahren verfügt über eine *sichere Schlüsselaktualisierung*, falls der Angriff nicht mehr Information gewinnt als ein Angreifer, der in den Intervallen $i-1$ und i einen Schlüssel-Angriff durchführt.

Definition 5.1.3. Ein aktualisierbares Public-Key-Verschlüsselungsverfahren Π hat eine *sichere Schlüsselaktualisierung*, falls die Sicht eines Angreifers $\mathcal{A}$, der im Intervall i einen temporären Schlüssel-Angriff durchführt, durch einen Angreifer $\mathcal{A}'$ perfekt simuliert werden kann, der einen Schlüssel-Angriff in den Intervallen $i-1$ und i durchführt.

Analog zu Definition 5.1.2 definiert man die Sicherheit für ein (t, N)-Key-Insulated Public-Key-Verschlüsselungsverfahren, falls vom Angreifer ein Generalschlüssel-Angriff durchgeführt wird und er somit über den Generalschlüssel sk^* verfügt.

Definition 5.1.4. Sei $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}^*, \mathsf{KeyUpdate}, \mathsf{Enc}, \mathsf{Dec})$ ein (t, N)-Key-Insulated Public-Key-Verschlüsselungsverfahren. Für einen Angreifer $\mathcal{A}$ sei

$$\begin{aligned}\mathsf{Succ}_{\mathcal{A},\Pi}(k) \;:=\; & \mathbf{P}[(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^k, N); b \leftarrow \{0,1\}^N; \\ & (i, b') \leftarrow \mathcal{A}^{\mathsf{LR}_{\mathsf{pk},b}(\cdot,\cdot,\cdot),O(\cdot)}(\mathsf{pk}, \mathsf{sk}^*) : b' = b_i],\end{aligned}$$

Für den Fall, dass $\mathcal{A}$ einen Angriff mit gewählten Klartexten durchführt, sei $O(\cdot) = \perp$. Falls $\mathcal{A}$ einen Angriff mit gewählten Geheimtexten durchführt, sei $O(\cdot) = \mathsf{Dec}^*_{\mathsf{sk}^*,\mathsf{sk}_0}(\cdot)$.
Das Verfahren Π heißt *starkes* (t, N)-*Key-Insulated Public-Key-Verschlüsselungsverfahren*, falls $|\mathsf{Succ}_{\mathcal{A},\Pi}(k) - \frac{1}{2}|$ für jeden effizienten Angreifer $\mathcal{A}$ in k vernachlässigbar ist.

Zu beachten ist, dass der Angreifer lediglich einen Generalschlüssel-Angriff durchführen darf. Denn falls der Angreifer sowohl einen Schlüssel- als auch einen Generalschlüssel-Angriff durchführen kann, verfügt er über den Generalschlüssel sk^* und einen privaten Schlüssel sk_i und kann somit für alle weiteren Intervalle j $(j > i)$ die privaten Schlüssel sk_j berechnen.

5.1.1 Ein konkretes Verfahren

Auf der modifizierten ElGamal-Verschlüsselung (vgl. Abschnitt 2.3.3) aufbauend, wurde von Y. Dodis *et al.* ein starkes (t, N)-Key-Insulated-Public-Key-Verschlüsselungsverfahren [DKXY02] entworfen, dessen Sicherheit auf der Decisional-Diffie-Hellman-Annahme beruht. Dieses Verfahren wird in den in Kapitel 6 entworfenen neuen, fairen elektronischen Geldsystemen dazu eingesetzt, der Bank Kunden- und Münztracing in Kooperation mit einem Deanonymisierer zu ermöglichen.

Schlüsselgenerierung: Bei Eingabe des Sicherheitsparameters 1^k und der Anzahl der Intervalle $N \in \mathbb{N}$ geht der Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k, N)$ wie folgt vor:

Schritt 1: Wähle zufällig zwei Primzahlen p, q mit $|q| = k$, $p = 2q + 1$ und bestimme zwei Generatoren g, h der Untergruppe $G_q \subset \mathbb{Z}_p^*$ mit $|G| = q$.

Schritt 2: Wähle $x_0^*, y_0^*, \ldots, x_t^*, y_t^* \in_{\mathcal{R}} \mathbb{Z}_q$. Bestimme die Polynome $f_x(\mu) = \sum_{i=0}^{t} x_i^* \mu^i$, $f_y(\mu) = \sum_{i=0}^{t} y_i^* \mu^i$ und berechne $z_i^* = g^{x_i^*} h^{y_i^*}$ für $i = 0, \ldots, t$ und $t < N$.

Die Ausgabe des Algorithmus $\mathsf{Gen}(1^k, N)$ ist das Tripel

$$(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = ((g, h, z_0^*, \ldots, z_t^*), (x_1^*, y_1^*, \ldots, x_t^*, y_t^*), (x_0^*, y_0^*)).$$

Schlüsselaktualisierung: Bei Eingabe der Indizes i, j und des Generalschlüssels sk^* berechnet der Algorithmus $\mathsf{KeyUpdate}^*\left((i, j), \mathsf{sk}^*\right)$ die Werte $x'_{i,j} = \sum_{k=1}^{t} x_k^*(j^k - i^k)$ und $y'_{i,j} = \sum_{k=1}^{t} y_k^*(j^k - i^k)$ und gibt $\mathsf{sk}'_{i,j} = (x'_{i,j}, y'_{i,j})$ aus.
Der Algorithmus $\mathsf{KeyUpdate}\left((i, j), \mathsf{sk}_i, \mathsf{sk}'_{i,j}\right)$ berechnet bei Eingabe der Indizes i, j, des privaten Schlüssels $\mathsf{sk}_i = (x_i, y_i) =: (f_x(i), f_y(i))$ und des temporären Schlüssels $\mathsf{sk}'_{i,j}$ die Werte $x_j = x_i + x'_{i,j}$ und $y_j = y_i + y'_{i,j}$ und gibt $\mathsf{sk}_j = (x_j, y_j) = (f_x(j), f_y(j))$ aus.

Verschlüsseln: Bei Eingabe eines Index i, des öffentlichen Schlüssels pk und der zu verschlüsselnden Nachricht m berechnet $\mathsf{Enc}(i, \mathsf{pk}, m)$ zunächst $z_i = \prod_{j=0}^{t} (z_j^*)^{i^j}$, wählt anschließend zufällig eine Zahl $r \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $C = (g^r, h^r, z_i^r m)$. Die Ausgabe von $\mathsf{Enc}(i, \mathsf{pk}, m)$ ist der Geheimtext $\langle i, C \rangle$.

Entschlüsseln: Der Entschlüsselungsalgorithmus $\mathsf{Dec}\left(\mathsf{sk}_i, \langle i, C \rangle\right)$ erhält als Eingabe den privaten Schlüssel sk_i, einen Geheimtext $\langle i, C \rangle$ mit $C = (u, v, w)$ und berechnet $m = wu^{-x_i} v^{-y_i}$. Die Ausgabe von $\mathsf{Dec}\left(\mathsf{sk}_i, \langle i, C \rangle\right)$ ist der Klartext m.

Das in diesem Abschnitt beschriebene aktualisierbare Public-Key-Verschlüsselungsverfahren ist sicher im Sinne von Definition 5.1.2 und 5.1.3.

Satz 5.1.1. Unter der Decisional-Diffie-Hellman-Annahme ist das vorgestellte aktualisierbare Public-Key-Verschlüsselungsverfahren unter einem Angriff mit gewählten Klartexten ein starkes (t, N)-Key-Insulated Public-Key-Verschlüsselungsverfahren mit sicherer Schlüsselaktualisierung.

Beweis. siehe [DKXY02]. □

5.2 Langfristig sichere blinde Signaturverfahren

Die Sicherheit elektronischer Geldsysteme beruht, wie bereits zu Beginn dieses Kapitels angesprochen, im Wesentlichen auf dem blinden Signaturverfahren, das dem Geldsystem zugrunde liegt. Aus diesem Grund wird in diesem Abschnitt das im vorangehenden Absatz beschriebene Konzept der aktualisierbaren Public-Key-Verschlüsselung, das in [DKXY03] auch für digitale Signaturverfahren vorgeschlagen wurde, aufgegriffen und auf blinde Signaturverfahren übertragen. Die neu formalisierten *aktualisierbaren blinden Signaturverfahren* können als grundlegender Baustein elektronischer Geldsysteme genutzt werden.
Aufbauend auf Definition 2.6.1 ist ein aktualisierbares blindes Signaturverfahren formal wie folgt definiert:

Definition 5.2.1 (**Aktualisierbare blinde Signaturverfahren**)**.** Es sei $k \in \mathbb{N}$, M die Menge aller Nachrichten und $N \in \mathbb{N}$ die polynomiell in k beschränkte Anzahl der Intervalle. Gegeben seien die folgenden polynomiellen Algorithmen und Protokolle:

1. Der *Schlüsselgenerierungsalgorithmus* Gen ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k und der Anzahl N der Intervalle einen öffentlichen Schlüssel pk, einen Generalschlüssel sk^* und einen ersten privaten Schlüssel sk_0 erzeugt. Die Ausgabe von $\mathsf{Gen}(1^k, N)$ ist das Tripel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0)$.

2. Das *Schlüsselaktualisierungsprotokoll* $\mathsf{KeyUpdate}$ ist ein interaktives Protokoll zwischen den probabilistischen, polynomiellen interaktiven Turing-Maschinen $\mathcal{S}$ und $\mathcal{T}$. Für den Generalschlüssel sk^*, die Indizes der Intervalle i, j und den im i-ten Intervall gültigen privaten Schlüssel sk_i ist i, j, sk_i die Eingabe des Signierers $\mathcal{S}$ und sk^* die Eingabe des manipulationssicheren Mediums $\mathcal{T}$. Die Ausgabe von $\mathcal{S}(i, j, \mathsf{sk}_i)$ ist der im j-ten Intervall gültige private Schlüssel sk_j oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{T}(\mathsf{sk}^*)$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

3. Das *Signaturprotokoll* Sign ist ein interaktives Protokoll zwischen den probabilistischen, polynomiellen interaktiven Turing-Maschinen $\mathcal{S}$ und $\mathcal{R}$, dessen Rundenzahl polynomiell in k ist. Für eine Nachricht $m \in M$ und den im i-ten Intervall gültigen privaten Schlüssel sk_i ist $i, \mathsf{pk}, \mathsf{sk}_i$ die Eingabe des Signierers $\mathcal{S}$ und i, pk, m die Eingabe des Empfängers $\mathcal{R}$. Die Ausgabe von $\mathcal{R}(i, \mathsf{pk}, m)$ ist die Signatur $\langle i, s\rangle$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{S}(i, \mathsf{pk}, \mathsf{sk}_i)$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

4. Der *Verifikationsalgorithmus* Verify ist ein deterministischer Algorithmus und erhält als Eingabe eine Nachricht $m \in M$, den öffentlichen Schlüssel pk sowie eine Signatur $\langle i, s\rangle$. Die Ausgabe von $\mathsf{Verify}(m, \mathsf{pk}, \langle i, s\rangle)$ ist der Wert 1 (= true) oder 0 (= false).

Das Tupel $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}, \mathsf{Sign}, \mathsf{Verify})$ heißt *aktualisierbares blindes Signaturverfahren*, falls folgende Bedingung erfüllt ist: Folgen $\mathcal{S}$ und $\mathcal{R}$ dem Signaturprotokoll, so erhält $\mathcal{R}$ als Ausgabe $\langle i, s\rangle$ und es gilt

$$\mathsf{Verify}(m, \mathsf{pk}, \langle i, s\rangle) = 1.$$

Analog zu den aktualisierbaren Public-Key-Verfahren [DKXY02], [DKXY03] wird der Generalschlüssel sk^* auf einem manipulationssicheren Medium gespeichert, das im weiteren Verlauf dieser Arbeit auch als *Tresor* $\mathcal{T}$ bezeichnet wird. Der für das Signieren von Nachrichten im i-ten Intervall benötigte Signaturschlüssel sk_i wird mit dem Schlüsselaktualisierungsprotokoll $\mathsf{KeyUpdate}$ berechnet, das zwischen dem Signierer $\mathcal{S}$ und dessen Tresor $\mathcal{T}$ ausgeführt wird. Eine blinde Signatur wird im i-ten Intervall wie gewohnt mit dem Signaturprotokoll Sign erstellt, das zwischen dem Signierer $\mathcal{S}$ (mit privatem Schlüssel sk_i) und dem Empfänger $\mathcal{R}$ durchgeführt wird.
Im Unterschied zu den aktualisierbaren Public-Key-Verschlüsselungsverfahren werden die beiden Schlüsselaktualisierungsalgorithmen aus Definition 5.1.1 in Definition 5.2.1 zu einem Schlüsselaktualisierungsprotokoll zusammengefasst. Die Definition des Schlüsselaktualisierungsprotokolls $\mathsf{KeyUpdate}$ aus Definition 5.2.1 fordert zusätzlich, dass eine direkte Schlüsselaktualisierung zwischen beliebigen Intervallen stets möglich ist.

5.2.1 Sicherheitsziele aktualisierbarer blinder Signaturen

Die Sicherheitsanalyse blinder Signaturverfahren beinhaltet im Wesentlichen zwei Gesichtspunkte: Unfälschbarkeit und Blindheit (vgl. Abschnitt 2.6). Berücksichtigen wir in einem aktualisierbaren blinden Signaturverfahren zusätzlich eine mögliche Kompromittierung des Signierers, so wirkt sich dies auf die Unfälschbarkeit, nicht aber auf die Blindheit des Verfahrens aus.

Unfälschbarkeit: Bei einer erfolgreichen Kompromittierung des Signierers ist der Angreifer im Besitz dessen Signaturschlüssels sk_i. In Abhängigkeit vom Zeitpunkt des Angriffs unterscheiden wir analog zu den aktualisierbaren Public-Key-Verschlüsselungsverfahren zwischen einem Schlüssel-Angriff und einem temporären Schlüssel-Angriff (vgl. Abschnitt 5.1).
Zur formalen Beschreibung eines Schlüssel-Angriffs erhält der Angreifer Zugriff auf ein Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$, das ihm die entsprechenden Signaturschlüssel liefert. Bei Eingabe eines Index i ist die Ausgabe von $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}(i)$ der im i-ten Intervall gültige Signaturschlüssel sk_i. Um die Sicherheit aktualisierbarer blinder Signaturverfahren in puncto Unfälschbarkeit zu formalisieren, betrachten wir einen *nicht adaptiven* und einen *adaptiven* Angreifer und greifen auf Spiel 2.6.1 aus Abschnitt 2.6 zurück. Der nicht adaptive Angreifer führt Spiel 2.6.1 in einem der N Intervalle durch, wobei es ihm erlaubt ist, zunächst maximal $t < N$ Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ zu stellen. Das heißt, der Angreifer führt das folgende Spiel aus:

Spiel 5.2.1 (Unfälschbarkeit).

Schritt 1: Der Signierer $\mathcal{S}$ erzeugt bei Eingabe des Sicherheitsparameters 1^k und der Anzahl $N \in \mathbb{N}$ der Intervalle das Tripel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = \mathsf{Gen}(1^k, N)$.

Schritt 2: Der Angreifer $\mathcal{A}$ stellt bis zu $t < N$ Anfragen an das Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ und wählt ein Intervall i, wobei i nicht als Anfrage an $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ gestellt wurde.

Schritt 3: Der Angreifer $\mathcal{A}$ führt im i-ten Intervall polynomiell in k viele parallele und nicht notwendig unabhängige Protokolle Sign mit identischen Kopien von $\mathcal{S}(i, \mathsf{pk}, \mathsf{sk}_i)$ aus. Sei ℓ die Anzahl der durchgeführten Protokolle, bei denen $\mathcal{S}$ am Ende $\mathsf{completed}$ ausgibt.

Schritt 4: Der Angreifer $\mathcal{A}$ gibt j im i-ten Intervall gültige Nachrichten-Signatur-Paare aus

$$(m_1, \langle i, s_1 \rangle), \ldots, (m_j, \langle i, s_j \rangle).$$

Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.2.1, falls die Zahl j der in Schritt 4 ausgegebenen gültigen Nachrichten-Signatur-Paare größer ist als die Anzahl ℓ der durchgeführten Protokolle.

Im Vergleich zu Spiel 5.2.1 hat der adaptive Angreifer die Möglichkeit, die Signaturprotokolle in beliebigen Intervallen und in beliebiger Reihenfolge durchzuführen, sowie in Abhängigkeit bereits durchgeführter Signaturprotokolle bis zu t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*, \mathsf{sk}_0}$ zu stellen. Das heißt, der Angreifer führt folgendes Spiel aus:

Spiel 5.2.2 (Starke Unfälschbarkeit).

Schritt 1: Analog zu Spiel 5.2.1.

Schritt 2: Der Angreifer $\mathcal{A}$ führt in unterschiedlichen Intervallen polynomiell in k viele und nicht notwendig unabhängige Protokolle Sign aus. Sei ℓ_i die Anzahl der im i-ten Intervall durchgeführten Protokolle, bei denen $\mathcal{S}$ am Ende $\mathsf{completed}$ ausgibt.

Schritt 3: Der Angreifer $\mathcal{A}$ gibt für jedes Intervall i, in dem er Protokolle durchgeführt hat, j_i gültige Nachrichten-Signatur-Paare aus

$$\left(m_1^{(i)}, \langle i, s_1^{(i)} \rangle\right), \ldots, \left(m_{j_i}^{(i)}, \langle i, s_{j_i}^{(i)} \rangle\right).$$

Dabei darf $\mathcal{A}$ die Protokolle in Schritt 2 in beliebigen Intervallen und in beliebiger Reihenfolge durchführen. In Abhängigkeit bereits durchgeführter Protokolle darf $\mathcal{A}$ bis zu t Anfragen an $\mathsf{Exp}_{\mathsf{sk}^*, \mathsf{sk}_0}$ stellen.
Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.2.2, falls für mindestens ein Intervall i die Zahl j_i der in Schritt 3 ausgegebenen gültigen Nachrichten-Signatur-Paare größer ist als die Anzahl ℓ_i der in diesem Intervall durchgeführten Protokolle und der Index i nicht als Anfrage an das Orakel $\mathsf{Exp}_{\mathsf{sk}^*, \mathsf{sk}_0}$ gestellt wurde.

Blindheit: Etwas salopp gesprochen, soll der Signierer in einem blinden Signaturverfahren anhand einer von ihm ausgestellten Signatur nicht entscheiden können, für wen er die Signatur ausgestellt hat. In einem aktualisierbaren blinden Signaturverfahren Π enthält die Signatur $\langle i, s \rangle$ einer Nachricht m neben der „eigentlichen Signatur“ s auch den Index i des Zeitintervalls, in dem sie erstellt wurde. Das heißt, gegenüber einem gewöhnlichen blinden Signaturverfahren gewinnt der Signierer Information und kann anhand

des Index zumindest all die Anwender als mögliche Empfänger ausschließen, mit denen er in diesem Intervall kein Signaturprotokoll durchgeführt hat. Betrachten wir Spiel 2.6.2 aus Abschnitt 2.6 und bieten dem Angreifer die Möglichkeit, die beiden Signaturprotokolle in unterschiedlichen Intervallen durchzuführen, so kann er anhand des in den Signaturen enthaltenen Index sehr wohl entscheiden, welche Nachricht er für welchen Empfänger signiert hat. Für ein aktualisierbares blindes Signaturverfahren fordern wir daher, dass die beiden Signaturprotokolle in Spiel 2.6.2 im gleichen Intervall ausgeführt werden. Das heißt, die Blindheit soll innerhalb jedes einzelnen Intervalls erfüllt sein.

Spiel 5.2.3 (**Blindheit**)**.**

Schritt 1. - Schritt 5. Analog zu Spiel 2.6.2. Lediglich die Signaturprotokolle im dritten Schritt werden innerhalb eines Intervalls durchgeführt.

Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.2.3, falls $\mathcal{A}$ nach Ablauf des Spiels entscheiden kann, welche Nachricht $\mathcal{A}$ für welchen Empfänger signiert hat.

Langfristig sichere blinde Signaturverfahren sind formal wie folgt definiert:

Definition 5.2.2. Ein aktualisierbares blindes Signaturverfahren $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}, \mathsf{Sign}, \mathsf{Verify})$ mit Sicherheitsparameter $k \in \mathbb{N}$ heißt (stark) (t, N)-*langfristig sicher*, falls die beiden folgenden Eigenschaften erfüllt sind:

- *Blindheit*: Für alle effizienten Algorithmen $\mathcal{A}$ existiert eine vernachlässigbare Funktion ν, so dass für die Erfolgswahrscheinlichkeit $\epsilon(k)$ von $\mathcal{A}$ in Spiel 5.2.3 gilt: $\epsilon(k) \leq \frac{1}{2} + \nu(k)$.
- *(Starke) Unfälschbarkeit*: Für alle effizienten Algorithmen $\mathcal{A}$, die höchstens t Anfragen an das Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellen, ist die Erfolgswahrscheinlichkeit $\epsilon(k)$ von $\mathcal{A}$ in Spiel 5.2.1 bzw. Spiel 5.2.2 vernachlässigbar in k.

Falls die Kompromittierung des Signierers während der Schlüsselaktualisierung zwischen den Intervallen i und j erfolgt, kennt der Angreifer den Signaturschlüssel sk_i sowie die Protokollansicht $\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}(\mathsf{KeyUpdate}(\mathcal{S}(i, j, \mathsf{sk}_i), \mathcal{T}(\mathsf{sk}^*)))$ des Signierers und kann damit den Signaturschlüssel sk_j berechnen. In Anlehnung an Definition 5.1.3 verfügt ein aktualisierbares blindes Signaturverfahren über eine sichere Schlüsselaktualisierung, falls es folgende Bedingung erfüllt:

Definition 5.2.3. Ein aktualisierbares blindes Signaturverfahren Π hat eine *sichere Schlüsselaktualisierung*, falls ein Angreifer $\mathcal{A}$, der in den Intervallen i, j einen Schlüssel-Angriff durchführt, die Protokollansicht $\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}(\mathsf{KeyUpdate}(\mathcal{S}(i, j, \mathsf{sk}_i), \mathcal{T}(\mathsf{sk}^*)))$ perfekt simulieren kann.

Bemerkung 5.2.1. Das in diesem Abschnitt definierte Sicherheitsmodell langfristig sicherer blinder Signaturverfahren berücksichtigt im Gegensatz zu [DKXY02], [DKXY03] keinen Generalschlüssel-Angriff, da dieser mit Blick auf die elektronischen Geldsysteme ein eher unwahrscheinliches Szenario darstellt (vgl. Bemerkung 5.3.2). Sicherheit gegen einen solchen Angriff kann aber mit den in [DKXY03] beschriebenen Methoden erreicht werden.

5.2.2 Ein generisches langfristig sicheres Verfahren

Prinzipiell kann allein der Tresor des Signierers dazu verwendet werden, die Signaturschlüssel der verschiedenen Intervalle ohne Interaktion mit dem Signierer zu generieren. In diesem Abschnitt wird nun gezeigt, wie ein perfekt blindes Signaturverfahren mit Sicherheitsparameter $k \in \mathbb{N}$, für das es keinen effizienten Angreifer gibt, der Spiel 2.6.1 mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt, dazu genutzt werden kann, ein $(N-1, N)$-langfristig sicheres blindes Signaturverfahren zu konstruieren. Ist $\Theta = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ ein solches perfekt blindes Signaturverfahren, so kann auf Grundlage von Θ ein $(N-1, N)$-langfristig sicheres blindes Signaturverfahren $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}, \mathsf{Sign}, \mathsf{Verify})$ konstruiert werden.
Dem Verfahren liegt folgende Idee zugrunde (vgl. Abbildung 5.1): Zur Erstellung blinder Signaturen wird in jedem Intervall das Signaturprotokoll $\mathtt{Sign}$ eingesetzt. Zunächst wird ein Schlüsselpaar $(pk, sk) = \mathtt{Gen}(1^k)$ erzeugt, sk als Generalschlüssel sk^* in einem Tresor $\mathcal{T}$ gespeichert und pk als öffentlicher Schlüssel pk verwendet. Um die privaten Schlüssel zu aktualisieren, wird zu Beginn eines neuen Intervalls i mit $\mathtt{Gen}$ ein Schlüsselpaar $(pk_i, sk_i) = \mathtt{Gen}(1^k)$ generiert, sk_i als Signaturschlüssel sk_i für das i-te Intervall benutzt und pk_i vom Tresor mit sk^* zertifiziert. Damit im i-ten Intervall erstellte Signaturen mit dem öffentlichen Schlüssel pk verifiziert werden können, wird das Zertifikat für den im i-ten Intervall gültigen „öffentlichen Schlüssel“ pk_i den Signaturen angehängt.

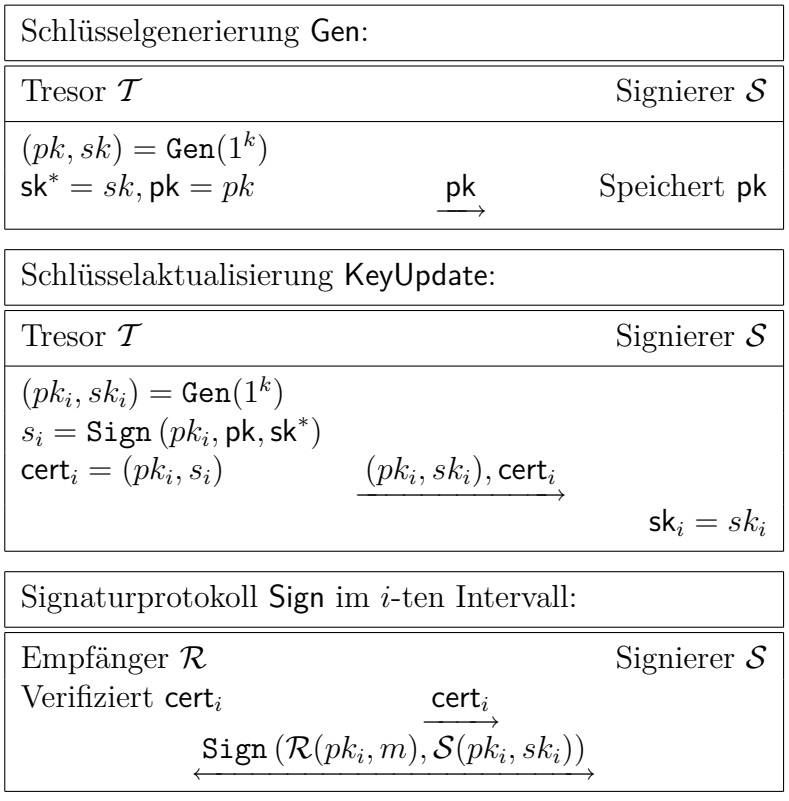

Abbildung 5.1: Die generische langfristig sichere blinde Signatur

Schlüsselgenerierung: Um einen Schlüssel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0)$ zu generieren, wird bei Eingabe des Sicherheitsparameters 1^k und der polynomiell in k beschränkten Anzahl $N \in \mathbb{N}$ der Zeitintervalle vom Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k, N)$ durch den Aufruf von $\mathtt{Gen}(1^k)$ das Schlüsselpaar $(pk, sk) = \mathtt{Gen}(1^k)$ erzeugt. Der private Schlüssel sk wird als Generalschlüssel sk^* in einem Tresor $\mathcal{T}$ gespeichert und anschließend gelöscht. Die Ausgabe von $\mathsf{Gen}(1^k, N)$ ist das Tripel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = (pk, sk, \perp)$.

Schlüsselaktualisierung: Zur Aktualisierung des privaten Schlüssels sk_i wird zu Beginn jedes neuen Intervalls j das Protokoll $\mathsf{KeyUpdate}(\mathcal{S}(i, j, \mathsf{sk}_i), \mathcal{T}(\mathsf{sk}^*))$ durchgeführt. Durch den Aufruf von $\mathtt{Gen}(1^k)$ generiert der Tresor $\mathcal{T}$ für das Intervall j ein neues Schlüsselpaar $(pk_j, sk_j) = \mathtt{Gen}(1^k)$ und erstellt das Zertifikat $\mathsf{cert}_j = (pk_j, s_j)$ mit $s_j = \mathtt{Sign}(\mathcal{T}(pk, pk_j), \mathcal{T}(pk, sk))$. Der Signierer $\mathcal{S}$ erhält den neuen privaten Schlüssel $\mathsf{sk}_j = (sk_j, \mathsf{cert}_j)$ und löscht den bisherigen privaten Schlüssel sk_i.

Signaturprotokoll: Um im i-ten Intervall eine Signatur auf die Nachricht $m \in M$ zu erstellen wird zwischen dem Empfänger $\mathcal{R}(i, \mathsf{pk}, m)$ und dem Signierer $\mathcal{S}(i, \mathsf{pk}, \mathsf{sk}_i)$ das Protokoll $\mathsf{Sign}(\mathcal{R}(i, \mathsf{pk}, m), \mathcal{S}(i, \mathsf{pk}, \mathsf{sk}_i))$ durchgeführt. Innerhalb von Sign wird das Protokoll $\mathtt{Sign}(\mathcal{R}(\mathsf{pk}, m), \mathcal{S}(pk, sk_i))$ ausgeführt, an dessen Ende $\mathcal{R}$ die Signatur $\tilde{s}$ auf m erhält. Die Ausgabe von $\mathcal{R}(i, \mathsf{pk}, m)$ im Protokoll Sign ist die Signatur $\langle i, s\rangle = \langle i, (\tilde{s}, \mathsf{cert}_i)\rangle$.

Verifikation einer Signatur: Um die im i-ten Intervall erstellte Signatur $\langle i, s\rangle$ einer Nachricht m zu verifizieren, wird bei Eingabe von m, pk und $\langle i, s\rangle$ von $\mathsf{Verify}(m, \mathsf{pk}, \langle i, s\rangle)$ anhand von pk zunächst die Gültigkeit des Zertifikats cert_i überprüft. Falls cert_i gültig ist, so gibt $\mathsf{Verify}(m, \mathsf{pk}, \langle i, s\rangle)$ entsprechend $\mathtt{Verify}(m, pk_i, \tilde{s})$ aus. Ist cert_i ungültig, wird 0 $(= \mathsf{false})$ ausgegeben.

Sicherheitsanalyse

Satz 5.2.1. Ist $\Theta = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ ein perfekt blindes Signaturverfahren mit Sicherheitsparameter k, für das es keinen effizienten Angreifer gibt, der Spiel 2.6.1 mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt, so ist das aktualisierbare blinde Signaturverfahren $\Pi = (\mathsf{Gen}, \mathsf{KeyUpdate}, \mathsf{Sign}, \mathsf{Verify})$ ein stark $(N-1, N)$-langfristig sicheres blindes Signaturverfahren mit sicherer Schlüsselaktualisierung.

Beweis. Wir zeigen zunächst, dass Π über eine sichere Schlüsselaktualisierung verfügt.

Sichere Schlüsselaktualisierung: Es genügt zu zeigen, dass ein effizienter Angreifer $\mathcal{A}$, der in den Intervallen i und j einen erfolgreichen Schlüssel-Angriff durchführt, die Protokollansicht $\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}(\mathsf{KeyUpdate}(\mathcal{S}(i, j, \mathsf{sk}_i), \mathcal{T}(\mathsf{sk}^*)))$ perfekt simulieren kann. Es ist

$$\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}(\mathsf{KeyUpdate}(\mathcal{S}(i, j, \mathsf{sk}_i), \mathcal{T}(\mathsf{sk}^*))) = \{\mathsf{sk}_j\} = \{(sk_j, \mathsf{cert}_j)\} = \{(sk_j, (pk_j, s_j))\}.$$

Da der Angreifer $\mathcal{A}$ in den Intervallen i und j einen erfolgreichen Schlüssel-Angriff durchführen kann, kennt $\mathcal{A}$ die beiden Signaturschlüssel sk_i und sk_j und kann somit auch die Protokollansicht perfekt simulieren.

Sei k der Sicherheitsparameter von Π. Für die starke $(N-1, N)$-langfristige Sicherheit zeigen wir zuerst, dass Π im Sinne von Definition 5.2.2 blind ist.

Blindheit: Angenommen, Π ist nicht blind im Sinne von Definition 5.2.2. Dann gibt es einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.2.3 mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ gewinnt, wobei ν eine nicht vernachlässigbare Funktion ist.
Da im dritten Schritt von Spiel 5.2.3 im i-ten Intervall innerhalb von Sign die Protokolle $\mathtt{Sign}(\mathcal{R}_0(\mathsf{pk}, m_b), \mathcal{S}(pk, sk_i))$ bzw. $\mathtt{Sign}(\mathcal{R}_1(\mathsf{pk}, m_{1-b}), \mathcal{S}(pk, sk_i))$ durchgeführt werden, kann $\mathcal{A}$ somit anhand von $(m_0, \tilde{s}_0)$ und $(m_1, \tilde{s}_1)$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, welche Nachricht $\mathcal{A}$ für welchen Empfänger signiert hat, im Widerspruch zur perfekten Blindheit von Θ.

Starke Unfälschbarkeit: Sei $\mathcal{A}$ ein effizienter Angreifer, der mit dem Signierer $\mathcal{S}$ kommuniziert und versucht die starke Unfälschbarkeit von Π zu brechen. Eine Fälschung tritt auf, falls in mindestens einem Intervall i nach ℓ_i durchgeführten Signaturprotokollen Sign mit $\mathcal{S}$ die Zahl der von $\mathcal{A}$ ausgegebenen gültigen Nachrichten-Signatur-Paare größer ist als die Anzahl ℓ_i der in diesem Intervall durchgeführten Protokolle und der Index i nicht als Anfrage an das Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ gestellt wurde.
Offensichtlich kann $\mathcal{A}$ im i-ten Intervall zwei Strategien verfolgen, um die starke Unfälschbarkeit von Π zu brechen:

1. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel pk_i mehr als ℓ_i gültige Nachrichten-Signatur-Paare zu erzeugen. Wobei ℓ_i die Anzahl der im i-ten Intervall durchgeführten Signaturprotokolle ist.

2. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel pk_i ohne Interaktion mit Signierer ein gültiges Nachrichten-Signatur-Paar zu erzeugen, oder $\mathcal{A}$ erzeugt sich ein eigenes Schlüsselpaar (pk_i, sk_i) und versucht das Zertifikat cert_i zu fälschen.

1.Fall: Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht, die Sicherheit von Θ im Sinne von Definition 2.6.2 zu brechen. Sei ℓ die in k polynomiell beschränkte Anzahl der Signaturprotokolle, die $\mathcal{A}$ insgesamt mit $\mathcal{S}$ durchführt. Der Angreifer $\mathcal{A}'$ interagiert mit dem Signierer $\mathcal{S}'$, der das Schlüsselpaar $(pk', sk') = \mathtt{Gen}(1^k)$ erzeugt hat und geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ generiert das Schlüsselpaar $(pk, sk) = \mathtt{Gen}(1^k)$ und setzt $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = (pk, sk, \perp)$. Der Angreifer $\mathcal{A}$ erhält pk als Eingabe (die Rolle des Signierers $\mathcal{S}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ werden von $\mathcal{A}'$ übernommen).

Schritt 2: Der Angreifer $\mathcal{A}'$ wählt zufällig einen Index $r \in_{\mathcal{R}} \{1, \ldots, \ell\}$. Sei i^* der Index des Intervalls, in dem $\mathcal{A}$ das insgesamt r-te Signaturprotokoll mit $\mathcal{S}$ durchgeführt hat. Falls $\mathcal{A}$ vor der Durchführung des r-ten Signaturprotokolls mit $\mathcal{S}$ bereits Signaturprotokolle im i^*-ten Intervall durchgeführt hat, bricht $\mathcal{A}'$ seinen Versuch ab.

Schritt 3: Der Angreifer $\mathcal{A}'$ verwendet den Signierer $\mathcal{S}'$ zur Durchführung des r-ten Signaturprotokolls mit $\mathcal{A}$. Sollte $\mathcal{A}$ in den darauffolgenden $r+1, \ldots, \ell$ Signaturprotokollen erneut Signaturprotokolle im i^*-ten Intervall durchführen, wird von $\mathcal{A}'$ genauso wie im r-ten Signaturprotokoll der Signierer $\mathcal{S}'$ zur Durchführung der Signaturprotokolle mit $\mathcal{A}$ verwendet.

Schritt 4: Der Angreifer $\mathcal{A}'$ bricht seinen Versuch ab, falls $\mathcal{A}$ den Index i^* als Anfrage an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellt.

Schritt 5: Alle anderen Signaturprotokolle mit $\mathcal{A}$ kann $\mathcal{A}'$ wie folgt durchführen: Für die entsprechenden Intervalle generiert $\mathcal{A}'$ neue Schlüsselpaare $(pk_j, sk_j) = \mathtt{Gen}(1^k)$ und generiert sich mit dem Schlüsselpaar (pk, sk) die Zertifikate $\mathsf{cert}_j = \mathtt{Sign}(\mathcal{A}'(pk, pk_j), \mathcal{A}'(pk, sk))$. Anschließend führt $\mathcal{A}'$ die Signaturprotokolle mit $\mathcal{A}$ aus. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ antworten.

Gibt $\mathcal{A}$ in einem Intervall i nach ℓ_i in diesem Intervall durchgeführten Signaturprotokollen mindestens $\ell_i + 1$ gültige Nachrichten-Signatur-Paare

$$(m_1, \langle i, \tilde{s}_1, \mathsf{cert}_i\rangle), \ldots, (m_{\ell_i+1}, \langle i, \tilde{s}_{\ell_i+1}, \mathsf{cert}_i\rangle)$$

aus, so gibt $\mathcal{A}'$ die ℓ_i+1 gültigen Nachrichten-Signatur-Paare $(m_1, \tilde{s}_1,), \ldots, (m_{\ell_i+1}, \tilde{s}_{\ell_i+1})$ aus.
Mit einer Wahrscheinlichkeit von $\frac{1}{\ell}$ wird der Versuch von $\mathcal{A}'$ nicht abgebrochen und es ist $i^* = i$. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im ersten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ das blinde Signaturverfahren Θ zu brechen:

$$\epsilon'(k) \geq \frac{\epsilon(k)}{\ell}.$$

Da ℓ polynomiell in k beschränkt und $\epsilon'(k)$ in k vernachlässigbar ist (Θ ist nach Voraussetzung ein sicheres blindes Signaturverfahren), folgt, dass $\epsilon(k)$ ebenfalls in k vernachlässigbar ist.

2.Fall: Erneut konstruieren wir einen effizienten Angreifer $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht, die Sicherheit von Θ im Sinne von Definition 2.6.2 zu brechen. Der Angreifer $\mathcal{A}'$ interagiert mit dem Signierer $\mathcal{S}'$, der das Schlüsselpaar $(pk', sk') = \mathtt{Gen}(1^k)$ erzeugt hat und geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ setzt $\mathsf{pk} = pk'$ und somit implizit $\mathsf{sk}^* = sk'$. Der Angreifer $\mathcal{A}$ erhält pk als Eingabe (die Rolle des Signierers $\mathcal{S}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ werden von $\mathcal{A}'$ übernommen).

Schritt 2: Da $\mathcal{A}$ für das i-te Intervall keine Signaturprotokolle mit $\mathcal{S}$ durchführt, kann $\mathcal{A}'$ alle Signaturprotokolle mit $\mathcal{A}$ wie folgt durchführen: Der Angreifer $\mathcal{A}'$ erzeugt sich die entsprechenden Schlüsselpaare $(pk_j, sk_j) = \mathtt{Gen}(1^k)$ und erstellt die Zertifikate cert_j, indem sich $\mathcal{A}'$ die öffentlichen Schlüssel pk_j von $\mathcal{S}'$ blind

signieren lässt. Sei ℓ die Anzahl der zwischen $\mathcal{A}'$ und $\mathcal{S}'$ durchgeführten blinden Signaturprotokolle. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ antworten.

Gibt $\mathcal{A}$ das Nachrichten-Signatur-Paar $(m, \langle i, \tilde{s}, \mathsf{cert}_i\rangle) = (m, \langle i, \tilde{s}, (pk_i, s_i)\rangle)$ aus, so gibt $\mathcal{A}'$ das Nachrichten-Signatur-Paar (pk_i, s_i) aus. Da $\mathcal{A}'$ sich den Schlüssel pk_i nicht von $\mathcal{S}'$ hat blind signieren lassen, hat $\mathcal{A}'$ nach ℓ durchgeführten blinden Signaturprotokollen mit $\mathcal{S}'$ mindestens $\ell+1$ gültige Nachrichten-Signatur-Paare erzeugt. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im zweiten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ somit $\epsilon'(k) = \epsilon(k)$. Da Θ nach Voraussetzung ein sicheres blindes Signaturverfahren ist, ist $\epsilon'(k)$ und damit auch $\epsilon(k)$ vernachlässigbar in k. □

5.2.3 Die langfristig sichere blinde Schnorr-Signatur

Die blinde Schnorr-Signatur (vgl. Abschnitt 2.6.1) bildet die Grundlage aller elektronischen Geldsysteme, die auf dem System von S. Brands [Bra94] aufbauen. In diesem Abschnitt wird eine (t, N)-langfristig sichere Variante der blinden Schnorr-Signatur entwickelt und analysiert. Sie dient gleichzeitig als ein grundlegender Baustein für die in Kapitel 6 entworfenen langfristig sicheren elektronischen Geldsysteme.

Schlüsselgenerierung: Um einen Schlüssel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0)$ zu generieren, werden bei Eingabe des Sicherheitsparameters 1^k und der polynomiell in k beschränkten Anzahl N der Intervalle vom Schlüsselgenerierungsalgorithmus $\mathsf{Gen}(1^k, N)$ folgende Schritte durchgeführt:

Schritt 1: Analog zu Schritt 1 der Schlüsselgenerierung aus Abschnitt 2.5.1.

Schritt 2: Wähle $t \in \mathbb{N}$ mit $t < N$.

Schritt 3: Wähle zufällig Zahlen $x_0^*, \ldots, x_t^* \in_{\mathcal{R}} \mathbb{Z}_q$, bestimme das Polynom $f(\mu) = \sum_{i=0}^{t} x_i^* \mu^i$ und berechne $y_i^* = g^{x_i^*}$ für $i = 0, \ldots, t$. Die Zahlen $x_1^*, \ldots, x_t^*$ werden in einem Tresor $\mathcal{T}$ gespeichert und anschließend gelöscht. Die Zahl x_0^* wird als Signaturschlüssel sk_0 für das 0-te Intervall gespeichert.

Schritt 4: Wähle eine kollisionsresistente Hashfunktion $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Die Ausgabe des Algorithmus $\mathsf{Gen}(1^k, N)$ ist das Tripel

$$(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = \left((p, q, g, y_0^*, \ldots, y_t^*, \mathrm{H}), (x_1^*, \ldots, x_t^*), x_0^*\right).$$

Schlüsselaktualisierung: Der für das Intervall j gültige private Schlüssel sk_j berechnet sich als $x_j = f(j)$. Ist das aktuelle Intervall das Intervall i, kann der Signierer $\mathcal{S}$ mit seinem gültigen Signaturschlüssel $\mathsf{sk}_i = x_i$ den neuen Signaturschlüssel x_j nicht berechnen; dazu müsste $\mathcal{S}$ mindestens t weitere Punkte des Polynoms f kennen. Da der Generalschlüssel $\mathsf{sk}^* = (x_1^*, \ldots, x_t^*)$ im Tresor $\mathcal{T}$ gespeichert ist, kann $\mathcal{S}$ den für das Intervall j benötigten Signaturschlüssel sk_j mit Hilfe von $\mathcal{T}$ berechnen. Zur

Aktualisierung des privaten Schlüssels wird das im Folgenden beschriebene Protokoll $\mathsf{KeyUpdate}\left(\mathcal{S}\left(i,j,\mathsf{sk}_i\right),\mathcal{T}\left(\mathsf{sk}^*\right)\right)$ durchgeführt (vgl. auch Abbildung 5.2), das im Wesentlichen auf dem Schlüsselaktualisierungsprotokoll aus Abschnitt 5.1.1 basiert.

Schritt 1: Der Signierer $\mathcal{S}$ sendet den Index i des aktuellen Intervalls i und den Index j des Intervalls, für den $\mathcal{S}$ den Signaturschlüssel $\mathsf{sk}_j = x_j$ berechnen möchte an seinen Tresor $\mathcal{T}$.

Schritt 2: Anhand des Generalschlüssels $\mathsf{sk}^* = (x_1^*, \ldots, x_t^*)$ berechnet $\mathcal{T}$ den temporären Schlüssel $x'_{i,j} = \sum_{k=1}^{t} x_k^*(j^k - i^k)$ und sendet $x'_{i,j}$ an $\mathcal{S}$.

Schritt 3: Mit dem aktuellen Signaturschlüssel x_i berechnet $\mathcal{S}$ den für das Intervall j gültigen Signaturschlüssel $x_j = x_i + x'_{i,j}$. Anschließend werden der für das Intervall i gültige Signaturschlüssel x_i und der temporäre Schlüssel $x_{i,j}$ von $\mathcal{S}$ gelöscht.

$\mathcal{S}(i,j,\mathsf{sk}_i)$		$\mathcal{T}(\mathsf{sk}^*)$
	$\xrightarrow{i,j}$	$x'_{i,j} = \sum_{k=1}^{t} x_k^*(j^k - i^k)$
	$\xleftarrow{x'_{i,j}}$	
$x_j = x_i + x'_{i,j}$		
Lösche $x_i, x'_{i,j}$		

Abbildung 5.2: Schlüsselaktualisierung der langfristig sicheren blinden Schnorr-Signatur

Das folgende Lemma zeigt, dass das Schlüsselaktualisierungsprotokoll KeyUpdate dem Signierer tatsächlich den gewünschten Signaturschlüssel liefert.

Lemma 5.2.2. Bei korrekter Durchführung von $\mathsf{KeyUpdate}\left(\mathcal{S}\left(i,j,\mathsf{sk}_i\right),\mathcal{T}\left(\mathsf{sk}^*\right)\right)$ erhält $\mathcal{S}$ als Ausgabe den im j-ten Intervall gültigen Signaturschlüssel sk_j.

Beweis. Verhalten sich der Signierer $\mathcal{S}$ und sein Tresor $\mathcal{T}$ gemäß des Protokolls, so gilt:

$$\begin{aligned} x_i + x'_{i,j} &= f(i) + x'_{i,j} = \sum_{i=0}^{t} x_k^* i^k + \sum_{k=1}^{t} x_k^*(j^k - i^k) = \sum_{k=1}^{t} x_k^* j^k + \sum_{k=0}^{t} x_k^* i^k - \sum_{k=1}^{t} x_k^* i^k \\ &= \sum_{k=1}^{t} x_k^* j^k + x_0^* = \sum_{k=0}^{t} x_k^* j^k = f(j) \end{aligned}$$

Das heißt, es ist $x_i + x'_{i,j} = f(j) = x_j = \mathsf{sk}_j$ der im j-ten Intervall gültige Signaturschlüssel. □

Signaturprotokoll: Um im Intervall i eine Signatur der Nachricht $m \in \{0,1\}^*$ zu erstellen, wird zwischen dem Empfänger $\mathcal{R}(i,\mathsf{pk},m)$ und dem Signierer $\mathcal{S}(i,\mathsf{pk},\mathsf{sk}_i)$ das in Abbildung 5.3 dargestellte Signaturprotokoll $\mathsf{Sign}(\mathcal{S}(i,\mathsf{pk},\mathsf{sk}_i),\mathcal{R}(i,\mathsf{pk},m))$ durchgeführt. Die Ausgabe von $\mathcal{R}(i,\mathsf{pk},m)$ ist die Signatur $\langle i,(c',r')\rangle$ der Nachricht m im Intervall i.

$\mathcal{R}(i, \mathsf{pk}, m)$		$\mathcal{S}(i, \mathsf{pk}, \mathsf{sk}_i)$
		$w \in_{\mathcal{R}} \mathbb{Z}_q$
	$\xleftarrow{a}$	$a = g^w$
$u, v \in_{\mathcal{R}} \mathbb{Z}_q$		
$a' = a^u g^v$		
$c' = \mathrm{H}(m, a')$		
$c = c' u^{-1} \bmod q$	$\xrightarrow{c}$	
	$\xleftarrow{r}$	$r = cx_i + w \bmod q$
$r' = ru + v \bmod q$		

Abbildung 5.3: Die langfristig sichere blinde Schnorr-Signatur

Verifikation einer Signatur: Um die im Intervall i erstellte Signatur $\langle i, (c', r') \rangle$ einer Nachricht m zu verifizieren, werden bei Eingabe von m, pk und $\langle i, (c', r') \rangle$ vom Verifikationsalgorithmus $\mathsf{Verify}\,(m, \mathsf{pk}, \langle i, (c', r') \rangle)$ folgende Werte berechnet:

$$y_i = \prod_{k=0}^{t} \left(y_i^*\right)^{i^k}, \ \tilde{a} = g^{r'} y_i^{-c'} \ \text{ und } \ \tilde{c} = \mathrm{H}(m, \tilde{a}).$$

Die Ausgabe des Algorithmus $\mathsf{Verify}(m, \mathsf{pk}, \langle i, (c', r') \rangle)$ ist 1, falls $\tilde{c} = c'$ und 0 sonst.

Sicherheitsanalyse

Wir zeigen zunächst, dass die in diesem Abschnitt vorgestellte aktualisierbare Variante der blinden Schnorr-Signatur über eine sichere Schlüsselaktualisierung verfügt und ein (t, N)-langfristig sicheres Verfahren ist. Dabei führen wir die langfristige Sicherheit des Verfahrens auf die Sicherheit der blinden Schnorr-Signatur (vgl. Abschnitt 2.6.1) zurück. Da für die blinde Schnorr-Signatur noch kein expliziter Sicherheitsbeweis bekannt ist, zeigen wir die langfristige Sicherheit der aktualisierbaren Variante unter folgender Annahme:

Annahme 5.1. Für die in Abschnitt 2.6.1 vorgestellte blinde Schnorr-Signatur mit Sicherheitsparameter $k \in \mathbb{N}$ gibt es keinen effizienten Angreifer, der Spiel 2.6.1 mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt.

Satz 5.2.3. Gegeben sei die in diesem Abschnitt vorgestellte aktualisierbare blinde Schnorr-Signatur Π mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 5.1 ist Π ein (t, N)-langfristig sicheres blindes Signaturverfahren und verfügt über eine sichere Schlüsselaktualisierung.

Beweis. Wir zeigen zunächst, dass die aktualisierbare blinde Schnorr-Signatur über eine sichere Schlüsselaktualisierung verfügt.

Sichere Schlüsselaktualisierung: Es genügt zu zeigen, dass ein effizienter Angreifer $\mathcal{A}$, der in den Intervallen i und j einen erfolgreichen Schlüssel-Angriff durchführt, die Protokollansicht $\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}\left(\mathsf{KeyUpdate}\left(\mathcal{S}\left(i,j,x_i\right),\mathcal{T}\left(\right)\right)\right)$ perfekt simulieren kann. Angreifer $\mathcal{A}$ kennt die beiden Signaturschlüssel $\mathsf{sk}_i = f(i) = x_i$ und $\mathsf{sk}_j = f(j) = x_j$ und kann damit die Protokollansicht

$$\mathsf{view}_{\mathcal{S}}^{\mathcal{T}}\left(\mathsf{KeyUpdate}\left(\mathcal{S}\left(i,j,\mathsf{sk}_i\right),\mathcal{T}\left(\mathsf{sk}^*\right)\right)\right) = \{x'_{i,j}\}$$

perfekt simulieren, denn es ist

$$x'_{i,j} = f(j) - f(i) = x_j - x_i.$$

Sei k der Sicherheitsparameter von Π. Für die (t, N)-langfristige Sicherheit zeigen wir zuerst, dass Π im Sinne von Definition 5.2.2 blind ist.

Blindheit: Angenommen Π ist nicht blind im Sinne von Definition 5.2.2, dann gibt es einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.2.3 mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ gewinnt. Wobei ν eine nicht vernachlässigbare Funktion ist.
Da in Spiel 5.2.3 im i-ten Intervall die Protokolle $\mathsf{Sign}(\mathcal{R}_0(i, \mathsf{pk}, m_b), \mathcal{A}(i, \mathsf{pk}, \mathsf{sk}_i))$ und $\mathsf{Sign}(\mathcal{R}_1(i, \mathsf{pk}, m_{1-b}), \mathcal{A}(i, \mathsf{pk}, \mathsf{sk}_i))$ durchgeführt werden, kann $\mathcal{A}$ anhand von $(m_0, s_0) = (m_0, (c'_0, r'_0))$ und $(m_1, s_1) = (m_1, (c'_1, r'_1))$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, welche Nachricht $\mathcal{A}$ für welchen Empfänger signiert hat. Dies ist aber ein Widerspruch zur perfekten Blindheit der Schnorr-Signatur (vgl. Abschnitt 2.6.1).

Unfälschbarkeit: Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der die Unfälschbarkeit von Π gemäß Definition 5.2.2 brechen kann. Dann gewinnt $\mathcal{A}$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ Spiel 5.2.1. Das heißt, nachdem $\mathcal{A}$ höchstens t Anfragen an $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ gestellt hat und anschließend im i-ten Intervall ℓ Signaturprotokolle mit dem Signierer $\mathcal{S}$ durchgeführt hat, ist die Anzahl l der von $\mathcal{A}$ im i-ten Intervall ausgegebenen Nachrichten-Signatur-Paare mit der Wahrscheinlichkeit $\epsilon(k)$ größer als ℓ. Durch die t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ erhält $\mathcal{A}$ lediglich t Funktionswerte von f. Da die Koeffizienten $x_0^*, \ldots, x_t^*$ des Polynoms f zufällig gewählt wurden, ist f ein zufällig gewähltes Polynom mit $\mathrm{Grad}(f) = t$ und der private Schlüssel $\mathsf{sk}_i = x_i = f(i)$ ist für $\mathcal{A}$ gleichverteilt in $\mathbb{Z}_q$ und stochastisch unabhängig von den t privaten Schlüsseln, die $\mathcal{A}$ von $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ erhalten hat. Das heißt, der Angreifer $\mathcal{A}$ erhält durch die t Anfragen an $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ keine Information über den privaten Schlüssel x_i.
Somit ist $\mathcal{A}$ bei Eingabe von pk mit der Wahrscheinlichkeit $\epsilon(k)$ in der Lage $l > \ell$ gültige Nachrichten-Signatur-Paare der Instanz $(\mathsf{pk}, \mathsf{sk}) = \left(\left(p, q, g, g^{f(i)}, \mathrm{H}\right), f\left(i\right)\right)$ des blinden Schnorr-Signaturverfahren auszugeben. Dies steht aber im Widerspruch zu Annahme 5.1. □

Natürlich kann die blinde Schnorr-Signatur mit dem in Abschnitt 5.2.2 vorgestellten generischen Verfahren unter Annahme 5.1 in ein stark $(N-1, N)$-langfristig sicheres blindes Signaturverfahren transformiert werden. Verglichen mit der blinden Schnorr-Signatur verdoppelt sich dabei allerdings der zur Verifikation einer Signatur benötigte Rechenaufwand. Im Gegensatz dazu ist die in diesem Abschnitt vorgestellte aktualisierbare blinde Schnorr-Signatur Π genauso effizient wie die blinde Schnorr-Signatur und

für $t = N-1$ ist auch Π unter Annahme 5.1 ein stark $(N-1, N)$-langfristig sicheres blindes Signaturverfahren.
Um dies zu beweisen führen wir die starke $(N-1, N)$-langfristige Sicherheit von Π auf die Sicherheit der blinden Schnorr-Signatur zurück. Wir zeigen, dass ein effizienter Angreifer $\mathcal{A}$, der die starke $(N-1, N)$-langfristige Sicherheit von Π bricht, von einem effizienten Angreifer $\mathcal{A}'$ dazu verwendet werden kann, die Unfälschbarkeit der blinden Schnorr-Signatur Θ zu brechen. Für den Beweis entscheidend ist, dass $\mathcal{A}'$ auf alle Anfragen antworten kann, die $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellt. Wie $\mathcal{A}'$ dies bewerkstelligen kann, wird im Folgenden dargestellt:
Der Angreifer $\mathcal{A}'$ führt im Sicherheitsparameter $k \in \mathbb{N}$ polynomiell viele Signaturprotokolle mit dem Signierer $\mathcal{S}'$ durch, der das Schlüsselpaar $((p, q, g, y = g^x, \mathrm{H}), x) = \mathsf{Gen}(1^k)$ erzeugt hat. Um die Unfälschbarkeit von Θ zu brechen, integriert $\mathcal{A}'$ den öffentlichen Schlüssel $y = g^x$ in den öffentlichen Schlüssel pk von Π. Der Angreifer $\mathcal{A}'$ wählt zufällig einen Index $i \in_{\mathcal{R}} \{0, \ldots, N-1\}$ und setzt $g^{f(i)} = y$. Damit legt $\mathcal{A}$ implizit den privaten Schlüssel $\mathsf{sk}_i = f(i) = x$ fest. Damit $\mathcal{A}'$ auf alle Anfragen antworten kann, die $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellt, muss $\mathcal{A}'$ die Funktionswerte des Polynoms f an den Stellen $0, \ldots, i-1$ und $i+1, \ldots, N-1$ kennen und y_j^* für $j = 0, \ldots, i-1, i+1, \ldots, N-1$ als Teil von pk korrekt berechnen. Dazu wählt $\mathcal{A}'$ zufällig Zahlen $x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{N-1} \in_{\mathcal{R}} \mathbb{Z}_q$ und setzt $f(j) = x_j$ für $j = 0, \ldots, i-1, i+1, \ldots, N-1$. Da der Angreifer $\mathcal{A}'$ lediglich $N-1$ Funktionswerte von f kennt, kann $\mathcal{A}'$ die Koeffizienten x_j^* von f nicht berechnen. Allerdings kennt $\mathcal{A}'$ den Wert $y = g^x$ und kann mit dem Newton-Interpolationsverfahren (siehe z.B. [FH07]) die im öffentlichen Schlüssel pk enthaltenen $g^{x_j^*}$ berechnen. Hierzu betrachtet $\mathcal{A}'$ das Polynom

$$f(\mu) = a_0 + a_1(\mu - 0) + a_2(\mu - 0)(\mu - 1) + \ldots + a_{N-1}(\mu - 0) \cdots (\mu - (N-2)).$$

Es gilt:

$$\begin{aligned} x = f(0) &= a_0 \bmod q \\ x_1 = f(1) &= a_0 + a_1 \bmod q \\ x_2 = f(2) &= a_0 + 2a_1 + 2a_2 \bmod q \\ &\vdots \\ x_{N-1} = f(N-1) &= a_0 + (N-1)a_1 + \ldots + (N-1)!a_{N-1} \bmod q \end{aligned}$$

Der Angreifer $\mathcal{A}'$ kennt die Funktionswerte $f(0), \ldots, f(i-1)$ und kann somit die Werte a_j und g^{a_j} für $j = 0, \ldots, i-1$ rekursiv berechnen. Da $\mathcal{A}'$ den Funktionswert $f(i)$ nicht kennt, kann $\mathcal{A}'$ die Werte a_j für $j = i, \ldots, N-1$ nicht direkt berechnen. Allerdings kennt $\mathcal{A}'$ den Wert $g^{f(i)} = g^x$ und kann somit die restlichen Werte g^{a_j} für $j = i, \ldots, N-1$ rekursiv berechnen. Multipliziert man das Polynom f aus, kann man anhand der a_j die gesuchten Koeffizienten x_j^* berechnen. Das heißt, der Angreifer $\mathcal{A}'$ kann anhand der g^{a_j} alle benötigten Werte $g^{x_j^*}$ berechnen.

Satz 5.2.4. Gegeben sei die aktualisierbare blinde Schnorr-Signatur Π mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 5.1 ist Π stark $(N-1, N)$-langfristig sicher und verfügt über eine sichere Schlüsselaktualisierung.

Beweis. Die Blindheit und die sichere Schlüsselaktualisierung folgen analog zu Satz 5.2.3. Bleibt die starke Unfälschbarkeit zu zeigen. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ die starke Unfälschbarkeit von Π gemäß Definition 5.2.2 brechen kann. Dann kann $\mathcal{A}$ von einem effizienten Angreifer $\mathcal{A}'$ dazu genutzt werden, die Unfälschbarkeit (vgl. Spiel 2.6.1) der blinden Schnorr-Signatur zu brechen, was ein Widerspruch zu Annahme 5.1 ist. Der Angreifer $\mathcal{A}'$ kommuniziert mit dem Signierer $\mathcal{S}'$ und erhält folgende Daten als Eingabe:

- Primzahlen p, q mit $|p-1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Einen Generator g der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.
- Einen öffentlichen Schlüssel $y = g^x$ mit unbekanntem $x \in \mathbb{Z}_q$.
- Eine kollisionsresistente Hashfunktion $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Zunächst übersetzt $\mathcal{A}'$ diese Daten wie folgt in das Tripel $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0)$:

- Der Angreifer $\mathcal{A}'$ übernimmt p, q, g, H.
- Der Angreifer wählt zufällig einen Index $i \in_{\mathcal{R}} \{0, \ldots, N-1\}$ und setzt $g^{f(i)} = y$. Implizit setzt $\mathcal{A}'$ damit $\mathsf{sk}_i = f(i) = x$.
- Der Angreifer $\mathcal{A}'$ wählt zufällig Zahlen $x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{N-1} \in_{\mathcal{R}} \mathbb{Z}_q$ und setzt $f(j) = x_j$ für $j = 0, \ldots, i-1, i+1, \ldots, N-1$. Anschließend berechnet $\mathcal{A}'$ wie beschrieben die Werte $y_j^* = g^{x_j^*}$ für $j = 0, \ldots, N-1$.
- Der Angreifer $\mathcal{A}'$ setzt $(\mathsf{pk}, \mathsf{sk}^*, \mathsf{sk}_0) = ((p, q, g, y_0^*, \ldots, y_t^*, \mathrm{H}), (x_1^*, \ldots, x_{N_1}^*), x_0^*)$.

Der Angreifer $\mathcal{A}$ kommuniziert mit dem Signierer $\mathcal{S}$. Sei ℓ die in k polynomiell beschränkte Anzahl der Signaturprotokolle, die Angreifer $\mathcal{A}$ insgesamt mit $\mathcal{S}$ durchführt. Um Θ im Sinne der Unfälschbarkeit zu brechen geht $\mathcal{A}'$ wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}$ erhält pk als Eingabe. Die Rolle des Signierers $\mathcal{S}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ werden von $\mathcal{A}'$ übernommen.

Schritt 2: Der Angreifer $\mathcal{A}'$ verwendet den Signierer $\mathcal{S}'$ zur Durchführung der Signaturprotokolle mit $\mathcal{A}$ im i-ten Intervall. Falls $\mathcal{A}$ den Index i als Anfrage an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ stellt, bricht $\mathcal{A}'$ seinen Versuch ab.

Schritt 3: Da $\mathcal{A}'$ die privaten Schlüssel aller anderen Intervalle kennt, kann $\mathcal{A}'$ in allen anderen Intervallen die Signaturprotokolle mit $\mathcal{A}$ durchführen. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ antworten.

Gibt $\mathcal{A}$ in einem Intervall i^* nach ℓ_{i^*} in diesem Intervall durchgeführten Signaturprotokollen mindestens $\ell_{i^*}+1$ gültige Nachrichten-Signatur-Paare aus, so gibt $\mathcal{A}'$ diese $\ell_{i^*}+1$ Nachrichten-Signatur-Paare aus. Mit einer Wahrscheinlichkeit von $\frac{1}{N}$ wird der Versuch von $\mathcal{A}'$ nicht abgebrochen und es ist $i = i^*$. Für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ gilt somit:

$$\epsilon'(k) \geq \frac{\epsilon(k)}{N}.$$

Dies ist aber ein Widerspruch zu Annahme 5.1. □

5.3 Langfristig sichere elektronische Geldsysteme

Basierend auf dem im vorhergehenden Abschnitt entwickelten Modell der aktualisierbaren blinden Signaturverfahren, wird in diesem Abschnitt das Modell der *aktualisierbaren elektronischen Geldsysteme* als Grundlage langfristig sicherer elektronischer Geldsysteme vorgestellt.
An einem aktualisierbaren elektronischen Geldsystem nehmen die Parteien teil, die sich auch in einem elektronischen Geldsystem wiederfinden (vgl. Definition 3.1.1). Die Bank verfügt zusätzlich über einen Tresor, wie er im vorangehenden Abschnitt eingeführt wurde. Aufbauend auf den Definitionen 3.1.1 und 5.2.1 ist ein aktualisierbares elektronisches Geldsystem formal wie folgt definiert:

Definition 5.3.1 (**Aktualisierbare elektronische Geldsysteme**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter, M die Menge aller Nachrichten und $N \in \mathbb{N}$ die polynomiell in k beschränkte Anzahl der Intervalle. Ein *aktualisierbares elektronisches (Offline-)Geldsystem* Ω besteht aus den folgenden Protokollen und Algorithmen:

1. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{BKeyGen}$ der Bank $\mathcal{B}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k und der Anzahl N der Intervalle einen öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$, einen Generalschlüssel $\mathsf{sk}^*_{\mathcal{B}}$ und einen ersten privaten Schlüssel $\mathsf{sk}^{(0)}_{\mathcal{B}}$ erzeugt. Die Ausgabe von $\mathsf{BKeyGen}(1^k, N)$ ist das Tripel $(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}})$.

2. Das *Schlüsselaktualisierungsprotokoll* $\mathsf{BKeyUpdate}$ ist ein interaktives Protokoll zwischen der Bank $\mathcal{B}$ und ihrem Tresor $\mathcal{T}$. Für den Generalschlüssel $\mathsf{sk}^*_{\mathcal{B}}$, die Indizes i, j und den im i-ten Intervall gültigen privaten Schlüssel $\mathsf{sk}^{(i)}_{\mathcal{B}}$ ist $i, j, \mathsf{sk}^{(i)}_{\mathcal{B}}$ die Eingabe der Bank $\mathcal{B}$ und $\mathsf{sk}^*_{\mathcal{B}}$ die Eingabe des Tresors $\mathcal{T}$. Die Ausgabe von $\mathcal{B}(i, j, \mathsf{sk}^{(i)}_{\mathcal{B}})$ ist der im j-ten Intervall gültige private Schlüssel $\mathsf{sk}^j_{\mathcal{B}}$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{T}(\mathsf{sk}^*_{\mathcal{B}})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

3. Das *Kontoeröffnungs-Protokoll* Reg ist ein interaktives Protokoll zwischen dem Kunden $\mathcal{K}$ und der Bank $\mathcal{B}$, bei dem $\mathsf{pk}_{\mathcal{B}}$ sowohl die Eingabe des Kunden $\mathcal{K}$ als auch der Bank $\mathcal{B}$ ist. Die Ausgabe von $\mathcal{K}(\mathsf{pk}_{\mathcal{B}})$ ist das Schlüsselpaar $(\mathsf{pk}_{\mathcal{K}}, \mathsf{sk}_{\mathcal{K}})$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{B}(\mathsf{pk}_{\mathcal{B}})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

4. Das *Abhebe-Protokoll* Withdrawal ist ein interaktives Protokoll zwischen dem Kunden $\mathcal{K}$ und der Bank $\mathcal{B}$. Im i-ten Intervall ist $i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}$ die Eingabe des Kunden $\mathcal{K}$ und $i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}^{(i)}$ die Eingabe der Bank $\mathcal{B}$. Die Ausgabe des Kunden $\mathcal{K}(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}})$ ist eine elektronische Münze $\mathsf{Coin} = (m, \langle i, s\rangle)$, bestehend aus einer Nachricht $m \in M$ und einer Signatur $\langle i, s\rangle$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{B}(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}^{(i)})$ completed oder not completed.

5. Das *Bezahl-Protokoll* Payment ist ein interaktives Protokoll zwischen dem Kunden $\mathcal{K}$ und dem Händler $\mathcal{H}$. Es ist $\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{Coin}$ die Eingabe des Kunden $\mathcal{K}$ und $\mathsf{pk}_{\mathcal{H}}, \mathsf{pk}_{\mathcal{B}}$ die Eingabe des Händlers $\mathcal{H}$. Die Ausgabe des Händlers $\mathcal{H}(\mathsf{pk}_{\mathcal{H}}, \mathsf{pk}_{\mathcal{B}})$ ist der Wert 1 (= true) oder 0 (= false). Entsprechend ist die Ausgabe von $\mathcal{K}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{Coin})$ completed oder not completed.

6. Das *Einlöse-Protokoll* Deposit ist ein interaktives Protokoll zwischen dem Händler $\mathcal{H}$ und der Bank $\mathcal{B}$. Dabei ist $\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{H}}, \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ die Eingabe des Händlers $\mathcal{H}$ und $\mathsf{pk}_{\mathcal{B}}$ die Eingabe der Bank $\mathcal{B}$. Die Ausgabe der Bank $\mathcal{B}\,(\mathsf{pk}_{\mathcal{B}})$ ist 1 (= true) oder 0 (= false). Entsprechend ist die Ausgabe von $\mathcal{H}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{H}}, \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}\,(P)\right)$ completed oder not completed.

7. Der *Identifizierungsalgorithmus* Identify der Bank $\mathcal{B}$ ist ein deterministischer Algorithmus. Bei Eingabe der Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}(D_1), \mathsf{view}_{\mathcal{B}}^{\mathcal{H}_2}(D_2)$ zweier Einlöse-Protokolle D_1 und D_2 gibt $\mathsf{Identify}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}\,(D_1)\,, \mathsf{view}_{\mathcal{B}}^{\mathcal{H}_2}\,(D_2)\right)$ die Identität id des betrügenden Kunden bzw. Händlers aus, falls in D_1 und D_2 dieselbe Münze eingelöst wurde und $\perp$ sonst.

Entsprechend Abschnitt 5.2 speichert die Bank $\mathcal{B}$ ihren Generalschlüssel $\mathsf{sk}_{\mathcal{B}}^{*}$ in ihrem Tresor $\mathcal{T}$ und kann ihren, zum Ausstellen elektronischer Münzen benötigten, Signaturschlüssel mit dem Schlüsselaktualisierungsprotokoll BKeyUpdate aktualisieren. Aus Sicht der Kunden und Händler ändert sich am Ablauf der restlichen Protokolle gegenüber einem „normalen“ elektronischen Geldsystem nichts.

Bemerkung 5.3.1. Durch die regelmäßige Aktualisierung ihres privaten Schlüssels und der damit verbundenen Intervalleinteilung entsteht für die Bank auf einfachste Weise und ohne zusätzlichen Aufwand die Möglichkeit, *kurzlebige* (zeitlich eingeschränkt gültige) Münzen auszugeben. Das heißt, die Bank kann die Gültigkeit elektronischer Münzen zeitlich begrenzen, indem sie die Gültigkeit der Münzen auf eine gewisse Anzahl von Intervallen einschränkt: Münzen, die im i-ten Intervall abgehoben wurden sind bis zum $i + x$-ten Intervall gültig. Wobei $x \in \mathbb{N}$ und $i + x \leq N$ ist. Diese Möglichkeit konnte bisher nur mit partiell blinden Signaturen (z.B. [AF96],[AO00]) realisiert werden.
Die Konstruktion kurzlebiger Münzen erschließt weitere Anwendungsgebiete, in denen der Einsatz elektronischer Geldsysteme denkbar ist. Beispielsweise können kurzlebige Münzen innerhalb von anonymisierten Prepaid-Services dazu genutzt werden, Kunden einen zeitlich beschränkten Zugriff auf Inhalte oder Services des Anbieters bereitzustellen. Kurzlebige Münzen werden dann nicht als Zahlungsmittel, sondern als zeitlich limitiertes anonymisiertes Authentifikations-Token eingesetzt.

5.3.1 Sicherheitsziele aktualisierbarer Geldsysteme

Die Sicherheitsanalyse elektronischer Geldsysteme umfasst gemäß Definition 3.1.2 folgende Eigenschaft: Identitäts-Nachweis, Nichterweiterbarkeit, Unfälschbarkeit und Anonymität. Berücksichtigen wir in einem aktualisierbaren elektronischen Geldsystem zusätzlich eine mögliche Kompromittierung der Bank, so wirkt sich dies auf die drei erstgenannten Eigenschaften aus.
Die Kompromittierung der Bank modellieren wir identisch zu den aktualisierbaren blinden Signaturen durch das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ und unterscheiden in Abhängigkeit vom Zeitpunkt des Angriffs zwischen einem Schlüssel-Angriff und einem temporären Schlüssel-Angriff (vgl. Abschnitt 5.2.1).

Identitäts-Nachweis: Da ein Angreifer nach der Kompromittierung der Bank in der Lage ist, sich selbst beliebige Münzen auszustellen, kann der Identitäts-Nachweis in allen kompromittierten Intervallen nicht mehr gewährleistet werden. Für die Sicherheit eines aktualisierbaren Geldsystems fordern wir, dass die Bank Double-Spender in den restlichen Intervallen identifizieren kann.

Nichterweiterbarkeit: Wir unterscheiden analog zu den aktualisierbaren blinden Signaturen zwischen einem adaptiven und einem nicht adaptiven Angreifer. Der nicht adaptive Angreifer darf zunächst maximal t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ stellen und versucht anschließend, die Nichterweiterbarkeit des Systems in einem von ihm gewählten Intervall zu brechen. Das heißt, der Angreifer führt folgendes Spiel durch:

Spiel 5.3.1 (**Nichterweiterbarkeit**)**.**

Schritt 1: Die Bank $\mathcal{B}$ erzeugt bei Eingabe des Sicherheitsparameters 1^k und der Anzahl $N \in \mathbb{N}$ der Intervalle das Tripel $(\mathsf{pk}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^0_{\mathcal{B}}) = \mathsf{BKeyGen}(1^k, N)$.

Schritt 2: Der Angreifer $\mathcal{A}$ stellt bis zu $t < N$ Anfragen an das Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ und wählt ein Intervall i, wobei i nicht als Anfrage an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ gestellt wurde.

Schritt 3: Der Angreifer $\mathcal{A}$ erhält den öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$ und ℓ Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1^{(i)}), \ldots, \mathsf{view}_{\mathcal{K}_\ell}(W_\ell^{(i)})$ von ℓ Abhebe-Protokollen, die im i-ten Intervall durchgeführt wurden. Die Anzahl ℓ der Ansichten ist dabei in k polynomiell beschränkt.

Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.3.2, falls $\mathcal{A}$ am Ende $\ell + 1$ gültige Münzen ausgibt, die erfolgreich ausgegeben werden können.

Analog zu Abschnitt 5.2.1 modellieren wir den adaptiven Angriff durch folgendes Spiel:

Spiel 5.3.2 (**Starke Nichterweiterbarkeit**)**.**

Schritt 1: Analog zu Spiel 5.3.1.

Schritt 2: Der Angreifer $\mathcal{A}$ erhält den öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$ und insgesamt ℓ Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)$ von ℓ Abhebe-Protokollen. Wobei ℓ in k polynomiell beschränkt ist.

In Schritt 2 hat $\mathcal{A}$ die Möglichkeit Protokollansichten von Abhebe-Protokollen beliebiger Intervalle und in beliebiger Reihenfolge anzufordern sowie in Abhängigkeit erhaltener Ansichten bis zu t Anfragen an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ zu stellen.
Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.3.1, falls $\mathcal{A}$ für ein Intervall i am Ende $\ell_i + 1$ gültige Münzen ausgibt, die erfolgreich ausgegeben werden können und der Index i nicht als Anfrage an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ gestellt wurde. Dabei ist ℓ_i die Anzahl der Ansichten, die $\mathcal{A}$ für das i-te Intervall erhalten hat.

Unfälschbarkeit: Auch hier unterscheiden wir zwischen einem nicht adaptiven und einem adaptiven Angreifer. Beide Angreifer modellieren wir analog zur Nichterweiterbarkeit durch die beiden folgenden Spiele:

Spiel 5.3.3 (**Unfälschbarkeit**)**.**

Schritt 1. - Schritt 3. Analog zu Spiel 5.3.1.

Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.3.2, falls $\mathcal{A}$ eine Münze Coin ausgibt, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank $\mathcal{B}$ die Identität des Betrügers aufdecken kann.

Spiel 5.3.4 (**Starke Unfälschbarkeit**)**.**

Schritt 1. - Schritt 3. Analog zu Spiel 5.3.2.

In Schritt 2 hat $\mathcal{A}$ die Möglichkeit, Ansichten von Abhebe-Protokollen beliebiger Intervalle und in beliebiger Reihenfolge zu erhalten, sowie in Abhängigkeit erhaltener Ansichten bis zu t Anfragen an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ zu stellen.
Der Angreifer $\mathcal{A}$ gewinnt Spiel 5.3.1, falls $\mathcal{A}$ für ein Intervall i eine Münze Coin ausgibt, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank $\mathcal{B}$ die Identität des Betrügers aufdecken kann und der Index i nicht als Anfrage an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ gestellt wurde.

Anonymität: In einem aktualisierbaren elektronischen Geldsystem enthält eine Münze $\mathsf{Coin} = (m, \langle i, s\rangle)$ neben der Signatur auch den Index i, in dem die Münze abgehoben wurde. Das heißt, anhand des Index i kann die Bank all die Kunden als Empfänger von Coin ausschließen, die im i-ten Intervall keine Münzen abgehoben haben. Für ein aktualisierbares elektronisches Geldsystem fordern wir daher, dass die Anonymität der Kunden innerhalb jedes einzelnen Intervalls gewährleistet bleibt (vgl. Abschnitt 5.2.1).

Langfristig sichere elektronische Geldsysteme sind formal wie folgt definiert:

Definition 5.3.2 (**Langfristig sichere elektronische Geldsysteme**)**.** Ein aktualisierbares elektronisches Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ heißt (stark) (t, N)-langfristig sicher, falls es die folgenden Eigenschaften erfüllt:

1. *Identitäts-Nachweis*: Es sei Coin eine von $\mathcal{K}$ in einem nicht kompromittierten Intervall bei $\mathcal{B}$ abgehobene Münze. Falls Coin zweimal bei $\mathcal{B}$ eingelöst wird, kann $\mathcal{B}$ anhand der Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_0}(D_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}(D_1)$ der beiden Einlöse-Protokolle D_0, D_1 den Betrug aufdecken und mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers effizient berechnen.

2. *(Starke) Nichterweiterbarkeit*: Für alle effizienten $\mathcal{A}$, die höchstens $t < N$ Anfragen an das Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*, \mathsf{sk}_{\mathcal{B}}^{(0)}}$ stellen, ist die Erfolgswahrscheinlichkeit von $\mathcal{A}$ in Spiel 5.3.1 bzw. Spiel 5.3.2 vernachlässigbar in k.

3. *(Starke) Unfälschbarkeit*: Für alle effizienten $\mathcal{A}$, die höchstens $t < N$ Anfragen an das Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*, \mathsf{sk}_{\mathcal{B}}^{(0)}}$ stellen, ist die Erfolgswahrscheinlichkeit von $\mathcal{A}$ in Spiel 5.3.3 bzw. Spiel 5.3.4 vernachlässigbar in k.

4. *Anonymität*: Es gibt keinen effizienten Angreifer $\mathcal{A}$, der bei Eingabe zweier im gleichen Intervall abgehobener Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ $(b, \bar{b} \in \{0,1\}, b \neq \bar{b})$, den Protokollansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ und den $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ entsprechenden Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört. Dabei ist ν eine nicht vernachlässigbare Funktion.

Analog zu Definition 5.2.3 gewährleistet ein aktualisierbares elektronisches Geldsystem eine sichere Schlüsselaktualisierung, falls folgende Bedingung erfüllt ist.

Definition 5.3.3. Ein aktualisierbares elektronisches Geldsystem Ω verfügt über eine *sichere Schlüsselaktualisierung*, falls $\mathsf{view}_{\mathcal{B}}^{\mathcal{T}}\left(\mathsf{BKeyUpdate}\left(\mathcal{B}\left(i, j, \mathsf{sk}_{\mathcal{B}}^{i}\right), \mathcal{T}\left(\mathsf{sk}_{\mathcal{B}}^{*}\right)\right)\right)$ von einem Angreifer $\mathcal{A}$, der in den beiden Intervallen i und j einen Schlüssel-Angriff durchführt, perfekt simuliert werden kann.

Bemerkung 5.3.2. Das in diesem Abschnitt definierte Sicherheitsmodell langfristig sicherer elektronischer Geldsysteme berücksichtigt keinen Generalschlüssel-Angriff (vgl. Abschnitt 5.1). Da der Tresor ausschließlich mit der Bank kommuniziert, ist die Schlüsselaktualisierung über eine direkte Verbindung zwischen Bank und Tresor möglich. Aus diesem Grund kann erreicht werden, dass lediglich die Bank Zugriff auf den Tresor erhält; insbesondere haben potentielle Angreifer keine Möglichkeit auf den Tresor zuzugreifen. In einem elektronischen Geldsystem stellt ein Generalschlüssel-Angriff daher ein sehr unwahrscheinliches Szenario dar. Gegen einen solchen Angriff kann man sich aber mit den in [DKXY03] beschriebenen Methoden absichern.

5.3.2 Generische langfristig sichere elektronische Geldsysteme

In Analogie zu Abschnitt 5.2.2 kann auch jedes beliebige elektronische Geldsystem, das im Sinne von Definition 3.1.2 sicher ist, dazu verwendet werden, ein stark $(N-1, N)$-langfristig sicheres elektronisches Geldsystem zu entwerfen. Die dem System zugrunde liegenden Ideen sind in Abbildung 5.4 skizziert.
Aufbauend auf einem elektronischen Geldsystem Θ mit Sicherheitsparameter $k \in \mathbb{N}$ und den Algorithmen bzw. Protokollen `BKeyGen`, `Reg`, `Withdrawal`, `Payment`, `Deposit` und `Identify` kann folgendermaßen ein stark $(N-1, N)$-langfristig sicheres elektronisches Geldsystem Ω entwickelt werden:

Schlüsselgenerierung BKeyGen:	
Tresor $\mathcal{T}$	Bank $\mathcal{B}$
$(pk, sk) = \texttt{BKeyGen}(1^k)$ $\mathsf{sk}^*_{\mathcal{B}} = sk, \mathsf{pk}_{\mathcal{B}} = pk$	$\xrightarrow{\mathsf{pk}_{\mathcal{B}}}$ Speichert $\mathsf{pk}_{\mathcal{B}}$

Schlüsselaktualisierung BKeyUpdate:	
Tresor $\mathcal{T}$	Bank $\mathcal{B}$
$(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}}) = \texttt{BKeyGen}(1^k)$, $s_i = \texttt{Sign}\left(pk^{(i)}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^*_{\mathcal{B}}\right)$ $\mathsf{cert}^{(i)}_{\mathcal{B}} = (pk^{(i)}_{\mathcal{B}}, s_i)$	$\xrightarrow{(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}}), \mathsf{cert}^{(i)}_{\mathcal{B}}}$ Speichert $\mathsf{cert}^{(i)}_{\mathcal{B}}$ $\mathsf{sk}^{(i)}_{\mathcal{B}} = sk^{(i)}_{\mathcal{B}}$

Abhebe-Protokoll Withdrawal im i-ten Intervall:	
Kunde $\mathcal{K}$	Bank $\mathcal{B}$
Verifiziere cert_i	$\xleftarrow{\mathsf{cert}^{(i)}_{\mathcal{B}}}$ $\xleftrightarrow{\texttt{Withdrawal}\left(\mathcal{K}(pk^{(i)}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}), \mathcal{B}(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}})\right)}$ $\mathsf{Coin} = (m, \langle i, s\rangle)$

Bezahl-Protokoll Payment im i-ten Intervall:	
Kunde $\mathcal{K}$	Händler $\mathcal{H}$
$\xleftrightarrow{\texttt{Payment}\left(\mathcal{K}\left(pk^{(i)}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{Coin}\right), \mathcal{H}\left(\mathsf{pk}_{\mathcal{H}}, pk^{(i)}_{\mathcal{B}}\right)\right)}$	

Abbildung 5.4: Das generische langfristig sichere elektronische Geldsystem

Schlüsselgenerierung: Um einen Schlüssel $(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}})$ zu generieren, wird bei Eingabe des Sicherheitsparameters 1^k und der polynomiell in k beschränkten Anzahl $N \in \mathbb{N}$ der Intervalle vom Schlüsselgenerierungsalgorithmus $\mathsf{BKeyGen}(1^k, N)$ der Bank $\mathcal{B}$ durch den Aufruf von $\mathtt{BKeyGen}(1^k)$ das Schlüsselpaar (pk, sk) erzeugt. Der private Schlüssel sk wird als Generalschlüssel $\mathsf{sk}^*_{\mathcal{B}}$ in dem Tresor $\mathcal{T}$ der Bank $\mathcal{B}$ gespeichert und anschließend gelöscht. Die Ausgabe von $\mathsf{Gen}(1^k, N)$ ist das Tripel

$$(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}}) = (pk, sk, \perp).$$

Schlüsselaktualisierung: Zur Aktualisierung des privaten Schlüssels $\mathsf{sk}^{(i)}_{\mathcal{B}}$ wird zu Beginn des j-ten Intervalls das Protokoll $\mathsf{BKeyUpdate}(\mathcal{B}(i, j, \mathsf{sk}^{(i)}_{\mathcal{B}}), \mathcal{T}(\mathsf{sk}^*_{\mathcal{B}}))$ durchgeführt. Durch den Aufruf von $\mathtt{BKeyGen}(1^k)$ generiert der Tresor $\mathcal{T}$ für das Intervall j ein neues Schlüsselpaar $(pk^{(j)}_{\mathcal{B}}, sk^{(j)}_{\mathcal{B}}) = \mathtt{BKeyGen}(1^k)$ und erstellt das Zertifikat $\mathsf{cert}^{(j)}_{\mathcal{B}} = (pk^{(j)}_{\mathcal{B}}, s_j)$ mit $s_j = \mathtt{Withdrawal}(\mathcal{T}(pk^{(j)}_{\mathcal{B}}, pk), \mathcal{T}(pk, sk))$. Die Bank $\mathcal{B}$ erhält den privaten Schlüssel $\mathsf{sk}^{(j)}_{\mathcal{B}} = (sk^{(j)}_{\mathcal{B}}, \mathsf{cert}^{(j)}_{\mathcal{B}})$ und löscht den bisherigen privaten Schlüssel $\mathsf{sk}^{(i)}_{\mathcal{B}}$.

Kontoeröffnung: Möchte ein Kunde $\mathcal{K}$ ein Konto bei der Bank $\mathcal{B}$ eröffnen, so wird zwischen $\mathcal{K}$ und der Bank $\mathcal{B}$ das Kontoeröffnungs-Protokoll $\mathsf{Reg}(\mathcal{K}(\mathsf{pk}_{\mathcal{B}}), \mathcal{B}(\mathsf{pk}_{\mathcal{B}}))$ durchgeführt, das der Kontoeröffnung $\mathtt{Reg}(\mathcal{K}(\mathsf{pk}_{\mathcal{B}}), \mathcal{B}(\mathsf{pk}_{\mathcal{B}}))$ entspricht. Der Kunde $\mathcal{K}$ erhält ein Schlüsselpaar $(\mathsf{pk}_{\mathcal{K}}, \mathsf{sk}_{\mathcal{K}})$.

Abheben einer Münze: Um im i-ten Intervall eine Münze Coin bei der Bank $\mathcal{B}$ abzuheben, wird zwischen dem Kunden $\mathcal{K}(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}})$ und der Bank $\mathcal{B}\left(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^{(i)}_{\mathcal{B}}\right)$ das Abhebe-Protokoll $\mathsf{Withdrawal}\left(\mathcal{K}\left(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}\right), \mathcal{B}\left(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^{(i)}_{\mathcal{B}}\right)\right)$ durchgeführt. Dazu führen $\mathcal{K}$ und $\mathcal{B}$ das Protokoll $\mathtt{Withdrawal}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}\right), \mathcal{B}\left(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}}\right)\right)$ aus, an dessen Ende $\mathcal{K}$ die elektronische Münze $\mathsf{Coin}' = (m, \tilde{s})$ erhält, die aus einer Nachricht m und der Signatur $\tilde{s}$ besteht. Am Ende des Protokolls $\mathsf{Withdrawal}\left(\mathcal{K}\left(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}\right), \mathcal{B}\left(i, \mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^{(i)}_{\mathcal{B}}\right)\right)$ erhält $\mathcal{K}$ die Münze $\mathsf{Coin} = (m, \langle i, s\rangle) = \left(m, \left\langle i, \left(\tilde{s}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$.

Die Gültigkeit einer Münze $\mathsf{Coin} = \left(m, \left\langle i, \left(\tilde{s}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$ wird durch den Algorithmus $\mathsf{Verify}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{Coin}\right)$ verifiziert. Zur Verifikation von Coin wird von $\mathsf{Verify}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{Coin}\right)$ anhand von $\mathsf{pk}_{\mathcal{B}}$ zunächst die Gültigkeit des Zertifikats $\mathsf{cert}^{(i)}_{\mathcal{B}}$ überprüft. Ist $\mathsf{cert}^{(i)}_{\mathcal{B}}$ gültig, so gibt $\mathsf{Verify}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{Coin}\right)$ entsprechend $\mathtt{Verify}\left(m, pk^{(i)}_{\mathcal{B}}, \tilde{s}\right)$ aus und 0 sonst.

Bemerkung 5.3.3. Da zur Verifikation der Münze $\mathsf{Coin} = \left(m, \left\langle i, \left(\tilde{s}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$ neben der „eigentlichen Signatur“ $\tilde{s}$ von Coin zusätzlich die Gültigkeit des Zertifikates $\mathsf{cert}^{(i)}_{\mathcal{B}}$ überprüft werden muss, verdoppelt sich im Vergleich zum Basisgeldsystem Θ der Rechenaufwand. Da Bank und Händler in praktischen Anwendungen relativ häufig Münzen auf ihre Echtheit überprüfen müssen, sollte die Verifikation elektronischer Münzen möglichst effizient realisiert werden. In Kapitel 6 werden langfristig sichere Geldsysteme entwickelt, in denen die Verifikation der Münzen genauso effizient realisiert werden kann, wie in dem System, auf dem sie basieren.

Bezahlen mit einer Münze: Falls der Kunde $\mathcal{K}$ mit einer elektronischen Münze $\mathsf{Coin} = (m, \langle i, s\rangle)$ bei einem Händler $\mathcal{H}$ bezahlen möchte, wird zwischen beiden Parteien das Bezahl-Protokoll $\mathsf{Payment}(\mathcal{K}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{Coin}), \mathcal{H}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{H}}))$ durchgeführt. Nachdem der Händler $\mathcal{H}$ die Gültigkeit der Münze Coin überprüft hat, wird das Protokoll $\mathtt{Payment}\left(\mathcal{K}\left(pk_{\mathcal{B}}^{(i)}, \mathsf{sk}_{\mathcal{K}}, (m, \tilde{s})\right), \mathcal{H}\left(\mathsf{pk}_{\mathcal{H}}, pk_{\mathcal{B}}^{(i)}\right)\right)$ ausgeführt.

Einlösen einer Münze: Möchte der Händler $\mathcal{H}$ die vom Kunden $\mathcal{K}$ erhaltene Münze Coin bei der Bank einlösen, wird das Protokoll $\mathsf{Deposit}(\mathcal{H}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{H}}, \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)), \mathcal{B}(\mathsf{pk}_{\mathcal{B}}))$ zwischen beiden Parteien ausgeführt. Die Durchführung des Protokolls entspricht dem Protokoll $\mathtt{Deposit}\left(\mathcal{H}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{H}}, \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}\right)\right)$.

Identifikation eines Double-Spenders: Um einen möglichen Double-Spender zu identifizieren, führt die Bank $\mathcal{B}$ den Algorithmus $\mathsf{Identify}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}(D_1), \mathsf{view}_{\mathcal{B}}^{\mathcal{H}_2}(D_2)\right)$ aus, der dem Algorithmus $\mathtt{Identify}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}_1}(D_1), \mathsf{view}_{\mathcal{B}}^{\mathcal{H}_2}(D_2)\right)$ entspricht.

Sicherheitsanalyse

Satz 5.3.1. Ist Θ ein im Sinne von Definition 3.1.2 sicheres elektronisches Geldsystem, so ist das aktualisierbare elektronische Geldsystem Ω ein stark $(N-1, N)$-langfristig sicheres elektronisches Geldsystem mit sicherer Schlüsselaktualisierung.

Beweis. Die sichere Schlüsselaktualisierung von Ω folgt direkt aus Satz 5.2.1.

Sei k der Sicherheitsparameter von Ω. Für die starke $(N-1, N)$-langfristige Sicherheit zeigen wir zunächst, dass Ω Anonymität im Sinne von Definition 5.3.2 erfüllt.

Anonymität: Angenommen Ω ist nicht anonym. Dann gibt es einen effizienten Angreifer $\mathcal{A}$, der bei Eingabe zweier Münzen

$$\mathsf{Coin}_b = \left(m_b, \left\langle i, \left(\tilde{s}_b, \mathsf{cert}_{\mathcal{B}}^{(i)}\right)\right\rangle\right) \text{ und } \mathsf{Coin}_{\bar{b}} = \left(m_{\bar{b}}, \left\langle i, \left(\tilde{s}_{\bar{b}}, \mathsf{cert}_{\mathcal{B}}^{(i)}\right)\right\rangle\right)$$

mit $b, \bar{b} \in \{0,1\}$ und $b \neq \bar{b}$, den Ansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ sowie den zu den beiden Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ gehörenden Ansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört. Dabei ist ν eine nicht vernachlässigbare Funktion.
Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der versucht unter Verwendung von $\mathcal{A}$ die Anonymität von Θ zu brechen. Der Angreifer $\mathcal{A}'$ erhält als Eingabe die beiden Münzen

$$\mathsf{Coin}'_b = (m'_b, s'_b) \text{ und } \mathsf{Coin}'_{\bar{b}} = (m'_{\bar{b}}, s'_{\bar{b}}),$$

die Protokollansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P'_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P'_{\bar{b}})$ und die zu $\mathsf{Coin}'_b, \mathsf{Coin}'_{\bar{b}}$ gehörenden Ansichten $\mathsf{view}_{\mathcal{B}'}^{\mathcal{K}_0}(W'_0)$, $\mathsf{view}_{\mathcal{B}'}^{\mathcal{K}_1}(W'_1)$. Die beiden Münzen $\mathsf{Coin}'_b, \mathsf{Coin}'_{\bar{b}}$ wurden bei der Bank $\mathcal{B}'$ abgehoben und dem Angreifer $\mathcal{A}'$ ist der öffentliche Schlüssel $pk_{\mathcal{B}'}$ der Bank $\mathcal{B}'$ bekannt. Um die Anonymität von Θ zu brechen geht $\mathcal{A}'$ wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ generiert das Schlüsselpaar $(pk, sk) = \texttt{BKeyGen}(1^k)$ und setzt $(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}})) = (pk, sk, \perp)$ sowie $pk^{(i)}_{\mathcal{B}} = pk_{\mathcal{B}'}$.

Schritt 2: Der Angreifer $\mathcal{A}'$ erstellt mit dem privaten Schlüssel sk das entsprechende Zertifikat $\mathsf{cert}^{(i)}_{\mathcal{B}} = (pk^{(i)}_{\mathcal{B}}, s_i)$ mit $s_i = \texttt{Withdrawal}(\mathcal{A}'(pk^{(i)}_{\mathcal{B}}, pk), \mathcal{A}'(pk, sk))$.

Schritt 3: Der Angreifer $\mathcal{A}'$ erstellt die Münzen $\mathsf{Coin}_b = \left(m'_b, \left\langle i, \left(s'_b, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$ und $\mathsf{Coin}_{\bar{b}} = \left(m'_{\bar{b}}, \left\langle i, \left(s'_{\bar{b}}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$. Anschließend werden von $\mathcal{A}'$ die entsprechenden Ansichten $\mathsf{view}^{\mathcal{K}_b}_{\mathcal{H}_b}(P_b) = \mathsf{view}^{\mathcal{K}_b}_{\mathcal{H}_b}(P'_b)$, $\mathsf{view}^{\mathcal{K}_{\bar{b}}}_{\mathcal{H}_{\bar{b}}}(P_{\bar{b}}) = \mathsf{view}^{\mathcal{K}_{\bar{b}}}_{\mathcal{H}_{\bar{b}}}(P'_{\bar{b}})$ und $\mathsf{view}^{\mathcal{K}_0}_{\mathcal{B}}(W_0) = \mathsf{view}^{\mathcal{K}_0}_{\mathcal{B}'}(W'_0), \mathsf{view}^{\mathcal{K}_1}_{\mathcal{B}}(W_1) = \mathsf{view}^{\mathcal{K}_1}_{\mathcal{B}'}(W'_1)$ erstellt, die $\mathcal{A}$ zusammen mit den Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ an $\mathcal{A}$ übergibt.

Mit der Wahrscheinlichkeit $\epsilon(k)$ kann $\mathcal{A}$ entscheiden, Coin_b zu W_0 oder W_1 gehört. Somit kann $\mathcal{A}'$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, ob Coin'_b zu W'_0 oder W'_1 gehört. Da $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ ist, ist dies ein Widerspruch zur Sicherheit von Θ.

Identitäts-Nachweis: Angenommen der Identitäts-Nachweis ist in Ω nicht gewährleistet, dann gibt es ein nicht kompromittiertes Intervall i und eine im i-ten Intervall bei $\mathcal{B}$ abgehobene Münze

$$\mathsf{Coin} = (m, \langle i, s\rangle) = \left(m, \left\langle i, \left(\tilde{s}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right),$$

die zweimal bei $\mathcal{B}$ eingelöst wurde, ohne dass $\mathcal{B}$ anhand der Ansichten $\mathsf{view}^{\mathcal{H}_0}_{\mathcal{B}}(D_0)$ und $\mathsf{view}^{\mathcal{H}_1}_{\mathcal{B}}(D_1)$ die Identität des Betrügers mit einer in k überwältigenden Wahrscheinlichkeit effizient berechnen kann.
Das heißt, die im i-ten Intervall durch $\texttt{Withdrawal}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}\right), \mathcal{B}\left(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}}\right)\right)$ abgehobene Münze $(m, \tilde{s})$, die anschließend mit `Payment` zweimal ausgegeben wurde, kann von $\mathcal{B}$ nicht mit einer in k überwältigenden Wahrscheinlichkeit dem Betrüger zugeordnet werden. Dies ist aber ein Widerspruch zur Sicherheit von Θ.

Starke Nichterweiterbarkeit: Angenommen Ω ist nicht sicher im Sinne der starken Nichterweiterbarkeit, dann gibt es einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.3.2 mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ gewinnt. Wir gehen analog zu Satz 5.2.1 vor. Im i-ten Intervall kann $\mathcal{A}$ zwei Strategien verfolgen, um die starke Nichterweiterbarkeit von Ω zu brechen.

1. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel $pk^{(i)}_{\mathcal{B}}$ mehr als ℓ_i gültige Münzen zu erzeugen. Wobei ℓ_i die Anzahl der von $\mathcal{A}$ für das i-te Intervall angeforderten ℓ_i Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W^{(i)}_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_{\ell_i}}(W^{(i)}_{\ell_i})$ ist.

2. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel $pk^{(i)}_{\mathcal{B}}$ ohne Interaktion mit $\mathcal{B}$ eine gültige Münze zu erzeugen, oder $\mathcal{A}$ erzeugt sich ein eigenes Schlüsselpaar $(pk^{(i)}_{\mathcal{B}}, sk^{(i)}_{\mathcal{B}})$ und versucht das Zertifikat cert_i zu fälschen.

1.Fall: Wir konstruieren einen Angreifer effizienten $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht die Sicherheit von Θ im Sinne von Definition 3.1.2 zu brechen. Sei ℓ die in k

polynomiell beschränkte Anzahl der Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)$ die $\mathcal{A}$ insgesamt während seines Angriffs erhält. Der Angreifer $\mathcal{A}'$ erhält Ansichten durchgeführter Abhebe-Protokolle mit der Bank $\mathcal{B}'$, die das Schlüsselpaar $(pk', sk') = \mathtt{BKeyGen}(1^k)$ erzeugt hat. Der Angreifer $\mathcal{A}'$ geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ generiert das Schlüsselpaar $(pk, sk) = \mathtt{BKeyGen}(1^k)$ und setzt $(\mathsf{pk}, \mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}}) = (pk, sk, \perp)$. Der Angreifer $\mathcal{A}$ erhält pk als Eingabe. Die Rolle der Bank $\mathcal{B}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}}}$ werden von $\mathcal{A}'$ übernommen.

Schritt 2: Der Angreifer $\mathcal{A}'$ wählt zufällig einen Index $r \in_{\mathcal{R}} \{1, \ldots, \ell\}$. Sei i^* der Index des Intervalls, für das $\mathcal{A}$ die insgesamt r-te Protokollansicht $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_r}(W_r)$ angefordert hat. Falls $\mathcal{A}$ vor der r-ten Anfrage bereits Protokollansichten für das i^*-te Intervall angefordert hat, bricht $\mathcal{A}'$ seinen Versuch ab.

Schritt 3: Der Angreifer $\mathcal{A}'$ verwendet die Bank $\mathcal{B}'$ zur Erstellung der r-ten Protokollansicht $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_r}(W_r)$. Sollte $\mathcal{A}$ in den darauffolgenden $\ell - r$ Anfragen erneut Protokollansichten für das i^*-te Intervall anfordern, wird von $\mathcal{A}'$ genauso wie bei der r-ten Anfrage die Bank $\mathcal{B}'$ zur Erstellung der Protokollansichten verwendet.

Schritt 4: Der Angreifer $\mathcal{A}'$ bricht seinen Versuch ab, falls $\mathcal{A}$ den Index i^* als Anfrage an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}}}$ stellt.

Schritt 5: Alle anderen Anfragen von $\mathcal{A}$ kann $\mathcal{A}'$ wie folgt beantworten: Für die entsprechenden Intervalle generiert $\mathcal{A}'$ Schlüsselpaare $(pk^{(j)}_{\mathcal{B}}, sk^{(j)}_{\mathcal{B}}) = \mathtt{BKeyGen}(1^k)$ und erstellt mit dem Schlüsselpaar (pk, sk) die zugehörigen Zertifikate $\mathsf{cert}^{(j)}_{\mathcal{B}} = \mathtt{Withdrawal}(\mathcal{A}'(pk, pk_j), \mathcal{A}'(pk, sk))$. Anschließend kann $\mathcal{A}'$ die angeforderten Protokollansichten erstellen. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}}, \mathsf{sk}^{(0)}_{\mathcal{B}}}$ antworten.

Gibt $\mathcal{A}$ nach ℓ_i in einem Intervall i angeforderten Protokollansichten mindestens $\ell_i + 1$ gültige Münzen

$$\mathsf{Coin}_1 = \left(m_1, \left\langle i, \left(\tilde{s}_1, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right), \ldots, \mathsf{Coin}_{\ell_i+1} = \left(m_{\ell_i+1}, \left\langle i, \left(\tilde{s}_{\ell_i+1}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$$

aus, so gibt $\mathcal{A}'$ die $\ell_i + 1$ gültigen Münzen $(m_1, \tilde{s}_1,), \ldots, (m_{\ell_i+1}, \tilde{s}_{\ell_i+1})$ aus.
Mit einer Wahrscheinlichkeit von $\frac{1}{\ell}$ wird der Versuch von $\mathcal{A}'$ nicht abgebrochen und es ist $i^* = i$. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im ersten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ die Sicherheit von Θ zu brechen:

$$\epsilon'(k) \geq \frac{\epsilon(k)}{\ell}.$$

Da ℓ polynomiell in k beschränkt und $\epsilon'(k)$ in k vernachlässigbar ist (Θ ist ein sicheres elektronisches Geldsystem), folgt, dass $\epsilon(k)$ ebenfalls in k vernachlässigbar ist. Dies steht aber im Widerspruch zur Annahme.

2.Fall: Erneut konstruieren wir einen effizienten Angreifer $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht, die Sicherheit von Θ im Sinne von Definition 3.1.2 zu brechen. Der Angreifer $\mathcal{A}'$ erhält Ansichten durchgeführter Abhebe-Protokolle mit der Bank $\mathcal{B}'$, die das Schlüsselpaar $(pk', sk') = \mathtt{BKeyGen}(1^k)$ erzeugt hat. Er geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ setzt $\mathsf{pk} = pk'$ und somit implizit $\mathsf{sk}^* = sk'$. Der Angreifer $\mathcal{A}$ erhält pk als Eingabe (die Rolle des Signierers $\mathcal{S}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}^*,\mathsf{sk}_0}$ werden von $\mathcal{A}'$ übernommen).

Schritt 2: Da $\mathcal{A}$ für das i-te Intervall keine Protokollansichten anfordert, kann $\mathcal{A}'$ alle von $\mathcal{A}$ angeforderten Protokollansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)$ wie folgt erstellen: Der Angreifer $\mathcal{A}'$ erzeugt sich die entsprechenden Schlüsselpaare $(pk_j, sk_j) = \mathtt{BKeyGen}(1^k)$ und erstellt die Zertifikate cert_j, indem $\mathcal{A}'$ sich pk_j von $\mathcal{B}'$ blind signieren lässt. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{B}},\mathsf{sk}^{(0)}_{\mathcal{B}}}$ antworten. Mit ℓ bezeichnen wir die Anzahl der von $\mathcal{A}'$ erstellten Zertifikate.

Gibt $\mathcal{A}$ die Münze $\mathsf{Coin} = (m, \langle i, (\tilde{s}, \mathsf{cert}_i)\rangle) = (m, \langle i, (\tilde{s}, (pk_i, s_i))\rangle)$ aus, so gibt $\mathcal{A}'$ die Münze (pk_i, s_i) aus. Da $\mathcal{A}'$ sich den Schlüssel pk_i nicht von $\mathcal{B}'$ hat blind signieren lassen, hat $\mathcal{A}'$ anhand von ℓ Protokollansichten mindestens $\ell + 1$ gültige Münzen erzeugt. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im zweiten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ das blinde Signaturverfahren Θ zu brechen:

$$\epsilon'(k) = \epsilon(k).$$

Da Θ ein sicheres blindes Signaturverfahren ist, ist $\epsilon'(k)$ und damit $\epsilon(k)$ vernachlässigbar in k, was im Widerspruch zur Annahme steht.

Starke Unfälschbarkeit: Angenommen Ω ist nicht sicher im Sinne der starken Unfälschbarkeit. Dann gibt es einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.3.4 mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ gewinnt. Das heißt, der Angreifer $\mathcal{A}$ kann mit der Wahrscheinlichkeit $\epsilon(k)$ eine gültige Münze Coin erstellen, die zweimal erfolgreich ausgegeben werden kann, ohne dass $\mathcal{B}$ mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers aufdecken kann. Genauso wie im Fall der starken Nichterweiterbarkeit kann $\mathcal{A}$ im i-ten Intervall die beiden folgenden Strategien verwenden:

1. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel pk_i eine solche Münze Coin zu erstellen, wobei $\mathcal{A}$ während seines Angriffs insgesamt ℓ_i Abhebe-Protokollansichten für das i-te Intervall angefordert hat.

2. Der Angreifer $\mathcal{A}$ versucht unter dem im i-ten Intervall gültigen öffentlichen Schlüssel pk_i ohne Abhebe-Protokollansichten des i-ten Intervalls eine solche Münze Coin zu erstellen, oder $\mathcal{A}$ erzeugt sich ein eigenes Schlüsselpaar (pk_i, sk_i) und versucht, das Zertifikat cert_i zu fälschen.

Im ersten Fall können wir analog zur starken Nichterweiterbarkeit einen effizienten Angreifer $\mathcal{A}'$ konstruieren, der unter Verwendung von $\mathcal{A}$ versucht, die Sicherheit von Θ im Sinne von Definition 3.1.2 zu brechen. Sei ℓ die in k polynomiell beschränkte Anzahl der Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)$, die $\mathcal{A}$ insgesamt während seines Angriffs erhält. Der Angreifer $\mathcal{A}'$ erhält Ansichten durchgeführter Abhebe-Protokolle mit der Bank $\mathcal{B}'$, die das Schlüsselpaar $(pk', sk') = \mathtt{BKeyGen}(1^k)$ erzeugt hat, und geht völlig analog zur starken Nichterweiterbarkeit vor.
Gibt $\mathcal{A}$ in einem Intervall i eine gültige Münze $\mathsf{Coin} = \left(m, \left\langle i, \left(\tilde{s}, \mathsf{cert}^{(i)}_{\mathcal{B}}\right)\right\rangle\right)$ aus, die zweimal erfolgreich ausgegeben werden kann, ohne dass $\mathcal{B}$ mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers aufdecken kann, so gibt $\mathcal{A}'$ die Münze $(m, \tilde{s})$ aus.
Mit einer Wahrscheinlichkeit von $\frac{1}{\ell}$ wird der Angriff von $\mathcal{A}'$ nicht abgebrochen und es ist $i^* = i$. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im ersten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ die Sicherheit von Θ zu brechen:

$$\epsilon'(k) \geq \frac{\epsilon(k)}{\ell}.$$

Da ℓ polynomiell in k beschränkt und $\epsilon'(k)$ in k vernachlässigbar ist (Θ ist ein sicheres elektronisches Geldsystem), folgt das $\epsilon(k)$ ebenfalls in k vernachlässigbar ist. Dies steht aber im Widerspruch zur Annahme.

Im zweiten Fall können wir ebenfalls einen effizienten Angreifer $\mathcal{A}'$ konstruieren, der unter Verwendung von $\mathcal{A}$ versucht, die Sicherheit von Θ im Sinne von Definition 3.1.2 zu brechen. Der Angreifer $\mathcal{A}'$ erhält Ansichten durchgeführter Abhebe-Protokolle mit der Bank $\mathcal{B}'$, die das Schlüsselpaar $(pk', sk') = \mathtt{BKeyGen}(1^k)$ erzeugt hat, und geht völlig analog zur starken Nichterweiterbarkeit vor.
Gibt $\mathcal{A}$ die Münze $\mathsf{Coin} = (m, \langle i, (\tilde{s}, \mathsf{cert}_i)\rangle) = (m, \langle i, (\tilde{s}, (pk_i, s_i))\rangle)$ aus, so gibt $\mathcal{A}'$ die Münze (pk_i, s_i) aus. Da $\mathcal{A}'$ sich den Schlüssel pk_i nicht von $\mathcal{B}'$ hat blind signieren lassen, hat $\mathcal{A}'$ anhand von ℓ Protokollansichten mindestens $\ell + 1$ gültige Münzen erzeugt. Ist $\epsilon(k)$ die Erfolgswahrscheinlichkeit von $\mathcal{A}$ im zweiten Fall, so gilt für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ das blinde Signaturverfahren Θ zu brechen:

$$\epsilon'(k) = \epsilon(k).$$

Da Θ ein sicheres blindes Signaturverfahren ist, ist $\epsilon'(k)$ und damit $\epsilon(k)$ vernachlässigbar in k, im Widerspruch zur Annahme. □

5.4 Langfristig sichere faire Geldsysteme

In einem fairen elektronischen Geldsystem werden Kunden- bzw. Münztracing durch die Verschlüsselung der Kunden-Identität bzw. eines Münzidentifikators realisiert. Damit in einem solchen System die Privatsphäre der Kunden möglichst optimal vor einer Kompromittierung des Deanonymisierers geschützt ist, ist es naheliegend, auch die privaten

Schlüssel des Deanonymisierers zu aktualisieren und aus diesem Grund für die Tracing-Mechanismen ein aktualisierbares Public-Key-Verschlüsselungsverfahren (vgl. Definition 5.1.1) einzusetzen. Ein faires elektronisches Geldsystem, dass diese Eigenschaft erfüllt und in dem zusätzlich die Bank ihren privaten Schlüssel regelmäßig aktualisiert, wird auch als *aktualisierbares faires elektronisches Geldsystem* bezeichnet und ist formal wie folgt definiert:

Definition 5.4.1 (**Aktualisierbare faire elektronische Geldsysteme**)**.** Es sei $k \in \mathbb{N}$ der Sicherheitsparameter, M die Menge aller Nachrichten und $N \in \mathbb{N}$ die polynomiell in k beschränkte Anzahl der Intervalle. Ein *aktualisierbares faires elektronisches Geldsystem* ist ein aktualisierbares elektronisches Geldsystem (vgl. Definition 5.3.1) und besteht zusätzlich aus folgenden Protokollen und Algorithmen:

1. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{DKeyGen}$ des Deanonymisierers $\mathcal{D}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k, der Anzahl N der Intervalle und des öffentlichen Schlüssels $\mathsf{pk}_\mathcal{B}$ der Bank $\mathcal{B}$ einen öffentlichen Schlüssel $\mathsf{pk}_\mathcal{D}$, einen Generalschlüssel $\mathsf{sk}^*_\mathcal{D}$ und einen privaten Schlüssel $\mathsf{sk}^0_\mathcal{D}$ erzeugt. Die Ausgabe von $\mathsf{DKeyGen}(1^k, N, \mathsf{pk}_\mathcal{B})$ ist das Tripel $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}^*_\mathcal{D}, \mathsf{sk}^0_\mathcal{D})$.

2. Das *Schlüsselaktualisierungsprotokoll* $\mathsf{DKeyUpdate}$ ist ein interaktives Protokoll zwischen dem Deanonymisierer $\mathcal{D}$ und seinem Tresor $\mathcal{T}_\mathcal{D}$. Für den Generalschlüssel $\mathsf{sk}^*_\mathcal{D}$, die Indizes der Intervalle i, j und den im i-ten Intervall gültigen privaten Schlüssel $\mathsf{sk}^{(i)}_\mathcal{D}$ ist $i, j, \mathsf{sk}^{(i)}_\mathcal{D}$ die Eingabe von $\mathcal{D}$ und $\mathsf{sk}^*_\mathcal{D}$ die Eingabe des Tresors $\mathcal{T}_\mathcal{D}$. Die Ausgabe von $\mathcal{D}(i, j, \mathsf{sk}^{(i)}_\mathcal{D})$ ist der im j-ten Intervall gültige private Schlüssel $\mathsf{sk}^j_\mathcal{D}$. Entsprechend ist die Ausgabe von $\mathcal{T}_\mathcal{D}(\mathsf{sk}^*_\mathcal{D})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

3. Das *Kundentracing-Protokoll* KTrace ist ein interaktives Protokoll zwischen der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$. Im i-ten Intervall ist $\mathsf{pk}_\mathcal{D}, \mathsf{sk}^{(i)}_\mathcal{D}$ die Eingabe des Deanonymisierers $\mathcal{D}$ und $\mathsf{view}^\mathcal{H}_\mathcal{B}(D)$ die Eingabe der Bank $\mathcal{B}$. Die Ausgabe der Bank $\mathcal{B}(\mathsf{view}^\mathcal{H}_\mathcal{B}(D))$ ist die Identität $\mathrm{id}_\mathcal{K}$ eines Kunden $\mathcal{K}$ oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{D}(\mathsf{pk}_\mathcal{D}, \mathsf{sk}^{(i)}_\mathcal{D})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

4. Das *Münztracing-Protokoll* MTrace ist ein interaktives Protokoll zwischen der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$. Im i-ten Intervall ist $\mathsf{view}^\mathcal{K}_\mathcal{B}(W)$ die Eingabe der Bank $\mathcal{B}$ und $\mathsf{pk}_\mathcal{D}, \mathsf{sk}^{(i)}_\mathcal{D}$ die Eingabe des Deanonymisierers $\mathcal{D}$. Die Ausgabe der Bank $\mathcal{B}(\mathsf{view}^\mathcal{K}_\mathcal{B}(W))$ ist die in W abgehobene Münze Coin oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{D}(\mathsf{pk}_\mathcal{D}, \mathsf{sk}^{(i)}_\mathcal{D})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

Bemerkung 5.4.1. In einigen fairen elektronischen Geldsystemen werden Kunden- und Münztracing vom Deanonymisierer mit unterschiedlichen privaten Schlüsseln durchgeführt, z.B. in [FTY98] oder den in den Abschnitten 6.2 und 6.3 entworfenen Systemen. Das heißt, in einem aktualisierbaren fairen Geldsystem kann der im i-ten Intervall gültige private Schlüssel $\mathsf{sk}^{(i)}_\mathcal{D}$ als Paar $\mathsf{sk}^{(i)}_\mathcal{D} = \left(\overline{\mathsf{sk}_\mathcal{D}}^{(i)}, \widehat{\mathsf{sk}_\mathcal{D}}^{(i)}\right)$ aufgefasst werden, wobei $\overline{\mathsf{sk}_\mathcal{D}}^{(i)}$

von $\mathcal{D}$ als Eingabe für KTrace und $\widehat{\mathsf{sk}_{\mathcal{D}}}^{(i)}$ als Eingabe für MTrace benutzt wird. Dementsprechend ist es sinnvoll, die privaten Schlüssel $\overline{\mathsf{sk}_{\mathcal{D}}}^{(i)}$ und $\widehat{\mathsf{sk}_{\mathcal{D}}}^{(i)}$ getrennt voneinander zu aktualisieren und das Schlüsselaktualisierungsprotokoll $\mathsf{DKeyUpdate}$ in zwei korrespondierende Aktualisierungsprotokolle $\mathsf{DKeyUpdate}_1$ und $\mathsf{DKeyUpdate}_2$ zu unterteilen.

Sicherheitsziele aktualisierbarer fairer Geldsysteme

Die Konsequenzen, die sich aus einer Kompromittierung des Deanonymisierers ergeben, wurden bereits in Abschnitt 4.1.2 umfassend angesprochen. Ziel eines sicheren aktualisierbaren fairen elektronischen Geldsystems soll es daher sein, neben den Interessen der Bank, auch die Privatsphäre der Kunden bestmöglich zu schützen. Das heißt, die Anonymität der Kunden soll, in den von einer Kompromittierung nicht betroffenen Intervallen, immer noch gewährleistet bleiben. Um die Kompromittierung des Deanonymisierers formal zu modellieren, erhält der Angreifer Zugriff auf das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{D}},\mathsf{sk}^0_{\mathcal{D}}}$, das ihm in üblicher Weise die entsprechenden privaten Schlüssel liefert.

Ein aktualisierbares faires elektronisches Geldsystem heißt *langfristig sicheres faires elektronisches Geldsystem*, falls es folgender Definition genügt:

Definition 5.4.2. Ein aktualisierbares faires elektronisches Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ heißt (stark) (t, N)-*langfristig sicheres faires elektronisches Geldsystem*, falls Ω ein (stark) (t, N)-langfristig sicheres elektronisches Geldsystem ist und es zusätzlich keinen effizienten Angreifer $\mathcal{A}$ gibt, der t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{D}},\mathsf{sk}^0_{\mathcal{D}}}$ stellt und bei Eingabe zweier im gleichen Intervall i abgehobener Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ $(b, \bar{b} \in \{0,1\}, b \neq \bar{b})$, den Protokollansichten $\mathsf{view}^{\mathcal{K}_b}_{\mathcal{H}_b}(P_b)$, $\mathsf{view}^{\mathcal{K}_{\bar{b}}}_{\mathcal{H}_{\bar{b}}}(P_{\bar{b}})$ und den $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ entsprechenden Protokollansichten $\mathsf{view}^{\mathcal{K}_0}_{\mathcal{B}}(W_0)$, $\mathsf{view}^{\mathcal{K}_1}_{\mathcal{B}}(W_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört. Dabei ist ν eine nicht vernachlässigbare Funktion und i wurde von $\mathcal{A}$ nicht als Anfrage an $\mathsf{Exp}_{\mathsf{sk}^*_{\mathcal{D}},\mathsf{sk}^0_{\mathcal{D}}}$ gestellt.

Wie man auf Basis des von Y. Frankel, Y. Tsiounis und M. Yung entwickelten fairen elektronischen Geldsystems [FTY98] und des in Abschnitt 5.1 beschriebenen Verschlüsselungsverfahrens ein (t, N)-langfristig sicheres faires elektronisches Geldsystem konstruieren kann, wird in Kapitel 6 gezeigt. Des Weiteren wird in Kapitel 7 ein generisches faires elektronisches Geldsystem vorgestellt, das unter Verwendung eines (t, N)-langfristig sicheren blinden Signaturverfahren und dem Einsatz eines geeigneten Public-Key-Verschlüsselungsverfahren (z.B. das Verfahren aus Abschnitt 5.1) in ein (t, N)-langfristig sicheres faires elektronisches Geldsystem transformiert werden kann.

Kapitel 6

Eine langfristig sichere faire elektronische Geldbörse

Für die Sicherheit fairer elektronischer Geldsysteme ist eine sichere Speicherung des Signaturschlüssels der Bank und des privaten Schlüssels des Deanonymisierers entscheidend. Welche Konsequenzen eine erfolgreiche Schlüsselkompromittierung in fairen elektronischen Geldsystemen nach sich zieht, wurde in Abschnitt 4.1 ausführlich diskutiert. Um diese Risiken im Sicherheitsmodell fairer elektronischer Geldsysteme zu berücksichtigen, wurde in Abschnitt 5.3 das Modell langfristig sicherer fairer elektronischer Geldsysteme eingeführt.
Ziel dieses Kapitels ist es, ein neues langfristig sicheres faires elektronisches Geldsystem zu entwickeln, das im Vergleich zu dem in Abschnitt 5.3.2 konstruierten generischen langfristig sicheren Geldsystem nicht nur vor der Kompromittierung der Bank, sondern auch vor der des Deanonymisierers schützt. Das neue langfristig sichere faire Geldsystem ist ein aktualisierbares faires Geldsystem. Es basiert auf dem fairen Geldsystem von Frankel *et al.* [FTY98] und verwendet als grundlegenden Baustein die langfristig sichere blinde Schnorr-Signatur aus Abschnitt 5.2.3. Im Gegensatz zu [FTY98] wird zur Realisierung des Kunden- und Münztracings nicht die ElGamal-Verschlüsselung, sondern das in [DKXY02] vorgestellte Key-Insulated Public-Key-Verschlüsselungsverfahren (vgl. Abschnitt 5.1) eingesetzt.
In Abschnitt 6.1 wird zunächst das System von Frankel *et al.* vorgestellt. Darauf aufbauend wird in Abschnitt 6.2 mit den bereits angesprochenen Techniken ein neues langfristig sicheres faires Geldsystem entwickelt und dessen Sicherheit analysiert. Damit Double-Spending von der Bank bereits im Voraus verhindert werden kann, wird dieses System in Abschnitt 6.3 zu einer langfristig sicheren fairen elektronischen Geldbörse erweitert.

6.1 Das System von Frankel, Tsiounis und Yung

Von den Autoren Y. Frankel, Y. Tsiounis und M. Yung wurde im Jahre 1998 ein faires elektronisches Geldsystem [FTY98] vorgestellt, das auf dem von S. Brands veröffentlich-

ten elektronischen Geldsystem [Bra94] basiert. Das System stellt eine Weiterentwicklung von [FTY96] dar und kommt ohne die dort eingeführten *Indirect Discourse Proofs* aus. Somit ist es aktuell neben [GT03] eines der effizientesten fairen elektronischen Geldsysteme.

6.1.1 Initialisierung des Systems

Das System wird durch die Bank $\mathcal{B}$ und den Deanonymisierer $\mathcal{D}$ initialisiert, die sich beide ein Schlüsselpaar $(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B})$ bzw. $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D})$ generieren und die öffentlichen Schlüssel $\mathsf{pk}_\mathcal{B}$ bzw. $\mathsf{pk}_\mathcal{D}$ publizieren. Anhand des öffentlichen Schlüssels $\mathsf{pk}_\mathcal{B}$ können Kunden und Händler die Echtheit elektronischer Münzen verifizieren. Der öffentliche Schlüssel $\mathsf{pk}_\mathcal{D}$ wird im Abhebe- und Bezahl-Protokoll dazu verwendet die Identität des Kunden bzw. einen Münz-Identifikator zu verschlüsseln, um bei begründetem Verdacht Kunden- und Münztracing mit Hilfe des Deanonymisierers $\mathcal{D}$ durchführen zu können.

Initialisierung der Bank: Zunächst initialisiert die Bank $\mathcal{B}$ durch $\mathsf{BKeyGen}(1^k)$ das System. Bei Eingabe des Sicherheitsparameters $k \in \mathbb{N}$ wählt $\mathcal{B}$ Primzahlen p und q mit $|p-1| = \delta + k$ für ein vorgegebenes $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein vorgegebenes $\gamma \in \mathbb{N}$. Die Bank $\mathcal{B}$ wählt zufällig Generatoren $g, g_1, \ldots, g_4$ der eindeutig bestimmten zyklischen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$. Anschließend wählt $\mathcal{B}$ zufällig einen Wert $X_\mathcal{B} \in_\mathcal{R} \mathbb{Z}_q$ und berechnet $h = g^{X_\mathcal{B}}, h_1 = g_1^{X_\mathcal{B}}, h_2 = g_2^{X_\mathcal{B}}, h_3 = g_3^{X_\mathcal{B}}$. Außerdem wählt $\mathcal{B}$ Hashfunktionen $\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$ aus einer Familie geeigneter kollisionsresistenter Hashfunktionen. Die Bank $\mathcal{B}$ veröffentlicht

$$\mathsf{pk}_\mathcal{B} = (p, q, g, g_1, \ldots, g_4, h, h_1, h_2, h_3, (\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots))$$

als ihren öffentlichen Schlüssel und speichert $\mathsf{sk}_\mathcal{B} = X_\mathcal{B}$ als ihren privaten Schlüssel.

Definition 6.1.1. Eine Signatur s der Bank $\mathcal{B}$ für ein Paar $(A, B) \in G_q \times G_q$ ist ein Tupel $(z, a, b, r) \in G_q \times G_q \times G_q \times \mathbb{Z}_q$, für das die Gleichungen $g^r = h^{\mathrm{H}(A,B,z,a,b)} a$ und $A^r = z^{\mathrm{H}(A,B,z,a,b)} b$ erfüllt sind. Eine Münze Coin ist ein Tripel $(A, B, (z, a, b, r))$.

Initialisierung des Deanonymisierers: Durch $\mathsf{DMKeyGen}(1^k, \mathsf{pk}_\mathcal{B})$ erzeugt der Deanonymisierer $\mathcal{D}$ ein Schlüsselpaar $(\mathsf{sk}_\mathcal{D}, \mathsf{pk}_\mathcal{D})$. Dazu wählt $\mathcal{D}$ zufällig einen Wert $X_\mathcal{D} \in_\mathcal{R} \mathbb{Z}_q$ und berechnet die beiden öffentlichen Schlüssel $f_2 = g_2^{X_\mathcal{D}}$ sowie $f_3 = g_3^{X_\mathcal{D}}$. Der Deanonymisierer $\mathcal{D}$ veröffentlicht $\mathsf{pk}_\mathcal{D} = (f_2, f_3)$ als seinen öffentlichen Schlüssel und speichert $\mathsf{sk}_\mathcal{D} = X_\mathcal{D}$ als seinen privaten Schlüssel.

6.1.2 Die Protokolle

Kontoeröffnung: Zur Kontoeröffnung eines neuen Kunden $\mathcal{K}$ wird zwischen $\mathcal{K}$ und $\mathcal{B}$ das Protokoll $\mathsf{Reg}(\mathcal{K}(\mathsf{pk}_\mathcal{B}), \mathcal{B}(\mathsf{pk}_\mathcal{B}))$ ausgeführt. Der Kunde $\mathcal{K}$ wählt zufällig einen Wert $u_1 \in_\mathcal{R} \mathbb{Z}_q$ und berechnet $I = g_1^{u_1}$. Ist $I g_2 \neq 1$, sendet $\mathcal{K}$ das Element I an $\mathcal{B}$ und $\mathcal{B}$ speichert I zusammen mit den persönlichen Daten von $\mathcal{K}$ in ihrer Datenbank. Um die in

[CFMT96] beschriebenen Angriffe zu vermeiden, beweist $\mathcal{K}$ der Bank $\mathcal{B}$ zusätzlich, dass er eine Darstellung von I bzgl. g_1 kennt; z.B. mit dem Schnorr-Identifikationsverfahren [Sch91] (vgl. Abschnitt 2.5.1).

Abheben einer Münze: Möchte $\mathcal{K}$ eine Münze Coin bei $\mathcal{B}$ abheben, authentifizieren sich $\mathcal{K}$ und $\mathcal{B}$ gegenseitig. Anschließend wird das in Abbildung 6.1 dargestellte Protokoll $\mathsf{Withdrawal}\left(\mathcal{K}\left(\mathsf{sk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right), \mathcal{B}\left(\mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right)\right)$ ausgeführt. Bei erfolgreicher Abwicklung des Protokolls erhält $\mathcal{K}$ eine gültige Münze $\mathsf{Coin} = (A, B, (z, a, b, r))$ in die $\mathcal{K}$'s Identität I integriert ist. Zusätzlich wird durch das Protokoll sichergestellt, dass $\mathcal{B}$ anhand von $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ in Kooperation mit $\mathcal{D}$ Münztracing durchführen kann.

$\mathcal{K}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{D}})$		$\mathcal{B}(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}})$
$m, s, t \in_R \mathbb{Z}_q$		
$I' = g_1^{u_1 s^{-1}} g_3^{s^{-1}} g_4^t, E_1 = g_2^s f_3^m, E_2 = g_3^m$		
$V_1 = \mathrm{NIZK}\left[(\alpha_1, \alpha_2) : E_1 = g_2^{\alpha_1} f_3^{\alpha_2} \wedge E_2 = g_3^{\alpha_2}\right]$		
$V_2 = \mathrm{NIZK}\left[(\beta_1, \ldots, \beta_6) : g_3 = (I')^{\beta_1} g_1^{\beta_2} g_4^{\beta_3} \wedge E_1 = g_2^{\beta_1} f_3^{\beta_4} \wedge I = (I')^{\beta_1} g_3^{\beta_5} g_4^{\beta_6}\right]$		
	$\xrightarrow{I', E_1, E_2, V_1, V_2}$	
		$E_2 \stackrel{?}{\neq} 1$
		Verifiziere V_1, V_2
		$w \in_R \mathbb{Z}_q$
		$a' = g^w$
		$b' = (I' g_2)^w$
	$\xleftarrow{a', b', b''}$	$b'' = g_4^w$
$A = (I' g_2 g_4^{-t})^s = g_1^{u_1} g_2^s g_3$		
$z = h_1^{u_1} h_2^s h_3 = A^{X_{\mathcal{B}}}$		
$x_1, x_2, u, v \in_{\mathcal{R}} \mathbb{Z}_q$		
$B = g_1^{x_1} g_2^{x_2}$		
$a = (a')^u g^v$		
$b = (b' b''^{-t})^{su} A^v$		
$c = \mathrm{H}(A, B, z, a, b)$		
$c' = cu^{-1} \bmod q$	$\xrightarrow{c'}$	
	$\xleftarrow{r'}$	$r' = c' X_{\mathcal{B}} + w \bmod q$
$r = r'u + v \bmod q$		
$g^r \stackrel{?}{=} h^c a$		
$A^r \stackrel{?}{=} z^c b$		

Abbildung 6.1: Abhebe-Protokoll im System von Frankel, Tsiounis und Yung

Um eine korrekte Durchführung des Münztracings zu garantieren, wird A während des Protokolls so konstruiert, dass neben der Identität I auch ein *Münz-Identifikator* in A

enthalten ist, mit dem $\mathcal{B}$ später Münztracing durchführen kann. Zur Konstruktion des Münz-Identifikators berechnet $\mathcal{K}$ zunächst $I' = I^{s^{-1}} g_3^{s^{-1}} g_4^t$ sowie die ElGamal-Verschlüsselung (E_1, E_2) von g_2^s und beweist $\mathcal{B}$ mit dem nichtinteraktiven Beweis

$$V_1 = \text{NIZK}\left[(\alpha_1, \alpha_2) : E_1 = g_2^{\alpha_1} f_3^{\alpha_2} \wedge E_2 = g_3^{\alpha_2}\right]$$

(vgl. auch Abschnitt 2.9.1), dass (E_1, E_2) eine ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel f_3 von $\mathcal{D}$ ist. Der Klartext g_2^s wird von $\mathcal{B}$ als Münz-Identifikator verwendet. Um sicherzustellen, dass g_2^s in A enthalten ist, beweist $\mathcal{K}$ der Bank $\mathcal{B}$ anhand des nichtinteraktiven Beweises

$$V_2 = \text{NIZK}\left[(\beta_1, \ldots, \beta_6) : g_3 = (I')^{\beta_1} g_1^{\beta_2} g_4^{\beta_3} \wedge E_1 = g_2^{\beta_1} f_3^{\beta_4} \wedge I = (I')^{\beta_1} g_3^{\beta_5} g_4^{\beta_6}\right],$$

dass E_1 bezüglich I' korrekt berechnet wurde. Anschließend wird I' von $\mathcal{B}$ blind signiert und $\mathcal{K}$ erhält die Münze $\mathsf{Coin} = (A, B, (z, a, b, r))$.
Während des Bezahl-Protokolls muss $\mathcal{K}$ mit der Münze $(A, B, (z, a, b, r))$ bezahlen, die einen Wert A der Form $A = I g_2^s g_3$ enthalten muss. Dadurch wird $\mathcal{K}$ bei der Berechnung von A gezwungen, den Wert s als Blendungsfaktor zu verwenden. Dies garantiert, dass A den für das Münztracing benötigten Münz-Identifikator g_2^s enthält.

Bezahlen mit einer Münze: Möchte $\mathcal{K}$ bei einem Händler $\mathcal{H}$ mit der Münze $\mathsf{Coin} = (A, B, (z, a, b, r))$ bezahlen, so wird das in Abbildung 6.2 dargestellte Bezahl-Protokoll $\mathsf{Payment}(\mathcal{K}(\mathsf{sk}_\mathcal{K}, \mathsf{Coin}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D}), \mathcal{H}(\mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D}))$ ausgeführt. Zur Gewährleistung der Anonymität von $\mathcal{K}$ wird vorausgesetzt, dass $\mathcal{K}$ und $\mathcal{H}$ über einen anonymen Kanal kommunizieren. Den Zeitpunkt des Bezahlvorgangs bezeichnen wir mit ts (engl. *timestamp*).

$\mathcal{K}(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{K}, \mathsf{Coin}, \mathsf{pk}_\mathcal{D})$	$\mathcal{H}(\mathsf{pk}_\mathcal{H}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$
$A_1 = I g_2^s = A g_3^{-1}$	
$A_2 = f_2^s$	
$V_3 = \text{NIZK}\left[(\gamma_1, \gamma_2) : A_1 = g_1^{\gamma_1} g_2^{\gamma_2} \wedge A_2 = f_2^{\gamma_2}\right](I_\mathcal{H}, \mathsf{ts})$	
In V_3 verwendet $\mathcal{K}$ den Wert B als Commitment auf A_1.	
$\xrightarrow{\mathsf{Coin}, A_1, A_2, V_3}$	$A_1 \stackrel{?}{\neq} 1$
	$A_1 g_3 \stackrel{?}{=} A$
	$\mathsf{Verify}(\mathsf{pk}_\mathcal{B}, \mathsf{Coin})$
	Verifiziere V_3

Abbildung 6.2: Bezahl-Protokoll im System von Frankel, Tsiounis und Yung

Das Bezahl-Protokoll stellt zum einen sicher, dass $\mathcal{K}$ im Falle eines Double-Spending durch $\mathcal{B}$ identifiziert werden kann. Zum anderen ist es so konstruiert, dass $\mathcal{B}$ nach dem Einlösen von Coin anhand von $\mathsf{view}_\mathcal{H}^\mathcal{K}(P)$ in Kooperation mit $\mathcal{D}$ Kundentracing durchführen kann. Um dies zu garantieren, berechnet $\mathcal{K}$ anhand der Münze Coin die inverse

ElGamal-Verschlüsselung ($A_1 = Ig_2^s, A_2 = f_2^s$) und erstellt den nichtinteraktiven Beweis

$$V_3 = \text{NIZK}\left[(\gamma_1, \gamma_2) : A_1 = g_1^{\gamma_1} g_2^{\gamma_2} \wedge A_2 = f_2^{\gamma_2}\right](I_{\mathcal{H}}, \mathsf{ts}).$$

Mit V_3 beweist $\mathcal{K}$ dem Händler $\mathcal{H}$, dass (A_1, A_2) eine inverse ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel f_2 von $\mathcal{D}$ ist. Da $\mathcal{K}$ während der Erstellung von V_3 den Wert B als Commitment auf A_1 verwenden muss, kann $\mathcal{K}$'s Identität bei Doppeltausgabe von Coin aufgedeckt werden. Die Konstruktion von $A = Ig_2^s g_3$ als Teil der Münze Coin und $A_1 = Ag_3^{-1}$ zwingt $\mathcal{K}$ während des Abhebe-Protokolls den Wert s als Blendungsfaktor zu verwenden. Unter der Annahme, dass das Abhebe-Protokoll restriktiv blind ist (vgl. Abschnitt 2.6.2), liefert (A_1, A_2) eine ElGamal-Verschlüsselung der Identität I von $\mathcal{K}$, anhand der Kundentracing durchgeführt werden kann.

Einlösen einer Münze: Um eine Münze Coin bei $\mathcal{B}$ einzulösen, wird das Protokoll $\mathsf{Deposit}\left(\mathcal{H}\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}\right)\right)$ durchgeführt. Zunächst authentifizieren sich beide Parteien gegenseitig. Anschließend sendet $\mathcal{H}$ die Ansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P) = \{\mathsf{Coin}, A_1, A_2, V_3\}$ an $\mathcal{B}$. Die Bank $\mathcal{B}$ überprüft die Signatur (z, a, b, r) der Münze Coin und den nichtinteraktiven Beweis V_3.

Kunden- und Münztracing: Zur Durchführung des Kundentracings wird das Protokoll $\mathsf{KTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)\right), \mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}\right)\right)$ ausgeführt. Die Bank $\mathcal{B}$ sendet die in $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ enthaltene inverse ElGamal-Verschlüsselung (A_1, A_2) an $\mathcal{D}$, der diese zu $I = g_1^{u_1}$ entschlüsselt und I an $\mathcal{B}$ sendet. Anhand von I sucht $\mathcal{B}$ den entsprechenden Kunden in ihrer Datenbank.
Für Münztracing wird das Protokoll $\mathsf{MTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)\right), \mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}\right)\right)$ durchgeführt. Die Bank $\mathcal{B}$ sendet die in $\mathsf{view}_{\mathcal{B}}(W)$ enthaltene ElGamal-Verschlüsselung (E_1, E_2) an $\mathcal{D}$, der diese zu g_2^s entschlüsselt und den Münz-Identifikator g_2^s an $\mathcal{B}$ sendet. Anhand der in $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ enthaltenen Identität I kann $\mathcal{B}$ die Münze $A = Ig_2^s g_3$ berechnen und somit die bereits eingelösten Münzen nach A durchsuchen.

6.1.3 Sicherheitsanalyse

Unfälschbarkeit, Nichterweiterbarkeit, Anonymität und Fairness sind die wichtigsten Sicherheitsaspekte eines fairen elektronischen Geldsystems. Von Frankel *et al.* wurde in [FTY98] gezeigt, dass das in diesem Abschnitt beschriebene faire elektronische Geldsystem *FOLC* (Fair Offline Electronic Cash) im Sinne von Definition 3.3.2 sicher ist.
Insbesondere bietet *FOLC* laut [FTY98] Anonymität im Sinne von Definition 3.1.2. Im Jahre 2003 wurde von M. Gaud und J. Traoré in [GT03] jedoch gezeigt, dass der in [FTY98] angegebene Sicherheitsbeweis in Bezug auf die Anonymität von *FOLC* fehlerhaft ist.
Die Anonymität von *FOLC* kann aber analog zu [GT03] mit dem von M. Gaud und J. Traoré eingeschlagenen Weg gezeigt werden, so dass *FOLC* tatsächlich ein im Sinne von Definition 3.3.2 sicheres faires elektronisches Geldsystem ist. Für eine ausführliche Diskussion sei an dieser Stelle auf [FTY98],[GT03] und die Sicherheitsanalyse des im

nächsten Abschnitt vorgestellten langfristig sicheren fairen elektronischen Geldsystems verwiesen.

Satz 6.1.1. Im Random-Oracle-Modell ist *FOLC* ein, im Sinne von Definition 3.3.2, sicheres faires elektronisches Geldsystem.

Beweis. siehe [FTY98] bzw. [GT03]. □

6.2 Ein langfristig sicheres faires System

Um die in Abschnitt 4.1 dargestellten Folgen einer Kompromittierung der Bank bzw. des Deanonymisierers zu minimieren, wird in diesem Abschnitt auf Basis von *FOLC* [FTY98] ein neues langfristig sicheres faires elektronisches Geldsystem entwickelt.
Da das neue System ein aktualisierbares faires elektronisches Geldsystem ist, verfügen sowohl die Bank $\mathcal{B}$ als auch der Deanonymisierer $\mathcal{D}$ über einen Tresor, auf dem sie ihren *Generalschlüssel* speichern (vgl. Abschnitt 5.4). Den Tresor der Bank $\mathcal{B}$ bezeichnen wir mit $\mathcal{T}_{\mathcal{B}}$ und den des Deanonymisierers $\mathcal{D}$ mit $\mathcal{T}_{\mathcal{D}}$. Zusätzlich können $\mathcal{B}$ und $\mathcal{D}$ ihre privaten Schlüssel mit den Protokollen BKeyUpdate bzw. DKeyUpdate zu Beginn jedes neuen Intervalls aktualisieren.
Im Vergleich zu *FOLC* ergeben sich neben dem Einsatz der auf den Tresoren gespeicherten Generalschlüssel und den beiden Schlüsselaktualisierungsprotokollen zwei signifikante Unterschiede: Zum einen wird die in *FOLC* für Kunden- und Münztracing verwendete ElGamal-Verschlüsselung durch das in Abschnitt 5.1 dargestellte Verschlüsselungsverfahren ersetzt. Zum anderen wird, anstelle der *FOLC* zugrunde liegenden blinden Schnorr-Signatur, die in Abschnitt 5.2.3 entwickelte langfristig sichere Variante der blinden Schnorr-Signatur eingesetzt. Diese beiden Unterschiede bewirken nicht nur, dass Bank und Deanonymisierer ihre privaten Schlüssel regelmäßig aktualisieren, sondern ziehen auch einige Änderungen an den Algorithmen BKeyGen und DKeyGen bzw. den Protokollen Withdrawal, Payment, MTrace und KTrace nach sich. Das neue langfristig sichere faire elektronische Geldsystem wird im Folgenden detailliert vorgestellt.

6.2.1 Initialisierung des Systems

Ebenso wie in *FOLC* wird das System durch die Bank $\mathcal{B}$ und den Deanonymisierer $\mathcal{D}$ initialisiert. Die Unterschiede zu *FOLC* ergeben sich dadurch, dass die Bank $\mathcal{B}$ und der Deanonymisierer $\mathcal{D}$ ihre privaten Schlüssel in regelmäßigen Intervallen aktualisieren. Im Einzelnen läuft die Initialisierung des Systems wie folgt ab:

6.2.1.1 Initialisierung der Bank

Zur Initialisierung des Systems wird von der Bank $\mathcal{B}$ der Algorithmus $\mathsf{BKeyGen}(1^k)$ aufgerufen. Bei Eingabe des Sicherheitsparameters $k \in \mathbb{N}$ werden von $\mathcal{B}$ die folgenden Schritte ausgeführt.

Schritt 1: Die Bank $\mathcal{B}$ wählt zwei Primzahlen p und q, so dass $|p-1| = \delta + k$ und $p = \gamma q + 1$ für vorgegebene $\delta, \gamma \in \mathbb{N}$.

Schritt 2: Die Bank $\mathcal{B}$ wählt $N, t \in \mathbb{N}$ mit $t < N$.

Schritt 3: Die Bank $\mathcal{B}$ wählt zufällig Generatoren $g, g_1, \ldots, g_8$ der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$. Weiter wählt $\mathcal{B}$ zufällig Zahlen $x_0^*, \ldots, x_t^* \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet

$$h_i^* = g^{x_i^*} \text{ sowie } h_{1,i}^* = g_1^{x_i^*}, h_{2,i}^* = g_2^{x_i^*}, h_{3,i}^* = g_3^{x_i^*} \text{ für } i = 0, \ldots, t.$$

Die Zahlen $x_1^*, \ldots, x_t^*$ speichert $\mathcal{B}$ als Generalschlüssel $\mathsf{sk}_{\mathcal{B}}^*$ in ihrem Tresor $\mathcal{T}_{\mathcal{B}}$. Anschließend werden $x_1^*, \ldots, x_t^*$ von $\mathcal{B}$ gelöscht.

Schritt 4: Die Bank $\mathcal{B}$ wählt Hashfunktionen $\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$ aus einer Familie geeigneter kollisionsresistenter Hashfunktionen, speichert x_0^* als ihren privaten Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(0)}$ für das 0-te Intervall und gibt ihren öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$ bekannt:

$$\mathsf{pk}_{\mathcal{B}} = (p, q, g, g_1, \ldots, g_8, h_0^*, \ldots, h_t^*, h_{1,0}^*, \ldots, h_{1,t}^*, h_{2,0}^*, \ldots, h_{2,t}^*, h_{3,0}^*, \ldots, h_{3,t}^*, \mathrm{H}, \mathrm{H}_0, \ldots).$$

Durch die Wahl von N bestimmt $\mathcal{B}$ die Anzahl der Intervalle, in die die Laufzeit des öffentlichen Schlüssels von $\mathcal{B}$ eingeteilt wird. Mit der Schwelle t fixiert $\mathcal{B}$ die Anzahl der Intervalle, die von einem Angreifer kompromittiert werden dürfen. Anhand von $x_0^*, \ldots, x_t^*$ legt sich $\mathcal{B}$ auf das Polynom

$$f(\mu) = \sum_{k=0}^{t} x_k^* \mu^k$$

mit $\mathrm{Grad}(f) = t$ fest und bestimmt somit durch f die Vorschrift nach der der private Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(i)}$ von $\mathcal{B}$ im i-ten Intervall berechnet wird: $\mathsf{sk}_{\mathcal{B}}^{(i)} = f(i)$. Der Einfachheit halber bezeichnen wir $f(i)$ auch mit x_i.
Zur Verifikation einer im i-ten Intervall erstellten Signatur wird ein dem i -ten Intervall angepasster Verifikationsschlüssel $h_i = g^{x_i}$ benötigt, der aus den im öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$ enthaltenen Werten $h_0^*, \ldots, h_t^*$ durch

$$h_i = \prod_{k=0}^{t} (h_k^*)^{i^k}$$

berechnet werden kann (vgl. Abschnitt 5.2.3). Somit ist eine Signatur $\langle i, s \rangle$ der Bank $\mathcal{B}$ im Intervall i durch die folgende Definition gegeben:

Definition 6.2.1. Eine Signatur $\langle i, s \rangle$ der Bank $\mathcal{B}$ für ein Paar (A, B) im Intervall i ist ein Tupel $(z, a, b, r) \in G_q \times G_q \times G_q \times \mathbb{Z}_q$, so dass Folgendes gilt:

$$g^r = h_i^{\mathrm{H}(A,B,z,a,b)} a, \tag{6.1}$$

$$A^r = z^{\mathrm{H}(A,B,z,a,b)} b, \tag{6.2}$$

wobei $h_i = \prod_{k=0}^{t} (h_k^*)^{i^k}$ ist. Eine Münze Coin ist ein Tupel $(A, B, \langle i, (z, a, b, r) \rangle)$. Kennt ein Kunde $\mathcal{K}$ eine Darstellung von A bzgl. (g_1, g_2, g_3) und eine Darstellung von B bzgl. (g_1, g_2), dann kennt $\mathcal{K}$ eine *Darstellung der Münze.*

Um später potenzielle Double-Spender anhand ihrer Kontonummer eindeutig identifizieren zu können und Kunden- bzw. Münztracing in Kooperation mit dem Deanonymisierer $\mathcal{D}$ erfolgreich durchführen zu können, legt $\mathcal{B}$ eine Datenbank an, in der $\mathcal{B}$ die Daten ihrer Kunden und alle eingelösten Münzen speichert.

Schlüsselaktualisierung der Bank: Im i-ten Intervall kann $\mathcal{B}$ mit ihrem privaten Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(i)} = x_i$ den im j-ten Intervall gültigen privaten Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(j)} = x_j$ nicht berechnen; dazu müsste $\mathcal{B}$ mindestens t weitere Punkte des Polynoms f kennen. Da der Generalschlüssel $\mathsf{sk}_{\mathcal{B}}^{*} = (x_1^*, \ldots, x_t^*)$ von $\mathcal{B}$ auf $\mathcal{T}_{\mathcal{B}}$ gespeichert wurde, kann $\mathcal{B}$ den privaten Schlüssel x_j mit Hilfe von $\mathcal{T}_{\mathcal{B}}$ berechnen. Zur Aktualisierung des privaten Schlüssels x_i wird das in Abbildung 6.3 dargestellte Protokoll $\mathsf{BKeyUpdate}(\mathcal{B}(i, j, \mathsf{sk}_{\mathcal{B}}^{(i)}), \mathcal{T}_{\mathcal{B}}(\mathsf{sk}_{\mathcal{B}}^{*}))$ durchgeführt.

Schritt 1: Die Bank $\mathcal{B}$ sendet den Index i des aktuellen Intervalls und den Index j des Intervalls, für den $\mathcal{B}$ den Signaturschlüssel $\mathsf{sk}_{\mathcal{B}}^{(j)} = x_j$ berechnen möchte, an $\mathcal{T}_{\mathcal{B}}$.

Schritt 2: Anhand des Generalschlüssels $\mathsf{sk}_{\mathcal{B}}^{*} = (x_1^*, \ldots, x_t^*)$ berechnet $\mathcal{T}_{\mathcal{B}}$ den temporären Schlüssel $x'_{i,j} = \sum_{k=1}^{t} x_k^*(j^k - i^k)$ und sendet $x'_{i,j}$ an $\mathcal{B}$.

Schritt 3: Mit dem Signaturschlüssel x_i berechnet $\mathcal{B}$ den für das j-te Intervall gültigen Signaturschlüssel $x_j = x_i + x'_{i,j}$. Anschließend werden x_i und $x'_{i,j}$ von $\mathcal{B}$ gelöscht.

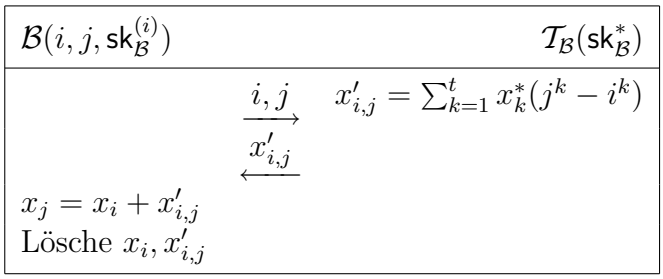

Abbildung 6.3: Aktualisierung der Bank im langfristig sicheren System

Bemerkung 6.2.1. Das Protokoll $\mathsf{BKeyUpdate}$ ermöglicht $\mathcal{B}$ eine direkte Schlüsselaktualisierung. Das heißt, die Bank $\mathcal{B}$ kann ihren privaten Schlüssel zwischen zwei beliebigen, nicht notwendiger Weise aufeinanderfolgenden Intervallen i und j aktualisieren (vgl. Abschnitt 5.3). Üblicherweise wird $\mathcal{B}$ das Protokoll zu Beginn jedes neuen Intervalls ausführen. Mit anderen Worten aktualisiert $\mathcal{B}$ ihren privaten Schlüssel zwischen den Intervallen $i-1$ und i durch $\mathsf{BKeyUpdate}\left(\mathcal{B}\left(i-1, i, \mathsf{sk}_{\mathcal{B}}^{(i-1)}\right), \mathcal{O}\left(\mathsf{sk}_{\mathcal{B}}^{*}\right)\right)$.

6.2.1.2 Initialisierung des Deanonymisierers

Im Gegensatz zu *FOLC* wird für Kunden- und Münztracing nicht die ElGamal-Verschlüsselung verwendet, sondern das in Abschnitt 5.1 beschriebene Verschlüsselungsverfahren [DKXY02]. Um die Schlüsselaktualisierung von $\mathcal{B}$ und $\mathcal{D}$ zu synchronisieren,

übernimmt $\mathcal{D}$ der Einfachheit halber die Laufzeit des öffentlichen Schlüssels, die Anzahl der Intervalle N und die Schwelle t von $\mathcal{B}$. Die Initialisierung des Deanonymisierers geschieht durch den Aufruf des Algorithmus $\mathsf{DMKeyGen}(1^k, N, \mathsf{pk}_{\mathcal{B}})$. Bei Eingabe des Sicherheitsparameters 1^k, der Anzahl der Intervalle N und des öffentlichen Schlüssels $\mathsf{pk}_{\mathcal{B}}$ werden von $\mathcal{D}$ die im Folgenden beschriebenen Schritte durchgeführt.

Schritt 1: Der Deanonymisierer $\mathcal{D}$ wählt zufällig Zahlen $\hat{x}_0^*, \ldots, \hat{x}_t^*, \hat{y}_0^*, \ldots, \hat{y}_t^* \in_{\mathcal{R}} \mathbb{Z}_q$ und $\bar{x}_0^*, \ldots, \bar{x}_t^*, \bar{y}_0^*, \ldots, \bar{y}_t^* \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $\hat{z}_i^* = g_5^{\hat{x}_i^*} g_6^{\hat{y}_i^*}$ sowie $\bar{z}_i^* = g_7^{\bar{x}_i^*} g_8^{\bar{y}_i^*}$ für $i = 0, \ldots, t$.

Schritt 2: Die Zahlen $\hat{x}_1^*, \ldots, \hat{x}_t^*, \hat{y}_1^*, \ldots, \hat{y}_t^*$ und $\bar{x}_1^*, \ldots, \bar{x}_t^*, \bar{y}_1^*, \ldots, \bar{y}_t^*$ werden von $\mathcal{D}$ als Generalschlüssel $\mathsf{sk}_{\mathcal{D}}^*$ in seinem Tresor $\mathcal{T}_{\mathcal{D}}$ gespeichert und anschließend gelöscht.

Schritt 3: Der Deanonymisierer $\mathcal{D}$ verwendet die beiden Paare $(\hat{x}_0^*, \hat{y}_0^*)$ und $(\bar{x}_0^*, \bar{y}_0^*)$ als private Schlüssel $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(0)} = (\hat{x}_0^*, \hat{y}_0^*)$ und $\overline{\mathsf{sk}}_{\mathcal{D}}^{(0)} = (\bar{x}_0^*, \bar{y}_0^*)$ für das 0-te Intervall und gibt folgendes Tupel als öffentlichen Schlüssel bekannt:

$$\mathsf{pk}_{\mathcal{D}} = (\hat{z}_0^*, \ldots, \hat{z}_t^*, \bar{z}_0^*, \ldots, \bar{z}_t^*).$$

Genauso wie $\mathcal{B}$ legt sich $\mathcal{D}$ durch die Wahl der Zahlen $\hat{x}_i^*, \hat{y}_i^*$ und $\bar{x}_i^*, \bar{y}_i^*$ für $i = 0, \ldots, t$ auf die folgenden vier Polynome

$$\hat{f}_x(\mu) = \sum_{k=0}^{t} \hat{x}_k^* \mu^k, \hat{f}_y(\mu) = \sum_{k=0}^{t} \hat{y}_k^* \mu^k \text{ und } \bar{f}_x(\mu) = \sum_{k=0}^{t} \bar{x}_k^* \mu^k, \bar{f}_y(\mu) = \sum_{k=0}^{t} \bar{y}_k^* \mu^k$$

vom Grad t fest. Die beiden Polynome $\hat{f}_x, \hat{f}_y$ bestimmen die Vorschrift nach der der private Schlüssel $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)}$ im i-ten Intervall berechnet wird. Entsprechend wird durch die beiden Polynome $\bar{f}_x, \bar{f}_y$ der Schlüssel $\overline{\mathsf{sk}}_{\mathcal{D}}^{(i)}$ bestimmt. Es ist $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)} = \left(\hat{f}_x(i), \hat{f}_y(i)\right)$ und analog $\overline{\mathsf{sk}}_{\mathcal{D}}^{(i)} = \left(\bar{f}_x(i), \bar{f}_y(i)\right)$. Abkürzend bezeichnen wir $\left(\hat{f}_x(i), \hat{f}_y(i)\right)$ mit $(\hat{x}_i, \hat{y}_i)$ und $\left(\bar{f}_x(i), \bar{f}_y(i)\right)$ mit $(\bar{x}_i, \bar{y}_i)$. Die zur Verschlüsselung im Intervall i benötigten temporären Schlüssel können mit Hilfe von $\mathsf{pk}_{\mathcal{D}}$ entsprechend berechnet werden und lauten:

$$\hat{z}_i = \prod_{k=0}^{t} (\hat{z}_k^*)^{i^k} \text{ sowie } \bar{z}_i = \prod_{k=0}^{t} (\bar{z}_k^*)^{i^k}.$$

Bemerkung 6.2.2. Der Schlüssel $\hat{z}_i$ wird von den Kunden der Bank im i-ten Intervall dazu verwendet einen Münz-Identifikator zu verschlüsseln, damit $\mathcal{B}$ später erfolgreich Münztracing durchführen kann. Zur Realisierung des Kundentracing wird im i-ten Intervall der Schlüssel $\bar{z}_i$ zur Verschlüsselung der Kontonummer eingesetzt.

Schlüsselaktualisierung des Deanonymisierers: Im i-ten Intervall kann der Deanonymisierer $\mathcal{D}$ die im j-ten Intervall gültigen privaten Schlüssel $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(j)} = (\hat{x}_j, \hat{y}_j)$ und $\overline{\mathsf{sk}}_{\mathcal{D}}^{(j)} = (\bar{x}_j, \bar{y}_j)$ nur mit Hilfe des in seinem Tresor $\mathcal{T}_{\mathcal{D}}$ gespeicherten Generalschlüssels $\mathsf{sk}_{\mathcal{D}}^*$ berechnen. Zur Aktualisierung des privaten Schlüssels $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)}$ wird das in Abbildung 6.4 dargestellte Protokoll $\mathsf{DKeyUpdate}_1\left(\mathcal{D}\left(i, j, \widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)}\right), \mathcal{T}_{\mathcal{D}}(\mathsf{sk}_{\mathcal{D}}^*)\right)$ ausgeführt.

Schritt 1: Der Deanonymisierer $\mathcal{D}$ sendet die Indizes i, j an seinen Tresor $\mathcal{T}_\mathcal{D}$.

Schritt 2: Da $\mathcal{T}_\mathcal{D}$ die Zahlen $\hat{x}^*_k, \hat{y}^*_k$ für $k = 1, \ldots, t$ kennt, berechnet $\mathcal{T}_\mathcal{D}$ die beiden temporären Schlüssel $\hat{x}'_{i,j} = \sum_{k=1}^{t} \hat{x}^*_k(j^k - i^k)$ und $\hat{y}'_{i,j} = \sum_{k=1}^{t} \hat{y}^*_k(j^k - i^k)$ und sendet $\hat{x}'_{i,j}, \hat{y}'_{i,j}$ an $\mathcal{D}$.

Schritt 3: Mit dem aktuellen privaten Schlüssel $(\hat{x}_i, \hat{y}_i)$ berechnet $\mathcal{D}$ dann $\hat{x}_j = \hat{x}_i + \hat{x}'_{i,j}$ und $\hat{y}_j = \hat{y}_i + \hat{y}'_{i,j}$. Anschließend werden $(\hat{x}_i, \hat{y}_i)$ und $\hat{x}'_{i,j}$ von $\mathcal{D}$ gelöscht. Der im j-ten Intervall gültige private Schlüssel ist $\widehat{\mathsf{sk}}^{(j)}_\mathcal{D} = (\hat{x}_j, \hat{y}_j)$.

$\mathcal{D}(i, j, \widehat{\mathsf{sk}}^{(i)}_\mathcal{D})$		$\mathcal{T}_\mathcal{D}(\mathsf{sk}^*_\mathcal{D})$
	$\xrightarrow{i, j}$	$\hat{x}'_{i,j} = \sum_{k=1}^{t} \hat{x}^*_k(j^k - i^k)$
	$\xleftarrow{\hat{x}'_{i,j}, \hat{y}'_{i,j}}$	$\hat{y}'_{i,j} = \sum_{k=1}^{t} \hat{y}^*_k(j^k - i^k)$
$\hat{x}_j = \hat{x}_i + \hat{x}'_{i,j}$		
$\hat{y}_j = \hat{y}_i + \hat{y}'_{i,j}$		
Lösche $\hat{x}_i, \hat{y}_i, \hat{x}'_{i,j}, \hat{y}'_{i,j}$		

$\mathcal{D}(i, j, \overline{\mathsf{sk}}^{(i)}_\mathcal{D})$		$\mathcal{T}_\mathcal{D}(\mathsf{sk}^*_\mathcal{D})$
	$\xrightarrow{i, j}$	$\bar{x}'_{i,j} = \sum_{k=1}^{t} \bar{x}^*_k(j^k - i^k)$
	$\xleftarrow{\bar{x}'_{i,j}, \bar{y}'_{i,j}}$	$\bar{y}'_{i,j} = \sum_{k=1}^{t} \bar{y}^*_k(j^k - i^k)$
$\bar{x}_j = \bar{x}_i + \bar{x}'_{i,j}$		
$\bar{y}_j = \bar{y}_i + \bar{y}'_{i,j}$		
Lösche $\bar{x}_i, \hat{y}_i, \bar{x}'_{i,j}, \bar{y}'_{i,j}$		

Abbildung 6.4: Aktualisierung des Deanonymisierers im langfristig sicheren System

Entsprechend wird $\overline{\mathsf{sk}}^{(i)}_\mathcal{D}$ durch das Protokoll $\mathsf{DKeyUpdate}_2(\mathcal{D}(i, j, \overline{\mathsf{sk}}^{(i)}_\mathcal{D}), \mathcal{T}_\mathcal{D}(\mathsf{sk}^*_\mathcal{D}))$ aktualisiert. Da das Protokoll analog zu $\mathsf{DKeyUpdate}_1(\mathcal{D}(i, j, \widehat{\mathsf{sk}}^{(i)}_\mathcal{D}), \mathcal{T}_\mathcal{D}(\mathsf{sk}^*_\mathcal{D}))$ abläuft, sei auf Abbildung 6.4 verwiesen.

Bemerkung 6.2.3. Die beiden Protokolle $\mathsf{DKeyUpdate}_1$ und $\mathsf{DKeyUpdate}_2$ ermöglichen $\mathcal{D}$ eine direkte Schlüsselaktualisierung. Da $\mathcal{D}$ zu jedem beliebigen Zeitpunkt auf die Anfragen von $\mathcal{B}$ antworten können muss, die $\mathcal{D}$ bei der Durchführung der Tracing-Protokolle KTrace und MTrace von $\mathcal{B}$ erhält, ist für $\mathcal{D}$ im Vergleich zur Bank wichtig, dass er seine privaten Schlüssel zwischen beliebigen Intervallen aktualisieren kann.

6.2.2 Kontoeröffnung

Möchte ein neuer Kunde $\mathcal{K}$ ein Konto bei der Bank $\mathcal{B}$ eröffnen, so muss sich $\mathcal{K}$ zunächst gegenüber $\mathcal{B}$ ausweisen, beispielsweise mit einem Personalausweis. Anschließend

wird das in Abbildung 6.5 dargestellte Protokoll $\mathsf{Reg}(\mathcal{K}(\mathsf{pk}_\mathcal{B}), \mathcal{B}(\mathsf{pk}_\mathcal{B}))$ zwischen $\mathcal{K}$ und $\mathcal{B}$ durchgeführt. Da zur Durchführung des Protokolls keine von einem Intervall abhängigen Schlüssel benötigt werden, kann die Kontoeröffnung wie in Abschnitt 6.1 realisiert werden. Das Protokoll läuft wie folgt ab:

Schritt 1: Der Kunde $\mathcal{K}$ wählt zufällig eine Zahl $u_1 \in_\mathcal{R} \mathbb{Z}_q$ und berechnet $I = g_1^{u_1}$. Anschließend speichert $\mathcal{K}$ das Schlüsselpaar $(\mathsf{pk}_\mathcal{K}, \mathsf{sk}_\mathcal{K}) = (I, u_1)$, erstellt den nichtinteraktiven Beweis $V_0 = \text{NIZK}\,[\alpha : I = g_1^\alpha]$ (vgl. Abschnitt 2.9.1) und sendet I, V_0 an die Bank $\mathcal{B}$.

Schritt 2: Die Bank $\mathcal{B}$ überprüft V_0 und die Konstruktion von I.

$\mathcal{K}(\mathsf{pk}_\mathcal{B})$		$\mathcal{B}(\mathsf{pk}_\mathcal{B})$
$u_1 \in_\mathcal{R} \mathbb{Z}_q$ $I = g_1^{u_1}$ $V_0 = \text{NIZK}\,[\alpha : I = g_1^\alpha]$		
	$\xrightarrow{I, V_0}$	Verifiziere V_0
		$Ig_2 \overset{?}{\neq} 1$

Abbildung 6.5: Die Kontoeröffnung im langfristig sicheren System

Bei erfolgreicher Durchführung des Protokolls legt $\mathcal{B}$ ein Konto für $\mathcal{K}$ an. Zusammen mit den persönlichen Daten von $\mathcal{K}$ speichert $\mathcal{B}$ die Zahl I in ihrer Datenbank. Anhand von I lässt sich $\mathcal{K}$ eindeutig identifizieren. Daher bezeichnen wir I auch als die *Kontonummer* bzw. *Identität* von $\mathcal{K}$. Die Zahl $u_1 = \log_{g_1} I$ dient dazu, $\mathcal{K}$ im Falle eines Double-Spending zu identifizieren.

Bemerkung 6.2.4. Der nichtinteraktive Beweis V_0 garantiert, dass $\mathcal{K}$ eine Darstellung von I bzgl. g_1 gewählt hat und verhindert die in [CFMT96] vorgestellten Angriffe. Würde $\mathcal{B}$ auf diesen Nachweis verzichten, könnte $\mathcal{K}$ auch eine Darstellung von I bzgl. (g_1, g_2) wählen. Im Falle eines Double-Spending könnte $\mathcal{K}$ von $\mathcal{B}$ dann nicht identifiziert werden.

6.2.3 Eine Münze abheben

Durch das Abhebe-Protokoll Withdrawal müssen sowohl die Interessen des Kunden $\mathcal{K}$, als auch die Interessen der Bank $\mathcal{B}$ gewahrt bleiben. Einerseits soll $\mathcal{K}$ am Ende des Protokolls eine gültige Münze Coin erhalten, die $\mathcal{B}$ dem Kunden $\mathcal{K}$ nicht zuordnen kann. Andererseits muss das Protokoll sicherstellen, dass $\mathcal{K}$ seine Kontonummer I in die Münze Coin integriert, damit $\mathcal{B}$ ihn im Falle eines Double-Spending identifizieren kann. In gleicher Weise muss das Protokoll garantieren, dass $\mathcal{B}$ anhand von $\mathsf{view}_\mathcal{B}^\mathcal{K}(W)$ in Kooperation mit $\mathcal{D}$ Münztracing durchführen kann.

Diese Sicherheitseigenschaften können mit den in *FOLC* verwendeten Techniken erreicht werden. Im Vergleich zu *FOLC* ergeben sich durch den Einsatz des in Abschnitt 5.1 beschriebenen Verschlüsselungsverfahrens und die Integration der langfristig sicheren Variante der blinden Schnorr-Signatur aus Abschnitt 5.2.3 einige Änderungen: Da die eingesetzte Verschlüsselung, anders als die ElGamal-Verschlüsselung, mit zwei Generatoren arbeitet, müssen die beiden nichtinteraktiven Zero-Knowledge-Beweise V_1 und V_2 (vgl. Abbildung 6.1) entsprechend angepasst werden. Auf Grund der regelmäßigen Aktualisierung des privaten Schlüssels $\mathsf{sk}_{\mathcal{B}}^{(i)} = x_i$, müssen die in Abhängigkeit von x_i berechneten Werte entsprechend angepasst werden.
Bevor im i-ten Intervall das Abhebe-Protokoll Withdrawal zwischen $\mathcal{K}$ und $\mathcal{B}$ durchgeführt wird, überprüfen beide Parteien, ob sie die für das i-te Intervall benötigten Schlüssel bereits berechnet haben. Falls nötig berechnet $\mathcal{K}$ anhand von $\mathsf{pk}_{\mathcal{D}}$ die zur Verschlüsselung eines Münz-Identifikators bzw. seiner Kontonummer benötigten temporären Schlüssel $\hat{z}_i = \prod_{k=0}^{t}(\hat{z}_k^*)^{i^k}$ und $\bar{z}_i = \prod_{k=0}^{t}(\bar{z}_k^*)^{i^k}$. Um eine gültige Münze zu erstellen, muss $\mathcal{K}$ zusätzlich mit Hilfe von $\mathsf{pk}_{\mathcal{B}}$ die folgenden temporären Schlüssel berechnen:

$$h_i = \prod_{k=0}^{t}(h_k^*)^{i^k},\ h_{1,i} = \prod_{k=0}^{t}(h_{1,k}^*)^{i^k},\ h_{2,i} = \prod_{k=0}^{t}(h_{2,k}^*)^{i^k},\ \text{und}\ h_{3,i} = \prod_{k=0}^{t}(h_{3,k}^*)^{i^k}.$$

Alternativ kann $\mathcal{K}$ sämtliche temporären Schlüssel $\hat{z}_j$ und $\bar{z}_j$ sowie h_j, h_{1j}, h_{2j} und h_{3j} für $j = 0, \ldots, N$ einmalig, zum Beispiel nach der Kontoeröffnung, berechnen. Die Bank $\mathcal{B}$ aktualisiert gegebenenfalls ihren Signaturschlüssel durch den Aufruf des Protokolls $\mathsf{BKeyUpdate}\left(\mathcal{B}\left(i-1, i, x_{i-1}\right), \mathcal{T}_{\mathcal{B}}\left(\mathsf{sk}_{\mathcal{B}}^*\right)\right)$ und berechnet ebenso wie $\mathcal{K}$ den temporären Schlüssel $\hat{z}_i = \prod_{k=0}^{t}(\hat{z}_k^*)^{i^k}$, den $\mathcal{B}$ zur Verifikation von V_1 und V_2 benötigt.
Um eine Münze Coin abzuheben, authentifizieren sich $\mathcal{K}$ und $\mathcal{B}$ zunächst gegenseitig. Anschließend wird das Protokoll $\mathsf{Withdrawal}\left(\mathcal{K}\left(u_1, \mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right), \mathcal{B}\left(x_i, \mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right)\right)$ durchgeführt (vgl. Abbildung 6.6):

Schritt 1: Der Kunde $\mathcal{K}$ wählt zufällig drei Zahlen $m, s, t \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet damit

$$I' = I^{s^{-1}} g_3^{s^{-1}} g_4^t \ (= g_1^{u_1 s^{-1}} g_3^{s^{-1}} g_4^t) \text{ sowie } E_1 = g_2^s \hat{z}_i^m, E_2 = g_5^m, E_3 = g_6^m.$$

Dabei ist $\langle i, (E_1, E_2, E_3)\rangle$ eine modifizierte ElGamal-Verschlüsselung des *Münz-Identifikators* g_2^s unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ von $\mathcal{D}$ im i-ten Intervall. Der Kunde $\mathcal{K}$ sendet I', das durch t perfekt geblendet ist, und $\langle i, (E_1, E_2, E_3)\rangle$ an $\mathcal{B}$. Anschließend beweist $\mathcal{K}$ der Bank $\mathcal{B}$ durch den nichtinteraktiven Zero-Knowledge-Beweis

$$V_1 = \text{NIZK}\left[(\alpha_1, \alpha_2) : E_1 = g_2^{\alpha_1} \hat{z}_i^{\alpha_2} \wedge E_2 = g_5^{\alpha_2} \wedge E_3 = g_6^{\alpha_2}\right],$$

dass es sich bei (E_1, E_2, E_3) tatsächlich um eine modifizierte ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel $\hat{z}_i$ von $\mathcal{D}$ im i-ten Intervall handelt (vgl. Abschnitt 2.7.1.1 und 2.9.1). Mit dem nichtinteraktiven Zero-Knowledge-Beweis

$$V_2 = \text{NIZK}\left[(\beta_1, \ldots, \beta_6) : g_3 = (I')^{\beta_1} g_1^{\beta_2} g_4^{\beta_3} \wedge E_1 = g_2^{\beta_1} \hat{z}_i^{\beta_4} \wedge I = (I')^{\beta_1} g_3^{\beta_5} g_4^{\beta_6}\right]$$

überzeugt $\mathcal{K}$ die Bank $\mathcal{B}$ zunächst davon, dass $\mathcal{K}$ den Wert E_1 bzgl. I' korrekt berechnet hat. Auf Grund der weiteren Konstruktion der Protokolle garantiert V_2

$\mathcal{K}(i, u_1, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$	$\mathcal{B}(i, x_i, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$
Einmal pro Intervall:	
$\hat{z}_i = \prod_{k=0}^{t} (\hat{z}_k^*)^{i^k}$	$\mathsf{BKeyUpdate}(\mathcal{B}((i-1), i, x_{i-1}), \mathcal{T}_\mathcal{B}(\mathsf{sk}_\mathcal{B}^*))$
$\bar{z}_i = \prod_{k=0}^{t} (\bar{z}_k^*)^{i^k}$	$\hat{z}_i = \prod_{k=0}^{t} (\hat{z}_k^*)^{i^k}$
$h_i = \prod_{k=0}^{t} (h_k^*)^{i^k}$	
$h_{1,i} = \prod_{k=0}^{t} (h_{1,k}^*)^{i^k}$	
$h_{2,i} = \prod_{k=0}^{t} (h_{2,k}^*)^{i^k}$	
$h_{3,i} = \prod_{k=0}^{t} (h_{3,k}^*)^{i^k}$	
$m, s, t \in_R \mathbb{Z}_q$	
$I' = I^{s^{-1}} g_3^{s^{-1}} g_4^t = g_1^{(u_1)s^{-1}} g_3^{s^{-1}} g_4^t$	
$E_1 = g_2^s \hat{z}_i^m, E_2 = g_5^m, E_3 = g_6^m$	
$V_1 = \mathrm{NIZK}[(\gamma_1, \gamma_2) : E_1 = g_2^{\gamma_1} \hat{z}_i^{\gamma_2} \wedge E_2 = g_3^{\gamma_2} \wedge E_3 = g_4^{\gamma_2}]$	
$V_2 = \mathrm{NIZK}[(\delta_1, \ldots, \delta_6) : g_3 = (I')^{\delta_1} g_1^{\delta_2} g_4^{\delta_3} \wedge E_1 = g_2^{\delta_1} \hat{z}_i^{\delta_4} \wedge I = (I')^{\delta_1} g_3^{\delta_5} g_4^{\delta_6}]$	
$\xrightarrow{I', E_1, E_2, E_3, V_1, V_2}$	
	$E_2 \overset{?}{\neq} 1$
	Verifiziere V_1, V_2
	$w \in_R \mathbb{Z}_q$
	$a' = g^w$
$\xleftarrow{a', b', b''}$	$b' = (I' g_2)^w, b'' = g_4^w$
$A = (I' g_2 g_4^{-t})^s = g_1^{u_1} g_2^s g_3$	
$z = h_{1,i}^{u_1} h_{2,i}^s h_{3,i} = A^{x_i}$	
$x_1, x_2, u, v, \in_\mathcal{R} \mathbb{Z}_q$	
$B = g_1^{x_1} g_2^{x_2}$	
$a = (a')^u g^v$	
$b = (b' b''^{-t})^{su} A^v$	
$c = \mathrm{H}(A, B, z, a, b)$	
$c' = cu^{-1} \bmod q$	
$\xrightarrow{c'}$	
$\xleftarrow{r'}$	$r' = c' x_i + w \bmod q$
$r = r'u + v \bmod q$	

Abbildung 6.6: Das Abhebe-Protokoll im langfristig sicheren System

der Bank $\mathcal{B}$, dass der für Münztracing benötigte *Münz-Identifikator* g_2^s verschlüsselt wurde.

Schritt 2: Nachdem sich $\mathcal{B}$ von der Korrektheit der beiden nichtinteraktiven Beweise V_1, V_2 überzeugt hat, wird anschließend I' von $\mathcal{B}$ blind signiert und damit eine Münze gemäß dem restriktiv blinden Signaturprotokoll von Brands [Bra94] erzeugt. Dazu wählt $\mathcal{B}$ zufällig eine Zahl $w \in_{\mathcal{R}} \mathbb{Z}_q$, berechnet damit die Werte

$$a' = g^w, b' = (I'g_2)^w \text{ und } b'' = g_4^w$$

und schickt a, b', b'' dem Kunden $\mathcal{K}$. Die *Restriktion* des blinden Signaturverfahrens gewährleistet, dass $\mathcal{K}$ seine Identität $I = g_1^{u_1}$ in die Münze integrieren muss.

Schritt 3: Der Kunde $\mathcal{K}$ berechnet dann

$$A = (I'g_2g_4^{-t})^s = Ig_2^sg_3.$$

Um auf die im Bezahl-Protokoll Payment geforderte Darstellung $(u_1, s, 1)$ von A bzgl. (g_1, g_2, g_3) zu kommen, muss $\mathcal{K}$ zur Blendung von A die bereits gewählte Zahl s verwenden. Das heißt, der für das Münztracing benötigte Münz-Identifikator g_2^s ist in A enthalten. Dies garantiert, dass $\mathcal{B}$ später in Kooperation mit $\mathcal{D}$ Münztracing durchführen kann. Mit Hilfe der aus dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}$ der Bank $\mathcal{B}$ berechneten Werte $h_{1,i}, h_{2,i}$ und $h_{3,i}$ berechnet $\mathcal{K}$ nun

$$z = h_{1,i}^{u_1}h_{2,i}^s h_{3,i} = A^{x_i},$$

wählt zufällig zwei weitere Zahlen $u, v \in_{\mathcal{R}} \mathbb{Z}_q$ und blendet damit

$$a' \text{ zu } a = a^u g^v \text{ sowie } b', b'' \text{ zu } b = (b'b''^{-t})^{su}A^v = A^{wu+v}.$$

Anschließend wählt $\mathcal{K}$ zufällig zwei weitere Zahlen $x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$, berechnet dann das Commitment B und die Challenge c mit

$$B = g_1^{x_1}g_2^{x_2} \text{ und } c = \mathrm{H}(A, B, z, a, b),$$

blendet den Wert c und erhält $c' = cu^{-1} \bmod q$. Die geblendete Challenge c' sendet $\mathcal{K}$ an $\mathcal{B}$.

Durch die Zahlen x_1, x_2, den privaten Schlüssel u_1 und der zum Blenden verwendeten Zahl s werden von $\mathcal{K}$ somit zwei Geraden mit Steigungen u_1 und s sowie Achsenabschnitten x_1 und x_2 festgelegt. Es ist wichtig, dass B in die Berechnung der Challenge c mit einfließt. Da c beim Bezahlen auch von $\mathcal{H}$ zur Verifikation der Signatur berechnet wird, muss $\mathcal{K}$ den Wert B auch beim Bezahlen verwenden. Das heißt, der Kunde $\mathcal{K}$ gibt beim einmaligen Bezahlen mit Coin jeweils einen Punkt der beiden Geraden preis. Bezahlt $\mathcal{K}$ erneut mit Coin, so gibt $\mathcal{K}$ zwei weitere Punkte der beiden Geraden preis und $\mathcal{B}$ kann die Identität von $\mathcal{K}$ aufdecken.

Schritt 4: Die Bank $\mathcal{B}$ berechnet dann $r' = c'x_i + w \bmod q$ und sendet die Response r' an $\mathcal{K}$ zurück. Dabei ist x_i der im Intervall i gültige private Schlüssel von $\mathcal{B}$.

Schritt 5: Zum Schluss des Protokolls berechnet $\mathcal{K}$ dann $r = r'u + v \bmod q$ und überprüft, ob er im Besitz einer gültigen Münze ist, indem er die beiden folgenden Gleichungen verifiziert:

$$g^r = h_i^c a,$$
$$A^r = z^c b.$$

Dabei handelt es sich bei $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ um eine gültige Münze, wie folgendes Lemma zeigt.

Lemma 6.2.1. Folgen der Kunde $\mathcal{K}$ und die Bank $\mathcal{B}$ dem in Abbildung 6.6 dargestellten Abhebe-Protokoll, so ist $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ eine gültige Münze, von der $\mathcal{K}$ eine Darstellung kennt.

Beweis. Damit $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ eine gültige Münze ist, müssen die beiden Gleichungen 6.1 und 6.2 aus Definition 6.2.1 erfüllt sein. Verhalten sich $\mathcal{K}$ und $\mathcal{B}$ dem Protokoll entsprechend, so gilt:

$$g^r = g^{r'u+v} = g^{r'u} g^v = g^{(c'x_i+w)u} g^v = (g^{x_i})^{c'u} (g^w)^u g^v = h_i^c a,$$
$$A^r = A^{r'u+v} = A^{r'u} A^v = A^{(c'x_i+w)u} A^v = (A^{x_i})^{c'u} A^{wu+v} = z_i^c b.$$

Somit ist $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ eine gültige Münze. Natürlich kennt $\mathcal{K}$ eine Darstellung von $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$, denn $\mathcal{K}$ kennt eine Darstellung von $B = g_1^{x_1} g_2^{x_2}$ bzgl. (g_1, g_2), da $\mathcal{K}$ die Darstellung (x_1, x_2) im Protokoll selbst wählt. Gleiches gilt für die Darstellung $(u_1, s, 1)$ von A bzgl (g_1, g_2, g_3). □

Da sich im Vergleich zu *FOLC* am Abhebe-Protokoll Withdrawal keine sicherheitsrelevanten Änderungen ergeben, ist die folgende Annahme, auf der die Identifikation eines Double-Spenders beruht, gerechtfertigt (vgl. Abschnitte 2.6.2 und 6.1.2).

Annahme 6.1. Das in Abbildung 6.6 dargestellte Protokoll ist ein restriktiv blindes Signaturprotokoll.

6.2.4 Mit einer Münze bezahlen

Während beim Abhebe-Protokoll Withdrawal sowohl die Interessen des Kunden $\mathcal{K}$, als auch die der Bank $\mathcal{B}$ gewahrt bleiben müssen, stehen beim Bezahl-Protokoll Payment neben den Interessen von $\mathcal{K}$ vor allem die Interessen des Händlers $\mathcal{H}$ und damit implizit auch die von $\mathcal{B}$ im Vordergrund. Auf der einen Seite muss aus Sicht von $\mathcal{K}$ dessen Anonymität gewährleistet bleiben. Auf der anderen Seite muss das Protokoll aus Sicht von $\mathcal{H}$ auch sicherstellen, dass $\mathcal{K}$ mit einer gültigen Münze Coin bezahlt. Damit $\mathcal{H}$ die Münze Coin später erfolgreich bei $\mathcal{B}$ einlösen kann, muss zusätzlich garantiert werden, dass die während des Protokolls von $\mathcal{K}$ erstellte Verschlüsselung $\langle j, (C_1, C_2, C_3)\rangle$ dessen Kontonummer I unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ chiffriert und $\mathcal{K}$ im Falle einer wiederholten Ausgabe von Coin ausreichend Information preisgibt, so dass $\mathcal{B}$ die Identität von $\mathcal{K}$ im Falle eines Double-Spending später aufdecken kann.

Während *FOLC* für die Konstruktion der für das Kundentracing benötigten Verschlüsselung von I die Münze selbst verwendet, kann dieser Ansatz hier nicht verwendet werden. Stattdessen erstellt $\mathcal{K}$ im j-ten Intervall die Verschlüsselung $\langle j, (C_1, C_2, C_3)\rangle$ seiner Kontonummer I und beweist während des Protokolls, dass $\langle j, (C_1, C_2, C_3)\rangle$ tatsächlich eine Verschlüsselung seiner Kontonummer ist. Diesen Beweis führt $\mathcal{K}$ mit dem in Abschnitt 2.7.1.2 vorgestellten Zero-Knowledge-Beweis

$$V_3 = \mathrm{HVZK}\left[(\lambda_1, \lambda_2, \lambda_3) : A_1 = g_1^{\lambda_1} g_2^{\lambda_2} \wedge C_1 = g_1^{\lambda_1} \bar{z}_j^{\lambda_3} \wedge C_2 = g_7^{\lambda_3} \wedge C_3 = g_8^{\lambda_3}\right].$$

Bevor das Bezahl-Protokoll durchgeführt wird, überprüfen $\mathcal{K}$ und $\mathcal{H}$, ob ihnen der temporäre Schlüssel $\bar{z}_j = \prod_{k=0}^{t} (\bar{z}_k^*)^{j^k}$ bereits vorliegt. Gegebenenfalls berechnen sie $\bar{z}_j$ anhand von $\mathsf{pk}_{\mathcal{D}}$. Anschließend wird das im Folgenden beschriebene Bezahl-Protokoll $\mathsf{Payment}(\mathcal{K}(u_1, \mathsf{Coin}, \mathsf{pk}_{\mathcal{D}}), \mathcal{H}(\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}))$ durchgeführt. Dabei wird zur Gewährleistung der Anonymität von $\mathcal{K}$ vorausgesetzt, dass $\mathcal{K}$ und $\mathcal{H}$ über einen anonymen Kanal kommunizieren.

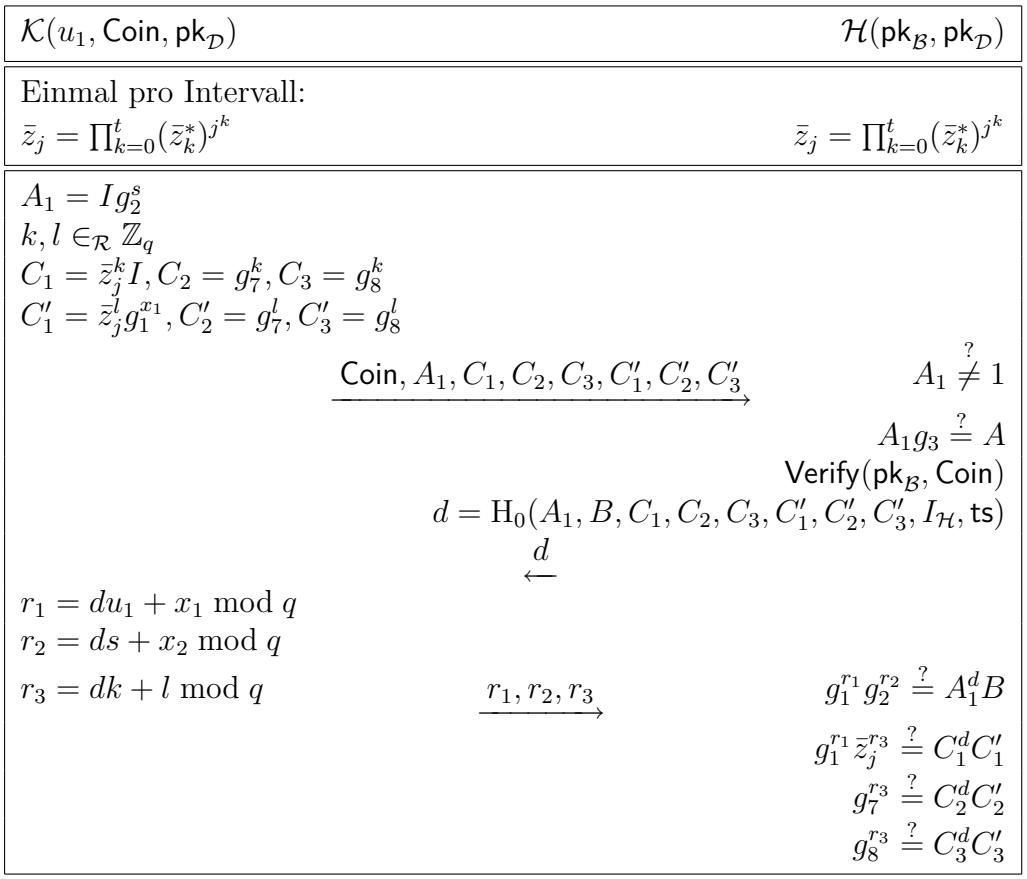

Abbildung 6.7: Das Bezahl-Protokoll im langfristig sicheren System

Schritt 1: Der Kunde $\mathcal{K}$ berechnet zunächst $A_1 = g_1^{u_1} g_2^s = A_3 g_3^{-1}$. Anschließend wählt $\mathcal{K}$ zufällig zwei Zahlen $k, l \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $C_1 = \bar{z}_j^k I$, $C_2 = g_7^k$ und $C_3 = g_8^k$.

Zusammen bilden $\langle j, (C_1, C_2, C_3)\rangle$ eine modifizierte ElGamal-Verschlüsselung der Identität $I = g_1^{u_1}$ des Kunden $\mathcal{K}$ im j-ten Intervall unter dem öffentlichen Schlüssel pk_{DM} des Deanonymisierers $\mathcal{D}$. Um den Händler $\mathcal{H}$ davon zu überzeugen, dass es sich bei $\langle j, (C_1, C_2, C_3)\rangle$ um eine modifizierte ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel von $\mathcal{D}$ handelt, berechnet $\mathcal{K}$ die Commitments $C_1' = \bar{z}_j^l g_1^{x_1}, C_2' = g_7^l, C_3' = g_8^l$. Der Kunde $\mathcal{K}$ sendet dann die Münze Coin, A_1, die modifizierte ElGamal-Verschlüsselung $\langle j, (C_1, C_2, C_3)\rangle$ sowie die Commitments C_1', C_2', C_3' an den Händler $\mathcal{H}$.

Schritt 2: Der Händler $\mathcal{H}$ überprüft, ob $A_1 \neq 1$ und $A_1 g_3 = A$ gilt, verifiziert die Signatur $\langle i, (z, a, b, r)\rangle$ der Münze Coin und berechnet die Challenge

$$\mathrm{H}_0(A_1, B, C_1, C_2, C_3, C_1', C_2', C_3', I_{\mathcal{H}}, \mathsf{ts}),$$

die $\mathcal{H}$ an $\mathcal{K}$ sendet. Da neben A_1, B, der modifizierten ElGamal-Verschlüsselung und den Commitments auch die Kontonummer $I_{\mathcal{H}}$ des Händlers $\mathcal{H}$ und der Zeitpunkt der Bezahlung ts in die Berechnung der Challenge einfließen, wird sichergestellt, dass verschiedene Händler mit überwältigender Wahrscheinlichkeit verschiedene Challenges berechnen. Genauso wird der gleiche Händler zu verschiedenen Zeitpunkten mit überwältigender Wahrscheinlichkeit verschiedene Challenges berechnen.

Schritt 3: Nachdem $\mathcal{K}$ die Challenge d empfangen hat, berechnet $\mathcal{K}$ die Responses $r_1 = du_1 + x_1 \bmod q, r_2 = ds + x_2 \bmod q$ und $r_3 = dk + l \bmod q$ und sendet r_1, r_2, r_3 an $\mathcal{H}$.

Schritt 4: Abschließend verifiziert der Händler $\mathcal{H}$ folgende Gleichungen:

$$\begin{aligned} g_1^{r_1} g_2^{r_2} &= A_1^d B, \\ g_1^{r_1} \bar{z}_j^{r_3} &= C_1^d C_1', \\ g_7^{r_3} &= C_2^d C_2', \\ g_8^{r_3} &= C_3^d C_3'. \end{aligned}$$

Gelingen diese Verifikationen, so hat $\mathcal{K}$ den Händler $\mathcal{H}$ davon überzeugt, dass $\mathcal{K}$ das Tripel (u_1, s, k) kennt, für das $A_1 = g_1^{u_1} g_2^s$ und $C_1 = g_1^{u_1} \bar{z}_j^k, C_2 = g_7^k, C_3 = g_8^k$ gelten. Damit hat $\mathcal{K}$ nachgewiesen, dass $\langle j, C_1, C_2, C_3\rangle$ im Intervall j eine modifizierte ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ ist. Da $\mathcal{K}$ während des Abhebe-Protokolls $\mathsf{Withdrawal}$ seine Identität $I = g_1^{u_1}$ zur Berechnung von A verwenden muss und $\mathcal{H}$ zu Beginn des Bezahl-Protokolls die Gleichung $A = A_1 g_3$ überprüft, handelt es sich bei $\langle j, (C_1, C_2, C_3)\rangle$ um eine modifizierte ElGamal-Verschlüsselung der Identität I von $\mathcal{K}$. Außerdem hat $\mathcal{H}$ von $\mathcal{K}$ jeweils einen Punkt der beiden, während des Abhebe-Protokoll $\mathsf{Withdrawal}$ festgelegten, Geraden erhalten. Somit kann $\mathcal{B}$ zusammen mit $\mathcal{D}$ Kundentracing durchführen und im Falle einer erneuten Ausgabe von Coin die Identität von $\mathcal{K}$ aufdecken.

Bemerkung 6.2.5. Kennt $\mathcal{K}$ die Kontonummer $I_{\mathcal{H}}$ des Händlers $\mathcal{H}$ und verfügen beide über eine synchronisierte Uhrzeit, so kann das Protokoll Payment nichtinteraktiv durchgeführt werden (vgl. Abschnitt 2.9.1):

$$\text{NIZK}\left[(\lambda_1,\lambda_2,\lambda_3) : A_1 = g_1^{\lambda_1} g_2^{\lambda_2} \wedge C_1 = g_1^{\lambda_1} \bar{z}_j^{\lambda_3} \wedge C_2 = g_7^{\lambda_3} \wedge C_3 = g_8^{\lambda_3}\right](\mathsf{Coin}, I_{\mathcal{H}}, \mathsf{ts}).$$

Folgendes Lemma zeigt, dass ehrliche Kunden ihre Münzen erfolgreich als Zahlungsmittel verwenden können und Angreifer abgefangene Münzen ohne die Kenntnis einer Darstellung nicht ausgeben können.

Lemma 6.2.2. Genau dann kann der Kunde $\mathcal{K}$ mit einer Münze Coin bezahlen, wenn er eine Darstellung von Coin kennt.

Beweis. Kennt der Kunde $\mathcal{K}$ eine Darstellung der Münze $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ und verhält er sich gemäß dem in Abbildung 6.7 dargestellten Protokoll, dann wird der Händler $\mathcal{H}$ die Münze akzeptieren.

$$\begin{aligned}
g_1^{r_1} g_2^{r_2} &= g_1^{du_1+x_1} g_2^{ds+x_2} = (g_1^{u_1})^d g_1^{x_1} (g_2^s)^d g_2^{x_2} = (g_1^{u_1} g_2^s)^d g_1^{x_1} g_2^{x_2} = A_1^d B \\
g_1^{r_1} \bar{z}_i^{r_3} &= g_1^{du_1+x_1} z_i^{dk+l} = (g_1^{u_1})^d g_1^{x_1} (\bar{z}_j^k)^d \bar{z}_j^l = (I \bar{z}_j^k)^d \bar{z}_j^l = C_1^d C_1' \\
g_7^{r_3} &= g_7^{dk+l} = C_2^d C_2' \\
g_8^{r_3} &= g_8^{dk+l} = C_3^d C_3'
\end{aligned}$$

Kann $\mathcal{K}$ umgekehrt mit der Münze Coin bezahlen, so muss er eine Darstellung von Coin kennen. Da das Bezahl-Protokoll im Wesentlichen dem Zero-Knowledge-Beweis aus Abschnitt 2.7.1.2 entspricht, folgt dies unmittelbar aus Satz 2.7.2. □

6.2.5 Eine Münze einlösen

Möchte der Händler $\mathcal{H}$, die vom Kunden $\mathcal{K}$ erhaltene Münze Coin bei der Bank $\mathcal{B}$ einlösen, wird das in Abbildung 6.8 dargestellte Protokoll $\mathsf{Deposit}\left(\mathcal{H}\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right)\right)$ ausgeführt. Zunächst authentifiziert sich $\mathcal{H}$ gegenüber $\mathcal{B}$ indem $\mathcal{H}$ den nichtinteraktiven Beweis $V_4 = \text{NIZK}\left[\delta : I_{\mathcal{H}} = g_1^{\delta}\right]\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right)$ erstellt und V_4 zusammen mit dem Zeitpunkt der Bezahlung ts und

$$\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P) = \{\mathsf{Coin}, A_1, C_1, C_2, C_3, C_1', C_2', C_3', r_1, r_2, r_3\}$$

an $\mathcal{B}$ sendet.
Ist $A_1 = 1$, so wird die Bank den Beleg nicht akzeptieren. Ist $A_1 \neq 1$, so berechnet $\mathcal{B}$ die Challenge $d = \text{H}_0(A_1, B, C_1, C_2, C_3, C_1', C_2', C_3', I_{\mathcal{H}}, \mathsf{ts})$ im vorher durchgeführten Bezahl-Protokoll zwischen $\mathcal{K}$ und $\mathcal{H}$ aus der $\mathcal{B}$ bekannten Kontonummer $I_{\mathcal{H}}$ des Händler $\mathcal{H}$ und dem übertragenen Zeitpunkt der Bezahlung ts, sowie $A_1, B, C_1, C_2, C_3, C_1', C_2', C_3'$. Anhand von $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ verifiziert $\mathcal{B}$ die Signatur $\langle i, (z, a, b, r)\rangle$ der Münze Coin und das zwischen $\mathcal{K}$ und $\mathcal{H}$ durchgeführte Bezahl-Protokoll, welches von $\mathcal{B}$ aus zwei Gründen überprüft wird: Zum einen möchte $\mathcal{B}$ im Falle eines Double-Spending der Münze Coin durch $\mathcal{K}$ dessen Identität I aufdecken und zum anderen will $\mathcal{B}$ sicherstellen, dass $\langle j, (C_1, C_2, C_3)\rangle$

tatsächlich eine Verschlüsselung der Kontonummer I des betreffenden Kunden $\mathcal{K}$ im Intervall j unter dem öffentlichen Schlüssel von $\mathcal{D}$ ist.
Gelingt der Bank eine der beiden Verifikationen nicht, so wird $\mathcal{B}$ den Beleg nicht akzeptieren. Andernfalls prüft die Bank, ob Coin bereits in der Datenbank enthalten ist. Ist Coin noch nicht in der Datenbank enthalten, so wird Coin zusammen mit $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ in der Münztabelle der Datenbank gespeichert. Der Wert der Münze wird dem Händler $\mathcal{H}$ gut geschrieben.

$\mathcal{H}\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right)$		$\mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}\right)$
$V_4 = \mathrm{NIZK}\left[\delta : I_{\mathcal{H}} = g_1^{\delta}\right]\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)\right)$		
	$\xrightarrow{\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P), V_4, \mathsf{ts}}$	Verifiziere V_4
		$A_1 \stackrel{?}{\neq} 1$
		$c = \mathrm{H}(A, B, z, a, b)$
		$g^r \stackrel{?}{=} h^c a'$
		$A^r \stackrel{?}{=} z'^c b'$
		$A_1 g_3 \stackrel{?}{=} A$
		$d = \mathrm{H}_0(A_1, B, C_1, C_2, C_3, C_1', C_2', C_3', I_{\mathcal{H}}, \mathsf{ts})$
		$g_1^{r_1} g_2^{r_2} \stackrel{?}{=} {A_1}^d B$
		$g_1^{r_1} \bar{z_i}^{r_3} \stackrel{?}{=} C_1^d C_1'$
		$g_7^{r_3} \stackrel{?}{=} C_2^d C_2'$
		$g_8^{r_3} \stackrel{?}{=} C_3^d C_3'$

Abbildung 6.8: Das Einlöse-Protokoll Deposit im langfristig sicheren System

6.2.5.1 Identifikation von Double-Spendern

Ist die Münze Coin bereits in der Datenbank enthalten, so gibt es zwei Möglichkeiten:

1. Der Kunde $\mathcal{K}$ hat Coin bereits zum zweiten Mal ausgegeben.
2. Der Händler $\mathcal{H}$ will Coin ein zweites Mal einlösen.

Versucht $\mathcal{H}$ die Münze Coin erneut einzulösen, muss $\mathcal{H}$ der Bank $\mathcal{B}$, um Coin ein weiteres Mal erfolgreich einzulösen, neben $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ eine weitere gültige Protokollansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P')$ präsentieren. Ohne eine Darstellung der Münze Coin zu kennen, kann $\mathcal{H}$ aber nur mit vernachlässigbarer Wahrscheinlichkeit eine zweite gültige Protokollansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P')$ präsentieren (vgl. Lemma 6.2.2). Das heißt, der Händler $\mathcal{H}$ kann lediglich $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ verwenden, um Coin erneut einzulösen. Da $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ bereits in der Datenbank gespeichert ist, kann $\mathcal{B}$ daraus schließen, dass $\mathcal{H}$ versucht die Münze Coin ein zweites Mal einzulösen. In diesem Fall wird $\mathcal{B}$ dem Händler $\mathcal{H}$ den Wert von Coin nicht gutschreiben.

Dementsprechend erkennt $\mathcal{B}$, dass $\mathcal{K}$ die Münze Coin ein zweites Mal ausgegeben hat, falls Coin bereits in der Datenbank gespeichert ist, aber $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ noch nicht. Da Coin schon eingelöst wurde, ist die Protokollansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(\tilde{P})$ eines Bezahl-Protokolls $\tilde{P}$ mit Coin in der Datenbank gespeichert. Anhand von $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ und $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(\tilde{P})$ kann $\mathcal{B}$ die Identität von $\mathcal{K}$ durch den Algorithmus $\mathsf{Identify}\left(\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}\left(\tilde{P}\right)\right)$ wie folgt aufdecken:
Anhand der in $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}\left(\tilde{P}\right)$ enthaltenen Paare $(r_1, d), (\tilde{r}_1, \tilde{d})$ berechnet $\mathcal{B}$ die Kontonummer

$$I = g_1^{\frac{r_1 - \tilde{r}_1}{d - \tilde{d}}}$$

und durchsucht ihre Datenbank nach dem Kunden $\mathcal{K}$ mit Kontonummer I. Lemma 6.2.7 zeigt, dass der Algorithmus $\mathsf{Identify}$ tatsächlich die Kontonummer des Double-Spenders liefert.

6.2.6 Kunden- und Münztracing

Kundentracing

Besteht zum Beispiel der dringende Verdacht, dass ein Händler $\mathcal{H}$ aus illegalen Geschäften stammendes Geld wäscht, so ist die Herkunft des Geldes für die Staatsanwaltschaft interessant. Da der in diesem Falle kriminelle Kunde $\mathcal{K}$ während der Durchführung des Bezahl-Protokolls $\mathsf{Payment}$ seine Kontonummer unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ verschlüsseln muss (andernfalls würde $\mathcal{B}$ die Münze Coin von $\mathcal{H}$ später nicht akzeptieren), kann die Herkunft der Münze Coin durch $\mathcal{B}$ in Kooperation mit $\mathcal{D}$ nachgewiesen werden. Dazu führen $\mathcal{B}$ und $\mathcal{D}$ das in Abbildung 6.9 dargestellte Protokoll $\mathsf{KTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)\right), \mathcal{D}\left(\overline{\mathsf{sk}}_{\mathcal{D}}^{(j)}, \mathsf{pk}_{\mathcal{D}}\right)\right)$ durch. Es wird angenommen, dass das Bezahl-Protokoll im i-ten Intervall durchgeführt wurde und das aktuelle Intervall das j-te Intervall ist.

Schritt 1: Die Bank $\mathcal{B}$ sendet die in $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)$ enthaltene modifizierte ElGamal-Verschlüsselung $\langle i, C_1, C_2, C_3\rangle$ an den Deanonymisierer $\mathcal{D}$.

Schritt 2: Um $\langle i, C_1, C_2, C_3\rangle$ zu entschlüsseln, benötigt $\mathcal{D}$ den im i-ten Intervall gültigen privaten Schlüssel $\overline{\mathsf{sk}}_{\mathcal{D}}^{(i)} = (\bar{x}_i, \bar{y}_i)$. Stimmen die Indizes i und j überein, kann $\mathcal{D}$ den Geheimtext mit seinem aktuellen privaten Schlüssel entschlüsseln. Andernfalls aktualisiert $\mathcal{D}$ den privaten Schlüssel durch $\mathsf{DKeyUpdate}_2\left(\mathcal{D}\left(j, i, \mathsf{sk}_{\mathcal{D}}^{(j)}\right), \mathcal{T}_{\mathcal{D}}\left(\mathsf{sk}_{\mathcal{D}}^{*}\right)\right)$. Anschließend entschlüsselt $\mathcal{D}$ den Geheimtext wie folgt:

$$C_1 C_2^{-\bar{x}_i} C_3^{-\bar{x}_i} = I \bar{z}_i^k g_7^{-\bar{x}_i k} g_8^{-\bar{y}_i k} = I.$$

Die Kontonummer I sendet $\mathcal{D}$ an $\mathcal{B}$.

Schritt 3: Anhand der Kontonummer I kann $\mathcal{B}$ durch Suche in ihrer Datenbank den verdächtigen Kunden $\mathcal{K}$ ermitteln.

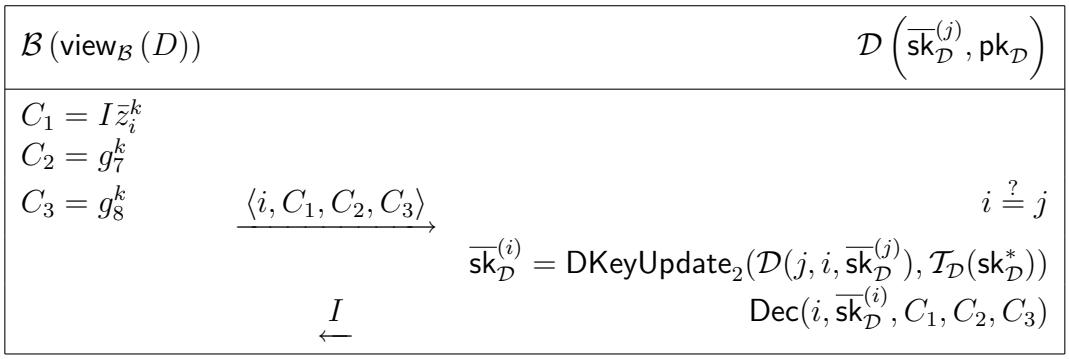

Abbildung 6.9: Kundentracing im langfristig sicheren System

Münztracing

Um Münzen eines Kunden $\mathcal{K}$, der auf Grund einer Erpressung durch einen Händler $\mathcal{H}$ mit seinen Münzen bei $\mathcal{H}$ bezahlt hat, zu erkennen bzw. das Ziel bestimmter Münzen ausfindig zu machen, wird während des Abhebe-Protokolls $\mathsf{Withdrawal}$ im i-ten Intervall eine modifizierte ElGamal-Verschlüsselung $\langle i, E_1, E_2, E_3\rangle$ des Münz-Identifikators g_2^s unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ von $\mathcal{D}$ erstellt. Anhand des Münz-Identifikators kann $\mathcal{B}$ mit Hilfe von $\mathcal{D}$ die Münzen beim Einzahlen ausfindig machen und so den Erpresser überführen. Dazu wird im j-ten Intervall das in Abbildung 6.10 dargestellte Protokoll $\mathsf{MTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)\right), \mathcal{D}\left(\widehat{\mathsf{sk}}_{\mathcal{D}}^{(j)}, \mathsf{pk}_{\mathcal{D}}\right)\right)$ durchgeführt.

Schritt 1: Die Bank $\mathcal{B}$ sendet die in $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ enthaltene modifizierte ElGamal-Verschlüsselung $\langle i, E_1, E_2, E_3\rangle$ an den Deanonymisierer $\mathcal{D}$.

Schritt 2: Der Deanonymisierer $\mathcal{D}$ benötigt, wie im Falle des Kundentracings, den für das i-te Intervall gültigen privaten Schlüssel $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)} = (\hat{x}_i, \hat{y}_i)$. Stimmen die Indizes i und j überein, kann $\mathcal{D}$ den Geheimtext mit seinem aktuellen privaten Schlüssel entschlüsseln. Andernfalls aktualisiert $\mathcal{D}$ den privaten Schlüssel durch $\mathsf{DKeyUpdate}_1\left(\mathcal{D}\left(j, i, \widehat{\mathsf{sk}}_{\mathcal{D}}^{(j)}\right), \mathcal{T}_{\mathcal{D}}(\mathsf{sk}_{\mathcal{D}}^*)\right)$. Mit dem privaten Schlüssel $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)}$ berechnet $\mathcal{D}$ den Klartext

$$E_1 E_2^{-\hat{x}_i} E_3^{-\hat{y}_i} = g_2^s$$

und sendet den Münz-Identifikator g_2^s an $\mathcal{B}$.

Schritt 3: Die Bank $\mathcal{B}$ berechnet anhand der in $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ enthaltenen Kontonummer I von $\mathcal{K}$ den Münzteil $A = I g_2^s g_3$, durchsucht die bereits eingelösten Münzen nach A und kann somit den Händler $\mathcal{H}$ bestimmen, bei dem mit der Münze bezahlt wurde.

$\mathcal{B}(\mathsf{view}_{\mathcal{B}}(W))$		$\mathcal{D}\left(\widehat{\mathsf{sk}}_{\mathcal{D}}^{(j)}, \mathsf{pk}_{\mathcal{D}}\right)$
$E_1 = g_2^s \hat{z}_j^m$ $E_2 = g_5^m$ $E_3 = g_6^m$	$\xrightarrow{\langle j, E_1, E_2, E_3\rangle}$	$i \stackrel{?}{=} j$
		$\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)} = \mathsf{DKeyUpdate}_1(\mathcal{D}(j, i, \widehat{\mathsf{sk}}_{\mathcal{D}}^{(j)}), \mathcal{T}_{\mathcal{D}}(\mathsf{sk}_{\mathcal{D}}^*))$
	$\xleftarrow{g_2^s}$	$\mathsf{Dec}(i, \widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)}, E_1, E_2, E_3)$
$A = I g_2^s g_3$		

Abbildung 6.10: Münztracing im langfristig sicheren System

6.2.7 Sicherheitsanalyse

Ziel der Sicherheitsanalyse ist es zu zeigen, dass das in diesem Abschnitt vorgestellte aktualisierbare faire elektronische Geldsystem Ω ein (t, N)-langfristig sicheres elektronisches Geldsystem im Sinne von Definition 5.3.2 ist. Um dies zu beweisen, zeigen wir zunächst, dass Ω ein $(0, 1)$-langfristig sicheres faires elektronisches Geldsystem ist. Das heißt, für $t = 0$ und $N = 1$ ist Ω gemäß Definition 3.3.2 ein sicheres elektronisches Geldsystem.

6.2.7.1 Sicherheitsanalyse für $t = 0$ und $N = 1$

Anonymität: Wir müssen zeigen, dass es keinen effizienten Angreifer $\mathcal{A}$ gibt, der bei Eingabe zweier gegebener Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0), \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$, der in W_0 und W_1 abgehobenen Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ und den Ansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ der entsprechenden Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört, wobei ν eine nicht vernachlässigbare Funktion ist. Dazu sind die folgenden Daten gegeben:

- Abheben: $\begin{cases} \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0) = \left\{g_1^{u_0}, E_1^{(0)}, E_2^{(0)}, E_3^{(0)}, V_1^{(0)}, V_2^{(0)}, c'^{(0)}\right\} \\ \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1) = \left\{g_1^{u_1}, E_1^{(1)}, E_2^{(1)}, E_3^{(1)}, V_1^{(1)}, V_2^{(1)}, c'^{(1)}\right\} \end{cases}$
- Bezahlen: $\begin{cases} \mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b) = \left\{A_1^{(b)}, A_2^{(b)}, B^{(b)}, C_1^{(b)}, C_2^{(b)}, C_3^{(b)}, (z^{(b)}, a^{(b)}, b^{(b)}, r^{(b)}), V_3^{(b)}\right\} \\ \mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}}) = \left\{A_1^{(\bar{b})}, A_2^{(\bar{b})}, B^{(\bar{b})}, C_1^{(\bar{b})}, C_2^{(\bar{b})}, C_3^{(\bar{b})}, (z^{(\bar{b})}, a^{(\bar{b})}, b^{(\bar{b})}, r^{(\bar{b})}), V_3^{(\bar{b})}\right\} \end{cases}$
  mit $b \neq \bar{b}$ und $b, \bar{b} \in \{0, 1\}$.

Um zu zeigen, dass es keinen effizienten Angreifer $\mathcal{A}$ geben kann, der anhand der gegebenen Daten mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, welche Münze zu welchem Abhebe-Protokoll gehört, können wir, wie bereits in Abschnitt 6.1.3 erwähnt, nicht auf den Sicherheitsbeweis von *FOLC* zurückgreifen. Stattdessen werden

wir die Anonymität des Systems mit der in [GT03] beschriebenen Vorgehensweise zeigen, wozu wir das so genannte *Matching-Withdrawal-Payment-Problem* und das *Decisional-Withdrawal-Payment-Problem* benötigen.

Definition 6.2.2. Für ein elektronisches Geldsystem seien Matching-Withdrawal-Payment- und Decisional-Withdrawal-Payment-Problem wie folgt definiert:

1. *Matching-Withdrawal-Payment-Problem*: Gegeben seien die Ansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_1)$, die in W_0 und W_1 abgehobenen Münzen Coin_b, $\mathsf{Coin}_{\bar{b}}$ ($b, \bar{b} \in \{0,1\}$ mit $b \neq \bar{b}$) sowie die entsprechenden Ansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}}(P_{\bar{b}})$. Bestimme b.

2. *Decisional-Withdrawal-Payment-Problem*: Gegeben sei eine von $\mathcal{K}$ abgehobene und ausgegebene Münze Coin, eine Ansicht $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ sowie die Ansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ in der mit Coin bezahlt wurde. Entscheide ob Coin in W abgehoben wurde.

Offensichtlich kann das Matching-Withdrawal-Payment-Problem gelöst werden, falls das Decisional-Withdrawal-Payment-Problem gelöst werden kann.

Lemma 6.2.3. Gegeben sei ein elektronisches Geldsystem mit Sicherheitsparameter $k \in \mathbb{N}$ und eine nicht vernachlässigbare Funktion ν. Falls das Decisional-Withdrawal-Payment-Problem mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ gelöst werden kann, kann das Matching-Withdrawal-Payment-Problem mit der Wahrscheinlichkeit $\epsilon(k)$ gelöst werden.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der das Decisional-Withdrawal-Payment-Problem mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ lösen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Dann kann $\mathcal{A}$ anhand einer Münze Coin, der Protokollansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ des Bezahl-Protokolls P in dem mit Coin bezahlt wurde und der Protokollansicht $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ eines Abhebe-Protokolls W mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, ob Coin in W abgehoben wurde.
Ein effizienter Angreifer $\mathcal{A}'$ kann $\mathcal{A}$ dazu verwenden, das Matching-Withdrawal-Payment-Problem mit der Wahrscheinlichkeit $\epsilon(k)$ zu lösen. Dazu übergibt $\mathcal{A}'$ die Ansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_0)$, $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}}(P_b)$ und die Münze Coin_b an $\mathcal{A}$, der mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden kann, ob Coin_b in W_0 abgehoben wurde. Folglich kann $\mathcal{A}'$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, ob Coin_b in W_0 oder W_1 abgehoben wurde. □

Um die Anonymität von Ω zu beweisen, zeigen wir zunächst, dass es keinen effizienten Angreifer gibt, der das Matching-Withdrawal-Payment-Problem mit einer Wahrscheinlich $\epsilon(k) \geq 1/2 + \nu(k)$ gewinnt.

Lemma 6.2.4. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ und eine nicht vernachlässigbare Funktion ν. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der das Matching-Withdrawal-Payment-Problem in Ω mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ lösen kann.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der das Matching-Withdrawal-Payment-Problem in Ω mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ lösen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Dann kann $\mathcal{A}$ von einem effizienten Angreifer $\mathcal{A}'$ dazu verwenden, die modifizierte ElGamal-Verschlüsselung im Sinne der polynomiellen Ununterscheidbarkeit zu brechen.
Der Angreifer $\mathcal{A}'$ versucht folgende Instanz des modifizierten ElGamal-Verschlüsselungsverfahren im Sinne der polynomiellen Ununterscheidbarkeit zu brechen.

- Primzahlen p, q mit $|p-1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Generatoren g_5, g_6 der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.
- $(\mathsf{pk}, \mathsf{sk}) = ((p, q, g_5, g_6, \hat{z}), (\hat{x}_\mathcal{D}, \hat{y}_\mathcal{D}))$ mit $\hat{z} = g_5^{\hat{x}_\mathcal{D}} g_6^{\hat{y}_\mathcal{D}}$ und $\hat{x}_\mathcal{D}, \hat{y}_\mathcal{D} \in_\mathcal{R} \mathbb{Z}_q$.

Um die Instanz $(\mathsf{pk}, \mathsf{sk})$ der modifizierten ElGamal-Verschlüsselung im Sinne der polynomiellen Ununterscheidbarkeit zu brechen wählt $\mathcal{A}'$ zufällig $\delta \in_\mathcal{R} \mathbb{Z}_q^*$ und berechnet $g_2 = g_5^\delta$. Weiter wählt $\mathcal{A}'$ die Werte $s_0, s_1 \in \mathbb{Z}_q$ und berechnet die Nachrichten $m_0 = g_2^{s_0}$ sowie $m_1 = g_2^{s_1}$. Es seien

$$E^{(0)} = (E_1^{(0)}, E_2^{(0)}, E_3^{(0)}) = (g_2^{s_b} \hat{z}^{t_b}, g_5^{t_b}, g_6^{t_b}) \text{ und } E^{(1)} = (E_1^{(1)}, E_2^{(1)}, E_3^{(1)}) = (g_2^{s_{\bar{b}}} \hat{z}^{t_{\bar{b}}}, g_5^{t_{\bar{b}}}, g_6^{t_{\bar{b}}})$$

die modifizierten ElGamal-Verschlüsselungen der beiden Nachrichten in einer durch $b \in_\mathcal{R} \{0,1\}$ zufällig gewählten Reihenfolge, wobei $t_b, t_{\bar{b}} \in_\mathcal{R} \mathbb{Z}_q^*$ und b dem Angreifer $\mathcal{A}'$ nicht bekannt ist.
Anhand der beiden Nachrichten m_0, m_1 und der beiden modifizierten ElGamal-Verschlüsselungen $E^{(0)}, E^{(1)}$ versucht $\mathcal{A}'$ zu entscheiden welche Verschlüsselung zu welcher Nachricht gehört. Um $\mathcal{A}$ dafür zu verwenden übersetzt $\mathcal{A}'$ die beiden modifizierten ElGamal-Verschlüsselungen $E^{(0)}, E^{(1)}$ wie folgt in eine Instanz des Matching-Withdrawal-Payment-Problem:

Schritt 1: Der Angreifer $\mathcal{A}'$ wählt zufällig $X_\mathcal{B}, \bar{x}_\mathcal{D}, \bar{y}_\mathcal{D} \in_\mathcal{R} \mathbb{Z}_q, g_1, g_3, g_4, g_7, g_8 \in_\mathcal{R} G_q$. Anschließend berechnet $\mathcal{A}'$ die Werte $h_1 = g_1^{X_\mathcal{B}}, h_2 = g_2^{X_\mathcal{B}}, h_3 = g_3^{X_\mathcal{B}}$ und die öffentlichen Schlüssel $\bar{z} = g_7^{\bar{x}_\mathcal{D}} g_8^{\bar{y}_\mathcal{D}}$. Außerdem wählt $\mathcal{A}'$ zufällig $u_1 \in_\mathcal{R} \mathbb{Z}_q^*$ und berechnet $I = g_1^{u_1}$.

Schritt 2: Der Angreifer $\mathcal{A}'$ simuliert für beide Münzen die Protokollansichten der Bank, indem er das Random-Oracle sowie die nichtinteraktiven Zero-Knowledge-Beweise $V_1^{(b)}, V_1^{(\bar{b})}, V_2^{(b)}, V_2^{(\bar{b})}$ simuliert und $c'_b, c'_{\bar{b}} \in_\mathcal{R} \mathbb{Z}_q^*$ wählt. Gemäß den Abschnitten 2.7.1.1, 2.7.1.2 und 2.9.1.1 kann $\mathcal{A}'$ das Random-Oracle und die Beweise perfekt simulieren. Die Simulationen von $V_1^{(b)}$ und $V_2^{(b)}$ sehen wie folgt aus:

$V_1^{(b)} = \text{NIZK}\left[(\gamma_1^{(b)}, \gamma_2^{(b)}) : E_1^{(b)} = g_2^{\gamma_1^{(b)}} \hat{z}^{\gamma_2^{(b)}} \wedge E_2^{(b)} = g_5^{\gamma_2^{(b)}} \wedge E_3^{(b)} = g_6^{\gamma_2^{(b)}}\right]$:

- Wähle zufällige Werte $c, r_1, r_2 \in_\mathcal{R} \mathbb{Z}_q$.
- Berechne $E'_1 = \hat{z}^{r_1} g_2^{r_2} E_1^{-c}, E'_2 = g_5^{r_1} E_2^{-c}, E'_3 = g_6^{r_2} E_3^{-c}$.
- Setze $\text{H}(E'_1, E'_2, E'_3) := c$.

- $V_1 = (c, r_1, r_2)$.

$V_2^{(b)} = \text{NIZK}\left[(\delta_1^{(b)}, \ldots, \delta_6^{(b)}) : g_3 = (I')^{\delta_1^{(b)}} g_1^{\delta_2^{(b)}} g_4^{\delta_3^{(b)}} \wedge \ldots \wedge I = (I')^{\delta_1^{(b)}} g_3^{\delta_5^{(b)}} g_4^{\delta_6^{(b)}}\right]$:

- Wähle zufällige Werte $c, r, r_1, r_2, r_3, \hat{r}_1, \hat{r}_3 \in \mathbb{Z}_q$.
- Berechne $\tilde{g_3} = I'^r g_1^{r_1} g_4^{\hat{r}_1} g_3^{-c}, \tilde{E}_1 = g_2^r \hat{z}^{r_2} E_1^{-c}, \tilde{I} = I'^r g_3^{r_3} g_4^{\hat{r}_3} I^{-c}$.
- Setze $\text{H}(\tilde{g_3}, \tilde{E}_1, \tilde{I}) := c$.
- $V_2 = (c, r, r_1, r_2, r_3, \hat{r}_1, \hat{r}_3)$.

Die nichtinteraktiven Zero-Knowledge-Beweise $V_1^{(\bar{b})}$ und $V_2^{(\bar{b})}$ lassen sich analog perfekt simulieren. Es ergeben sich folgende Sichten der Bank $\mathcal{B}$:

$$\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_b) = \{I, E_1^{(b)}, E_2^{(b)}, E_3^{(b)}, V_1^{(b)}, V_2^{(b)}, c'_b\}$$
$$\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_{\bar{b}}) = \{I, E_1^{(\bar{b})}, E_2^{(\bar{b})}, E_3^{(\bar{b})}, V_1^{(\bar{b})}, V_2^{(\bar{b})}, c'_{\bar{b}}\}$$

Schritt 3: Da $\mathcal{A}'$ die Werte s_0 und s_1 kennt, kann $\mathcal{A}'$ die Werte $A_1 = g_1^{u_1} g_2^{s_0}$ und $\hat{A}_1 = g_1^{u_1} g_2^{s_1}$ berechnen. Außerdem kann $\mathcal{A}'$ die beiden modifizierten ElGamal-Verschlüsselungen $(C_1, C_2, C_3), (\hat{C}_1, \hat{C}_2, \hat{C}_3)$ berechnen indem er zufällig $\alpha, \hat{\alpha} \in_{\mathcal{R}} \mathbb{Z}_q$ wählt und $C_1 = g_7^{\alpha}, C_2 = g_8^{\alpha}, C_3 = g_1^{u_1} \bar{z}^{\alpha}$ bzw. $\hat{C}_1 = g_7^{\hat{\alpha}}, \hat{C}_2 = g_8^{\hat{\alpha}}, \hat{C}_3 = g_1^{u_1} \bar{z}^{\hat{\alpha}}$ berechnet. Mit diesen Werten kann $\mathcal{A}'$ die beiden Bezahl-Protokolle V_3 und $\hat{V}_3$ gemäß den Abschnitten 2.7.1.3 und 2.9.1.1 perfekt simulieren:

$V_3 = \text{NIZK}\left[(\lambda_1, \lambda_2, \lambda_3) : A_1 = g_1^{\lambda_1} g_2^{\lambda_2} \wedge C_1 = g_1^{\lambda_1} \bar{z}^{\lambda_3} \wedge C_2 = g_7^{\lambda_3} \wedge C_3 = g_8^{\lambda_3}\right](\cdots)$:

- Wähle zufällige Werte $d, r_1, r_2, r_3 \in_{\mathcal{R}} \mathbb{Z}_q$.
- Berechne $C'_1 = g_7^{r_3} C_1^{-d}, C'_2 = g_8^{r_3} C_2^{-d}, C'_3 = g_1^{r_1} \bar{z}^{r_3} C_3^{-d}, B = g_1^{r_1} g_2^{r_2} A_1^{-d}$.
- Setze $\text{H}(A, B, C_1, C_2, C_3, C'_1, C'_2, C'_3, I_{\mathcal{H}}, \mathsf{ts}) := d$.

Der nichtinteraktive Zero-Knowledge-Beweis $\hat{V}_3$ lässt sich analog simulieren.

Schritt 4: Der Angreifer $\mathcal{A}'$ berechnet die Werte $A = A_1 g_3$ und $z = (A_1 g_3)^{X_{\mathcal{B}}}$ sowie $\hat{A} = \hat{A}_1 g_3$ und $\hat{z} = (\hat{A}_1 g_3)^{X_{\mathcal{B}}}$. Schließlich kann $\mathcal{A}'$ die Signaturen der beiden Münzen mit dem Signaturschlüssel $X_{\mathcal{B}}$ perfekt simulieren. Die Signatur für A, B lässt sich wie folgt perfekt simulieren:

- Wähle zufällige Werte $R, c \in_{\mathcal{R}} \mathbb{Z}_q$.
- Berechne $a = g^R, r = c' X_{\mathcal{B}} + R, b = A^R$
- Setze $\text{H}(A, B, z, a, b) := c$
- $\mathsf{Coin} = (A, B, (z, a, b, r))$

Die Signatur für $\hat{A}, \hat{B}$ lässt sich analog simulieren. Es ergeben sich folgende Sichten der Händler $\mathcal{H}$ und $\hat{\mathcal{H}}$:

$$\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P) = \{A, B, \mathsf{Coin}, V_3\}$$
$$\mathsf{view}^{\mathcal{K}}_{\hat{\mathcal{H}}}(\hat{P}) = \{\hat{A}, \hat{B}, \widehat{\mathsf{Coin}}, \hat{V}_3\}$$

Insgesamt hat $\mathcal{A}'$ somit aus den Verschlüsselungen $E^{(0)}, E^{(1)}$ eine perfekt simulierte Instanz des Matching-Withdrawal-Payment-Problem für Ω erstellt.

Schritt 5: Der Angreifer $\mathcal{A}'$ übergibt die Instanz $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W_b), \mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W_{\bar{b}}), \mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$, $\mathsf{view}^{\mathcal{K}}_{\hat{\mathcal{H}}}(\hat{P})$ des Matching-Withdrawal-Payment-Problem an den Angreifer $\mathcal{A}$.

Da $\mathcal{A}$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden kann, ob die in $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ ausgegebene Münze Coin in W_b oder $W_{\bar{b}}$ abgehoben wurde, kann $\mathcal{A}'$ mit der gleichen Wahrscheinlichkeit $\epsilon(k)$ entscheiden, welcher der beiden Klartexte m_0 und m_1 in $E^{(0)}$ verschlüsselt wurde. Dies ist aber ein Widerspruch zur polynomiellen Ununterscheidbarkeit der modifizierten ElGamal-Verschlüsselung.

□

Der Beweis zu Lemma 6.2.4 zeigt, dass sich die nichtinteraktiven Zero-Knowledge-Beweise V_1, V_2 und der Wert c in der Protokollansicht $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W)$ von $\mathcal{B}$ im Random-Oracle-Modell perfekt simulieren lassen. Analog lassen sich in der Ansicht $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ von $\mathcal{H}$ der nichtinteraktive Zero-Knowledge-Beweis V_3 und die Signatur der Münze Coin im Random-Oracle-Modell perfekt simulieren. Das heißt, ein Angreifer kann aus diesen Daten keinerlei Information gewinnen und letztlich nur die modifizierten ElGamal-Verschlüsselungen verwenden. Deshalb können wir diese Daten für die weitere Sicherheitsanalyse auch vernachlässigen. Im Weiteren sind die Protokollansichten $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W)$ und $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ durch folgende Daten gegeben:

$$\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W) = \{I, E_1, E_2, E_3\},$$
$$\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P) = \{A_1, C_1, C_2, C_3\}.$$

Entsprechend lassen sich auch die Ansichten $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W_F)$ und $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P_F)$ der Bank $\mathcal{B}$ bzw. des Händler $\mathcal{H}$ in *FOLC* perfekt simulieren (siehe [FTY98]), so dass diese im Weiteren wie folgt gegeben sind:

$$\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W_F) = \{I_F, E_{1,F}, E_{2,F}\},$$
$$\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P_F) = \{A_{1,F}, A_{2,F}\}.$$

Mit Hilfe dieser Ansichten können wir die Anonymität von Ω auf die Anonymität von *FOLC* zurückführen. Da *FOLC* gemäß Satz 6.1.1 ein sicheres faires elektronisches Geldsystem ist, gibt es keinen effizienten Angreifer, der die Anonymität von *FOLC* im Sinne von Definition 3.1.2 brechen kann. Insbesondere gibt es keinen effizienten Angreifer, der in *FOLC* das Decisional-Withdrawal-Payment-Problem mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ lösen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Wir zeigen zunächst, dass es auch in Ω keinen solchen effizienten Angreifer gibt.

Lemma 6.2.5. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ und eine nicht vernachlässigbare Funktion ν. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der das Decisional-Withdrawal-Payment-Problem in Ω mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ lösen kann.

Beweis. Angenommen, es gibt einen effizienten Angreifer $\mathcal{A}$, der das Decisional-Withdrawal-Payment-Problem in Ω mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ lösen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Dann kann $\mathcal{A}$ von einem effizienten Angreifer $\mathcal{A}'$ dazu verwendet werden, das Decisional-Withdrawal-Payment-Problem in *FOLC* zu lösen. Dies ist aber ein Widerspruch zur Sicherheit von *FOLC* (vgl. Satz 6.1.1). Folgende Daten sind in *FOLC* öffentlich bekannt:

- Primzahlen p, q mit $|p-1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Generatoren $g, g_1, \ldots, g_4$ der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.
- $h = g^{X_\mathcal{B}}, h_1 = g_1^{X_\mathcal{B}}, h_2 = g_2^{X_\mathcal{B}}, h_3 = g_3^{X_\mathcal{B}}$ mit $X_\mathcal{B} \in \mathbb{Z}_q$.
- $f_2 = g_2^{X_\mathcal{D}}, f_3 = g_3^{X_\mathcal{D}}$ mit $X_\mathcal{D} \in \mathbb{Z}_q$.
- Geeigneten Hashfunktion $\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$.

Für einen Kunden $\mathcal{K}$ mit Kontonummer $I_F = g_1^{u_1}$ ist eine Instanz des Decisional-Withdrawal-Payment-Problem in *FOLC* durch folgende Daten gegeben:

- $\mathsf{view}_\mathcal{B}^\mathcal{K}(W_F) = \{I_F, E_{1,F} = g_2^s f_3^m, E_{2,F} = g_3^m\}$ für $s, m \in \mathbb{Z}_q$.
- $\mathsf{view}_\mathcal{H}^\mathcal{K}(P_F) = \{A_{1,F} = I_F g_2^{s'}, A_{2,F} = f_2^{s'}\}$ für $s' \in \mathbb{Z}_q$.

Um das Decisional-Withdrawal-Payment-Problem in *FOLC* zu brechen, übersetzt $\mathcal{A}'$ die Instanz $\mathsf{view}_\mathcal{B}^\mathcal{K}(W_F)$, $\mathsf{view}_\mathcal{H}^\mathcal{K}(P_F)$ von *FOLC* in eine Instanz $\mathsf{view}_\mathcal{B}^\mathcal{K}(W), \mathsf{view}_\mathcal{H}^\mathcal{K}(P)$ von Ω und verwendet anschließend $\mathcal{A}$. Der Angreifer $\mathcal{A}'$ geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ simuliert anhand der in $\mathsf{view}_\mathcal{B}^\mathcal{K}(W_F)$ enthaltenen ElGamal-Verschlüsselung $(E_{1,F}, E_{2,F})$ die mod. ElGamal-Verschlüsselung (E_1, E_2, E_3) (vgl. Satz 2.3.2):

- Der Angreifer $\mathcal{A}'$ wählt zufällige Werte $r_1, r_2 \in_\mathcal{R} \mathbb{Z}_q$ und berechnet die Generatoren $g_5 = g_3^{r_1}$ und $g_6 = g_5^{r_2}$.
- Der Angreifer $\mathcal{A}'$ wählt einen zufälligen Wert $r_3 \in_\mathcal{R} \mathbb{Z}_q$ und berechnet den öffentlichen Schlüssel $\hat{z} = f_3 g_6^{r_3}$ der modifizierten ElGamal-Verschlüsselung.
- Der Angreifer $\mathcal{A}'$ berechnet die modifizierte ElGamal-Verschlüsselung:

$$
\begin{aligned}
E_2 &= E_{2,F}^{r_1} = (g_3^m)^{r_1} = g_5^m, \\
E_3 &= E_2^{r_2} = (g_5^m)^{r_2} = (g_5 r_2)^m = g_6^m, \\
E_1 &= E_{1,F} E_3^{r_3} = g_2^s f_3^m (g_6^m)^{r_3} = g_2^s f_3^m (g_6^{r_3})^m = g_2^s (f_3 g_6^{r_3})^m = g_2^s \hat{z}^m.
\end{aligned}
$$

Schritt 2: Der Angreifer $\mathcal{A}'$ simuliert anhand der in $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(\tilde{W})$ enthaltenen Kontonummer $\tilde{I}$ von $\mathcal{K}$ die modifizierte ElGamal-Verschlüsselung (C_1, C_2, C_3):

- Der Angreifer $\mathcal{A}'$ wählt zufällig Generatoren $g_7, g_8 \in_{\mathcal{R}} G_q$ und zufällige Werte $r_7, r_8 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $\bar{z} = g_7^{r_7} g_8^{r_8}$.
- Der Angreifer $\mathcal{A}'$ wählt $r_4 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die modifizierte ElGamal-Verschlüsselung $(C_1, C_2, C_3) = (I_F \bar{z}, g_7^{r_4}, g_8^{r_4})$ von I_F.
- Der Angreifer $\mathcal{A}'$ setzt $I := I_F$ und $A_1 := A_{1,F}$.

Insgesamt hat $\mathcal{A}'$ somit die beiden modifizierten ElGamal-Verschlüsselungen perfekt simuliert. Für die Instanz des Decisional-Withdrawal-Payment-Problem in Ω ergeben sich folgende Daten:

$$\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W) = \{I, E_1, E_2, E_3\}$$
$$\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P) = \{A_1, C_1, C_2, C_3\}$$

Schritt 3: Der Angreifer $\mathcal{A}'$ übergibt die Instanz $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ an $\mathcal{A}$.

Da der Angreifer $\mathcal{A}$ das Decisional-Withdrawal-Payment-Problem in Ω mit der Wahrscheinlichkeit $\epsilon(k)$ lösen kann, kann auch $\mathcal{A}'$ das Decisional-Withdrawal-Payment-Problem in *FOLC* mit der Wahrscheinlichkeit $\epsilon(k)$ lösen. Dies ist aber ein Widerspruch zur Sicherheit von *FOLC*. □

Lemma 6.2.6. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der die Anonymität von Ω im Sinne von Definition 3.1.2 brechen kann.

Beweis. Angenommen, es gibt einen effizienten Angreifer $\mathcal{A}$, der die Anonymität von Ω mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ brechen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Der Angreifer $\mathcal{A}$ kann von einem effizienten Angreifer $\mathcal{A}'$ dazu verwenden werden, die Anonymität von *FOLC* zu brechen. Eine Instanz von *FOLC* ist durch folgende Daten gegeben:

$$\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_F^{(b)}) = \{I_F^{(b)}, E_{1,F}^{(b)}, E_{2,F}^{(b)}\} \quad \text{und} \quad \mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_F^{(\bar{b})}) = \{I_F^{(\bar{b})}, E_{1,F}^{(\bar{b})}, E_{2,F}^{(\bar{b})}\},$$
$$\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P_F) = \{A_{1,F}, A_{2,F}\} \quad \text{und} \quad \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(\hat{P}_F) = \{\hat{A}_{1,F}, \hat{A}_{2,F}\}.$$

Um die Anonymität von *FOLC* zu brechen, übersetzt der Angreifer $\mathcal{A}'$ die Instanz $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_F^{(b)}), \mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_F^{(\bar{b})}), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P_F), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(\hat{P}_F)$ von *FOLC* in eine Instanz von Ω.
Der Angreifer $\mathcal{A}'$ geht ähnlich wie im Beweis zu Lemma 6.2.5 vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ simuliert die beiden modifizierten ElGamal-Verschlüsselungen $(E_1^{(b)}, E_2^{(b)}, E_3^{(b)})$ und $(E_1^{(\bar{b})}, E_2^{(\bar{b})}, E_3^{(\bar{b})})$ analog zu Lemma 6.2.5.

- Der Angreifer $\mathcal{A}'$ wählt zufällig Zahlen $r_1, r_2 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die Generatoren $g_5 = g_3^{r_1}$ und $g_6 = g_5^{r_2}$.
- Für die Berechnung des öffentlichen Schlüssels wählt $\mathcal{A}'$ zufällig eine Zahl $r_3 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $\hat{z} = f_3 g_6^{r_3}$.
- Anschließend berechnet $\mathcal{A}'$ die Werte $E_2^{(b)} = (E_{2,F}^{(b)})^{r_1}, E_3^{(b)} = (E_2^{(b)})^{r_2}, E_1^{(b)} = E_{1,F}^{(b)}(E_3^{(b)})^{r_3}$ sowie $E_2^{(\bar{b})} = (E_{2,F}^{(\bar{b})})^{r_1}, E_3^{(\bar{b})} = (E_2^{(\bar{b})})^{r_2}, E_1^{(\bar{b})} = E_{1,F}^{(\bar{b})}(E_3^{(\bar{b})})^{r_3}$.

Schritt 2: Der Angreifer $\mathcal{A}'$ simuliert die beiden modifizierten ElGamal-Verschlüsselungen (C_1, C_2, C_3) und $(\hat{C}_1, \hat{C}_2, \hat{C}_3)$ wie folgt (vgl. Satz 2.3.3):

- Der Angreifer $\mathcal{A}'$ setzt $g_7 := f_2$, wählt $r_4 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $g_8 = g_7^{r_4}$.
- Für die Berechnung des öffentlichen Schlüssels wählt $\mathcal{A}'$ zufällig $r_5 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $\bar{z} = g_2 g_8^{r_5}$.
- Der Angreifer $\mathcal{A}'$ setzt $C_2 := A_{2,F}$ und $\hat{C}_2 := \hat{A}_{2,F}$. Anschließend berechnet $\mathcal{A}'$ die Werte $C_3 = A_{2,F}^{r_4}, \hat{C}_3 = \hat{A}_{2,F}^{r_4}$ und $C_1 = A_{1,F} C_2^{r_5}, \hat{C}_1 = \hat{A}_{1,F} \hat{C}_2^{r_5}$.

Insgesamt hat $\mathcal{A}'$ die modifizierten ElGamal-Verschlüsselungen $(E_1^{(b)}, E_2^{(b)}, E_3^{(b)})$, $(E_1^{(\bar{b})}, E_2^{(\bar{b})}, E_3^{(\bar{b})})$ und (C_1, C_2, C_3), $(\hat{C}_1, \hat{C}_2, \hat{C}_3)$ perfekt simuliert. Für die Instanz von Ω ergeben sich folgende Daten:

$$\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_b) = \{I^{(b)}, E_1^{(b)}, E_2^{(b)}, E_3^{(b)}\} \quad \text{und} \quad \mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W_{\bar{b}}) = \{I^{(\bar{b})}, E_1^{(\bar{b})}, E_2^{(\bar{b})}.E_3^{(\bar{b})}\},$$
$$\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P) = \{A_1, C_1, C_2, C_3\} \quad \text{und} \quad \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(\hat{P}) = \{\hat{A}_1, \hat{C}_1, \hat{C}_2, \hat{C}_3\}.$$

Schritt 3: Der Angreifer $\mathcal{A}'$ übergibt die Instanz an $\mathcal{A}$.

Da der Angreifer $\mathcal{A}$ die Anonymität von Ω mit der Wahrscheinlichkeit $\epsilon(k)$ brechen kann, kann auch $\mathcal{A}'$ die Anonymität von *FOLC* mit der Wahrscheinlichkeit $\epsilon(k)$ brechen. Dies steht im Widerspruch zur Anonymität von *FOLC*. □

Identitätsnachweis: Im Falle eines Double-Spending muss die Bank die Identität des Betrügers mit einer überwältigenden Wahrscheinlichkeit aufdecken können. Dies folgt aus der Restriktivität des Abhebe-Protokolls (vgl. Annahme 6.1) und der Korrektheit des während des Bezahl-Protokolls durchgeführten Zero-Knowledge-Beweises (vgl. Satz 2.7.3).

Lemma 6.2.7. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 6.1 kann die Bank im Falle eines Double-Spending mit der Münze Coin die Identität I des betrügerischen Kunden $\mathcal{K}$ mit überwältigender Wahrscheinlichkeit aufdecken.

Beweis. Wurde mit der Münze Coin bei zwei verschiedenen Händlern $\mathcal{H}, \tilde{\mathcal{H}}$ bezahlt, so befinden sich in den Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D), \mathsf{view}_{\mathcal{B}}^{\tilde{\mathcal{H}}}(\tilde{D})$ der Bank $\mathcal{B}$ folgende Daten:

$$\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P) = \{\mathsf{Coin}, A_1, C_1, C_2, C_3, C_1', C_2', C_3', r_1, r_2, r_3\},$$
$$\mathsf{view}_{\tilde{\mathcal{H}}}^{\mathcal{K}}(\tilde{P}) = \{\mathsf{Coin}, A_1, \tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \tilde{C}_1', \tilde{C}_2', \tilde{C}_3', \tilde{r}_1, \tilde{r}_2, \tilde{r}_3\},$$

sowie die Kontonummern der Händler $\mathcal{H}, \tilde{\mathcal{H}}$ und die Zeitpunkte $\mathsf{ts}, \tilde{\mathsf{ts}}$ der beiden Bezahlvorgänge. Anhand von $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P), \mathsf{view}^{\mathcal{K}}_{\tilde{\mathcal{H}}}(\tilde{P})$ und $\mathsf{ts}, \tilde{\mathsf{ts}}$ kann $\mathcal{B}$ die Challenges d und $\tilde{d}$ berechnen. Da H_0 eine kollisionsresistente Hashfunktion ist, gilt $d = \tilde{d}$ nur mit vernachlässigbarer Wahrscheinlichkeit und $\mathcal{B}$ kann $g_1^{\frac{r_1 - \tilde{r}_1}{d - \tilde{d}}}$ berechnen. Da beide Bezahl-Protokolle erfolgreich durchgeführt wurden, gilt $B = g_1^{r_1} g_2^{r_2} A_1^{-d}$ und $B = g_1^{\tilde{r}_1} g_2^{\tilde{r}_2} A_1^{\tilde{d}}$ woraus $A_1 = g_1^{\frac{r_1 - \tilde{r}_1}{d - \tilde{d}}} g_2^{\frac{r_2 - \tilde{r}_2}{d - \tilde{d}}}$ folgt. Auf Grund der Korrektheit des im Bezahl-Protokoll durchgeführten Zero-Knowledge-Beweises muss $\mathcal{K}$ eine Darstellung von A_1 bzgl. (g_1, g_2) kennen und mit der Restriktivität des Abhebe-Protokolls folgt $g_1^{\frac{r_1 - \tilde{r}}{d - \tilde{d}}} = I$ mit überwältigender Wahrscheinlichkeit. □

Nichterweiterbarkeit: Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der bei Eingabe von $\mathbf{W}_\ell = \{\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)\}$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\ell + 1$ gültige Münzen ausgibt. Da sich in Ω im Vergleich zu *FOLC* am Abhebe-Protokoll Withdrawal lediglich Änderungen an den für das Münz-Tracing benötigten Zero-Knowledge-Beweisen V_1, V_2 ergeben und das „eigentliche blinde Signaturprotokoll“ unverändert bleibt, folgt die Nichterweiterbarkeit von Ω direkt aus der Nichterweiterbarkeit von *FOLC* (vgl. Satz 6.1.1).

Lemma 6.2.8. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter k. Dann gibt es keinen effizienten Angreifer $\mathcal{A}$, der die Nichterweiterbarkeit von Ω brechen kann.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der in Ω bei Eingabe von $\mathbf{W}_\ell$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ mindestens $\ell + 1$ gültige Münzen ausgibt. Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der $\mathcal{A}$ verwendet, um die Nichterweiterbarkeit von *FOLC* zu brechen. Folgende Daten sind in *FOLC* veröffentlicht:

- Primzahlen p, q mit $|p - 1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Generatoren $g, g_1, \ldots, g_4$ der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.
- $h = g^{X_{\mathcal{B}}}, h_1 = g_1^{X_{\mathcal{B}}}, h_2 = g_2^{X_{\mathcal{B}}}, h_3 = g_3^{X_{\mathcal{B}}}$ mit $X_{\mathcal{B}} \in \mathbb{Z}_q$.
- $f_2 = g_2^{X_{\mathcal{D}}}, f_3 = g_3^{X_{\mathcal{D}}}$ mit $X_{\mathcal{D}} \in \mathbb{Z}_q$.
- Geeignete Hashfunktionen $\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$.

Um in *FOLC* bei gegebenen Protokollansichten $\mathbf{W}_{\ell,F}$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\ell + 1$ gültige Münzen zu produzieren, übersetzt $\mathcal{A}'$ die Instanz $\mathbf{W}_{\ell,F}$ in eine Instanz $\mathbf{W}_\ell$ von Ω. In *FOLC* besteht eine Protokollansicht $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}}(W_F)$ eines Kunden $\mathcal{K}$ aus folgenden Werten:

$$\mathsf{view}^{\mathcal{B}}_{\mathcal{K}}(W_F) = \{I_F, m, s, t, I'_F, E_{1,F}, E_{2,F}, V_{1,F}, V_{2,F}, a', b', b'', x_1, x_2, u, v, r'\}$$

Der Angreifer $\mathcal{A}'$ geht wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}'$ übernimmt die Generatoren $g, g_1, \ldots, g_4$ und die öffentlichen Schlüssel $h = g^{X_\mathcal{B}}, h_1 = g_1^{X_\mathcal{B}}, h_2 = g_2^{X_\mathcal{B}}, h_3 = g_3^{X_\mathcal{B}}$. Anschließend wählt $\mathcal{A}'$ zufällig Werte $\delta_5, \ldots, \delta_8 \in \mathbb{Z}_q$ und berechnet die Generatoren $g_5 = g^{\delta_5}, \ldots, g_8 = g^{\delta_8}$.

Schritt 2: Der Angreifer $\mathcal{A}'$ wählt zufällig Werte $\hat{x}, \hat{y}, \bar{x}, \bar{y} \in_\mathcal{R} \mathbb{Z}_q$ und berechnet die öffentlichen Schlüssel $\hat{z} = g_5^{\hat{x}} g_6^{\hat{y}}$ und $\bar{z} = g_7^{\bar{x}} g_8^{\bar{y}}$.

Schritt 3: Für eine gegebene Protokollansicht $\mathsf{view}_\mathcal{K}^\mathcal{B}(W_F)$ von *FOLC* kann $\mathcal{A}'$ die Protokollansicht $\mathsf{view}_\mathcal{K}^\mathcal{B}(W)$ für Ω wie folgt erstellen:

- Der Angreifer $\mathcal{A}'$ setzt $I := I_F$ und $I' := I'_F$.
- Anhand der gegebenen Werte berechnet $\mathcal{A}'$ die modifizierte ElGamal-Verschlüsselung $(E_1, E_2, E_3) = (g_2^s \hat{z}^m, g_5^m, g_6^m)$ und erstellt die beiden nichtinteraktiven Zero-Knowledge-Beweise V_1, V_2.
- Da sich der Rest des Abhebe-Protokolls in Ω nicht von *FOLC* unterscheidet, übernimmt $\mathcal{A}'$ die restlichen Daten aus $\mathsf{view}_\mathcal{K}^\mathcal{B}(W_F)$. Die transformierte Protokollansicht $\mathsf{view}_\mathcal{K}^\mathcal{B}(W)$ lautet:

$$\mathsf{view}_\mathcal{K}^\mathcal{B}(W) = \{I, m, s, t, I', E_1, E_2, E_3, V_1, V_2, a', b', b'', x_1, x_2, u, v, r'\}$$

Schritt 4: Der Angreifer $\mathcal{A}'$ übergibt $\mathbf{W}_\ell$ an $\mathcal{A}$.

Mit der Wahrscheinlichkeit $\epsilon(k)$ gibt $\mathcal{A}$ anhand von $\mathbf{W}_\ell$ mindestens $\ell+1$ gültige Münzen aus. Diese $\ell + 1$ gültigen Münzen sind auch in *FOLC* gültige Münzen. Somit kann $\mathcal{A}'$ die Nichterweiterbarkeit von *FOLC* mit der Wahrscheinlichkeit $\epsilon(k)$ brechen. Dies steht aber im Widerspruch zur Sicherheit von *FOLC*. □

Unfälschbarkeit Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der bei Eingabe von $\mathbf{W}_\ell$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit eine gültige Münze ausgibt, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers aufdecken kann. Dies folgt im Wesentlichen aus der Restriktivität des Abhebe-Protokolls Withdrawal und der Unfälschbarkeit von *FOLC*.

Lemma 6.2.9. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 6.1 gibt es keinen effizienten Angreifer, der die Unfälschbarkeit von Ω brechen kann.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der in Ω bei Eingabe von $\mathbf{W}_\ell$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ eine solche gültige Münze ausgibt. Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der $\mathcal{A}$ verwendet, um die Unfälschbarkeit von *FOLC* zu brechen. Der Angreifer $\mathcal{A}'$ geht analog zum Beweis von Lemma 6.2.8 vor.
Mit der Wahrscheinlichkeit $\epsilon(k)$ gibt $\mathcal{A}$ anhand von $\mathbf{W}_\ell$ eine gültige Münze Coin aus, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank die Identität des

Betrüger mit einer in k überwältigenden Wahrscheinlichkeit aufdecken kann. Die Münze Coin ist auch in *FOLC* eine gültige Münze mit besagter Eigenschaft. Somit kann $\mathcal{A}'$ die Unfälschbarkeit von *FOLC* mit der Wahrscheinlichkeit $\epsilon(k)$ brechen. Dies ist aber ein Widerspruch zur Sicherheit von *FOLC*. □

Münz- und Kundentracing: Wir müssen zeigen, dass der Deanonymisierer anhand der Ansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W), \mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ Münz- bzw. Kundentracing durchführen kann. Dies folgt im Wesentlichen aus der Restriktivität von Withdrawal und den im Abhebe- bzw. Bezahl-Protokoll durchgeführten Zero-Knowledge-Beweisen V_1, V_2 und V_3.

Lemma 6.2.10. Gegeben sei das faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann kann der Deanonymisierer Münz- und Kundentracing unter Annahme 6.1 korrekt durchführen.

Beweis. Wir müssen zeigen, dass der Teil A der Münze $\mathsf{Coin} = (A, B, (z, a, b, r))$ während des Abhebe-Protokolls Withdrawal in der für beide Tracingmechanismen notwendigen Form berechnet wird.
Während des Abhebe-Protokolls Withdrawal zeigt der Kunde $\mathcal{K}$ durch den Zero-Knowledge-Beweis V_1, dass $E_1 = g_2^s \hat{z}^m, E_2 = g_5^m$ und $E_3 = g_6^m$ für s, m aus $\mathbb{Z}_q$ gemeinsam eine Verschlüsselung von g_2^s unter dem öffentlichen Schlüssel $\hat{z}$ von $\mathcal{D}$ bilden. Durch V_2 beweist $\mathcal{K}$ der Bank $\mathcal{B}$, dass folgende Gleichungen erfüllt sind:

$$\begin{aligned} g_3 &= (I')^v g_1^w g_4^t \text{ für } v, w, t \in \mathbb{Z}_q, \\ E_1 &= g_2^v \hat{z}^u \text{ für } u \in \mathbb{Z}_q, \\ I &= (I')^v g_3^r g_4^z \text{ für } r, z \in \mathbb{Z}_q. \end{aligned}$$

Mit dem Beweis V_1 hat $\mathcal{K}$ bereits gezeigt, dass $E_1 = g_2^s \hat{z}^m$ gilt. Da $\mathcal{K}$ nicht effizient eine andere Darstellung von E_1 berechnen kann, folgt $v = s, u = m$ und somit auch:

$$\begin{aligned} g_3 &= (I')^s g_1^w g_4^t, \\ I &= (I')^s g_3^r g_4^z. \end{aligned}$$

Aus diesen beiden Gleichungen folgt:

$$\begin{aligned} I' &= g_3^{s^{-1}} g_1^{-ws^{-1}} g_4^{-ts^{-1}}, \\ I' &= I^{s^{-1}} g_3^{-rs^{-1}} g_4^{-zs^{-1}}. \end{aligned}$$

Während der Kontoeröffnung hat $\mathcal{K}$ der Bank $\mathcal{B}$ gezeigt, dass $\mathcal{K}$ eine Darstellung von $I = g_1^{u_1}$ bzgl. g_1 kennt, woraus $-w = u_1$ folgt. Somit ist

$$\begin{aligned} I' &= g_1^{u_1 s^{-1}} g_3^{s^{-1}} g_4^{-ts^{-1}}, \\ I' &= g_1^{u_1 s^{-1}} g_3^{-rs^{-1}} g_4^{-zs^{-1}}. \end{aligned}$$

Gemäß Annahme 6.1 ist das Abhebe-Protkoll ein restriktiv blindes Signaturprotokoll, d.h. A muss von der Form $A = (I' g_2)^u g^v$ sein, für $u, v \in \mathbb{Z}_q$. Woraus sich

$$A = g_1^{u_1 s^{-1} u} g_3^{s^{-1} u} g_4^{-ts^{-1} u + v} g_2^u$$

ergibt. Beim Bezahlen beweist $\mathcal{K}$ dem Händler $\mathcal{H}$ für $k, x, y \in \mathbb{Z}_q$, dass $A_1 = g_1^x g_2^y$ und $C_1 = \bar{z}^k g_1^x, C_2 = g_7^k, C_3 = g_8^k$ eine Verschlüsselung unter dem öffentlichen Schlüssel $\hat{z}$ von $\mathcal{D}$ ist. Zusätzlich prüft $\mathcal{H}$, ob $A = A_1 g_3$ gilt. Das heißt, der Angreifer A ist von der Form $A = g_1^x g_2^y g_3$. Wir vergleichen die Exponenten der beiden Darstellungen von A:

$$A = g_1^{u_1 s^{-1} u} g_3^{s^{-1} u} g_4^{-t s^{-1} u + v} g_2^u,$$
$$A = g_1^x g_2^y g_3.$$

Es folgt $s^{-1}u \equiv 1 \bmod q$ und damit $u \equiv s \bmod q$ sowie $-ts^{-1}u + v \equiv 0 \bmod q$. Einsetzen in $A = g_1^{u_1 s^{-1} u} g_3^{s^{-1} u} g_4^{-t s^{-1} u + v} g_2^v$ liefert $A = g_1^{u_1} g_2^s g_3$ und $A_1 = g_1^{u_1} g_2^s$. Damit ist $C_1 = \bar{z} g_1^{u_1}, C_2 = g_7^k, C_2 = g_8^k$ die für das Kundentracing geforderte Verschlüsselung der Identität $I = g_1^{u_1}$ unter dem öffentlichen Schlüssel $\hat{z}$ und $E_1 = g_2^s \hat{z}^m, E_2 = g_5^m, E_3 = g_6^m$ die für Münztracing benötigte Verschlüsselung unter dem öffentlichen Schlüssel $\bar{z}$. □

Insgesamt folgt aus den Lemmata 6.2.6 bis 6.2.10 folgender Satz:

Satz 6.2.11. Im Random-Oracle-Modell ist das aktualisierbare faire elektronische Geldsystem Ω ein $(0, 1)$-langfristig sicheres faires elektronisches Geldsystem und somit gemäß Definition 3.3.2 ein sicheres faires elektronisches Geldsystem.

6.2.7.2 Sicherheitsanalyse für $t \geq 1$ und $N > 1$

Wir zeigen, dass das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ für $N > 1$ und $1 \leq t < N$ ein (t, N)-langfristig sichere faires elektronisches Geldsystem ist. Im Weiteren sei Ω' die nicht aktualisierbare Variante von Ω ($t = 0$ und $N = 1$).

Sichere Schlüsselaktualisierung: In Ω wird der private Schlüssel der Bank auf die gleiche Weise aktualisiert wie in der langfristig sicheren Variante der blinden Schnorr-Signatur (vgl. Abschnitt 5.2.3). Aus der sicheren Schlüsselaktualisierung der langfristig sicheren blinden Schnorr-Signatur folgt sofort die sichere Schlüsselaktualisierung von Ω.

Lemma 6.2.12. Das aktualisierbare faire elektronische Geldsystem Ω verfügt über eine sichere Schlüsselaktualisierung.

Beweis. Analog zu Satz 5.2.3. □

Münz- und Kundentracing: Für $t = 0$ und $N = 1$ wurde dies in Lemma 6.2.10 gezeigt. Da in den Beweisen V_1, V_2 und V_3 die im i-ten Intervall gültigen öffentlichen Schlüssel $\hat{z}_i$ bzw. $\bar{z}_i$ verwendet werden, folgt dies für $N > 1$ und $1 \leq t < N$ analog.

Lemma 6.2.13. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann kann der Deanonymisierer Münz- und Kundentracing unter Annahme 6.1 in jedem Intervall $0 \leq i \leq N - 1$ korrekt durchführen.

Beweis. Analog zu Lemma 6.2.10. Für jedes Intervall $0 \leq i \leq N$ ersetze im Beweis zu Lemma 6.2.10 die beiden öffentlichen Schlüssel $\hat{z}, \bar{z}$ durch $\hat{z}_i$ bzw. $\bar{z}_i$. □

Identitätsnachweis: Für $t = 0$ und $N = 1$ wurde dies in Lemma 6.2.7 gezeigt. Da der Identitätsnachweis aus der Restriktivität des Abhebe-Protokolls (vgl. Annahme 6.1) und der Korrektheit des während des Bezahl-Protokolls durchgeführten Zero-Knowledge-Beweises (vgl. Satz 2.7.3) folgt, kann die Bank die Identität von Double-Spendern auch in Ω aufdecken.

Lemma 6.2.14. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 6.1 kann die Bank im Falle eines Double-Spending mit der Münze Coin die Identität I des betrügerischen Kunden $\mathcal{K}$ mit überwältigender Wahrscheinlichkeit aufdecken.

Beweis. Analog zu Lemma 6.2.7. □

Anonymität: Für $t = 0$ und $N = 1$ wurde dies in Lemma 6.2.6 gezeigt. Ein effizienter Angreifer der in Ω bei Eingabe zweier im gleichen Intervall entstandenen Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0), \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$, der in W_0 und W_1 abgehobenen Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ und den Protokollansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b), \mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ der entsprechenden Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört, kann dies auch in Ω', im Widerspruch zu Lemma 6.2.6. Wir zeigen nun, dass die Anonymität von Ω auch im Sinne von Definition 5.4.2 gewährleistet ist.

Lemma 6.2.15. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der die Anonymität von Ω im Sinne von Definition 5.4.2 brechen kann.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{D}}^*, \mathsf{sk}_{\mathcal{D}}^0}$ stellt und mit einer Wahrscheinlichkeit $\epsilon(k) \geq 1/2 + \nu(k)$ die Anonymität von Ω im Sinne von Definition 5.4.2 brechen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Das heißt, der Angreifer $\mathcal{A}$ kann nach t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{D}}^*, \mathsf{sk}_{\mathcal{D}}^0}$ bei Eingabe zweier im gleichen Intervall i entstandenen Protokollansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0), \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$, der in W_0 und W_1 abgehobenen Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ und den Protokollansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b), \mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ der entsprechenden Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ mit der Wahrscheinlichkeit $\epsilon(k)$ entscheiden, ob Coin_b zu W_0 oder W_1 gehört.
Durch die t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{D}}^*, \mathsf{sk}_{\mathcal{D}}^0}$ erhält $\mathcal{A}$ lediglich t Funktionswerte der Polynome $\bar{f}_x, \bar{f}_y, \hat{f}_x, \hat{f}_y$. Da die Koeffizienten $\bar{x}_0^*, \ldots, \bar{x}_t^*, \bar{y}_0^*, \ldots, \bar{y}_t^*, \hat{x}_0^*, \ldots, \hat{x}_t^*$ und $\hat{y}_0^*, \ldots, \hat{y}_t^*$ der Polynome $\bar{f}_x, \bar{f}_y, \hat{f}_x, \hat{f}_y$ zufällig gewählt wurden, sind $\bar{f}_x, \bar{f}_y, \hat{f}_x, \hat{f}_y$ zufällig gewählte Polynome mit Grad t und die privaten Schlüssel $\bar{x}_i = \bar{f}_x(i)$, $\bar{y}_i = \bar{f}_y(i)$, $\hat{x}_i = \hat{f}_x(i)$, $\hat{y}_i = \hat{f}_y(i)$ des i-ten Intervalls sind für $\mathcal{A}$ gleichverteilt in $\mathbb{Z}_q$ und stochastisch unabhängig von den t privaten Schlüsseln, die $\mathcal{A}$ von $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{D}}^*, \mathsf{sk}_{\mathcal{D}}^0}$ erhalten hat. Das heißt, der Angreifer $\mathcal{A}$ erhält durch die t Anfragen an $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{D}}^*, \mathsf{sk}_{\mathcal{D}}^0}$ keine Information über die privaten Schlüssel $\bar{x}_i, \bar{y}_i, \hat{x}_i, \hat{y}_i$.

Der Angreifer $\mathcal{A}$ kann somit lediglich anhand seiner Eingabe $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$, Coin_b, $\mathsf{Coin}_{\bar{b}}$, $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ entscheiden, ob Coin_b in W_0 oder W_1 abgehoben wurde. Dies steht aber im Widerspruch zu Lemma 6.2.6. □

Nichterweiterbarkeit: Für $t = 0$ und $N = 1$ wurde dies in Lemma 6.2.8 gezeigt. Ist $1 \leq t < N$, darf der Angreifer die Bank in t der insgesamt N Intervalle kompromittieren. Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der nach t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*,\mathsf{sk}_{\mathcal{B}}^{(0)}}$ bei Eingabe von ℓ, in einem von ihm gewählten Intervall i, entstandenen Protokollansichten $\mathbf{W}_\ell = \{\mathsf{view}_{\mathcal{K}_1}^{\mathcal{B}}(W_1), \ldots, \mathsf{view}_{\mathcal{K}_\ell}^{\mathcal{B}}(W_\ell)\}$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\ell+1$, im i-ten Intervall entstandene, gültige Münzen ausgibt. Da $\mathcal{A}$ durch die t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*,\mathsf{sk}_{\mathcal{B}}^{(0)}}$ keine Information über den im i-ten Intervall gültigen privaten Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(i)} = f(i)$ erhält, folgt dies direkt aus Lemma 6.2.8.

Lemma 6.2.16. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es keinen effizienten Angreifer, der Spiel 5.3.1 für Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt.

Beweis. Analog zu Lemma 6.2.15. □

Unfälschbarkeit: Für $t = 0$ und $N = 1$ wurde dies in Lemma 6.2.8 gezeigt. Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der nach t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*,\mathsf{sk}_{\mathcal{B}}^{(0)}}$ bei Eingabe von ℓ, in einem von ihm gewählten Intervall i, entstandenen Protokollansichten $\mathbf{W}_\ell$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit eine, im i-Intervall entstandene, gültige Münze ausgibt, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank mit einer in k überwältigenden Wahrscheinlichkeit die Identität des Betrügers aufdecken kann. Da $\mathcal{A}$ durch die t Anfragen an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*,\mathsf{sk}_{\mathcal{B}}^{(0)}}$ keine Information über den im i-ten Intervall gültigen privaten Schlüssel $\mathsf{sk}_{\mathcal{B}}^{(i)} = f(i)$ erhält, folgt dies direkt aus Lemma 6.2.9.

Lemma 6.2.17. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 6.1 gibt es keinen effizienten Angreifer, der Spiel 5.3.3 in Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt.

Beweis. Analog zu Lemma 6.2.15. □

Insgesamt folgt aus den Lemmata 6.2.12 bis 6.2.17 folgender Satz:

Satz 6.2.18. Im Random-Oracle-Modell ist das aktualisierbare faire elektronische Geldsystem Ω ein (t, N)-langfristig sicheres faires elektronisches Geldsystem mit sicherer Schlüsselaktualisierung.

6.2.7.3 Sicherheitsanalyse für $t = N - 1$ und $N > 1$

Analog zu Abschnitt 5.2.3 kann auch *FOLC*, auf dem das aktualisierbare faire elektronische Geldsystem Ω basiert, mit dem in Abschnitt 5.3.2 dargestellten generischen Ansatz in ein stark $(N-1, N)$-langfristig sicheres elektronisches Geldsystem umgewandelt werden. Allerdings wird durch diese generische Transformation zum einen die Kompromittierung des Deanonymisierers nicht berücksichtigt und zum anderen erhöht sich dadurch der zur Verifikation einer elektronischen Münze benötigte Rechenaufwand.
Im Gegensatz dazu ist Ω, das genauso effizient wie *FOLC* arbeitet und die Kompromittierung des Deanonymisierers berücksichtigt, für $t = N - 1$ automatisch ein starkes $(N-1, N)$-langfristig sicheres faires elektronisches Geldsystem. Das heißt, für $t = N-1$ ist Ω auch gegen die in den Spielen 5.3.2 und 5.3.4 beschriebenen adaptiven Angriffe (starke Nichterweiterbarkeit und starke Unfälschbarkeit) sicher.

Starke Nichterweiterbarkeit: Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der in Ω Spiel 5.3.2 mit einer im Sicherheitsparameter $k \in \mathbb{N}$ nicht vernachlässigbaren Wahrscheinlichkeit gewinnen kann. Dazu können wir im Wesentlichen so vorgehen, wie im Beweis zu Satz 5.3.1. Entscheidend ist, dass auf alle Anfragen geantwortet werden kann, die der Angreifer an das Schlüssel-Orakel stellt. Um dies sicherzustellen verwenden wir die in Abschnitt 5.2.3 vorgestellte Methode und führen die starke Nichterweiterbarkeit von Ω auf die Nichterweiterbarkeit der nicht aktualisierbaren Variante von Ω ($t = 0$ und $N = 1$) zurück, die wir im Weiteren mit Ω' bezeichnen.

Lemma 6.2.19. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es keinen effizienten Angreifer, der Spiel 5.3.2 in Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt.

Beweis. Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.3.2 in Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ gewinnt.
Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht die Sicherheit von Ω' im Sinne von Definition 3.1.2 zu brechen. Sei ℓ die in k polynomiell beschränkte Anzahl der Ansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}_1}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}_\ell}(W_\ell)$ die $\mathcal{A}$ insgesamt während seines Angriffs erhält. Der Angreifer $\mathcal{A}'$ erhält Ansichten durchgeführter Abhebe-Protokolle mit der Bank $\mathcal{B}'$, die gemeinsam mit dem Deanonymisierer $\mathcal{D}'$ folgende Daten veröffentlicht hat:

- Primzahlen p, q mit $|p-1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Generatoren $g, g_1, \ldots, g_8$ der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.
- $h = g^x, h_1 = g_1^x, h_2 = g_2^x, h_3 = g_3^x$ für ein $x \in \mathbb{Z}_q$.
- Geeignete kollisionsresistente Hashfunktionen $\mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$
- $\hat{z} = g_5^{\hat{x}} g_6^{\hat{y}}, \bar{z} = g_5^{\bar{x}} g_6^{\bar{y}}$ für $\hat{x}, \hat{y}, \bar{x}, \bar{y} \in \mathbb{Z}_q$.

Zunächst transformiert $\mathcal{A}'$ die Daten in die öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}$:

- Der Angreifer $\mathcal{A}'$ übernimmt $p, q, g, g_1, \ldots, g_8, \mathrm{H}, \mathrm{H}_0, \mathrm{H}_1, \ldots$.
- Der Angreifer $\mathcal{A}'$ wählt zufällig einen Index $i \in \{0, \ldots, N-1\}$ und setzt $h_i^* = h$, $h_{1,i}^* = h_1$, $h_{2,i}^* = h_2$, $h_{3,i}^* = h_3$, $\hat{z}_i^* = g_5^{\hat{x}} g_6^{\hat{y}}$ und $\bar{z}_i^* = g_5^{\bar{x}} g_6^{\bar{y}}$. Somit setzt $\mathcal{A}'$ implizit $\mathsf{sk}_{\mathcal{B}}^{(i)} = x$ und $\widehat{\mathsf{sk}}_{\mathcal{D}}^{(i)} = (\hat{x}, \hat{y}), \overline{\mathsf{sk}}_{\mathcal{D}}^{(i)} = (\bar{x}, \bar{y})$.
- Für $j = 0, \ldots, i-1, i+1, \ldots, N-1$ wählt $\mathcal{A}'$ zufällig Zahlen $x_j \in_{\mathcal{R}} \mathbb{Z}_q$ und setzt $f(j) = x_j$. Anschließend berechnet $\mathcal{A}$ wie in Abschnitt 5.2.3 beschrieben die Werte $h_j^*, h_{1j}^*, h_{2j}^*, h_{3j}^*$ für $j = 0, \ldots, N-1$.
- Für $j = 0, \ldots, i-1, i+1, \ldots, N-1$ wählt $\mathcal{A}'$ zufällig Zahlen $\hat{x}_j, \hat{y}_j, \bar{x}_j, \bar{y}_j \in_{\mathcal{R}} \mathbb{Z}_q$, setzt $\hat{f}_x(j) = \hat{x}_j, \hat{f}_y(j) = \hat{y}_j, \bar{f}_x(j) = \bar{x}_j, \bar{f}_y(j) = \bar{y}_j$ und berechnet wie in Abschnitt 5.2.3 beschrieben die Werte $\hat{z}_j^*, \bar{z}_j^*$ für $j = 0, \ldots, N-1$.
- Der Angreifer setzt $\mathsf{pk}_{\mathcal{B}} = (p, q, g, g_1, \ldots, g_8, \ldots)$ und $\mathsf{pk}_{\mathcal{D}} = (\hat{z}_0^*, \ldots, \bar{z}_{N-1}^*)$.

Um die Nichterweiterbarkeit von Ω' zu brechen geht $\mathcal{A}'$ wie folgt vor:

Schritt 1: Der Angreifer $\mathcal{A}$ erhält $\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}$ als Eingabe. Die Rolle der Bank $\mathcal{B}$ und die des Schlüssel-Orakels $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*, \mathsf{sk}_{\mathcal{B}}^{(0)}}$ werden von $\mathcal{A}'$ übernommen.

Schritt 2: Der Angreifer $\mathcal{A}'$ verwendet die Bank $\mathcal{B}'$ um die Protokollansicht für $\mathcal{A}$ im i-ten Intervall zu erstellen. Falls $\mathcal{A}$ den Index i als Anfrage an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*, \mathsf{sk}_{\mathcal{B}}^{(0)}}$ stellt, bricht $\mathcal{A}'$ seinen Versuch ab.

Schritt 3: Da $\mathcal{A}'$ die privaten Schlüssel der Bank $\mathcal{B}$ in allen anderen Intervalle kennt, kann $\mathcal{A}'$ in allen anderen Intervallen die Protokollansichten für $\mathcal{A}$ selbst erstellen. Entsprechend kann $\mathcal{A}'$ auf die Anfragen von $\mathcal{A}$ an das Schlüssel-Orakel $\mathsf{Exp}_{\mathsf{sk}_{\mathcal{B}}^*, \mathsf{sk}_{\mathcal{B}}^{(0)}}$ antworten.

Gibt $\mathcal{A}$ in einem Intervall i^* nach ℓ_{i^*} in diesem Intervall angeforderten Protokollansichten mindestens $\ell_{i^*}+1$ gültige Münzen $\mathsf{Coin}_1, \ldots, \mathsf{Coin}_{\ell_{i^*}+1}$ aus, so gibt $\mathcal{A}'$ ebenfalls die $\ell_{i^*}+1$ gültigen Münzen $\mathsf{Coin}_1, \ldots, \mathsf{Coin}_{\ell_{i^*}+1}$ aus.
Mit einer Wahrscheinlichkeit von $\frac{1}{N}$ wird der Versuch von $\mathcal{A}'$ nicht abgebrochen und es ist $i = i^*$. Für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ gilt somit

$$\epsilon'(k) \geq \frac{\epsilon(k)}{N}.$$

Dies ist aber ein Widerspruch zu Satz 6.2.11. □

Starke Unfälschbarkeit: Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der in Ω Spiel 5.3.4 mit einer im Sicherheitsparameter $k \in \mathbb{N}$ nicht vernachlässigbaren Wahrscheinlichkeit gewinnen kann. Wir können die starke Unfälschbarkeit von Ω wieder auf die Unfälschbarkeit von Ω' zurückführen und gehen analog zur starken Nichterweiterbarkeit vor.

Lemma 6.2.20. Gegeben sei das aktualisierbare faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 6.1 gibt es keinen effizienten Angreifer, der Spiel 5.3.4 in Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit gewinnt.

Beweis. Im Weiteren sei Ω' die nicht aktualisierbare Variante von Ω ($t = 0$ und $N = 1$). Angenommen es gibt einen effizienten Angreifer $\mathcal{A}$, der Spiel 5.3.4 in Ω mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ gewinnt. Wir konstruieren einen effizienten Angreifer $\mathcal{A}'$, der unter Verwendung von $\mathcal{A}$ versucht die Sicherheit von Ω' im Sinne von Definition 3.1.2 zu brechen. Der Angreifer $\mathcal{A}'$ geht analog zum Beweis von Lemma 6.2.19 vor.
Gibt $\mathcal{A}$ in einem Intervall i^* nach ℓ_{i^*} in diesem Intervall angeforderten Protokollansichten eine gültige Münzen Coin aus, die zweimal erfolgreich ausgegeben werden kann, ohne dass die Bank die Identität des Betrügers mit einer in k überwältigenden Wahrscheinlichkeit aufdecken kann, so gibt $\mathcal{A}'$ ebenfalls die Münze Coin aus.
Mit einer Wahrscheinlichkeit von $\frac{1}{N}$ wird der Versuch von $\mathcal{A}'$ nicht abgebrochen und es ist $i = i^*$. Für die Erfolgswahrscheinlichkeit $\epsilon'(k)$ von $\mathcal{A}'$ gilt somit

$$\epsilon'(k) \geq \frac{\epsilon(k)}{N}.$$

Dies ist aber ein Widerspruch zu Satz 6.2.11. □

Aus Satz 6.2.18 und den beiden Lemmata 6.2.19 und 6.2.20 ergibt sich folgender Satz:

Satz 6.2.21. Im Random-Oracle-Modell ist das aktualisierbare faire elektronische Geldsystem Ω ein starkes $(N-1, N)$-langfristig sicheres faires elektronisches Geldsystem mit sicherer Schlüsselaktualisierung.

6.3 Eine langfristig sichere faire Geldbörse

Grundsätzlich können Kunden in einem elektronischen Geldsystem abgehobene Münzen vervielfältigen und mehrfach ausgeben. Zwar kann die Bank die Identität der Double-Spender im Nachhinein aufdecken, die Mehrfachausgabe der Münzen aber mit rein kryptografischen Maßnahmen nicht verhindern. Aus diesem Grund wird in diesem Abschnitt das in Abschnitt 6.2 entwickelte System zu einer *langfristig sicheren fairen elektronischen Geldbörse* erweitert. Das heißt, auf Kundenseite wird ein Observer in das System integriert, mit dem die Mehrfachausgabe der Münzen verhindert wird (vgl. Abschnitt 3.2). Gleichzeitig bietet der Einsatz zusätzlicher manipulationssicherer Hardware den Kunden außerdem Schutz vor Angriffen durch Schadsoftware wie beispielsweise Trojaner, Viren oder Würmer.
Die Integration des Observers kann mit der bereits in [Bra94] vorgestellten und auch in [NS03] angewandten Methode realisiert werden. Jeder Kunde erhält während der Kontoeröffnung von der Bank einen Observer. Im Vergleich zu Abschnitt 6.2 wird die Kontonummer I zwischen Kunde $\mathcal{K}$ und Observer $\mathcal{O}$ aufgeteilt, wodurch $\mathcal{K}$ den diskreten Logarithmus $\log_{g_1} I$ nicht mehr kennt. Da $\mathcal{K}$ daher keine Darstellungen seiner

abgehobenen Münzen kennt, kann $\mathcal{K}$ diese nur mit Hilfe von $\mathcal{O}$ ausgeben (vgl. Lemma 6.2.2) und $\mathcal{O}$ muss entsprechend an den Protokollen beteiligt werden. Neben den Änderungen an den Protokollen Withdrawal und Payment, die sich durch die in [Bra94] beschriebene Technik ergeben, muss $\mathcal{O}$ zusätzlich in den für das Münztracing benötigten nichtinteraktiven Beweis V_2 integriert werden, der Teil von Withdrawal ist. Aus Sicht der Bank und der Händler ändert sich das System nicht, denn die Protokolle müssen nur auf Kundenseite angepasst werden.

6.3.1 Initialisierung des Systems

Da sich an der Initialisierung des Systems gegenüber Abschnitt 6.2 keine Änderungen ergeben, sei an dieser Stelle auf Abschnitt 6.2.1 verwiesen. Eine gültige Münze ist weiterhin durch Definition 6.2.1 gegeben.

6.3.2 Kontoeröffnung

Möchte ein neuer Kunde $\mathcal{K}$ im i-ten Intervall ein Konto bei der Bank $\mathcal{B}$ eröffnen, so muss sich $\mathcal{K}$ genauso wie in Abschnitt 6.2 zunächst gegenüber der Bank $\mathcal{B}$ ausweisen. Anschließend wird folgendes Protokoll durchgeführt, in dem $\mathcal{K}$ von $\mathcal{B}$ einen Observer $\mathcal{O}$ ausgehändigt bekommt.

Schritt 1: Der Kunde $\mathcal{K}$ wählt zufällig eine Zahl $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $A_{\mathcal{K}} = g_1^{u_1}$. Die Zahl u_1 speichert $\mathcal{K}$ als privaten Schlüssel $\mathsf{sk}_{\mathcal{K}}$ und sendet $A_{\mathcal{K}}$ an $\mathcal{B}$. Im Gegensatz zu Abschnitt 6.2 wird $A_{\mathcal{K}}$ von $\mathcal{B}$ nicht als Kontonummer von $\mathcal{K}$ verwendet.

Schritt 2: Zur Berechnung der Kontonummer I werden von $\mathcal{B}$ auf einem Observer $\mathcal{O}$ eine zufällig gewählte Zahl $o_1 \in_{\mathcal{R}} \mathbb{Z}_q$ sowie $A_{\mathcal{O}} = g_1^{o_1}$ gespeichert. Die Zahl o_1 wird im nichtauslesbaren Teil des Speichers von $\mathcal{O}$ hinterlegt, so dass o_1 dem Kunden $\mathcal{K}$ nicht bekannt ist. Anhand von $A_{\mathcal{O}}$ und $A_{\mathcal{K}}$ berechnet $\mathcal{B}$ die Kontonummer $I = A_{\mathcal{K}} A_{\mathcal{O}} = g_1^{u_1+o_1}$ und überprüft, ob $Ig_2 \neq 1$ gilt. Bevor $\mathcal{B}$ den Observer $\mathcal{O}$ an $\mathcal{K}$ überreicht, berechnet $\mathcal{B}$ mit ihrem aktuellen privaten Schlüssel x_i den Wert $z_i' = I^{x_i}$ und sendet z_i' sowie $A_{\mathcal{O}}$ an $\mathcal{K}$.

Schritt 3: Abschließend beweist $\mathcal{K}$ der Bank $\mathcal{B}$, dass $\mathcal{K}$ und $\mathcal{O}$ zusammen eine Darstellung von I bzgl. g_1 kennen. Dazu führt $\mathcal{K}$ in Kooperation mit $\mathcal{O}$ das in Abbildung 6.11 dargestellte Protokoll mit der Bank $\mathcal{B}$ aus.

Bemerkung 6.3.1. Alternativ kann z_i' auch von $\mathcal{K}$ zusammen mit $\mathcal{O}$ berechnet werden. Dazu sendet $\mathcal{K}$ den Wert $h_{1,i} = \prod_{k=0}(h_{1,k})^{i^k}$ an $\mathcal{O}$. Anhand von $h_{1,i}$ berechnet $\mathcal{O}$ den Wert $h_{\mathcal{O}}^{(i)} = h_{1,i}^{o_1}$ und sendet $h_{\mathcal{O}}^{(i)}$ an $\mathcal{K}$. Der Kunde $\mathcal{K}$ berechnet dann $z_i' = h_{1,i}^{u_1} h_{\mathcal{O}}^{(i)} = I^{x_i}$.

Analog zum System aus Abschnitt 6.2 dient die Kontonummer I der Bank $\mathcal{B}$ später zur Identifikation von Double-Spendern. Im Unterschied zu Abschnitt 6.2 kennt $\mathcal{K}$ keine Darstellung seiner Kontonummer I bzgl. g_1 und kann, unter der Voraussetzung, dass $\mathcal{O}$ nicht kompromittiert werden kann, seine Münzen nur zusammen mit $\mathcal{O}$ ausgeben (vgl. auch Abschnitt 6.2.4).

$\mathcal{O}(o_1)$		$\mathcal{K}(u_1)$		$\mathcal{B}(\mathsf{pk}_\mathcal{B})$
$o_1' \in_\mathcal{R} \mathbb{Z}_q$				
$\tilde{A}_\mathcal{O} = g_1^{o_1'}$	$\xrightarrow{\tilde{A}_\mathcal{O}}$	$u_1', u_2' \in_\mathcal{R} \mathbb{Z}_q$		
		$\tilde{I} = g_1^{u_1'} \tilde{A}_\mathcal{O} A_\mathcal{O}^{u_2'}$	$\xrightarrow{\tilde{I}}$	
				$c \in_\mathcal{R} \mathbb{Z}_q$
			$\xleftarrow{c}$	
	$\xleftarrow{c'}$	$c' = c + u_2'$		
$r' = c' o_1 + o_1'$	$\xrightarrow{r'}$	$g_1^{r'} \stackrel{?}{=} A_\mathcal{O}^{c'} \tilde{A}_\mathcal{O}$		
		$r = r' + c u_1 + u_1'$	$\xrightarrow{r}$	
				$g_1^r \stackrel{?}{=} I^c \tilde{I}$

Abbildung 6.11: Die Kontoeröffnung in der Variante mit Observer

Annahme 6.2. Der von der Bank $\mathcal{B}$ an den Kunden $\mathcal{K}$ ausgegebene Observer $\mathcal{O}$ ist manipulationssicher. Insbesondere lässt sich der Speicherinhalt des Observers $\mathcal{O}$ von $\mathcal{K}$ nicht auslesen.

6.3.3 Eine Münze abheben

Damit Double-Spending von vornherein verhindert werden kann, wird der Observer mit der in [Bra94] beschriebenen Methode in das neue Abhebe-Protokoll Withdrawal integriert, das auf dem Protokoll aus 6.2.3 aufbaut. Durch die Integration des Observers ergeben sich im Vergleich zu Abschnitt 6.2.3 einige Änderungen, die im Folgenden detailliert beschrieben werden.
Genauso wie in Abschnitt 6.2 überprüfen $\mathcal{K}$ und $\mathcal{B}$ vor der Durchführung des Abhebe-Protokolls Withdrawal, ob sie die für das i-te Intervall benötigten temporären Schlüssel bereits berechnet haben. Ist dies nicht der Fall, berechnet $\mathcal{K}$ analog zu Abschnitt 6.2 die temporären Schlüssel $\hat{z}_i, \bar{z}_i$ und $h_i, h_{2,i}, h_{3,i}$. Gegebenenfalls aktualisiert $\mathcal{B}$ ihren privaten Schlüssel und berechnet $\hat{z}_i$.
Da $\mathcal{K}$ keine Darstellung von I bzgl. g_1 kennt, kann der im Abhebe-Protokoll Withdrawal zu berechnende Wert z nicht wie in Abschnitt 6.2.3 berechnet werden. Deshalb berechnet $\mathcal{B}$ außerdem einmal pro Intervall $z' = I^{x_i}$ und sendet z' an $\mathcal{K}$. Alternativ kann z' auch von $\mathcal{K}$ und $\mathcal{O}$ berechnet werden (vgl. Abschnitt 6.3.2). Um eine Münze Coin bei $\mathcal{B}$ abzuheben, authentifiziert sich $\mathcal{K}$ mit Hilfe von $\mathcal{O}$ zunächst gegenüber $\mathcal{B}$ (vgl. Abbildung 6.11). Anschließend wird das in Abbildung 6.12 beschriebene Protokoll $\mathsf{Withdrawal}(\mathcal{O}(o_1), \mathcal{K}(u_1, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D}), \mathcal{B}(x_i, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D}))$ durchgeführt:

Schritt 1: Der Kunde $\mathcal{K}$ wählt zufällig drei Zahlen $m, s, t \in_\mathcal{R} \mathbb{Z}_q$ und berechnet damit $I' = (I g_3)^{s^{-1}} g_4^t$ sowie $E_1 = g_2^s \hat{z}_i^m, E_2 = g_5^m, E_3 = g_6^m$. Anschließend sendet $\mathcal{K}$ die

Verschlüsselung $\langle i, (E_1, E_2, E_3)\rangle$ und I' an $\mathcal{B}$ und führt wie in Abschnitt 6.2.3 die beiden nichtinteraktiven Zero-Knowledge-Beweise

$$V_1 = \text{NIZK}\left[(\alpha_1, \alpha_2) : E_1 = g_2^{\alpha_1}\hat{z}_i^{\alpha_2} \wedge E_2 = g_5^{\alpha_1} \wedge E_3 = g_6^{\alpha_2}\right]$$

und

$$V_2 = \text{NIZK}\left[(\beta_1, \ldots, \beta_6) : g_3 = (I')^{\beta_1} g_1^{\beta_2} g_4^{\beta_3} \wedge E_1 = g_2^{\beta_1}\hat{z}_i^{\beta_4} \wedge I = (I')^{\beta_1} g_3^{\beta_5} g_4^{\beta_6}\right]$$

aus. Den Beweis V_1 kann $\mathcal{K}$ alleine erstellen. Für den Beweis V_2 muss $\mathcal{K}$ eine Darstellung von I bzgl. g_1 kennen. Da $\mathcal{K}$ diese nicht kennt, kann $\mathcal{K}$ den Beweis V_2 nur mit Hilfe von $\mathcal{O}$ erstellen. Wie V_2 von $\mathcal{K}$ und $\mathcal{O}$ gemeinsam erstellt werden kann, ist in Abschnitt 2.9.1 erläutert (vgl. Abbildung 2.6).

Schritt 2: Nachdem sich $\mathcal{B}$ von der Korrektheit der beiden Beweise V_1 und V_2 überzeugt hat, berechnet $\mathcal{B}$ die Werte $a' = g^w$, $b' = (I'g_2)^w$ und $b'' = g_4^w$, wobei w gemäß einer Gleichverteilung auf $\mathbb{Z}_q$ gewählt wird. Die Bank $\mathcal{B}$ sendet a', b', b'' an $\mathcal{K}$.

Schritt 3: Der Observer $\mathcal{O}$ wählt zufällig eine Zahl $o_2 \in_{\mathcal{R}} \mathbb{Z}_q$, speichert o_2 zusammen mit der Zahl $B_{\mathcal{O}} = g_1^{o_2}$ in seinem geschützten Speicher und sendet $B_{\mathcal{O}}$ an $\mathcal{K}$. Der Wert $B_{\mathcal{O}}$ ist Teil des Zero-Knowledge-Beweises den $\mathcal{O}$ und $\mathcal{K}$ während des Abhebens und Bezahlens von Coin durchführen. Falls $\mathcal{K}$ mit der Münze Coin bezahlt, wird $\mathcal{O}$ während des Bezahl-Protokolls Payment die gespeicherte Zahl o_2 löschen und auf diese Weise eine erneute Ausgabe der Münze Coin verhindern.

Schritt 4: Analog zu Abschnitt 6.2.3 berechnet $\mathcal{K}$

$$A = (I'g_2g_4^{-t})^s = Ig_2^s g_3 \quad \text{sowie} \quad z = z'h_{2,i}^s h_{3,i} = A^{x_i}.$$

Anschließend blendet $\mathcal{K}$ die Zahlen a', b', b'' mit Hilfe zufällig gewählter Zahlen $u, v \in_{\mathcal{R}} \mathbb{Z}_q$ zu $a = (a')^u g^v$ und $b = (b'b''^{-t})A^v$. In die Berechnung des Wertes B, der zur *Double-Spending-Detection* benötigt wird, fließen neben dem aus zwei zufällig gewählten Zahlen $x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$ berechneten Wert $g_1^{x_1}g_2^{x_2}$ nun auch der von $\mathcal{O}$ berechnete Wert $B_{\mathcal{O}}$, sowie der durch s und eine weitere zufällig gewählte Zahl $e \in_{\mathcal{R}} \mathbb{Z}_q$ geblendete Wert $A_{\mathcal{O}}^{es}$ ein. Der Kunde $\mathcal{K}$ berechnet das Commitment B und die Challenge c wie folgt:

$$B = g_1^{x_1}g_2^{x_2}A_{\mathcal{O}}^{es}B_{\mathcal{O}} \text{ und } \mathrm{H}(A, B, z, a, b).$$

Anhand von u blendet $\mathcal{K}$ die Challenge c und sendet $c' = cu^{-1} \bmod q$ an $\mathcal{B}$. Da in die Berechnung von A die Kontonummer I von $\mathcal{K}$ mit einfließt, von der $\mathcal{K}$ keine Darstellung bzgl. g_1 kennt, wird $\mathcal{K}$ die Münze Coin nur mit Hilfe von $\mathcal{O}$ ausgeben können.

Schritt 5: Die Bank $\mathcal{B}$ berechnet $r' = c'x_i + w \bmod q$ und schickt die Response r an $\mathcal{K}$.

Schritt 6: Zum Schluss des Protokolls berechnet $\mathcal{K}$ den Wert $r = r'u + v \bmod q$ und überprüft, ob er im Besitz einer gültigen Münze ist, indem er die beiden Gleichungen

$$g^r = h_i^c a \text{ und } A^r = z^c b$$

verifiziert.

Im Vergleich zu Abschnitt 6.2 ergeben sich am Abhebe-Protokoll der Variante mit Observer keine sicherheitsrelevanten Änderungen, wodurch die folgende Annahme gerechtfertigt ist.

Annahme 6.3. Das in Abbildung 6.12 dargestellte Abhebe-Protokoll ist ein restriktiv blindes Signaturprotokoll.

Lemma 6.3.1. Folgen der Observer $\mathcal{O}$, der Kunde $\mathcal{K}$ und die Bank $\mathcal{B}$ dem Protokoll Withdrawal, so ist $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ eine gültige Münze. Der Kunde $\mathcal{K}$ kennt zusammen mit $\mathcal{O}$ eine Darstellung von Coin. Unter Annahme 6.3 kennt $\mathcal{K}$ selbst keine Darstellung der Münze Coin.

Beweis. Sicherlich ist $\mathsf{Coin} = (A, B, \langle i, (z, a, b, r)\rangle)$ eine gültige Münze (vgl. Lemma 6.2.1) und $\mathcal{K}$ und $\mathcal{O}$ kennen gemeinsam eine Darstellung der Münze Coin. Unter Annahme 6.3 muss die Kontonummer I in A integriert sein. Das heißt, es gilt $A = Ig_2^s g_3$. Da $\mathcal{K}$ keine Darstellung von I bzgl. g_1 kennt, kann er auch keine Darstellung von A bzgl. (g_1, g_2, g_3) kennen. Genauso wenig kennt $\mathcal{K}$ eine Darstellung von B bzgl. (g_1, g_2). □

6.3.4 Mit einer Münze bezahlen

Das Bezahl-Protokoll entspricht im Wesentlichen dem Protokoll aus Abschnitt 6.2.4. Da $\mathcal{K}$ die Münze Coin nur dann ausgeben kann, wenn $\mathcal{K}$ eine Darstellung von Coin kennt (vgl. Lemma 6.2.2), wird $\mathcal{K}$ das Bezahlen mit Coin nur mit Hilfe von $\mathcal{O}$ gelingen. Analog zum Abhebe-Protokoll Withdrawal kann der Observer $\mathcal{O}$, mit der in [Bra94] vorgestellten Technik, in das Bezahl-Protokoll Payment integriert werden.
Bevor im j-ten Intervall das eigentliche Bezahl-Protokoll Payment durchgeführt wird, überprüfen $\mathcal{K}$ und $\mathcal{H}$, ob ihnen der temporäre Schlüssel $\bar{z}_j$ bereits vorliegt. Gegebenenfalls berechnen sie $\bar{z}_j$ anhand von $\mathsf{pk}_\mathcal{D}$. Um mit Coin zu bezahlen, wird anschließend über einen anonymen Kanal zwischen den Parteien $\mathcal{O}$, $\mathcal{K}$ und $\mathcal{H}$ das im Folgenden beschriebene Bezahl-Protokoll $\mathsf{Payment}(\mathcal{O}(o_1, o_2), \mathcal{K}(u_1, \mathsf{Coin}, \mathsf{pk}_\mathcal{D}), \mathcal{H}(\mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D}))$ ausgeführt.

Schritt 1: Genauso wie in Abschnitt 6.2.4 berechnet $\mathcal{K}$ zunächst

$$A_1 = Ig_2^s = A_3 g_1^{-1}.$$

Anschließend berechnet $\mathcal{K}$ eine modifizierte ElGamal-Verschlüsselung seiner Identität I im Intervall j unter $\mathsf{pk}_\mathcal{D}$, indem $\mathcal{K}$ zufällig eine Zahl $k \in_\mathcal{R} \mathbb{Z}_q$ wählt und

$$C_1 = \bar{z}_j^k I, C_2 = g_7^k \text{ und } C_3 = g_8^k$$

$\mathcal{O}(o_1)$	$\mathcal{K}(u_1, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$	$\mathcal{B}(x_i, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$
Einmal pro Intervall:		
	$\hat{z}_i = \prod_{j=0}^{t} (\hat{z}_j^*)^{i^j}$	$\mathsf{BKeyUpdate}(\mathcal{B}(x_{i-1}, (i-1), i), \mathcal{T}_\mathcal{B}(\mathsf{sk}_\mathcal{B}^*))$
	$h_i = \prod_{k=0}^{t} (h_i^*)^{i^k}$	$\hat{z}_i = \prod_{j=0}^{t} (\hat{z}_j^*)^{i^j}$
	$h_{2,i} = \prod_{k=0}^{t} (h_{2,i}^*)^{i^k}$	$z' = I^{x_i}$
	$\xleftarrow{z'}$	
	$h_{3,i} = \prod_{k=0}^{t} (h_{3,i}^*)^{i^k}$	
	$m, s, t, \in_R \mathbb{Z}_q$	
	$I' = I^{s^{-1}} g_3^{s^{-1}} g_4^t = g_1^{(u_1+o_1)s^{-1}} g_3^{s^{-1}} g_4^t$	
	$E_1 = g_2^s \hat{z}_i^m, E_2 = g_5^m, E_3 = g_6^m$	
	$V_1 = \mathrm{NIZK}[(\gamma_1, \gamma_2) : E_1 = g_2^{\gamma_1} \hat{z}_i^{\gamma_2} \wedge E_2 = g_3^{\gamma_2} \wedge E_3 = g_4^{\gamma_2}]$	
	$V_2 = \mathrm{NIZK}[(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6) : g_3 = (I')^{\delta_1} g_1^{\delta_2} g_4^{\delta_3} \wedge$	
	$E_1 = g_2^{\delta_1} \hat{z}_i^{\delta_4} \wedge I = (I')^{\delta_1} g_3^{\delta_5} g_4^{\delta_6}]$	
	$\xrightarrow{I', E_1, E_2, E_3, V_1, V_2}$	
		$E_2 \overset{?}{\neq} 1$
		Verifiziere V_1, V_2
		$w \in_R \mathbb{Z}_q$
		$a' = g^w$
	$\xleftarrow{a', b', b''}$	$b' = (I' g_2)^w, b'' = g_4^w$
$o_2 \in_\mathcal{R} \mathbb{Z}_q$		
$B_\mathcal{O} = g_1^{o_2}$ $\xrightarrow{B_\mathcal{O}}$		
	$A = (I' g_2 g_4^{-t})^s = g_1^{u_1+o_1} g_2^s g_3$	
	$z = z' h_{2,i}^s h_{3,i} = A^{x_i}$	
	$x_1, x_2, u, v, e \in_\mathcal{R} \mathbb{Z}_q$	
	$B = g_1^{x_1} g_2^{x_2} A_\mathcal{O}^{se} B_\mathcal{O}$	
	$a = (a')^u g^v$	
	$b = (b' b''^{-t})^{su} A^v$	
	$c = \mathrm{H}(A, B, z, a, b)$	
	$c' = c/u \bmod q$	
	$\xrightarrow{c'}$	
	$\xleftarrow{r'}$	$r' = c' x_i + w \bmod q$
	$r = r'u + v \bmod q$	

Abbildung 6.12: Das Abhebe-Protokoll in der Variante mit Observer

berechnet. Um $\mathcal{H}$ davon zu überzeugen, dass $\langle j, (C_1, C_2, C_3)\rangle$ wirklich eine Verschlüsselung unter $\mathsf{pk}_{\mathcal{D}}$ ist, wählt $\mathcal{K}$ zufällig eine Zahl $l \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die Commitments $C_1' = \bar{z}_j^l g_1^{x_1} A_{\mathcal{O}}^{se} B_{\mathcal{O}}, C_2' = g_7^l, C_3' = g_8^l$. Anschließend sendet $\mathcal{K}$ die Münze Coin, die modifizierte ElGamal-Verschlüsselung $\langle j, (C_1, C_2, C_3)\rangle$, die Commitments C_1', C_2', C_3' und A_1 an $\mathcal{H}$.

Schritt 2: Der Händler $\mathcal{H}$ überprüft, ob $A_1 \neq 1$ und $A_1 g_3 = A$ gilt, verifiziert die Signatur $\langle i, (z, a, b, r)\rangle$ der Münze Coin und berechnet die Challenge

$$d = \mathrm{H}(A_1, B, C_1, C_2, C_3, C_1', C_2', C_3', I_{\mathcal{H}}, \mathsf{ts})$$

die $\mathcal{H}$ an $\mathcal{K}$ sendet.

Schritt 3: Da $\mathcal{K}$ ohne die Hilfe von $\mathcal{O}$ nicht korrekt auf die Challenge d antworten kann, blendet $\mathcal{K}$ die Challenge d zu $d' = d + es$ und sendet d' an $\mathcal{O}$.

Schritt 4: Der Observer $\mathcal{O}$ überprüft, ob sich o_2 noch in seinem Speicher befindet. Ist dies der Fall, so weiß $\mathcal{O}$, dass die Münze Coin noch nicht ausgegeben wurde und $\mathcal{O}$ antwortet $\mathcal{K}$ mit der Response $r_1' = d'o_1 + o_2 \bmod q$. Anschließend löscht $\mathcal{O}$ die Zahl o_2 aus seinem Speicher. Wurde o_2 bereits von $\mathcal{O}$ gelöscht, so wird $\mathcal{O}$ die Response r_1' nicht berechnen und die Kommunikation zu $\mathcal{K}$ abbrechen, da $\mathcal{K}$ die Münze bereits einmal ausgegeben hat. Ohne die Response r_1' kann $\mathcal{K}$ jedoch nicht auf die Challenge d von $\mathcal{H}$ antworten und somit wird $\mathcal{K}$ an einer erneuten Ausgabe von Coin durch $\mathcal{O}$ gehindert.

Schritt 5: Der Kunde $\mathcal{K}$ überprüft ob $g_1^{r_1'} = A_{\mathcal{O}}^{d'} B_{\mathcal{O}}$ gilt und berechnet dann seine Responses $r_1 = r_1' + du_1 + x_1 \bmod q, r_2 = ds + x_2 \bmod q, r_3 = dk + l \bmod q$, die $\mathcal{K}$ dann an $\mathcal{H}$ sendet.

Schritt 6: Abschließend verifiziert $\mathcal{H}$ genauso wie in Abschnitt 6.2.4 die folgenden Gleichungen:

$$g_1^{r_1} g_2^{r_2} = A_1^d B,\ g_1^{r_1} \bar{z}_j^{r_3} = C_1^d C_1',\ g_7^{r_3} = C_2^d C_2' \text{ und } g_8^{r_3} = C_3^d C_3'.$$

Analog zu Lemma 6.2.2 zeigt das folgende Lemma, dass ein ehrlicher Kunde Münzen nur in Kooperation mit seinem Observer erfolgreich ausgeben kann.

Lemma 6.3.2. Wird das Bezahl-Protokoll gemäß Abbildung 6.13 durchgeführt, so wird der Händler $\mathcal{H}$ die Münze Coin des Kunden $\mathcal{K}$ akzeptieren. Unter Annahme 6.2 kann $\mathcal{K}$ die Münze Coin nur in Kooperation mit seinem Observer $\mathcal{O}$ ausgeben. Versucht $\mathcal{K}$ die Münze Coin erneut auszugeben, so wird dies durch $\mathcal{O}$ verhindert.

Beweis. Verhalten sich $\mathcal{O}$, $\mathcal{K}$ und $\mathcal{H}$ dem Protokoll entsprechend, so akzeptiert $\mathcal{H}$ die Münze Coin. Da sich $\mathcal{O}$ dem Protokoll entsprechend verhält, gelingt die Verifikation der ersten beiden Gleichungen:

$$\begin{aligned} g_1^{r_1} g_2^{r_2} &= g_1^{r_1' + du_1 + x_1} g_2^{ds + x_2} = g_1^{d'o_1 + o_2} (g_1^{u_1})^d g_1^{x_1} (g_2^s)^d g_2^{x_2} = (g_1^{u_1} g_2^s)^d g_1^{(d+se)o_1 + o_2} g_1^{x_1} g_2^{x_2} \\ &= (g_1^{u_1} g_2^s)^d (g_1^{o_1})^d (g_1^{o_1})^{se} g_1^{o_2} g_1^{x_1} g_2^{x_2} = A_1^d B \end{aligned}$$

$\mathcal{O}(o_1, o_2)$	$\mathcal{K}(u_1, \mathsf{Coin}, \mathsf{pk}_\mathcal{D})$	$\mathcal{H}(\mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{D})$
Einmal pro Intervall:		
	$\bar{z}_j = \prod_{k=0}^{t} (\bar{z}_k^*)^{j^k}$	$\bar{z}_j = \prod_{k=0}^{t} (\bar{z}_k^*)^{j^k}$
	$A_1 = I g_2^s$	
	$k, l \in_\mathcal{R} \mathbb{Z}_q$	
	$C_1 = \bar{z}_j^k I, C_2 = g_7^k, C_3 = g_8^k$	
	$C_1' = \bar{z}_j^l g_1^{x_1} A_\mathcal{O}^{se} B_\mathcal{O}, C_2' = g_7^l, C_3' = g_8^l$	
	$\xrightarrow{\mathsf{Coin}, A_1, C_1, C_2, C_3, C_1', C_2', C_3'}$	$A_1 \stackrel{?}{\neq} 1$
		$A_1 g_3 \stackrel{?}{=} A$
		$\mathsf{Verify}(\mathsf{pk}_\mathcal{B}, \mathsf{Coin})$
		$d = \mathrm{H}_0(A_1, B, C_1, C_2, C_3, C_1', C_2', C_3', I_\mathcal{H}, \mathsf{ts})$
o_2 im Speicher?		$\xleftarrow{d}$
$r_1' = d' o_1 + o_2 \bmod q$	$d' = d + es$	
	$\xrightarrow{r_1'}$ $g_1^{r_1'} \stackrel{?}{=} A_\mathcal{O}^{d'} B_\mathcal{O}$	
	$r_1 = r_1' + d u_1 + x_1 \bmod q$	
	$r_2 = ds + x_2 \bmod q$	
	$r_3 = dk + l \bmod q$	$\xrightarrow{r_1, r_2, r_3}$ $g_1^{r_1} g_2^{r_2} \stackrel{?}{=} A_1^d B$
		$g_1^{r_1} \bar{z}_j^{r_3} \stackrel{?}{=} C_1^d C_1'$
		$g_7^{r_3} \stackrel{?}{=} C_2^d C_2'$
		$g_8^{r_3} \stackrel{?}{=} C_3^d C_3'$

Abbildung 6.13: Das Bezahl-Protokoll in der Variante mit Observer

und entsprechend

$$\begin{aligned} g_1^{r_1} \bar{z}_j^{r_3} &= g_1^{r_1' + d u_1 + x_1} \bar{z}_j^{dk+l} = (g_1^{o_1})^d (g_1^{o_1})^{se} g_1^{o_2} (g_1^{u_1})^d g_1^{x_1} (\bar{z}_j^k)^d \bar{z}_j^l \\ &= (I \bar{z}_j^k)^d A_\mathcal{O}^{se} B_\mathcal{O} g_1^{x_1} \bar{z}_j^l = C_3^d C_3' \end{aligned}$$

Verhält sich $\mathcal{K}$ gemäß dem Protokoll so gilt außerdem

$$\begin{aligned} g_7^{r_3} &= g_7^{dk+l} = C_1^d C_1', \\ g_8^{r_3} &= g_8^{dk+l} = C_1^d C_1'. \end{aligned}$$

Aus Sicht von $\mathcal{H}$ hat sich das Protokoll im Vergleich zu Abschnitt 6.2.4 nicht geändert. Somit folgt aus Lemma 6.2.2, dass $\mathcal{K}$ mit Coin nur dann bezahlen kann, wenn $\mathcal{K}$ eine Darstellung von Coin kennt. Aus Lemma 6.3.1 folgt, dass $\mathcal{K}$ unter Annahme 6.2 selbst keine Darstellung von Coin kennt. Daher kann $\mathcal{K}$ die Münze Coin nur mit Hilfe von $\mathcal{O}$ ausgeben.

Da $\mathcal{K}$ die Münze Coin nur mit Hilfe von $\mathcal{O}$ ausgeben kann und $\mathcal{O}$ bei einem erneuten Versuch von $\mathcal{K}$, mit Coin zu bezahlen, nicht auf die Challenge d' von $\mathcal{K}$ antworten wird, verhindert $\mathcal{O}$ dadurch die erneute Ausgabe von Coin. □

Bemerkung 6.3.2. Natürlich kann auch hier das Bezahl-Protokoll $\mathsf{Payment}$, genauso wie in Abschnitt 6.2.4 nichtinteraktiv durchgeführt werden, wenn $\mathcal{K}$ die Kontonummer $I_{\mathcal{H}}$ des Händlers $\mathcal{H}$ kennt. Im Unterschied zu Abschnitt 6.2.4 muss $\mathcal{K}$ das Protokoll in Kooperation mit $\mathcal{O}$ durchführen. Die nichtinteraktive Variante ergibt sich durch

$$\text{NIZK}\left[(\lambda_1,\lambda_2,\lambda_3) : A_1 = g_1^{\lambda_1} g_2^{\lambda_2} \wedge C_1 = g_1^{\lambda_1} \bar{z}_j^{\lambda_3} \wedge C_2 = g_7^{\lambda_3} \wedge C_3 = g_8^{\lambda_3}\right](\mathsf{Coin}, I_{\mathcal{H}}, \mathsf{ts}).$$

6.3.5 Eine Münze einlösen

Aus Sicht des Händlers $\mathcal{H}$ ergeben sich im Vergleich zu Abschnitt 6.2 keine Änderungen. Somit kann $\mathcal{H}$ eine Münze Coin bei der Bank $\mathcal{B}$ genauso wie in Abschnitt 6.2.5 einlösen. Zwischen beiden Parteien wird das in Abschnitt 6.2.5 vorgestellte Einlöse-Protokoll $\mathsf{Deposit}(\mathcal{H}(\mathsf{view}_{\mathcal{H}}(P)), \mathcal{B}(\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{D}}))$ durchgeführt (vgl. Abbildung 6.8).

6.3.6 Kunden- und Münztracing

Auch Kunden- und Münztracing können auf Grund der Tatsache, dass sich die Sichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ und $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ der Bank $\mathcal{B}$ bzw. des Händlers $\mathcal{H}$ gegenüber dem in Abschnitt 6.2 entwickelten System nicht geändert haben, auf die gleiche Weise durchgeführt werden. Entsprechend werden die in Abschnitt 6.2.5 vorgestellten Protokolle $\mathsf{KTrace}(\mathcal{B}(\mathsf{view}_{\mathcal{B}}(D)), \mathcal{D}(\overline{\mathsf{sk}}_{\mathcal{D}}^{j}, \mathsf{pk}_{\mathcal{D}}))$ und $\mathsf{MTrace}(\mathcal{B}(\mathsf{view}_{\mathcal{B}}(W)), \mathcal{D}(\widehat{sk}_{DM}^{j}, \mathsf{pk}_{\mathcal{D}})$ zwischen der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$ durchgeführt (vgl. Abbildungen 6.9 und 6.10).

6.3.7 Sicherheitsanalyse

Durch die Integration von Observern sollen die Kunden an der Mehrfachausgabe ihrer elektronischen Münzen gehindert werden. Solange die Observer nicht kompromittiert werden, folgt dies aus Lemma 6.3.2. Prinzipiell ist es aber nicht auszuschließen, dass ein Kunde $\mathcal{K}$ mit dem notwendigen technischen Wissen seinen Observer $\mathcal{O}$ kompromittiert. Falls eine Kunde $\mathcal{K}$ mit Kontonummer I in der hier vorgestellten aktualisierbaren fairen elektronischen Geldbörse Π seinen Observer $\mathcal{O}$ kompromittiert, kennt $\mathcal{K}$ den auf $\mathcal{O}$ gespeicherten Anteil $o_1 \in \mathbb{Z}_q$ der Darstellung von I bzgl g_1. Mit der Kenntnis von $\log_{g_1} I$ ist $\mathcal{K}$ nicht mehr auf die Kooperation von $\mathcal{O}$ angewiesen und kann seine Münzen mehrfach ausgeben. Da Π auf dem in Abschnitt 6.2 entwickelten System Ω aufbaut, stimmen die Protokolle $\mathsf{Withdrawal}$ und $\mathsf{Payment}$ im Falle einer Kompromittierung des Observer mit den Protokollen aus Abschnitt 6.2 überein. Das heißt, die Sicherheit von Π beruht im Wesentlichen auf der Sicherheit von Ω.

Sichere Schlüsselaktualisierung: Die privaten Schlüssel der Bank werden gemäß Abschnitt 6.2.1 aktualisiert. Die sichere Schlüsselaktualisierung folgt mit Satz 5.2.3.

Identitätsnachweis: Da der Kunde $\mathcal{K}$ im Falle einer Kompromittierung seines Observers $\mathcal{O}$ eine Darstellung von I bzgl. g_1 kennt, folgt dies direkt aus Lemma 6.2.7.

Nichterweiterbarkeit und Unfälschbarkeit: Falls der Kunde $\mathcal{K}$ seinen Observer $\mathcal{O}$ kompromittiert, stimmen die Protokollansichten $\mathsf{view}^{\mathcal{B}}_{\mathcal{K}}(W_1), \ldots, \mathsf{view}^{\mathcal{B}}_{\mathcal{K}}(W_\ell)$ mit den entsprechenden Ansichten aus Abschnitt 6.2 überein. Somit folgen Nichterweiterbarkeit und Unfälschbarkeit direkt aus den Lemmata 6.2.16 und 6.2.17.

Münz- und Kundentracing: Da sich die Protokollansichten $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W)$ und $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ im Vergleich zu den entsprechenden Ansichten aus Abschnitt 6.2 nicht ändern, können Münz- und Kundentracing erfolgreich durchgeführt werden. Dies folgt direkt aus Lemma 6.2.10 (mit $u_1 + o_1$ anstelle von u_1).

Anonymität: Damit die Anonymität der Kunden auch beim Einsatz von Observern gewährleistet bleibt, müssen Abhebe- und Bezahl-Protokoll so konzipiert sein, dass sie sowohl eingehende als auch ausgehende Information zwischen Bank und Observer verhindern (vgl. Abschnitt 3.2). Das heißt, die Bank kann auch anhand der auf den Observern gespeicherten Daten die Anonymität der Kunden nicht aufheben.

Lemma 6.3.3. Folgt der Kunde $\mathcal{K}$ den Protokollen Withdrawal and Payment, dann sind die Protokollansichten der Unterprotokolle die zwischen $\mathcal{K}$ und seinem Observer $\mathcal{O}$ ausgeführt werden, von den Sichten der Unterprotokolle die zwischen $\mathcal{K}$ und der Bank $\mathcal{B}$ bzw. $\mathcal{K}$ und dem Händler $\mathcal{H}$ ausgeführt werden stochastisch unabhängig.

Beweis. Wir zeigen, dass es bei vorliegenden Protokollansichten $\mathsf{view}^{\mathcal{K}}_{\mathcal{O}}$, $\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W)$ und $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ für den Kunde $\mathcal{K}$ bei der Durchführung der Protokolle Withdrawal und Payment genau eine mögliche Wahl von $m, s, t, x_1, x_2, u, v, e,\ y, y_1, \ldots, y_6, k, l$ gibt, so dass $\mathcal{O}$, $\mathcal{B}$ und $\mathcal{H}$ genau diese Ansichten erhalten. Die Protokollansichten des Observers $\mathcal{O}$, der Bank $\mathcal{B}$ und des Händlers $\mathcal{H}$ sind durch folgende Daten gegeben:

$$\begin{aligned}
\mathsf{view}^{\mathcal{K}}_{\mathcal{O}} &= \{A_{\mathcal{O}}, B_{\mathcal{O}}, C_{\mathcal{O}}, \hat{c}', \hat{r}'_1, d', r'_1 : g_1^{\hat{r}'} = A_{\mathcal{O}} C_{\mathcal{O}}^{\hat{c}'}, g_1^{r'_1} = A_{\mathcal{O}} B_{\mathcal{O}}^{d'}\},\\
\mathsf{view}^{\mathcal{K}}_{\mathcal{B}}(W) &= \{A_{\mathcal{K}}, I, I', E_1, E_2, E_3, \hat{c}, \hat{r}, \hat{r}_1, \ldots, \hat{r}_5, a', b', b'', c', r'\},\\
\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P) &= \{A, B, (z, a, b, r), A_1, C_1, C_2, C_3, d, r_1, r_2, r_3\}.
\end{aligned}$$

Die von $\mathcal{K}$ während des Abhebe-Protokolls Withdrawal gewählten Zahlen m, s, t sind wie folgt eindeutig bestimmt: Die Zahl m ist durch $m = \log_{g_5} E_2$ eindeutig bestimmt. Während des Bezahl-Protokolls Payment berechnet $\mathcal{K}$ den Wert $A_1 = I g_2^s$, wodurch

$$s = \log_{g_2}(A_1 \cdot I^{-1}) = \log_{g_2} A_1 - \log_{g_2} I$$

eindeutig bestimmt wird. Dadurch wird auch

$$t = \log_{g_4}(I' I^{-s^{-1}} g_3^{-s^{-1}}) = \log_{g_4} I' - s^{-1} \log_{g_4} I - s^{-1} \log_{g_4} g_3$$

eindeutig festgelegt. Die während der Durchführung von V_2 zufällig gewählten Zahlen y, $y_1, \ldots, y_6$ sind somit wie folgt eindeutig bestimmt:

$$y = \hat{r} - \hat{c}s \bmod q, \quad y_1 = \hat{r}_1 - \hat{r}'_1 + cu_1 \bmod q \quad \text{und} \quad y_2 = \hat{r}_2 + \hat{c}ts \bmod q$$

sowie

$$y_3 = \hat{r}_3 + c \bmod q, \quad y_4 = \hat{r}_4 + \hat{c}ts \bmod q, \quad y_5 = \hat{r}_5 - \hat{c}m \bmod q \quad \text{und} \quad y_6 = \hat{c}' + \hat{c}.$$

Weiter sind die während des Abhebe-Protokolls Withdrawal gewählten Zahl u und v durch

$$u = c \cdot c'^{-1} \bmod q \quad \text{bzw.} \quad v = r - r'u \bmod q$$

eindeutig bestimmt. Da u_1 und s bereits eindeutig bestimmt sind, werden anhand der in $\mathsf{view}^{\mathcal{K}}_{\mathcal{H}}(P)$ enthaltenen Zahlen r_1 und r_2 und der in $\mathsf{view}^{\mathcal{K}}_{\mathcal{O}}$ enthaltenen Zahl r'_1, d' die von $\mathcal{K}$ gewählten Zahlen x_1, x_2 und e durch

$$x_1 = r_1 - r'_1 - du_1 \bmod q \quad x_2 = r_2 - ds \bmod q \quad \text{und} \quad e = (d' - d)s^{-1} \bmod q$$

eindeutig bestimmt. Schließlich sind auch die von $\mathcal{K}$ während des Bezahl-Protokolls Payment gewählten Zahlen k und l durch

$$k = \log_{g_7} C_2 = \log_{g_8} C_3 \quad \text{und} \quad l = r_3 - dk \bmod q$$

eindeutig bestimmt.
Es bleibt zu zeigen, dass für diese Wahl die Protokolle Withdrawal und Payment erfolgreich durchgeführt werden können. Es ist

$$\begin{aligned} a &= (a')^u g^v = (a')^{cc'^{-1}} g^{r-r'u} = (g^w)^{cc'^{-1}} g^{r-c'x_iu-uw} = (g^w)^{cc'^{-1}} g^{r-cx_i-cc'^{-1}w} \\ &= (g^w)^{cc'^{-1}} g^r (g^{x_i})^{-c} (g^w)^{-cc'^{-1}} = g^r h_i^{-c} \end{aligned}$$

Woraus $g^r = h_i^c a$ folgt. Weiter gilt

$$\begin{aligned} A &= I g_2^s g_3 = I g_2^{\log_{g_2} A_1 - \log g_2 I} g_3 = I A_1 I^{-1} g_3 = A_1 g_3, \\ z &= z' h_{2,i}^s h_{3,i} = z' h_{2,i}^{\log_{g_2} A_1 - \log_{g_2} I} h_{3,i} = z' (g_2^{x_i})^{\log_{g_2} A_1 - \log_{g_2} I} h_{3,i} \\ &= z' (g_2^{\log_{g_2} A_1 - \log_{g_2} I})^{x_i} h_{3,i} = z' A_1^{x_i} I^{-x_i} h_{3,i} = A_1^{x_i} h_{3,i} = (A_1 g_3)^{x_i} = A^{x_i}, \\ b &= (b' b''^{-t})^{su} A^v = ((I' g_2)^w (g_4^w)^{-t})^{su} A^v = ((I^{s^{-1}} g_3^{s^{-1}} g_4^{-t} g_2)^w (g_4^t)^w)^{su} A^v \\ &= (I g_2^s g_3)^{wu} A^v = (I g_2^{\log_{g_2} A_1 - \log_{g_2} I} g_3)^{wu} A^v = A^{wu+v} = A^{wcc'^{-1}+r-r'u} \\ &= A^{wcc'^{-1}+r-(c'x_i+w)cc'^{-1}} = A^{r-cx_i} = A^r (A^{x_i})^{-c} = A^r z^{-c}. \end{aligned}$$

Woraus $z^c b = z^c A^r z^{-c} = A^r$ folgt. Außerdem ist

$$\begin{aligned} g_1^{x_1} A_{\mathcal{O}}^{es} B_{\mathcal{O}} &= g_1^{r_1 - r'_1 - du_1} A_{\mathcal{O}}^{(d'-d)s^{-1}s} B_{\mathcal{O}} = g_1^{r_1} g_1^{-r'_1 - du_1} A_{\mathcal{O}}^{d'-d} B_{\mathcal{O}} \\ &= g_1^{r_1} g_1^{-r'_1} g_1^{-du_1} A_{\mathcal{O}}^{d'} B_{\mathcal{O}} A_{\mathcal{O}}^{-d} = g_1^{r_1} (g_1^{\log_{g_1} A_{\mathcal{K}}})^{-d} A_{\mathcal{O}}^{-d} = g_1^{r_1} I^{-d} \\ B &= g_1^{x_1} g_2^{x_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}} = g_1^{r_1} I^{-d} g_2^{r_2 - ds} = g_1^{r_1} g_2^{r_2} I^{-d} (g_2^{\log_{g_2} A_1 - \log_{g_2} I})^{-d} = g_1^{r_1} g_2^{r_2} A_1^{-d}. \end{aligned}$$

Woraus $A_1^d B = g_1^{r_1} g_2^{r_2}$ folgt. Weiterhin gilt

$$\begin{aligned}
C_1' &= \bar{z}_i^l g_1^{x_1} A_{\mathcal{O}}^{es} B_{\mathcal{O}} = \bar{z}_i^{r_3 - dk} g_1^{r_1} I^{-d} = g_1^{r_1} \bar{z}_i^{r_3} (\bar{z}_i{}^k I)^{-d} = g_1^{r_1} \bar{z}_i^{r_3} C_1^{-d}, \\
C_2' &= g_7^l = g_7^{r_3 - dk} = g_7^{r_3} (g_7^k)^{-d} = g_7^{r_3} C_2^{-d}, \\
C_3' &= g_8^l = g_8^{r_3 - dk} = g_8^{r_3} (g_8^k)^{-d} = g_8^{r_3} C_3^{-d}.
\end{aligned}$$

Woraus $C_1^d C_1' = g_1^{r_1} \bar{z}_i^{r_3}$, $C_2^d C_2' = g_7^{r_3}$ und $C_3^d C_3' = g_8^{r_3}$ folgt. Ferner ist

$$\begin{aligned}
\tilde{I} &= (I')^y g_3^{y_3} g_4^{y_4} = (I')^{\hat{r}} (I^{s^{-1}} g_3^{s^{-1}} g_4^t)^{-\hat{c}s} g_3^{\hat{r}_3 + \hat{c}} g_4^{\hat{r}_4 + \hat{c}ts} = (I')^{\hat{r}} I^{-\hat{c}} g_3^{-\hat{c}} g_4^{-\hat{c}ts} g_3^{\hat{r}_3} g_3^{\hat{c}} g_4^{\hat{r}_4} g_4^{\hat{c}ts} \\
&= (I')^{\hat{r}} g_3^{\hat{r}_3} g_4^{\hat{r}_4} I^{-\hat{c}}
\end{aligned}$$

Woraus $I^{\hat{c}} \tilde{I} = (I')^{\hat{r}} g_3^{\hat{r}_3} g_4^{\hat{r}_4}$ folgt. Mit

$$\begin{aligned}
\tilde{E}_1 &= g_2^{\hat{r} - \hat{c}s} \hat{z}_i^{\hat{r}_5 - \hat{c}m} = g_2^{\hat{r}} (g_2^s)^{-\hat{c}} \hat{z}_i^{\hat{r}_5} (\hat{z}_i^m)^{-\hat{c}} = g_2^{\hat{r}} \hat{z}_i^{\hat{r}_5} (\hat{z}_i^m)^{-\hat{c}} (A_1 I^{-1})^{-\hat{c}} \\
&= g_2^{\hat{r}} \hat{z}_i^{\hat{r}_5} (\hat{z}_i^m g_2^s)^{-\hat{c}} = \hat{z}_i^{\hat{r}_5} E_1^{-\hat{c}}
\end{aligned}$$

folgt $\tilde{E}_1 E_1^{\hat{c}} = g_2^{\hat{r}} \hat{z}_i^{\hat{r}_5}$. Schließlich ist

$$\begin{aligned}
\tilde{g}_3 &= (I')^y A_{\mathcal{O}}^{y_6} C_{\mathcal{O}} g_1^{y_1} g_4^{y_2} = (I')^{\hat{r}} I^{-\hat{c}} g_3^{-\hat{c}} g_4^{-\hat{c}ts} A_{\mathcal{O}}^{\hat{c}' + \hat{c}} C_{\mathcal{O}} g_1^{\hat{r}_1 - \hat{r}_1' + \hat{c}u_1} g_4^{\hat{r}_2 + \hat{c}ts} \\
&= (I')^{\hat{r}} g_1^{\hat{r}_1} g_4^{\hat{r}_2} I^{-\hat{c}} (g_1^{u_1} A_{\mathcal{O}})^{\hat{c}} g_3^{-\hat{c}} = (I')^{\hat{r}} g_1^{\hat{r}_1} g_4^{\hat{r}_2} g_3^{-\hat{c}}
\end{aligned}$$

und es folgt $\tilde{g}_3 g_3^{\hat{c}} = (I')^{\hat{r}} g_1^{\hat{r}_1} g_4^{\hat{r}_2}$. Womit auch die Durchführbarkeit der beiden Protokolle Withdrawal und Payment gezeigt ist. □

Lemma 6.3.3 zeigt, dass die Kommunikation zwischen $\mathcal{K}$ und $\mathcal{B}$ bzw. $\mathcal{K}$ und $\mathcal{H}$ nicht als verdeckter Kanal zwischen $\mathcal{O}$ und $\mathcal{B}$ genutzt werden kann. Da sich die Sichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ und $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ gegenüber dem in Abschnitt 6.2 beschriebenen System nicht ändern, folgt aus Lemma 6.3.3 gleichzeitig die Anonymität der aktualisierbaren fairen elektronischen Geldbörse Π. Insgesamt ergibt sich aus der Sicherheitsanalyse von Π folgender Satz:

Satz 6.3.4. Die aktualisierbare faire elektronische Geldbörse Π ist im Random-Oracle-Modell ein (t, N)-langfristig sicheres faires elektronisches Geldsystem mit sicherer Schlüsselaktualisierung.

Bemerkung 6.3.3. Analog zu Abschnitt 6.2 folgt, dass die aktualisierbare faire elektronische Geldbörse eine stark $(N-1, N)$-langfristig sichere faire elektronische Geldbörse ist.

Kapitel 7

Deanonymisierer als Observer ausgebende Instanz

In elektronischen Geldsystemen werden Observer dazu eingesetzt, die Mehrfachausgabe elektronischer Münzen zu verhindern. Üblicherweise werden die Observer von der Bank an ihre Kunden ausgehändigt. Es ist aber durchaus denkbar, dass die Observer in einem fairen elektronischen Geldsystem nicht von der Bank, sondern vom Deanonymisierer an die Kunden verteilt werden; ein Ansatz, der in der Literatur bisher nicht diskutiert wurde. Ziel dieses Kapitels ist es, ein solches faires elektronisches Geldsystem zu entwickeln und zu untersuchen, welche Anforderungen in einem solchen System an den Observer und die eingesetzten kryptografischen Bausteine zu stellen sind. Des Weiteren soll analysiert werden, ob sich auf diese Weise ggf. effizientere Protokolle entwerfen lassen.

Zunächst wird in Abschnitt 7.1 ein generisches faires System vorgestellt, das von uns in [NS06] veröffentlicht wurde und in dem jedes sichere blinde Signaturverfahren als grundlegender Baustein eingesetzt werden kann. Dadurch, dass das System auf einem beliebigen, sicheren blinden Signaturverfahren aufgebaut werden kann, lässt es sich durch den Einsatz eines langfristig sicheren blinden Signaturverfahrens zu einem langfristig sicheren elektronischen Geldsystem erweitern. Verwendet man für Kunden- und Münztracing zusätzlich ein geeignetes Verschlüsselungsverfahren, beispielsweise das in Abschnitt 5.1 beschriebene Verschlüsselungsverfahren [DKXY02], so kann das System darüber hinaus in ein generisches langfristig sicheres faires elektronisches Geldsystem transformiert werden.

Es werden unterschiedliche Ansätze diskutiert, mit denen in diesem System Kunden- und Münztracing unter Einbezug des Observers realisiert werden können, ohne dadurch die Anonymität der Kunden unter bestimmten Voraussetzungen zu gefährden. Es zeigt sich, dass einer der beschriebenen Ansätze im Wesentlichen den von S. Kim *et al.* vorgestellten Gruppensignaturen mit Observern [KPW97] entspricht.

Um diese als Baustein für das in Abschnitt 7.1 konstruierte generische System zu nutzen, werden im darauffolgenden Abschnitt 7.2 Schwachstellen der von S. Kim *et al.* vorgestellten Gruppensignaturen mit Observern angesprochen, und es wird ein verbessertes Sicherheitsmodell für Gruppensignaturen mit Observern vorgeschlagen. Abschließend

wird in Abschnitt 7.3 ein neues Gruppensignaturverfahren mit Observern konstruiert, das sich zur Realisierung von Kunden- und Münztracing im generischen fairen System aus Abschnitt 7.1 eignet und den neuen Sicherheitsanforderungen entspricht.

7.1 Ein generisches faires elektronisches Geldsystem

Um die Mehrfachausgabe elektronischer Münzen nicht nur im Nachhinein aufdecken zu können, sondern diese bereits im Voraus zu verhindern, werden in einem elektronischen Geldsystem, wie bereits erwähnt, die Kunden von der Bank mit einem Observer ausgestattet. Damit die Anonymität der Kunden in einem Geldsystem, in dem die Observer von der Bank verteilt werden, garantiert werden kann, müssen Abhebe- und Bezahl-Protokoll so konzipiert werden, dass die Bank auch mit den auf den Observern gespeicherten Daten die Anonymität der Kunden nicht aufheben kann (vgl. Abschnitt 3.2).

In diesem Abschnitt wird ein generisches faires elektronisches Geldsystem entworfen, in dem die Observer nicht von der Bank, sondern vom Deanonymisierer an die Kunden ausgegeben werden. Das heißt, vor einer Kontoeröffnung bei der Bank müssen sich die Kunden zunächst beim Deanonymisierer registrieren, der ihnen den zur Kontoeröffnung benötigten Observer aushändigt. Da der Deanonymisierer eine unabhängig von der Bank handelnde Partei ist, die das Vertrauen aller Kunden genießt, kann man davon ausgehen, dass die von den Observern während des Abhebe- bzw. Bezahl-Protokolls berechneten und gesendeten Daten der Bank keine effizient berechenbare Information liefern, die die Bank möglicherweise dazu verwenden könnte, die Anonymität der Kunden aufzuheben, das heißt ein- und ausgehende Information zwischen der Bank und den Observern kann nicht entstehen (vgl. Abschnitt 3.2). Außerdem ist nicht zu erwarten, dass die vom Deanonymisierer ausgegebenen Observer in den Besitz der Bank geraten und somit kann vorausgesetzt werden, dass die Bank die auf den Observern gespeicherten Daten nicht auslesen kann.

Diese Voraussetzungen ermöglichen es, auf den Observern der Kunden auch Daten zu speichern, die es der Bank sonst ermöglichen würden, mit Hilfe dieser Information die Anonymität der Kunden aufzuheben. Insbesondere können auf diese Weise Kunden- und Münztracing mit den Observern umgesetzt werden, wodurch die üblicherweise als Baustein für Kunden- und Münztracing verwendeten, rechenintensiven Zero-Knowledge-Beweise ersetzt werden können. Allerdings ist anzumerken, dass hier ein hohes Maß an Vertrauen in die Manipulationssicherheit des Observers gesetzt wird, da von ihr letztlich die Sicherheit des Systems abhängt.

Das in diesem Abschnitt vorgestellte System wurde von uns in [NS06] veröffentlicht. Neben der Möglichkeit deutlich effizientere Protokolle zu implementieren, bietet das System vor allem den Vorteil, dass sich durch den Einsatz aktualisierbarer blinder Signaturverfahren und aktualisierbarer Public-Key-Verschlüsselungsverfahren somit auch ein generisches langfristig sicheres faires elektronisches Geldsystem angeben lässt.

7.1.1 Initialisierung des Systems und Kontoeröffnung

Initialiserung des Systems: Zur Initialisierung des Systems wählt die Bank $\mathcal{B}$ ein geeignetes blindes Signaturverfahren $\Pi = (\mathsf{Gen}, \mathsf{Sign}(\mathcal{K}(\cdot), \mathcal{B}(\cdot)), \mathsf{Verify})$ mit Sicherheitsparameter $k \in \mathbb{N}$ und berechnet das Schlüsselpaar $(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B}) = \mathsf{Gen}(1^k)$. Ihren öffentlichen Schlüssel $\mathsf{pk}_\mathcal{B}$ und alle erforderlichen Systemparameter publiziert $\mathcal{B}$.
Damit Kunden- und Münztracing durchgeführt werden können, wählt der Deanonymisierer $\mathcal{D}$ ein geeignetes probabilistisches Verschlüsselungsverfahren $\Sigma = (\mathsf{Gen}_\Sigma, \mathsf{Enc}, \mathsf{Dec})$ mit Sicherheitsparameter k, erstellt das Schlüsselpaar $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D}) = \mathsf{Gen}_\Sigma(1^k)$ und gibt den öffentlichen Schlüssel $\mathsf{pk}_\mathcal{D}$ bekannt. Zusätzlich wählt $\mathcal{D}$ zwei adäquate Signaturverfahren $\Pi_1 = (\mathsf{Gen}_1, \mathsf{Sign}_1, \mathsf{Verify}_1)$ und $\Pi_2 = (\mathsf{Gen}_2, \mathsf{Sign}_2, \mathsf{Verify}_2)$ mit Sicherheitsparameter k.

Registrierung beim Deanonymisierer: Um sich als Kunde $\mathcal{K}$ bei $\mathcal{D}$ zu registrieren, werden auf einem manipulationssicheren Speichermedium, dem Observer $\mathcal{O}$, die beiden Schlüsselpaare $(\mathsf{pk}_\mathcal{K}, \mathsf{sk}_\mathcal{K}) = \mathsf{Gen}_1(1^k)$ und $(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) = \mathsf{Gen}_2(1^k)$ generiert und gespeichert. Die beiden privaten Schlüssel $\mathsf{sk}_\mathcal{K}$ und $\mathsf{sk}_\mathcal{O}$ können weder von $\mathcal{K}$ noch von $\mathcal{D}$ ausgelesen werden. Der Deanonymisierer $\mathcal{D}$ speichert die persönlichen Daten von $\mathcal{K}$ zusammen mit $\mathsf{pk}_\mathcal{K}, \mathsf{pk}_\mathcal{O}$ in seiner Datenbank, veröffentlicht $\mathsf{pk}_\mathcal{K}, \mathsf{pk}_\mathcal{O}$ und überreicht den Observer $\mathcal{O}$ an $\mathcal{K}$.

Kontoeröffnung: Zur Kontoeröffnung wird das Protokoll $\mathsf{Reg}(\mathcal{K}(\mathsf{pk}_\mathcal{B}), \mathcal{B}(\mathsf{pk}_\mathcal{B}))$ zwischen $\mathcal{K}$ und $\mathcal{B}$ durchgeführt: Der Kunde $\mathcal{K}$ authentifiziert sich gegenüber $\mathcal{B}$, indem er nachweist, dass er im Besitz eines gültigen von $\mathcal{D}$ ausgegebenen Observers $\mathcal{O}$ ist (z.B. durch eine von $\mathcal{O}$ mit $\mathsf{sk}_\mathcal{K}$ signierte und von $\mathcal{B}$ zufällig gewählte Nachricht m). Bei erfolgreicher Verifikation speichert $\mathcal{B}$ die persönlichen Daten von $\mathcal{K}$ zusammen mit $\mathsf{pk}_\mathcal{K}$ in ihrer Datenbank. Anhand des öffentlichen Schlüssels $\mathsf{pk}_\mathcal{K}$ von $\mathcal{K}$ kann $\mathcal{B}$ im Falle eines Double-Spendings bzw. Kundentracings die gesuchte Identität aufdecken.

7.1.2 Eine Münze abheben

Möchte der Kunde $\mathcal{K}$ eine Münze Coin bei der Bank $\mathcal{B}$ abheben, authentifizieren sich $\mathcal{K}$ und $\mathcal{B}$ zunächst gegenseitig. Anschließend wird das im Folgenden dargestellte Protokoll $\mathsf{Withdrawal}(\mathcal{O}(\mathsf{pk}_\mathcal{K}, \mathsf{sk}_\mathcal{K}, \mathsf{pk}_\mathcal{D}), \mathcal{K}(\mathsf{pk}_\mathcal{K}, \mathsf{pk}_\mathcal{B}), \mathcal{B}(\mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{K}))$ durchgeführt (vgl. Abbildung 7.1):

Schritt 1: Der Kunde $\mathcal{K}$ wählt zufällig einen Münz-Identifikator $m \in_\mathcal{R} M$, wobei M die Nachrichtenmenge des blinden Signaturverfahren Π ist, und sendet m an $\mathcal{O}$.

Schritt 2: Der Observer $\mathcal{O}$ speichert m, berechnet den Geheimtext $c_1 = \mathsf{Enc}(\mathsf{pk}_\mathcal{D}, m)$ des Münz-Identifikators, sowie die Signatur $s_1 = \mathsf{Sign}_1(c_1, \mathsf{sk}_\mathcal{K}, \mathsf{pk}_\mathcal{K})$ und sendet c_1, s_1 an $\mathcal{K}$. Anhand des Geheimtextes c_1 kann $\mathcal{B}$ in Kooperation mit $\mathcal{D}$ Münztracing durchführen. Da die Signatur s_1 nur von $\mathcal{O}$ erstellt werden kann (der Signaturschlüssel $\mathsf{sk}_\mathcal{K}$ ist auf dem Observer $\mathcal{O}$ gespeichert), garantiert s_1 der Bank $\mathcal{B}$, dass c_1 eine Verschlüsselung des Münz-Identifikators m unter $\mathsf{pk}_\mathcal{D}$ ist.

Schritt 3: Der Kunde $\mathcal{K}$ überprüft die Signatur s_1 und sendet c_1, s_1 an $\mathcal{B}$.

Schritt 4: Die Bank $\mathcal{B}$ verifiziert die Signatur s_1. Bei erfolgreicher Verifikation führen $\mathcal{K}$ und $\mathcal{B}$ im Anschluss das blinde Signaturprotokoll $\mathsf{Sign}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, m\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}\right)\right)$ aus.

Wird das blinde Signaturprotokoll $\mathsf{Sign}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, m\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}\right)\right)$ erfolgreich durchgeführt, besitzt $\mathcal{K}$ eine Signatur s der Bank $\mathcal{B}$ für die Nachricht m und somit eine Münze $\mathsf{Coin} = (m, s)$.

$\mathcal{O}\left(\mathsf{pk}_{\mathcal{K}}, \mathsf{sk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{D}}\right)$		$\mathcal{K}\left(\mathsf{pk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{B}}\right)$	$\mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{K}}\right)$
	$\xleftarrow{m}$	$m \in_{\mathcal{R}} M$	
$c_1 = \mathsf{Enc}\left(\mathsf{pk}_{\mathcal{D}}, m\right)$			
$s_1 = \mathsf{Sign}_1\left(c_1, \mathsf{sk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{K}}\right)$	$\xrightarrow{c_1, s_1}$	$\mathsf{Verify}_1\left(c_1, \mathsf{pk}_{\mathcal{K}}, s_1\right)$	
		$\xrightarrow{c_1, s_1}$	$\mathsf{Verify}_1\left(c_1, \mathsf{pk}_{\mathcal{K}}, s_1\right)$
		$\mathsf{Sign}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, m\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}\right)\right)$	$\xrightarrow{\qquad\qquad\qquad\qquad}$
		$\mathsf{Coin} = (m, s)$	

Abbildung 7.1: Das Abhebe-Protokoll im generischen fairen System

7.1.3 Mit einer Münze bezahlen

Um mit einer Münze Coin bei einem Händler $\mathcal{H}$ zu bezahlen, wird das im Folgenden dargestellte Protokoll $\mathsf{Payment}\left(\mathcal{O}\left(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{K}}, \mathsf{pk}_{\mathcal{D}}\right), \mathcal{K}\left(\mathsf{Coin}, \mathsf{pk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{B}}\right), \mathcal{H}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{O}}\right)\right)$ durchgeführt (vgl. Abbildung 7.2):

Schritt 1: Der Kunde $\mathcal{K}$ sendet den Münz-Identifikator m, den Zeitpunkt ts des Bezahlvorgangs und den öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{H}}$ des Händlers $\mathcal{H}$ an $\mathcal{O}$.

Schritt 2: Der Observer $\mathcal{O}$ überprüft, ob der Münz-Identifikator m noch gespeichert ist. Falls ja, berechnet $\mathcal{O}$ den Geheimtext $c_2 = \mathsf{Enc}\left(\mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{K}}\right)$ und die Signatur $s_2 = \mathsf{Sign}_2\left((m, c_2), \mathsf{sk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{K}}\right)$. Anschließend löscht $\mathcal{O}$ den Münz-Identifikator m und sendet c_2, s_2 an $\mathcal{K}$. Ist der Münz-Identifikator m bereits gelöscht, bricht $\mathcal{O}$ die Kommunikation mit $\mathcal{K}$ ab. Anhand des Geheimtextes c_2 kann $\mathcal{B}$ in Kooperation mit $\mathcal{D}$ Kundentracing durchführen. Da die Signatur s_2 nur von $\mathcal{O}$ erstellt werden kann, garantiert s_2 der Bank $\mathcal{B}$, dass c_2 eine Verschlüsselung der Identität $\mathsf{pk}_{\mathcal{K}}$ unter $\mathsf{pk}_{\mathcal{D}}$ ist. Gleichzeitig stellt die Signatur s_2 auf (m, c_2) sicher, dass $\mathcal{K}$ die Münze m nicht ohne den Observer ausgeben kann.

Schritt 3: Der Kunde $\mathcal{K}$ überprüft die Signatur s_2 und sendet $\mathsf{Coin}, c_2, s_2, \mathsf{ts}$ an $\mathcal{H}$.

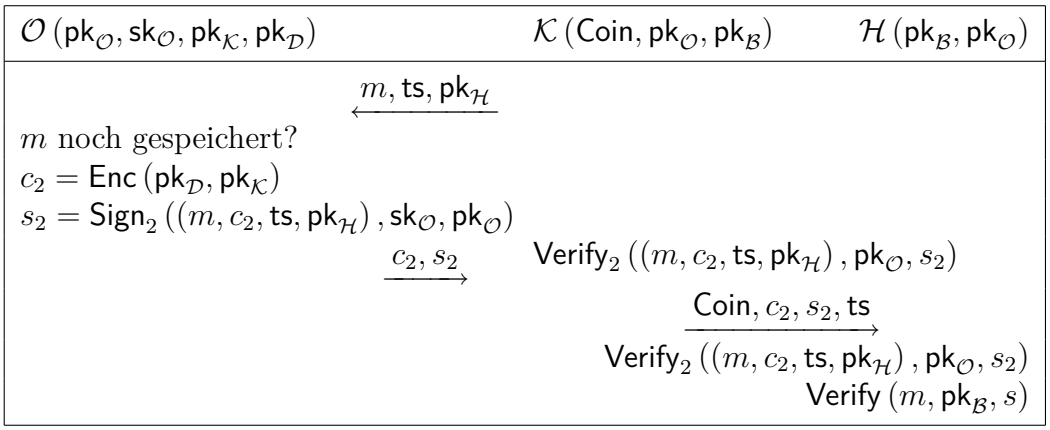

Abbildung 7.2: Das Bezahl-Protokoll im generischen fairen System

Schritt 4: Der Händler $\mathcal{H}$ verifiziert die beiden Signaturen s und s_2.

Für die Anonymität des Systems ist es entscheidend, dass in Schritt 2 des Bezahl-Protokolls Payment ein probabilistisches Verschlüsselungsverfahren Σ eingesetzt wird, da im Falle eines deterministischen Verschlüsselungsverfahrens die Identität $\mathsf{pk}_\mathcal{K}$ von $\mathcal{K}$ bei jedem von $\mathcal{K}$ durchgeführten Bezahl-Protokoll gleich verschlüsselt würde. Das heißt, bei jedem Bezahlvorgang würde derselbe Geheimtext c_2 übertragen und die Münzen von $\mathcal{K}$ wären somit verknüpfbar. Außerdem darf der zur Verifikation von s_2 benötigte öffentliche Schlüssel $\mathsf{pk}_\mathcal{O}$ nicht direkt mit der Identiät von $\mathcal{K}$ in Verbindung gebracht werden. Auf welche Art und Weise man dies realisieren kann, wird in Abschnitt 7.1.6 erörtert.

7.1.4 Eine Münze einlösen

Möchte $\mathcal{H}$ die Münze Coin bei $\mathcal{B}$ einlösen, authentifizieren sich beide zunächst gegenseitig. Anschließend wird das Protokoll $\mathsf{Deposit}\left(\mathcal{H}\left(\mathsf{view}_\mathcal{H}^\mathcal{K}(P)\right), \mathcal{B}(\mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{O})\right)$ durchgeführt. Der Händler $\mathcal{H}$ sendet $\mathsf{view}_\mathcal{H}^\mathcal{K}(P)$ an $\mathcal{B}$. Die Bank verifiziert die in $\mathsf{view}_\mathcal{H}^\mathcal{K}(P)$ enthaltenen Signaturen s und s_2. Gelingen beide Verifikationen sucht $\mathcal{B}$ in ihrer Datenbank nach Coin und speichert Coin gegebenenfalls. Ist Coin schon in der Datenbank enthalten, so versucht entweder $\mathcal{H}$ die Münze erneut einzulösen oder $\mathcal{K}$ hat die Münze ein weiteres Mal ausgegeben. Falls s_2 bereits in der Datenbank gespeichert ist, versucht $\mathcal{H}$ zu betrügen. Andernfalls hat $\mathcal{K}$ die Münze mehrfach ausgegeben und $\mathcal{B}$ kann die Identität von $\mathcal{K}$ mit Hilfe von $\mathcal{D}$ aufdecken, indem $\mathcal{B}$ Kundentracing durchführt (vgl. Abschnitt 7.1.5).

7.1.5 Kunden- und Münztracing

Um in bestimmten Verdachtsmomenten Kunden- und Münztracing durchzuführen, werden von der Bank $\mathcal{B}$ und dem Deanonymisierer $\mathcal{D}$ die beiden in Abbildung 7.3 dargestellten Protokolle ausgeführt.

$\mathsf{KTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D)\right), \mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}\right)\right)$:	
$\mathcal{B}(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D))$	$\mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}\right)$
$\xrightarrow{c_2}$ $\underset{\mathsf{pk}_{\mathcal{K}}}{\longleftarrow}$	$\mathsf{pk}_{\mathcal{K}} = \mathsf{Dec}\left(\mathsf{sk}_{\mathcal{D}}, c_2\right)$
Suche $\mathsf{pk}_{\mathcal{K}}$ in der Datenbank.	

$\mathsf{MTrace}\left(\mathcal{B}\left(\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)\right), \mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}\right)\right)$:	
$\mathcal{B}(\mathsf{view}_{\mathcal{B}}^{\mathcal{H}}(D))$	$\mathcal{D}\left(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}\right)$
$\xrightarrow{c_1}$ $\underset{m}{\longleftarrow}$	$m = \mathsf{Dec}\left(\mathsf{sk}_{\mathcal{D}}, c_1\right)$
Suche m in der Datenbank.	

Abbildung 7.3: Kunden- und Münztracing im generischen fairen System

7.1.6 Anforderungen an die Signaturverfahren Π_1 und Π_2

Im Wesentlichen werden die beiden Signaturverfahren Π_1 und Π_2 dazu eingesetzt, mit Hilfe des Observers Kunden- und Münztracing zu garantieren bzw. die Mehrfachausgabe elektronischer Münzen zu verhindern. Um Münztracing durchführen zu können, wird der Geheimtext c_1 des verschlüsselten Münz-Identifikators m während des Abhebe-Protokolls vom Observer $\mathcal{O}$ mit $\mathsf{sk}_{\mathcal{K}}$ signiert und die Signatur s_1 wird anschließend von der Bank $\mathcal{B}$ verifiziert. Da $\mathcal{B}$ weiß, mit welchem Kunden $\mathcal{K}$ die Bank $\mathcal{B}$ während des Abhebe-Protokolls kommuniziert, kann der zu $\mathsf{sk}_{\mathcal{K}}$ gehörende öffentliche Schlüssel $\mathsf{pk}_{\mathcal{K}}$ unter der Identität von $\mathcal{K}$ veröffentlicht werden. Im Gegensatz dazu kann der öffentliche Schlüssel $\mathsf{pk}_{\mathcal{O}}$, der während des Bezahlens mit der Münze Coin zur Verifikation der Signatur s_2 benötigt wird, nicht unter der Identität von $\mathcal{K}$ publiziert werden. Denn anhand von $\mathsf{pk}_{\mathcal{K}}$ könnte $\mathcal{B}$ ansonsten beim Einlösen von Coin die Identität von $\mathcal{K}$ feststellen und somit die Anonymität von $\mathcal{K}$ aufheben. Das heißt, der öffentliche Schlüssel $\mathsf{pk}_{\mathcal{O}}$ bzw. das Signaturverfahren Π_2 muss so gewählt werden, dass die Anonymität der Kunden gewährleistet bleibt. Dies kann auf unterschiedliche Art und Weise realisiert werden:

Pseudonyme: Für jeden Kunden $\mathcal{K}$ generiert der Deanonymisierer $\mathcal{D}$ während der Registrierung ein Pseudonym $\mathsf{pseudo}_{\mathcal{K}}$. Der öffentliche Schlüssel $\mathsf{pk}_{\mathcal{O}}$, des auf dem Observer $\mathcal{O}$ generierten Schlüsselpaars $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}) = \mathsf{Gen}_2(1^k)$ wird von $\mathcal{D}$ unter dem Pseudonym $\mathsf{pseudo}_{\mathcal{K}}$ publiziert. Auf diese Weise können sowohl der Händler $\mathcal{H}$ als auch die Bank $\mathcal{B}$ verifizieren, dass die während des Bezahlens mit der Münze Coin erstellte Signatur s_2 von $\mathcal{O}$ erstellt wurde. Da $\mathcal{B}$ lediglich das Pseudonym $\mathsf{pseudo}_{\mathcal{K}}$ kennt, nicht aber die Identität des Kunden $\mathcal{K}$, der sich hinter dem Pseudonym verbirgt, kann $\mathcal{B}$ zwar alle Münzen, die von $\mathcal{K}$ abgehoben wurden, miteinander verknüpfen, diese aber nicht der tatsächlichen Identität von $\mathcal{K}$ zuordnen. Das heißt, dieser Ansatz bietet zwar keine Anonymität im Sinne von Definition 3.1.2, man erreicht aber noch Pseudonymität.

Gruppensignaturen: Anstatt ein „gewöhnliches“ Signaturverfahren zu verwenden, kann für Π_2 auch ein sogenanntes *Gruppensignaturverfahren* eingesetzt werden. Gruppensignaturen wurden im Jahre 1991 von D. Chaum und E. van Heyst vorgeschlagen [CH91] und ermöglichen es Mitgliedern einer Gruppe, im Namen der Gruppe anonym Signaturen zu erstellen. Die Anonymität der Mitglieder kann bei Bedarf durch eine Trus-

ted-Third-Party aufgehoben werden (vgl. auch Abschnitt 3.3). In unserem Fall besteht die Gruppe aus allen beim Deanonymisierer registrierten Kunden bzw. aus allen von $\mathcal{D}$ ausgegebenen Observern. Durch den Einsatz eines Gruppensignaturverfahrens (z.B. [ACJT00], [CG04]) wird während des Bezahlens mit der Münze Coin anhand der Gruppensignatur s_2 sichergestellt, dass der für Kundentracing benötigte Geheimtext c_2 von einem von $\mathcal{D}$ ausgegebenen Observer erstellt und signiert wurde. Da die Gruppensignatur s_2 anonym ist, erhält die Bank $\mathcal{B}$ durch s_2 keine Information über den Kunden, der Coin bei $\mathcal{B}$ abgehoben hat und man kann auf diese Weise die Anonymität des Systems erreichen.

Gruppenschlüssel: Der Deanonymisierer $\mathcal{D}$ generiert während der Initialisierung des Systems einmalig das Schlüsselpaar $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}) = \mathsf{Gen}_2(1^k)$ und publiziert den öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{O}}$. Für jeden Kunden $\mathcal{K}$ wird das Schlüsselpaar $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}})$ während der Registrierung auf dessen Observer $\mathcal{O}$ gespeichert. Durch das Schlüsselpaar $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}})$ wird gewissermaßen sichergestellt, dass $\mathcal{K}$ bei $\mathcal{D}$ registriert ist. Somit können sowohl der Händler $\mathcal{H}$ als auch die Bank $\mathcal{B}$ verifizieren, dass die während des Bezahlens mit der Münze Coin erstellte Signatur s_2 von einem Observer mit $\mathsf{sk}_{\mathcal{O}}$ angefertigt wurde. Da der öffentliche Schlüssel $\mathsf{pk}_{\mathcal{O}}$ für alle Kunden gültig ist, erhält $\mathcal{B}$ durch die Signatur s_2 keine Information über den Kunden, der Coin bei $\mathcal{B}$ abgehoben hat. Das heißt, durch den Einsatz eines „Gruppenschlüssels“ kann man Anonymität im Sinne von Definition 3.1.2 erreichen.
Der Ansatz, die Identität $\mathsf{pk}_{\mathcal{K}}$ des Kunden $\mathcal{K}$ auf dem Observer $\mathcal{O}$ zu verschlüsseln und den für das Kundentracing benötigten Geheimtext c_2 anschließend mit dem auf $\mathcal{O}$ gespeicherten Gruppenschlüssel $\mathsf{sk}_{\mathcal{O}}$ zu signieren, stellt im Prinzip eine der einfachsten Methoden dar, mit Hilfe eines Observers Gruppensignaturen zu erstellen und entspricht im Wesentlichen den von Kim *et al.* vorgestellten *Gruppensignaturen mit Observern* [KPW97]. Vor- und Nachteile dieses Ansatzes liegen gleichermaßen auf der Hand. Ein Vorzug ist seine Einfachheit und die daraus resultierenden sehr effizient zu berechnenden Gruppensignaturen. Die Schwachstelle ist der auf allen Observern gespeicherte Gruppenschlüssel $\mathsf{sk}_{\mathcal{O}}$. Sobald ein ausgegebener Observer kompromittiert wird, ist die Sicherheit des Verfahrens nicht mehr gewährleistet. Das heißt, falls in dem, in diesem Abschnitt vorgestellten, generischen fairen Geldsystem für Π_2 ein Gruppensignaturverfahren mit Observern verwendet wird, ist die Sicherheit des gesamten Systems nicht mehr gewährleistet, sobald nur ein von $\mathcal{D}$ ausgegebener Observer kompromittiert wird. Dies verdeutlicht den bereits angesprochenen Zusammenhang zwischen der Sicherheit des generischen Geldsystems und der Manipulationssicherheit des Observers.

7.1.7 Sicherheitsanalyse

Die Sicherheitsanalyse eines fairen elektronischen Geldsystems umfasst Eigenschaften wie Anonymität, Unfälschbarkeit und Nichterweiterbarkeit (vgl. Definition 3.3.2). Da für die Sicherheit entscheidende Teile des Abhebe- bzw. Bezahl-Protokolls auf dem Observer ausgeführt werden, hängt die Sicherheit des in diesem Abschnitt vorgestellten

generischen fairen elektronischen Geldsystem Ω maßgeblich von der Manipulationssicherheit des Observers ab. Das heißt, die Sicherheit basiert im Wesentlichen auf folgender Annahme:

Annahme 7.1. Der vom Deanonymisierer $\mathcal{D}$ an den Kunden $\mathcal{K}$ ausgegebene Observer $\mathcal{O}$ ist manipulationssicher und wird von $\mathcal{D}$ unabhängig von der Bank $\mathcal{B}$ an $\mathcal{K}$ ausgegeben. Insbesondere lässt sich der gesamte Speicherinhalt von $\mathcal{O}$ weder von $\mathcal{K}$ noch von der Bank $\mathcal{B}$ auslesen und es entsteht keine ein- bzw. ausgehende Information zwischen $\mathcal{O}$ und $\mathcal{B}$.

Für die nun folgende Sicherheitsanalyse von Ω setzen wir voraus, dass die eingesetzten Bausteine folgende Bedingungen erfüllen:

- Die beiden Signaturverfahren Π_1, Π_2 sind nicht existentiell fälschbar unter einem adaptiven Angriff mit gewählten Nachrichten.
- Das blinde Signaturverfahren Π ist perfekt blind und sicher gemäß Definition 2.6.2.
- Das probabilistische Public-Key-Verschlüsselungsverfahren Σ ist semantisch sicher.
- Für das Signaturverfahren Π_2 wird vom Deanonymisierer $\mathcal{D}$ ein Schlüsselpaar $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}) = \mathsf{Gen}_2(1^k)$ generiert, dass auf allen Observern gespeichert wird. Der öffentliche Schlüssel $\mathsf{pk}_{\mathcal{O}}$ wird von $\mathcal{D}$ publiziert.

Identitätsnachweis: Im Falle eines Double-Spendings muss die Bank $\mathcal{B}$ die Identität des Betrügers mit einer überwältigenden Wahrscheinlichkeit feststellen können. Die Bank $\mathcal{B}$ wird eine vorliegende Münze $\mathsf{Coin} = (m, s)$ nur dann einlösen, wenn s eine gültige Signatur des Münz-Identifikators m ist und die in der Ansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ enthaltene Signatur s_2 eine gültige Signatur der Nachricht $(m, c_2, \mathsf{ts}, \mathsf{pk}_{\mathcal{H}})$ ist. Unter Annahme 7.1 kann die Signatur s_2 nur von dem Observer $\mathcal{O}$ des Kunden $\mathcal{K}$ erstellt werden, der die Münze Coin bei $\mathcal{B}$ abgehoben hat. Die Signatur s_2 garantiert, dass der Geheimtext c_2 von $\mathcal{O}$ erstellt und die Identität $\mathsf{pk}_{\mathcal{K}}$ von $\mathcal{K}$ unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ verschlüsselt wurde. Dies impliziert, dass $\mathcal{B}$ in Kooperation mit dem Deanonymisierer $\mathcal{D}$ die Identität des Betrügers aufdecken kann. Der Identitätsnachweis folgt somit aus der Fälschungssicherheit von Π_2.

Nichterweiterbarkeit: Eine gültige Münze $\mathsf{Coin} = (m, s)$ besteht aus einem Münz-Identifikator m und einer Signatur s, die während des zwischen dem Kunden $\mathcal{K}$ und der Bank $\mathcal{B}$ durchgeführten Abhebe-Protokolls $\mathsf{Withdrawal}$ mit dem blinden Signaturprotokoll $\mathsf{Sign}\left(\mathcal{K}\left(\mathsf{pk}_{\mathcal{B}}, m\right), \mathcal{B}\left(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}}\right)\right)$ erstellt wird. Die Nichterweiterbarkeit resultiert daher direkt aus der Sicherheit des blinden Signaturverfahrens Π.

Unfälschbarkeit: Folgt analog zum Identitätsnachweis. Ist $\mathsf{Coin} = (m, s)$ eine der Bank $\mathcal{B}$ vorliegende gültige Münze, so kann die in der Ansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ enthaltene Signatur s_2 unter Annahme 7.1 nur von dem Observer $\mathcal{O}$ des Kunden $\mathcal{K}$ erstellt werden,

der die Münze Coin bei $\mathcal{B}$ abgehoben hat. Somit ist sichergestellt, dass der ebenfalls in $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ enthaltene Geheimtext c_2 die Identität $\mathsf{pk}_{\mathcal{K}}$ von $\mathcal{K}$ verschlüsselt und $\mathcal{B}$ in Kooperation mit $\mathcal{D}$ die Identität des Betrügers aufdecken kann. Die Unfälschbarkeit folgt also ebenfalls aus der Fälschungssicherheit von Π_2.

Anonymität: Die Anonymität des Systems ist gewährleistet, falls die Bank $\mathcal{B}$ anhand ihr vorliegender Ansichten $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0), \mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$, der in W_0 und W_1 abgehobenen Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ und den Ansichten $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ der entsprechenden Münzen $\mathsf{Coin}_b, \mathsf{Coin}_{\bar{b}}$ nicht entscheiden kann, ob Coin_b zu W_0 oder W_1 gehört. Da Π ein perfekt blindes Signaturverfahren ist, liefern die beiden Münzen $\mathsf{Coin}_b = (m_b, s_b)$ und $\mathsf{Coin}_{\bar{b}} = (m_{\bar{b}}, s_{\bar{b}})$ der Bank $\mathcal{B}$ keine Information darüber, ob Coin_b in W_0 oder W_1 abgehoben wurde. Unter Annahme 7.1 und der Voraussetzung, dass für das Signaturverfahren Π_2 auf allen Observern dasselbe Schlüsselpaar gespeichert ist, liefern auch die in $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_0}(W_0)$, $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}_1}(W_1)$ bzw. $\mathsf{view}_{\mathcal{H}_b}^{\mathcal{K}_b}(P_b)$, $\mathsf{view}_{\mathcal{H}_{\bar{b}}}^{\mathcal{K}_{\bar{b}}}(P_{\bar{b}})$ enthaltenen Signaturen $s_1^{(0)}, s_1^{(1)}$ bzw. $s_2^{(b)}, s_2^{\bar{b}}$ keine Information. Da Σ ein semantisch sicheres Verschlüsselungsverfahren ist, kann $\mathcal{B}$ auch anhand der vorliegendene Geheimtexte $c_1^{(0)}, c_1^{(1)}$ und $c_2^{(b)}, c_2^{\bar{b}}$ nicht entscheiden, ob Coin_b in W_0 oder W_1 abgehoben wurde. Das heißt, die Anonymität des Systems basiert auf der perfekten Blindheit von Π und der semantischen Sicherheit von Σ.

Münz- und Kundentracing: In bestimmten Verdachtsmomenten muss die Bank $\mathcal{B}$ in Kooperation mit dem Deanonymisierer $\mathcal{D}$ Kunden- und Münztracing durchführen können. Unter Annahme 7.1 kann die in der Ansicht $\mathsf{view}_{\mathcal{B}}^{\mathcal{K}}(W)$ enthaltene Signatur s_1 nur von $\mathcal{K}$'s Observer $\mathcal{O}$ erstellt werden. Dies garantiert, dass der Geheimtext c_1 von $\mathcal{O}$ erstellt und der Münz-Identifikator m unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ verschlüsselt wurde. Mit dem gleichen Argument folgt, dass der in der Ansicht $\mathsf{view}_{\mathcal{H}}^{\mathcal{K}}(P)$ enthaltene Geheimtext c_2 unter Annahme 7.1 die Identität $\mathsf{pk}_{\mathcal{K}}$ des Kunden $\mathcal{K}$ unter dem öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{D}}$ verschlüsselt. Das heißt, der Deanonymisierer $\mathcal{D}$ kann die beiden Geheimtexte c_1 und c_2 korrekt entschlüsseln, und $\mathcal{B}$ kann anhand von m bzw. $\mathsf{pk}_{\mathcal{K}}$ Münz- bzw. Kundentracing durchführen. Münz- und Kundentracing ergeben sich somit unmittelbar aus der Fälschungssicherheit der beiden Signaturverfahren Π_1 und Π_2.

Insgesamt folgt somit aus obiger Sicherheitsanalyse des generischen fairen elektronischen Geldsystems Ω der folgende Satz:

Satz 7.1.1. Unter Annahme 7.1 ist das in diesem Abschnitt vorgestellte generische faire elektronische Geldsystem Ω mit Sicherheitsparameter $k \in \mathbb{N}$ ein gemäß Definition 3.3.2 sicheres faires elektronisches Geldsystem.

7.1.8 Effizienz

Durch den Einsatz von Observern und der Auslagerung der für Kunden- und Münztracing relevanten Teile des Abhebe- bzw. Bezahl-Protokolls auf die Observer ist im

Vergleich zu anderen fairen elektronischen Geldsystemen unter Annahme 7.1 die Entwicklung effizienterer Protokolle möglich. Allerdings beruht die Sicherheit des Systems dagegen zum größten Teil auf der Manipulationssicherheit der Observer.
Um die Effizienz des in diesem Abschnitt vorgeschlagenen generischen fairen elektronischen Geldsystems Ω mit anderen fairen elektronischen Geldsystemen zu vergleichen, betrachten wir Geldsysteme, die auf der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$ arbeiten (p, q sind entsprechend gewählte Primzahlen). Zu den effizientesten Systemen dieser Klasse zählt zum einen das in Abschnitt 6.1 vorgestellte faire Geldsystem *FOLC* [FTY98] und zum anderen das von M. Gaud und J. Traoré entwickelte System [GT03]. Für den Effizienzvergleich betrachten wir die modularen Exponentiationen, die in den drei Systemen während der Protokolle Withdrawal und Payment durchgeführt werden müssen (vgl. Tabelle 7.1). Um einen möglichst sinnvollen Vergleich zu erreichen sei Ω durch die folgenden Verfahren gegeben:

- Blindes Signaturverfahren Π: Blinde Schnorr-Signatur (vgl. Abschnitt 2.6.1),
- SignaturverfahrenΠ_1, Π_2: Schnorr-Signatur (vgl. Abschnitt 2.5.1),
- Verschlüsselungsverfahren Σ: ElGamal-Verschlüsselung (vgl. Abschnitt 2.3.2).

Protokoll	**Modulare Exponentiationen**					
	Instanz Ω		Frankel *et al.*.		Gaud *et al.*.	
	$\mathcal{K} + \mathcal{O}$	$\mathcal{B}$ bzw. $\mathcal{H}$	$\mathcal{K}$	$\mathcal{B}$ bzw. $\mathcal{H}$	$\mathcal{K}$	$\mathcal{B}$ bzw. $\mathcal{H}$
Abhebe-Protokoll Withdrawal:	7	3	25	19	16	8
Bezahl-Protokoll Payment:	5	4	2	9	-	9

Tabelle 7.1: Effizienzvergleich ausgewählter fairer elektronischer Geldsysteme

Tabelle 7.1 zeigt, dass man mit dem in diesem Abschnitt beschriebenen generischen Ansatz im Vergleich zu [FTY98] bzw. [GT03] besonders die Effizienz des Abhebe-Protokolls deutlich verbessern kann. Diese Effizienzsteigerung wird vor allem dadurch erreicht, dass die für das Münztracing relevanten Teile des Abhebe-Protokolls auf den Observer verlagert wurden. Auf diese Weise kann man aufwändige Zero-Knowledge-Beweise, wie sie in [FTY98] (die beiden nichtinteraktiven Beweise V_1, V_2 aus Abschnitt 6.1) und [GT03] verwendet werden, durch eine deutlich effizienter zu erstellende Signatur des Observers ersetzen (die Signatur s_1 aus Schritt 2 des Abhebe-Protokolls). Aus Tabelle 7.1 ergibt sich, dass das hier vorgestellte generische faire elektronische Geldsystem Ω in puncto Rechenaufwand ungefähr doppelt so effizient wie [GT03] und beinahe dreimal so effizient wie [FTY98] ist.
Abschließend bleibt festzuhalten, dass sich mit dem in diesem Abschnitt verfolgten Ansatz, die Observer vom Deanonymisierer an die Kunden auszugeben und wesentliche Teile des Abhebe- bzw. Bezahl-Protokolls auf den Observer zu verlagern, im Vergleich

zu anderen fairen Geldsystemen unter Annahme 7.1 effizientere Abhebe- bzw. Bezahl-Protokolle entwickeln lassen. Dabei beruht die Sicherheit des vorgestellten generischen Verfahrens Ω zum einen auf der Sicherheit der eingesetzten kryptografischen Bausteine (vgl. Abschnitt 7.1.7) und zum anderen auf der Manipulationssicherheit des Observers.

7.2 Gruppensignaturen mit Observern

Das Konzept der von Kim *et al.* vorgeschlagenen *Gruppensignaturen mit Observern* [KPW97] wurde schon in Abschnitt 7.1.6 angedeutet und es wurde gezeigt, dass sich Gruppensignaturen mit Observern auf Grund ihrer Einfachheit sehr gut als Baustein für das in Abschnitt 7.1 entwickelte generische faire elektronische Geldsystem eignen. Gleichzeitig wurde in Abschnitt 7.1.6 bereits auf die Schwächen dieses Ansatzes hingewiesen.
In diesem Abschnitt werden Gruppensignaturen mit Observern als Baustein des generischen fairen Geldsystems genauer analysiert. Um den angesprochenen Schwächen von [KPW97] Rechnung zu tragen, bzw. die Sicherheit des im vorangegangenen Abschnitt konstruierten generischen fairen elektronischen Geldsystems zu steigern, wird in diesem Abschnitt der Ansatz von Kim *et al.* beschrieben und ein verbessertes Sicherheitsmodell für Gruppensignaturen mit Observern vorgestellt. Anschließend wird in Abschnitt 7.3 ein neues Gruppensignaturverfahren mit Observern entwickelt, das den neuen Sicherheitsanforderungen genügt und zur Realisierung von Kundentracing im generischen fairen Geldsystem aus Abschnitt 7.1 eingesetzt werden kann.

7.2.1 Das Verfahren von Kim *et al.*

Wie bereits in Abschnitt 7.1.6 erwähnt, wurden Gruppensignaturen im Jahre 1991 von D. Chaum und E. van Heyst [CH91] vorgestellt und ermöglichen es Mitgliedern einer Gruppe, im Namen der Gruppe anonym Signaturen zu erstellen. Die Anonymität der Mitglieder kann bei Bedarf durch eine Trusted-Third-Party, dem Gruppenmanager, aufgehoben werden. Die von Kim *et al.* konzipierten Gruppensignaturen mit Observern stellen generell einen der einfachsten generischen Ansätze dar, mit Hilfe eines manipulationssicheren Speichermediums Gruppensignaturen zu erstellen und können grundsätzlich mit einem beliebigen Signaturverfahren Π und einem geeigneten probabilistischen Public-Key-Verschlüsselungsverfahren Σ realisiert werden.
Dem Verfahren liegen folgende Ideen zugrunde (vgl. Abbildung 7.4): Zur Initialisierung des Verfahrens wählt der Gruppenmanager $\mathcal{G}$ zunächst ein geeignetes Signaturverfahren $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ mit Sicherheitsparameter $k \in \mathbb{N}$ und ein geeignetes probabilistisches Public-Key-Verschlüsselungsverfahren $\Sigma = (\mathsf{Gen}_\Sigma, \mathsf{Enc}, \mathsf{Dec})$, ebenfalls mit Sicherheitsparameter k. Anschließend erzeugt der Gruppenmanager $\mathcal{G}$ die beiden Schlüsselpaare $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G}) = \mathsf{Gen}(1^k)$ und $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D}) = \mathsf{Gen}_\Sigma(1^k)$. Das Schlüsselpaar $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G})$ wird dazu verwendet, Gruppensignaturen zu erstellen bzw. diese zu verifizieren. Mit dem Schlüsselpaar $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D})$ wird die Identität der Gruppenmitglieder verschlüsselt bzw. entschlüsselt, damit $\mathcal{G}$ ggf. die Anonymität der Mitglieder aufheben kann.

Um sich als neues Gruppenmitglied bei $\mathcal{G}$ zu registrieren, erzeugt sich das neue Mitglied $\mathcal{M}$ ein Schlüsselpaar $(\mathsf{pk}_\mathcal{M}, \mathsf{sk}_\mathcal{M}) = \mathsf{Gen}(1^k)$. Der öffentliche Schlüssel $\mathsf{pk}_\mathcal{M}$ wird von $\mathcal{G}$ zusammen mit den persönlichen Daten von $\mathcal{M}$ in einer Datenbank hinterlegt. Im Anschluss daran speichert $\mathcal{G}$ das Schlüsselpaar $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G})$ und die öffentlichen Schlüssel $\mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{M}$ auf einem manipulationssicheren Speichermedium, dem Observer $\mathcal{O}$, und übergibt den Observer $\mathcal{O}$ an das neue Mitglied $\mathcal{M}$. Der öffentliche Schlüssel $\mathsf{pk}_\mathcal{M}$ dient als Identität von $\mathcal{M}$ und wird von $\mathcal{G}$ dazu verwendet, in bestimmten Verdachtsmomenten vorliegende Gruppensignaturen ihren Erstellern zuzuordnen. Da der private Schlüssel $\mathsf{sk}_\mathcal{G}$ auf dem Observer $\mathcal{O}$ gespeichert ist, kann das Mitglied $\mathcal{M}$ ohne seinen Observer $\mathcal{O}$ keine Gruppensignaturen erstellen.
Damit $\mathcal{M}$ im Namen der Gruppe Signaturen erstellen kann, wird das in Abbildung 7.4 dargestellte Protokoll zwischen $\mathcal{M}$ und $\mathcal{O}$ durchgeführt. Da $\mathcal{M}$ den Signaturschlüssel $\mathsf{sk}_\mathcal{G}$ nicht kennt, wird die Gruppensignatur $s = \mathsf{Sign}((m, c), \mathsf{sk}_\mathcal{G}, \mathsf{pk}_\mathcal{G})$ auf dem Observer $\mathcal{O}$ berechnet. Der Geheimtext $c = \mathsf{Enc}(\mathsf{pk}_\mathcal{D}, s_\mathcal{M})$, der neben der Nachricht m zur Berechnung der Signatur s verwendet wird, stellt sicher, dass $\mathcal{G}$ bei Bedarf die Anonymität der Mitglieder aufheben und die Signatur s ihrem Ersteller $\mathcal{M}$ zuordnen kann. Die von $\mathcal{M}$ berechnete Signatur $s_\mathcal{M}$ garantiert, dass weder der Gruppenmanager $\mathcal{G}$ noch andere Mitglieder im Namen von $\mathcal{M}$ Signaturen erstellen können.

7.2.2 Sicherheitsziele von Gruppensignaturen mit Observern

Die offensichtliche Schwachstelle der von Kim *et al.* vorgeschlagenen Gruppensignaturen mit Observern ist der auf allen Observern gespeicherte Signaturschlüssel $\mathsf{sk}_\mathcal{G}$. Sobald der Observer eines Mitglieds kompromittiert wird, können mit Hilfe des Signaturschlüssels $\mathsf{sk}_\mathcal{G}$ auch Nichtmitglieder Signaturen im Namen der Gruppe erstellen. Außerdem ist nicht mehr sichergestellt, dass der Gruppenmanager bei Bedarf eine vorliegende Gruppensignatur ihrem Ersteller zuordnen kann. Das heißt, die Sicherheit des gesamten Verfahrens ist im Falle einer Kompromittierung nicht mehr gewährleistet.
Um das Verfahren besser gegen eine mögliche Kompromittierung der Observer zu schützen, ist es naheliegend, anstelle des kompletten Signaturschlüssels $\mathsf{sk}_\mathcal{G}$ nur Teile von $\mathsf{sk}_\mathcal{G}$ auf den Observern zu speichern; dieser Ansatz soll im Weiteren verfolgt werden. Bevor wir auf die Sicherheitsziele von Gruppensignaturen mit Observern eingehen, werden Gruppensignaturverfahren mit Observern zunächst formal definiert:

Definition 7.2.1. Es sei $k \in \mathbb{N}$ der Sicherheitsparameter. Ein *Gruppensignaturverfahren mit Observern* besteht aus zwei probabilistischen, polynomiellen interaktiven Turing-Maschinen, dem Gruppenmanager $\mathcal{G}$ und dem Deanonymisierer $\mathcal{D}$, einer Menge von probabilistischen, polynomiellen interaktiven Turing-Maschinen M, den Gruppenmitgliedern und folgenden polynomiellen Algorithmen bzw. Protokollen:

1. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{GKeyGen}$ des Gruppenmanagers $\mathcal{G}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k den öffentlichen Gruppenschlüssel $\mathsf{pk}_\mathcal{G}$ und den privaten Schlüssel $\mathsf{sk}_\mathcal{G}$ erzeugt. Die Ausgabe von $\mathsf{GKeyGen}(1^k)$ ist das Schlüsselpaar $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G})$.

Initialisierung des Verfahrens:		
$\mathcal{G}(1^k)$		
Wähle $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ und $\Sigma = (\mathsf{Gen}_\Sigma, \mathsf{Enc}, \mathsf{Dec})$ $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G}) = \mathsf{Gen}(1^k)$ $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D}) = \mathsf{Gen}_\Sigma(1^k)$		

Registrierung eines neuen Mitglieds $\mathcal{M}$:		
$\mathcal{M}(1^k, \mathsf{pk}_\mathcal{G})$		$\mathcal{G}(\mathsf{pk}_\mathcal{G})$
$(\mathsf{pk}_\mathcal{M}, \mathsf{sk}_\mathcal{M}) = \mathsf{Gen}(1^k)$	$\xrightarrow{\mathsf{pk}_\mathcal{M}}$	Speichert $(\mathcal{M}, \mathsf{pk}_\mathcal{M})$
		Speichert $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G}), \mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{M}$ auf $\mathcal{O}$
	$\xleftarrow{\text{Übergebe } \mathcal{O} \text{ an } \mathcal{M}}$	

Signaturprotokoll zwischen $\mathcal{M}$ und $\mathcal{O}$:		
$\mathcal{M}(m, \mathsf{sk}_\mathcal{M}, \mathsf{pk}_\mathcal{M}, \mathsf{pk}_\mathcal{G})$		$\mathcal{O}(\mathsf{sk}_\mathcal{G}, \mathsf{pk}_\mathcal{G}, \mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{M})$
$s_\mathcal{M} = \mathsf{Sign}(m, \mathsf{sk}_\mathcal{M}, \mathsf{pk}_\mathcal{M})$	$\xrightarrow{m, s_\mathcal{M}}$	$\mathsf{Verify}(m, \mathsf{pk}, s_\mathcal{M})$
		$c = \mathsf{Enc}(\mathsf{pk}_\mathcal{D}, s_\mathcal{M})$
	$\xleftarrow{c, s}$	$s = \mathsf{Sign}((m, c), \mathsf{sk}_\mathcal{G}, \mathsf{pk}_\mathcal{G})$
$\mathsf{Verify}((m, c), \mathsf{pk}_\mathcal{G}, s)$		

Abbildung 7.4: Das Gruppensignaturverfahren mit Observern von Kim *et al.*

2. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{DKeyGen}$ des Deanonymisierers $\mathcal{D}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k und des öffentlichen Schlüssels $\mathsf{pk}_\mathcal{G}$ einen öffentlichen Schlüssel $\mathsf{pk}_\mathcal{D}$ und einen privaten Schlüssel $\mathsf{sk}_\mathcal{D}$ erzeugt. Die Ausgabe von $\mathsf{DKeyGen}(1^k, \mathsf{pk}_\mathcal{G})$ ist das Schlüsselpaar $(\mathsf{pk}_\mathcal{D}, \mathsf{sk}_\mathcal{D})$

3. Der *Schlüsselgenerierungsalgorithmus* $\mathsf{MKeyGen}$ des Mitgliedes $\mathcal{M}$ ist ein probabilistischer Algorithmus, der bei Eingabe des Sicherheitsparameters 1^k und des öffentlichen Schlüssels $\mathsf{pk}_\mathcal{G}$ einen öffentlichen Schlüssel $\mathsf{pk}_\mathcal{M}$ und einen privaten Schlüssel $\mathsf{sk}_\mathcal{M}$ erzeugt. Die Ausgabe von $\mathsf{MKeyGen}(1^k, \mathsf{pk}_\mathcal{G})$ ist das Schlüsselpaar $(\mathsf{pk}_\mathcal{M}, \mathsf{sk}_\mathcal{M})$

4. Das *Registrierungsprotokoll* Reg ist ein interkatives Protokoll, dass zwischen einem Mitglied $\mathcal{M}$ und dem Gruppenmanager $\mathcal{G}$ durchgeführt wird. Bei Eingabe des Schlüsselpaars $(\mathsf{pk}_\mathcal{M}, \mathsf{sk}_\mathcal{M})$ sendet $\mathcal{M}$ den öffentlichen Schlüssel $\mathsf{pk}_\mathcal{K}$ an $\mathcal{G}$. Der Gruppenmanager $\mathcal{G}$ erzeugt bei Eingabe von $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G})$ die beiden Schlüsselpaare $(\mathsf{sk}'_\mathcal{M}, \mathsf{pk}'_\mathcal{M})$ und $(\mathsf{sk}'_\mathcal{O}, \mathsf{pk}'_\mathcal{O})$. Das Paar $(\mathsf{pk}'_\mathcal{O}, \mathsf{sk}'_\mathcal{O})$ wird zusammen mit $\mathsf{pk}_\mathcal{G}, \mathsf{pk}_\mathcal{K}$ auf einem manipulationssicheren Medium, dem Observer $\mathcal{O}$, gespeichert. Der Observer

$\mathcal{O}$ und $(\mathsf{pk}'_{\mathcal{M}}, \mathsf{sk}'_{\mathcal{M}})$ werden von $\mathcal{G}$ an $\mathcal{M}$ übergeben.

5. Das *Signaturprotokoll* Sign ist ein interaktives Protokoll zwischen einem Mitglied $\mathcal{M}$ und dessen Observer $\mathcal{O}$. Es sind $m \in M$, $(\mathsf{pk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}})$, $(\mathsf{pk}'_{\mathcal{M}}, \mathsf{sk}'_{\mathcal{M}})$ und $\mathsf{pk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{G}}$ die Eingabe von $\mathcal{M}$ und $(\mathsf{pk}'_{\mathcal{O}}, \mathsf{sk}'_{\mathcal{O}})$ sowie $\mathsf{pk}_{\mathcal{G}}$ die Eingabe von $\mathcal{O}$. Die Ausgabe von $\mathcal{M}\,(m, \mathsf{pk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}}, \mathsf{pk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{G}})$ ist eine Signatur s oder $\perp$. Entsprechend ist die Ausgabe von $\mathcal{O}\,(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{G}})$ $\mathsf{completed}$ oder $\mathsf{not\ completed}$.

6. Der *Verifikationsalgorithmus* Verify ist ein deterministischer Algorithmus und erhält als Eingabe eine Nachricht m, den öffentlichen Schlüssel $\mathsf{pk}_{\mathcal{G}}$, sowie eine Signatur s. Die Ausgabe von $\mathsf{Verify}(m, \mathsf{pk}_{\mathcal{G}}, s)$ ist der Wert 1 (= true) oder 0 (= false).

7. Der *Identifikationsalgorithmus* $\mathsf{Identify}$ ist ein deterministischer Algorithmus und erhält als Eingabe ein Nachrichten-Signaturpaar (m, s), den Schlüssel $\mathsf{pk}_{\mathcal{G}}$ und das Schlüsselpaar $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$. Die Ausgabe von $\mathsf{Identify}\,(m, s, \mathsf{pk}_{\mathcal{G}}, \mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ ist die Identität $\mathsf{pk}_{\mathcal{M}}$ eines Mitglieds $\mathcal{M}$ oder $\perp$.

Die Sicherheitsanalyse von Gruppensignaturen mit Observern ist komplex und umfasst unterschiedliche Aspekte. Aus Sicht des Gruppenmanagers müssen Gruppensignaturen in Analogie zu „gewöhnlichen“ Signaturen unfälschbar sein, d.h. gültige Gruppensignaturen dürfen nur von registrierten Mitgliedern erstellt werden. Für die Mitglieder wiederum müssen die erstellten Signaturen, ähnlich wie elektronische Münzen, anonym und nichtverknüpfbar sein. Damit die Anonymität der Mitglieder von den Mitgliedern nicht missbraucht werden kann, fordert man zusätzlich, dass die Anonymität in bestimmten Verdachtsmomenten vom Deanonymisierer wieder aufgehoben werden kann. Außerdem dürfen weder der Gruppenmanager noch einzelne Mitglieder Gruppensignaturen im Namen anderer Mitglieder erstellen. Dies bedeutet insbesondere, dass der Gruppenmanager mit Hilfe der ihm vorliegenden privaten Schlüssel $\mathsf{sk}'_{\mathcal{M}}, \mathsf{sk}'_{\mathcal{O}}$ nicht im Namen von Mitglied $\mathcal{M}$ Signaturen erstellen kann. Genauso darf es auch einer Koalition aus n Mitgliedern nicht möglich sein im Namen anderer Mitglieder Nachrichten zu signieren.
Eine wesentliche Schwachstelle der von Kim *et al.* vorgestellten Gruppensignaturen mit Observern stellt das Risiko einer Kompromittierung der Observer dar. Da der Signaturschlüssel $\mathsf{sk}_{\mathcal{G}}$ der Gruppe auf jedem Observer gespeichert ist, kann ein Angreifer nach der Kompromittierung eines Observers beliebige Gruppensignaturen erstellen, deren korrekte Deanonymisierung durch den Deanonymisierer $\mathcal{D}$ nicht mehr gewährleistet ist. Das heißt, falls ein Mitglied $\mathcal{M}$ seinen Observer $\mathcal{O}$ kompromittiert, kann $\mathcal{M}$ nicht deanonymisierbare Gruppensignaturen erstellen. Um das Verfahren außerdem vor dem Risiko einer Kompromittierung der Observer zu schützen, fordern wir für die Sicherheit eines Gruppensignaturverfahrens mit Observern zusätzlich, dass ein Angreifer, der eine Menge von Observern $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ kompromittiert, anhand der auf den Observern gespeicherten privaten Schlüsseln $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ keine gültigen Gruppensignaturen erstellen kann. Man beachte, dass dem Angreifer lediglich die privaten Schlüssel $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ der Observer, nicht aber die entsprechenden privaten Schlüssel $\mathsf{sk}'_{\mathcal{M}_1}, \ldots, \mathsf{sk}'_{\mathcal{M}_n}$ der Mitglieder zu Verfügung stehen.

Unsere formale Definition der Sicherheit von Gruppensignaturen von Observern basiert auf [KPW97],[ACJT00] und umfasst zusätzlich die in diesem Abschnitt angesprochenen neuen Sicherheitsaspekte. Ein Gruppensignaturverfahren mit Observern heißt sicher, falls es folgender Definition genügt:

Definition 7.2.2. Ein Gruppensignaturverfahren mit Observern mit Sicherheitsparameter $k \in \mathbb{N}$ heißt *sicher*, falls es die folgenden Bedingungen erfüllt:

1. *Durchführbarkeit*: Für ein Nachrichten-Signaturpaar (m, s), das von einem registrierten Mitglied $\mathcal{M}$ mit Sign erstellt wurde, ist $\mathsf{Verify}(m, \mathsf{pk}_{\mathcal{G}}, s) = 1$.

2. *Unfälschbarkeit*: Nur registrierte Mitglieder können in Kooperation mit ihren Observern gültige Gruppensignaturen erstellen.

3. *Anonymität*: Es gibt keinen effizienten Angreifer $\mathcal{A}$, der bei Eingabe zweier öffentlicher Schlüssel $\mathsf{pk}_{\mathcal{M}_0}, \mathsf{pk}_{\mathcal{M}_1}$ sowie zwei von $\mathcal{M}_0$ und $\mathcal{M}_1$ erstellten Nachrichten-Signaturpaaren $(m_b, s_b), (m_{\bar{b}}, s_{\bar{b}})$ $\left(b, \bar{b} \in \{0,1\}, b \neq \bar{b}\right)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob (m_b, s_b) von $\mathcal{M}_0$ oder $\mathcal{M}_1$ erstellt wurde. Wobei ν eine in k nicht vernachlässigbare Funktion ist.

4. *Nichtverknüpfbarkeit*: Es gibt keinen effizienten Algorithmus $\mathcal{A}$, der bei Eingabe zweier Nachrichten-Signaturpaare $(m_0, s_0), (m_1, s_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob s_0 und s_1 vom selben Mitglied erstellt wurden. Wobei ν eine in k nicht vernachlässigbare Funktion ist.

5. *Identitätsbindung*: Weder ein einzelnes Mitglied $\mathcal{M}$ noch der Gruppenmanager $\mathcal{G}$ kann im Namen anderer Mitglieder Nachrichten signieren.

6. *Koalitionsresistenz*: Eine Koalition aus mehreren Mitgliedern kann keine gültige Gruppensignatur erstellen, die durch den Deanonymisierer nicht einem der koalierenden Mitgliedern zugeordnet werden kann.

7. *Kompromittierungsschutz*: Es gibt keinen effizienten Angreifer $\mathcal{A}$, der bei Eingabe des öffentlichen Gruppenschlüssels $\mathsf{pk}_{\mathcal{G}}$ und privaten Schlüsseln $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ mit einer in k nicht vernachlässigbaren Wahrscheinlichkeit $\epsilon(k)$ eine gültige Gruppensignatur ausgibt.

8. *Deanonymisierung*: Der Deanonymisierer kann anhand einer gültigen Gruppensignatur s, mit einer in k überwältigenden Wahrscheinlichkeit, die Identität des signierenden Mitglieds feststellen.

7.3 Ein neues Gruppensignaturverfahren mit Observern

In diesem Abschnitt wird ein neues Gruppensignaturverfahren mit Observern entwickelt, das den von uns neu definierten Sicherheitsanforderungen für Gruppensignaturen mit Observern genügt (vgl. Definition 7.2.2) und damit den bereits angesprochenen

Schwachstellen von [KPW97] entgegnet. Das neue Verfahren basiert im Wesentlichen auf der Schnorr-Signatur sowie der ElGamal-Verschlüssselung und eignet sich auf Grund seiner Eigenschaften als Baustein für das in Abschnitt 7.1 vorgeschlagene generische faire elektronische Geldsystem (Signaturverfahren Π_2).
Dem neuen Verfahren liegen folgende Ideen zugrunde: Mit dem Schlüsselgenerierungsalgorithmus der Schnorr-Signatur erzeugt sich der Gruppenmanager $\mathcal{G}$ zunächst das Schlüsselpaar $(\mathsf{pk}_\mathcal{G}, \mathsf{sk}_\mathcal{G}) = ((p, q, g, y, \mathrm{H}), x)$. Der private Schlüssel $\mathsf{sk}_\mathcal{G} = x$ wird als Signaturschlüssel der Gruppe verwendet. Damit nicht der komplette Signaturschlüssel x auf den von $\mathcal{G}$ ausgegebenen Observern gespeichert wird, teilt $\mathcal{G}$ den Signaturschlüssel x während der Registrierung eines neuen Mitglieds $\mathcal{M}$ in zwei Teilgeheimnisse $\mathsf{sk}'_\mathcal{M} = x_\mathcal{M}$ und $\mathsf{sk}'_\mathcal{O} = x_\mathcal{O}$. Dazu wählt $\mathcal{G}$ für jedes neue Mitglied $\mathcal{M}$ zufällig eine Zahl $x_\mathcal{M} \in_\mathcal{R} \mathbb{Z}_q$ und berechnet $x_\mathcal{O} = x - x_\mathcal{M}$. Das Teilgeheimnis $x_\mathcal{O}$ wird anschließend auf dem Observer $\mathcal{O}$ von $\mathcal{M}$ gespeichert. Um eine Nachricht im Namen der Gruppe zu signieren, muss $\mathcal{K}$ die Kenntnis des diskreten Logarithmus $\log_g y$ nachweisen, was $\mathcal{M}$ ohne den Observer $\mathcal{O}$ nicht kann. Mit der bereits in Abschnitt 6.3.2 angewendeten Methode kann das Signaturprotokoll so konstruiert werden, dass $\mathcal{M}$ nur in Kooperation mit $\mathcal{O}$ Signaturen erstellen kann und $\mathcal{O}$ das Teilgeheimnis $x_\mathcal{O}$ während des Protokolls nicht preisgeben muss.
Damit der Deanonymisierer $\mathcal{D}$ anhand einer gegebenen Gruppensignatur die Identität des signierenden Mitglieds $\mathcal{M}$ aufdecken kann, erstellt $\mathcal{M}$ während des Signaturprotokolls die ElGamal-Verschlüsselung $(A_1, A_2) = (Ig_2^t, f_2^t)$ ($I = g_1^{u_1}$ ist die Identität von $\mathcal{M}$ und f_2 der öffentliche Schlüssel von $\mathcal{D}$), die anschließend von $\mathcal{O}$ verifiziert wird. Nach erfolgreicher Verifikation erstellen $\mathcal{M}$ und $\mathcal{O}$ gemeinsam die Schnorr-Signatur, die $\mathcal{M}$ als registriertes Mitglied identifiziert. Da die ElGamal-Verschlüsselung (A_1, A_2) von $\mathcal{O}$ überprüft wird, ist für $\mathcal{D}$ sichergestellt, dass tatsächlich die Identität von $\mathcal{M}$ verschlüsselt wurde.
Um zu verhindern, dass der Gruppenmanager oder andere Gruppenmitglieder im Namen von $\mathcal{M}$ Gruppensignaturen erstellen können, beweist $\mathcal{M}$ mit der Signatur zusätzlich, dass $\mathcal{M}$ Darstellungen von A_1, A_2 bzgl. (g_1, g_2) bzw. f_2 kennt. Insgesamt besteht eine Gruppensignatur aus folgendem nichtinteraktiven Zero-Knowledge-Beweis, der von $\mathcal{M}$ und $\mathcal{O}$ gemeinsam erstellt wird:

$$\mathrm{NIZK}\left[(\alpha, \beta, \gamma) : y = g^\alpha \wedge A_1 = g_1^\beta g_2^\gamma \wedge A_2 = f_2^\gamma\right](m).$$

7.3.1 Initialisierung des Systems

Die Initialisierung des Systems besteht aus zwei Phasen. Zunächst führt der Gruppenmanager $\mathcal{G}$ den Schlüsselgenerierungsalgorithmus $\mathsf{GMKeyGen}$ aus, anschließend wird vom Deanonymisier $\mathcal{D}$ der Algorithmus $\mathsf{DMKeyGen}$ ausgeführt.

Gruppenmanager: Zunächst initialisiert der Gruppenmanager $\mathcal{G}$ durch $\mathsf{GKeyGen}(1^k)$ das System. Bei Eingabe des Sicherheitsparameters $k \in \mathbb{N}$ werden von $\mathcal{G}$ folgende Schritte ausgeführt:

Schritt 1: Der Gruppenmanager $\mathcal{G}$ wählt zwei Primzahlen p und q, so dass $|p-1| = \delta+k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.

Schritt 2: Der Gruppenmanager $\mathcal{G}$ bestimmt Generatoren g, g_1, g_2 der eindeutigen Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$. Weiter wählt $\mathcal{G}$ zufällig ein Element $x \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $y = g^x$. Es ist $\mathsf{sk}_{\mathcal{G}} = x$ der Signaturschlüssel der Gruppe.

Schritt 3: Der Gruppenmanager $\mathcal{G}$ wählt eine kollisionsresistente Hashfunktion $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$ und veröffentlicht $\mathsf{pk}_{\mathcal{G}} = (p, q, g, y, g_1, g_2, \mathrm{H})$ als öffentlichen Schlüssel der Gruppe.

Deanonymisierer: Die Initialisierung des Deanonymisierers $\mathcal{D}$ erfolgt durch den Algorithmus $\mathsf{DKeyGen}(1^k)$. Der Deanonymisierer $\mathcal{D}$ wählt zufällig einen Wert $x_{\mathcal{D}} \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $f_2 = g_2^{x_{\mathcal{D}}}$. Es ist $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}}) = (f_2, x_{\mathcal{D}})$.

Definition 7.3.1. Eine Gruppensignatur s für die Nachricht $m \in \{0,1\}^*$ ist ein Tupel $(A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$, für das $c' = \mathrm{H}(m)$ gilt und folgende Gleichungen für $c = \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$ erfüllt sind:

$$g^d = y^c r, \quad g_1^{d_1} g_2^{d_2} = A_1^c A_1', \quad f_2^{d_2} = A_2^c A_2'.$$

7.3.2 Registierung neuer Gruppenmitglieder

Um sich als neues Mitglied zu registrieren, generiert sich das Mitglied $\mathcal{M}$ mit dem Algorithmus $\mathsf{MKeyGen}(1^k, \mathsf{pk}_{\mathcal{G}})$ zunächst das Schlüsselpaar $(\mathsf{pk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}})$. Dazu wählt $\mathcal{M}$ zufällig einen Wert $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $I = g_1^{u_1}$. Es ist $(\mathsf{pk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}}) = (I, u_1)$. Anschließend führen der Gruppenmanager $\mathcal{G}$ und $\mathcal{M}$ das im Folgenden beschriebene Protokoll $\mathsf{Reg}\left(\mathcal{M}\left(\mathsf{pk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}}\right), \mathcal{G}\left(\mathsf{pk}_{\mathcal{G}}, \mathsf{sk}_{\mathcal{G}}\right)\right)$ aus:

Schritt 1: Der Gruppenmanager $\mathcal{G}$ wählt zufällig eine Zahl $x_{\mathcal{M}} \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $x_{\mathcal{O}} = x - x_{\mathcal{M}} \bmod q$. Außerdem berechnet $\mathcal{G}$ die Werte $y_{\mathcal{M}} = g^{x_{\mathcal{M}}}$ und $y_{\mathcal{O}} = g^{x_{\mathcal{O}}}$. Die Werte $x_{\mathcal{M}}, y_{\mathcal{M}}$ und $y_{\mathcal{O}}$ sendet $\mathcal{G}$ an $\mathcal{M}$ und speichert $x_{\mathcal{O}}$ und $y_{\mathcal{O}}$ auf einem Observer $\mathcal{O}$.

Schritt 2: Das Mitglied $\mathcal{M}$ sendet I an $\mathcal{G}$ und beweist $\mathcal{G}$, dass er $\log_{g_1} I$ kennt. Der Wert I ist die Identität von $\mathcal{M}$.

Schritt 3: Der Gruppenmanager speichert $(I, \mathcal{M})$ in seiner Datenbank und I zusätzlich auf dem Observer $\mathcal{O}$, der anschließend an $\mathcal{M}$ übergeben wird.

7.3.3 Erstellen und Verifizieren einer Gruppensignatur

Erstellen einer Gruppensignatur: Da das Mitglied $\mathcal{M}$ den Signaturschlüssel x der Gruppe nicht kennt, kann $\mathcal{M}$ eine Nachricht $m \in \{0,1\}^*$ nur mit Hilfe seines Observers

$\mathcal{O}$ signieren. Um eine Gruppensignatur für die Nachricht m zu generieren, erstellen $\mathcal{M}$ und $\mathcal{O}$ gemeinsam den nichtinteraktiven Zero-Knowledge-Beweis

$$\text{NIZK}\left[(\alpha, \beta, \gamma) : y = g^{\alpha} \wedge A_1 = g_1^{\beta} g_2^{\gamma} \wedge A_2 = f_2^{\gamma}\right](m).$$

Das Mitglied $\mathcal{M}$ und der Observer $\mathcal{O}$ führen zur Erstellung des Beweises das im Folgenden beschriebene Signaturprotokoll $\mathsf{Sign}(\mathcal{M}(m, x_{\mathcal{M}}, u_1, y_{\mathcal{O}}, \mathsf{pk}_{\mathcal{G}}), \mathcal{O}(x_{\mathcal{O}}, I, \mathsf{pk}_{\mathcal{G}}))$ aus (vgl. Abbildung 7.5).

$\mathcal{M}\,(m, x_{\mathcal{M}}, u_1, y_{\mathcal{O}}, \mathsf{pk}_{\mathcal{G}})$		$\mathcal{O}\,(x_{\mathcal{O}}, I, \mathsf{pk}_{\mathcal{G}})$
		$w_2 \in \mathbb{Z}_q$
		$\tilde{r} = g^{w_2}$
	$\xleftarrow{\tilde{r}}$	
$t, w_1, x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$		
$A_1 = I g_2^t, A_2 = f_2^t$		
$A_1' = g_1^{x_1} g_2^{x_2}, A_2' = f_2^{x_2}$		
$r = g^{w_1} \tilde{r}$		
$c' = \mathrm{H}(m)$		
$c = \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$	$\xrightarrow{t, r, c, c', A_1, A_2, A_1', A_2'}$	$A_1 \stackrel{?}{=} I g_2^t, A_2 \stackrel{?}{=} f_2^t$
		$c \stackrel{?}{=} \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$
		$\tilde{d} = c x_{\mathcal{O}} + w_2 \bmod q$
	$\xleftarrow{\tilde{d}}$	
$g^{\tilde{d}} \stackrel{?}{=} y_{\mathcal{O}}^c \tilde{r}$		
$d_1 = c u_1 + x_1 \bmod q$		
$d_2 = c t + x_2 \bmod q$		
$d = c x_{\mathcal{M}} + w_1 + \tilde{d} \bmod q$		

Abbildung 7.5: Erstellen einer Gruppensignatur

Schritt 1: Der Observer $\mathcal{O}$ wählt einen zufälligen Wert $w_2 \in \mathbb{Z}_q$, berechnet das Commitment $\tilde{r} = g^{w_2}$ und sendet $\tilde{r}$ an $\mathcal{M}$.

Schritt 2: Das Mitglied $\mathcal{M}$ wählt zufällige Werte $t, w_1, x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet die ElGamal-Verschlüsselung $(A_1, A_2) = (I g_2^t, f_2^t)$ seiner Identität I. Anschließend berechnet $\mathcal{M}$ die Commitments $A_1' = g_1^{x_1} g_2^{x_2}, A_2' = f_2^{x_2}, r = g^{w_1} \tilde{r}$ sowie die Hashwerte $c' = \mathrm{H}(m)$ und $c = \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$ und sendet $m, t, r, c, c', A_1, A_2, A_1', A_2'$ an $\mathcal{O}$.

Schritt 3: Der Observer $\mathcal{O}$ überprüft anhand von t, ob (A_1, A_2) eine ElGamal-Verschlüsselung von I unter f_2 ist und verfiziert, ob $c = \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$ gilt. Gelingt die Verifikation, berechnet $\mathcal{O}$ den Wert $\tilde{d} = c x_{\mathcal{O}} + w_2 \bmod q$ und sendet $\tilde{d}$ an $\mathcal{M}$. Andernfalls bricht er das Protokoll ab.

Schritt 4: Das Mitglied $\mathcal{M}$ verifiziert die Gleichung $g^{\tilde{d}} = y_{\mathcal{O}}^{c} r$ und berechnet die folgenden Werte: $d_1 = cu_1 + x_1 \bmod q, d_2 = ct + x_2 \bmod q$ sowie $d = cx_{\mathcal{M}} + w_1 + \tilde{d} \bmod q$.

Dabei handelt es sich bei $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ um eine gültige Gruppensignatur der Nachricht m (vgl. Lemma 7.3.1).

Verifizieren einer Gruppensignatur: Mit Hilfe von $\mathsf{Verify}\,(m, \mathsf{pk}_{\mathcal{G}}, s)$ kann die Gültigkeit einer Gruppensignatur s überprüft werden. Der Algorithmus Verify gibt true aus, falls das Nachrichten-Signaturpaar (m, s) Definition 7.3.1 genügt und false sonst.

7.3.4 Deanonymisieren eines Mitglieds

In bestimmten Verdachtsmomenten kann der Gruppenmanager $\mathcal{G}$ anhand einer ihm vorliegenden Signatur $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ der Nachricht m mit Hilfe des Deanonymisierers $\mathcal{D}$ die Identität des signierenden Mitglieds $\mathcal{M}$ feststellen. Mit dem Algorithmus $\mathsf{Identify}(m, s, \mathsf{pk}_{\mathcal{G}}, \mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$ kann $\mathcal{D}$ die Identität wie folgt aufdecken:

Schritt 1: Der Deanonymisierer $\mathcal{D}$ entschlüsselt die in s enthaltene ElGamal-Verschlüsselung (A_1, A_2) zu $A_1 A_2^{-x_{\mathcal{D}}^{-1}} = I g_2^t (f_2^t)^{-x_{\mathcal{D}}^{-1}} = I g_2^t g_2^{-t} = I$.

Schritt 2: Der Deanonymisierer $\mathcal{D}$ sendet $I = \mathsf{pk}_{\mathcal{M}}$ an $\mathcal{G}$, der mittels $\mathsf{pk}_{\mathcal{M}}$ das Mitglied $\mathcal{M}$ in seiner Datenbank finden kann.

7.3.5 Sicherheitsanalyse

Unfälschbarkeit, Anonymität und Koalitionsresistenz sind wichtige Sicherheitsmerkmale von Gruppensignaturen mit Observern. Ziel der Sicherheitsanalyse ist es, zu zeigen, dass das in diesem Abschnitt vorgestellte Gruppensignaturverfahren mit Observern Π ein gemäß Definition 7.2.2 sicheres Gruppensignaturverfahren mit Observern ist. Für die Sicherheitsanalyse entscheidend ist, dass die vom Gruppenmanager ausgegebenen Observer manipulationssicher sind.

Annahme 7.2. Der vom Gruppenmanager $\mathcal{G}$ an das Mitglied $\mathcal{M}$ ausgegebene Observer $\mathcal{O}$ ist manipulationssicher. Insbesondere kann $\mathcal{M}$ den Speicherinhalt von $\mathcal{O}$ nicht auslesen.

Durchführbarkeit: Folgendes Lemma zeigt, dass die von einem Mitglied mit dem Signaturprotokoll Sign erstellten Gruppensignaturen gültige Signaturen sind.

Lemma 7.3.1. Folgen das Mitglied $\mathcal{M}$ und der Observer $\mathcal{O}$ dem in Abbildung 7.5 dargestellten Protokoll, so ist $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ eine gültige Gruppensignatur der Nachricht $m \in \{0,1\}^*$.

Beweis. Für das Tupel $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ sind folgende Gleichungen erfüllt, falls $\mathcal{M}$ und $\mathcal{O}$ dem Signaturprotokoll folgen:

$$\begin{aligned} c' &= \mathrm{H}(m) \\ g^d &= g^{cx_{\mathcal{M}}+w_1+\tilde{d}} = (g^{x_{\mathcal{M}}})^c g^{w_1} (g^{x_{\mathcal{O}}})^c g^{w_2} = (g^{x_{\mathcal{M}}+x_{\mathcal{O}}})^c g^{w_1+w_2} = y^c r, \\ g_1^{d_1} g_2^{d_2} &= g_1^{cu_1+x_1} g_2^{ct+x_2} = (I g_2^t)^c g_1^{x_1} g_2^{x_2} = A_1^c A_1', \\ f_2^{d_2} &= f_2^{ct+x_2} = (f_2^t)^c f_2^{x_2} = A_2^c A_2'. \end{aligned}$$

Somit ist s gemäß Definition 7.3.1 eine gültige Gruppensignatur der Nachricht m. □

Unfälschbarkeit: Wir müssen zeigen, dass nur registrierte Mitglieder in Kooperation mit ihren Observern gültige Gruppensignaturen erstellen können. Insbesondere können Nichtmitglieder keine Gruppensignaturen erstellen.

Lemma 7.3.2. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$. Im Random-Oracle-Modell ist Π unter Annahme 7.2 gemäß Definition 7.2.2 unfälschbar.

Beweis. Da die Mitglieder den diskreten Logarithmus $\log_g y$ unter Annahme 7.2 nicht kennen, können sie nur in Kooperation mit ihren Observern Gruppensignaturen erstellen. Dass Nichtmitglieder keine Gruppensignaturen erstellen können, folgt direkt aus der Sicherheit der Schnorr-Signatur (vgl. Satz 2.5.1). □

Anonymität: Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der bei Eingabe zweier öffentlicher Schlüssel $\mathsf{pk}_{\mathcal{M}_0}, \mathsf{pk}_{\mathcal{M}_1}$ sowie zwei von $\mathcal{M}_0$ und $\mathcal{M}_1$ erstellten Nachrichten-Signaturpaaren $(m_b, s_b), (m_{\bar{b}}, s_{\bar{b}})$ $\left(b, \bar{b} \in \{0,1\}, b \neq \bar{b}\right)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob (m_b, s_b) von $\mathcal{M}_0$ oder $\mathcal{M}_1$ erstellt wurde, wobei ν eine nicht vernachlässigbare Funktion ist.

Lemma 7.3.3. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der die Anonymität von Π gemäß Definition 7.2.2 brechen kann.

Beweis. Angenommen, es gibt einen effizienten Angreifer $\mathcal{A}$, der die Anonymität von Π mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ brechen kann, wobei ν eine nicht vernachlässigbare Funktion ist. Dann kann $\mathcal{A}$ von einem Angreifer $\mathcal{A}'$ dazu verwendet werden, die ElGamal-Verschlüsselung im Sinne der polynomiellen Ununterscheidbarkeit zu brechen. Der Angreifer $\mathcal{A}'$ versucht folgende Instanz des ElGamal-Verschlüsselungsverfahrens im Sinne der polynomiellen Ununterscheidbarkeit zu brechen:

- Primzahlen p, q mit $|p-1| = \delta + k$ für ein $\delta \in \mathbb{N}$ und $p = \gamma q + 1$ für ein $\gamma \in \mathbb{N}$.
- Generator g_2 der eindeutig bestimmten Untergruppe G_q der Ordnung q von $\mathbb{Z}_p^*$.

- $(\mathsf{pk}, \mathsf{sk}) = ((p, q, g_2, f_2), x_{\mathcal{D}})$ mit $f_2 = g_2^{x_{\mathcal{D}}}$ und $x_{\mathcal{D}} \in \mathbb{Z}_q$.

Um die Instanz $(\mathsf{pk}, \mathsf{sk})$ der ElGamal-Verschlüsselung im Sinne der polynomiellen Ununterscheidbarkeit zu brechen, wählt $\mathcal{A}'$ zufällig $\delta \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnet $g_1 = g_2^{\delta}$. Weiter wählt $\mathcal{A}'$ zufällige Zahlen $u_0, u_1 \in \mathbb{Z}_q$ und berechnet die Nachrichten $m_0 = g_1^{u_0}, m_1 = g_1^{u_1}$. Es seien $A^{(0)} = (A_1^{(0)}, A_2^{(0)}) = (m_b g_2^{t_b}, f_2^{t_b})$ und $A^{(1)} = (A_1^{(1)}, A_2^{(1)}) = (m_{\bar{b}} g_2^{t_{\bar{b}}}, f_2^{t_{\bar{b}}})$ die ElGamal-Verschlüsselungen der beiden Nachrichten in einer durch $b \in_{\mathcal{R}} \{0, 1\}$ zufällig gewählten Reihenfolge, wobei $t_b, t_{\bar{b}} \in \mathbb{Z}_q$ und b dem Angreifer $\mathcal{A}'$ nicht bekannt ist.
Anhand der beiden Nachrichten m_0, m_1 und der beiden ElGamal-Verschlüsselungen $A^{(0)}, A^{(1)}$ versucht $\mathcal{A}'$ zu entscheiden, welche Verschlüsselung zu welcher Nachricht gehört. Um $\mathcal{A}$ dafür zu verwenden, übersetzt $\mathcal{A}'$ die beiden Verschlüsselungen $A^{(0)}, A^{(1)}$ wie folgt in eine Instanz von Π:

Schritt 1: Der Angreifer $\mathcal{A}'$ wählt zufällig $x \in_{\mathcal{R}} \mathbb{Z}_q$, $g \in_{\mathcal{R}} G_q$ und berechnet $y = g^x$.

Schritt 2: Der Angreifer setzt $\mathsf{pk}_{\mathcal{M}_0} = m_0$ und $\mathsf{pk}_{\mathcal{M}_1} = m_1$, wählt Nachrichten $n_b, n_{\bar{b}} \in \{0, 1\}^*$ und kann die beiden Nachrichten-Signaturpaare $(n_b, s_b), (n_{\bar{b}}, s_{\bar{b}})$ im Random-Oracle-Modell wie folgt perfekt simulieren.

Simulation (n_b, s_b):

- Wähle zufällige Werte $c^{(b)}, c'^{(b)}, d^{(b)}, d_1^{(b)}, d_2^{(b)} \in \mathbb{Z}_q$.
- Berechne $r^{(b)} = g^{d^{(b)}} y^{-c^{(b)}}$.
- Berechne $(A_1')^{(b)} = g_1^{d_1^{(b)}} g_2^{d_2^{(b)}} (A_1^{(b)})^{-c^{(b)}}$ und $(A_2')^{(b)} = f_2^{d_2^{(b)}} (A_2^{(b)})^{-c^{(b)}}$.
- Ersetze $\mathrm{H}(A_1^{(b)}, A_2^{(b)}, (A_1')^{(b)}, (A_2')^{(b)}, c'^{(b)}, r^{(b)})$ durch $c^{(b)}$ und $\mathrm{H}(n_b)$ durch $c'^{(b)}$.

Das Nachrichten-Signaturpaar $(n_{\bar{b}}, s_{\bar{b}})$ lässt sich analog simulieren.

Schritt 3: Der Angreifer $\mathcal{A}'$ übergibt $\mathsf{pk}_{\mathcal{M}_0}, \mathsf{pk}_{\mathcal{M}_1}$ und die beiden Nachrichten-Signaturpaare $(n_b, s_b), (n_{\bar{b}}, s_{\bar{b}})$ an $\mathcal{A}$.

Mit der Wahrscheinlichkeit $\epsilon(k)$ gibt $\mathcal{A}$ das korrekte Bit b aus. Der Angreifer $\mathcal{A}'$ kann daher mit der gleichen Wahrscheinlichkeit $\epsilon(k)$ entscheiden, welcher der beiden Klartexte m_0 und m_1 in $A^{(0)}$ verschlüsselt wurde. Dies ist aber ein Widerspruch zur polynomiellen Ununterscheidbarkeit der ElGamal-Verschlüsselung. □

Nichtverknüpfbarkeit: Wir müssen zeigen, dass es keinen effizienten Algorithmus $\mathcal{A}$, der bei Eingabe zweier Nachrichten-Signaturpaare $(m_0, s_0), (m_1, s_1)$ mit einer Wahrscheinlichkeit $\epsilon(k) \geq \frac{1}{2} + \nu(k)$ entscheiden kann, ob σ_0 und σ_1 vom selben Mitglied erstellt wurden. Dabei ist ν eine nicht vernachlässigbare Funktion.

Lemma 7.3.4. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$. Dann gibt es im Random-Oracle-Modell unter der Decisional-Diffie-Hellman-Annahme keinen effizienten Angreifer, der die Nichtverknüpfbarkeit von Π gemäß Definition 7.2.2 brechen kann.

Beweis. In [Tsi97] wurde gezeigt, dass Anonymität (wie sie in Definition 7.2.2 definiert ist) Nichtverknüpfbarkeit impliziert. Somit folgt aus Lemma 7.3.3 auch die Nichtverknüpfbarkeit. □

Identitätsbindung: Wir müssen zeigen, dass weder ein einzelnes Mitglied noch der Gruppenmanager im Namen eines anderen Mitglieds Signaturen ausstellen kann. Im Falle des Gruppenmanagers bedeutet dies, dass der Gruppenmanager auch mit Hilfe der von ihm generierten Teilgeheimnisse $\mathsf{sk}'_{\mathcal{M}}, \mathsf{sk}'_{\mathcal{O}}$ nicht im Namen des Mitglieds $\mathcal{M}$ Signaturen generieren kann.

Lemma 7.3.5. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$. Unter Annahme 7.2 ist Π gemäß Definition 7.2.2 identitätsbindend.

Beweis. Gegeben sei die Identität $I = g_1^{u_1}$ eines Mitglieds $\mathcal{M}$ und der Observer $\mathcal{O}$ von $\mathcal{M}$. Es reicht zu zeigen, dass der Gruppenmanager $\mathcal{G}$ nicht im Namen des Mitglieds $\mathcal{M}$ signieren kann. Angenommen $\mathcal{G}$ kann eine gültige Gruppensignatur im Namen von $\mathcal{M}$ erstellen. Dann kennt $\mathcal{G}$ ein Tupel

$$s = (A_1, A_2, A'_1, A'_2, c', r, d, d_1, d_2),$$

das Definition 7.3.1 genügt. Da das Mitglied $\mathcal{M}$ unter Annahme 7.2 nur mit Hilfe des Observers $\mathcal{O}$ gültige Gruppensignaturen erstellen kann, muss die Identität I von $\mathcal{M}$ in der ElGamal-Verschlüsselung (A_1, A_2) enthalten sein. Folglich ist

$$(A_1, A_2) = (I g_2^t, f_2^t)$$

für ein zufällig gewähltes $t \in_{\mathcal{R}} \mathbb{Z}_q$. Als Gruppenmanager kennt $\mathcal{G}$ den privaten Schlüssel x der Gruppe und kann daher auch $r, d \in \mathbb{Z}_q$ mit

$$g^d = y^{\mathrm{H}(A_1, A_2, A'_1, A'_2, c', r)} r$$

bestimmen. Mit den Gleichungen

$$\begin{aligned} g_1^{d_1} g_2^{d_2} &= A_1^{\mathrm{H}(A_1, A_2, A'_1, A'_2, c', r)} A'_1, \\ f_2^{d_2} &= A_2^{\mathrm{H}(A_1, A_2, A'_1, A'_2, c', r)} A'_2. \end{aligned}$$

zeigt $\mathcal{M}$, dass (A_1, A_2) eine ElGamal-Verschlüsselung von I unter f_2 ist. Insbesondere weisst $\mathcal{M}$ dadurch die Kenntnis einer Darstellung von A_1 bzgl. (g_1, g_2) nach. Die interaktive Variante dieses Beweises ist aber ein Honest-Verifier-Zero-Knowledge-Beweis. Das heißt, für die erfolgreiche Durchführung dieses Beweises muss $\mathcal{M}$ den diskreten Logarithmus $\log_{g_1} I = u_1$ kennen. Um eine gültige Gruppensignatur im Namen von $\mathcal{M}$ zu erzeugen, muss demnach auch $\mathcal{G}$ den diskreten Logarithmus $\log_{g_1} I$ kennen, im Widerspruch zur Diskreter-Logarithmus-Annahme (vgl. Annahme 2.1).
Der Besitz des Observers $\mathcal{O}$ von $\mathcal{M}$ hilft $\mathcal{G}$ nicht weiter, da auf dem Observer nach der Registrierung $x_{\mathcal{O}}$ und I gespeichert sind, aber nicht $\log_{g_1} I = u_1$. Sind zusätzlich noch die

von $\mathcal{O}$ während mehrerer Signaturprotokolle zwischen $\mathcal{M}$ und $\mathcal{O}$ erhaltenen Daten auf $\mathcal{O}$ gespeichert, so kann $\mathcal{G}$ aus diesen Daten lediglich die Identität I von $\mathcal{M}$ berechnen, denn während des Signaturprotokolls erhält $\mathcal{O}$ von $\mathcal{M}$ eine ElGamal-Verschlüsselung (A_1, A_2) von I. Die restlichen Werte, die $\mathcal{O}$ von $\mathcal{M}$ erhält, sind zufällig gewählte Werte. Somit kann $\mathcal{G}$ aus diesen Werten keinerlei Rückschlüsse auf $\log_{g_1} I = u_1$ machen. □

Koalitionsresistenz: Wir müssen zeigen, dass selbst eine Koalition aus mehreren Mitgliedern nicht in der Lage ist, eine Gruppensignatur zu erstellen, die vom Deanonymisierer nicht einem der koalierenden Mitglieder zugeordnet werden kann.

Lemma 7.3.6. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$ sowie die Mitglieder $\mathcal{M}_1, \ldots, \mathcal{M}_n$. Unter Annahme 7.2 kann eine Koalition aus $\mathcal{M}_1, \ldots, \mathcal{M}_n$ zusammen keine gültige Gruppensignatur erstellen, die durch den Deanonymisierer nicht einem der koalierenden Mitglieder $\mathcal{M}_1, \ldots, \mathcal{M}_n$ zugeordnet werden kann.

Beweis. Ziel der Mitglieder $\mathcal{M}_1, \ldots, \mathcal{M}_n$ ist es, eine Gruppensignatur s zu produzieren, die vom Deanonymisierer $\mathcal{D}$ keinem der koalierenden Mitglieder zugeordnet werden kann. Sicher können $\mathcal{M}_1, \ldots, \mathcal{M}_n$ ein Tupel $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ erstellen, für das die beiden folgenden Gleichungen erfüllt sind:

$$g_1^{d_1} g_2^{d_2} = A_1^{\mathrm{H}(A_1, A_2, A_1', A_2', c', r)} A_1',$$
$$f_2^{d_2} = A_2^{\mathrm{H}(A_1, A_2, A_1', A_2', c', r)} A_2'.$$

Dazu wählen sie $\tilde{u}_1, t, w_1, x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$ und berechnen die ElGamal-Verschlüsselung $A_1 = g^{\tilde{u}_1} g_2^t$, $A_2 = f_2^t$ der von ihnen gewählten Identität $\tilde{I} = g_1^{\tilde{u}_1}$ sowie $A_1' = g_1^{x_1} g_2^{x_2}$, $A_2' = f_2^{x_2}$, $r = g^{w_1}$, $c' = \mathrm{H}(m)$ und $c = \mathrm{H}(A_1, A_2, A_1', A_2', c', r)$. Weiter berechnen sie $d_1 = c\tilde{u}_1 + x_1 \bmod q$ und $d_2 = ct + x_2 \bmod q$. Dann gilt:

$$\begin{aligned} g_1^{d_1} g_2^{d_2} &= A_1^c A_1', \\ f_2^{d_2} &= A_2^c A_2'. \end{aligned}$$

Damit s eine gültige Gruppensignatur ist, muss zusätzlich die Gleichung

$$g^d = y^{\mathrm{H}(A_1, A_2, A_1', A_2', c', r)} r$$

erfüllt sein. Das heißt, die Mitglieder müssen die Kenntnis des disreten Logarithmus $\log_g y$ nachweisen. Unter Annahme 7.2 kennen sie $\log_g y$ nicht und sind gezwungen einen ihrer Observer zum Erstellen der Signatur zu verwenden. Dies impliziert aber, dass sie eine korrekte Verschlüsselung einer ihrer Identitäten in den Hashwert einfließen lassen müssen. □

Kompromittierungsschutz: Wir müssen zeigen, dass es keinen effizienten Angreifer gibt, der bei Eingabe des öffentlichen Gruppenschlüssels $\mathsf{pk}_{\mathcal{G}}$ und den privaten Schlüsseln $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ eine gültige Gruppensignatur ausgibt. Wir zeigen zunächst, dass ein Angreifer, der den Observer $\mathcal{O}$ eines Mitglieds $\mathcal{M}$ kompromittiert, durch die Kompromittierung von $\mathcal{O}$ keine zusätzliche Information gewinnt.

Lemma 7.3.7. Gegeben sei das Gruppensignaturverfahren mit Observern Π mit Sicherheitsparameter $k \in \mathbb{N}$. Weiter sei $\mathcal{M}$ ein Mitglied, $\mathcal{O}$ der Observer von $\mathcal{M}$ und $\mathcal{A}$ ein effizienter Angreifer. Im Random-Oracle-Modell erhält $\mathcal{A}$ unter der Voraussetzung, dass $\mathcal{A}$ und $\mathcal{M}$ nicht kooperieren, durch die Kompromittierung von $\mathcal{O}$ keine zusätzliche Information.

Beweis. Wir zeigen, dass sich der Inhalt eines Observers $\mathcal{O}$ im Random-Oracle-Modell perfekt simulieren lässt. Nach der Registrierung von $\mathcal{M}$ sind auf $\mathcal{O}$ die Werte $x_{\mathcal{O}} \in \mathbb{Z}_q$ und $I = g_1^{u_1}$. Da $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$ von $\mathcal{M}$ zufällig gewählt wurde, ist auch $I \in_{\mathcal{R}} G_q$ zufällig gewählt. Weiter ist $x_{\mathcal{O}} = x - x_{\mathcal{M}}$ mit $x_{\mathcal{M}} \in_{\mathcal{R}} \mathbb{Z}_q$, also ist auch $x_{\mathcal{O}} \in_{\mathcal{R}} \mathbb{Z}_q$ zufällig.

Simulation der Registrierung:

1. Wähle $x_{\mathcal{O}}, u_1 \in_{\mathcal{R}} \mathbb{Z}_q$.
2. Setze $I = g_1^{u_1}$.

Während des Signaturprotokolls erhält $\mathcal{O}$ von $\mathcal{M}$ folgende Daten: $A_1, A_2, A_1', A_2', r, t, c, c'$. Da $t, x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$ zufällig gewählt wurden, sind auch A_1, A_2 und A_1', A_2' zufällig gewählt. Der Wert r berechnet sich im Protokoll als $r = g^{w_1}\tilde{r}$ mit zufällig gewähltem $w_1 \in \mathbb{Z}_q$, also ist auch r zufällig gewählt.

Simulation des Signaturprotokolls:

1. Wähle $t \in_{\mathcal{R}} \mathbb{Z}_q$ und berechne $A_1 = Ig_2^t, A_2 = f_2^t$.
2. Wähle $x_1, x_2, w, c, c' \in_{\mathcal{R}} \mathbb{Z}_q$.
3. Berechne $A_1' = g_1^{x_1} g_2^{x_2}, A_2' = f_2^x, r = g^w$.
4. Ersetze $\mathrm{H}(A_1, A_2, A_1', A_2', c', r)$ durch c und $\mathrm{H}(m)$ durch c'.

Womit die Simulation im Random-Oracle-Modell abgeschlossen ist. □

Aus Lemma 7.3.7 folgt, dass ein Angreifer durch die privaten Schlüssel $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ keine zusätzliche Information erhält. Damit ist ein Angreifer, der $\mathsf{pk}_{\mathcal{G}}$ und $\mathsf{sk}'_{\mathcal{O}_1}, \ldots, \mathsf{sk}'_{\mathcal{O}_n}$ als Eingabe erhält mit einem Angreifer gleichzusetzen, der lediglich $\mathsf{pk}_{\mathcal{G}}$ kennt. Das heißt, der Kompromittierungsschutz folgt direkt aus der Unfälschbarkeit von Π.

Deanonymisierung: Wir müssen zeigen, dass der Deanonymisierer mittels einer gültigen Gruppensignatur die Identität des Signierers feststellen kann.

Lemma 7.3.8. Unter Annahme 7.2 kann der Deanonymisier $\mathcal{D}$ anhand einer Gruppensignatur $s = (A_1, A_2, A_1', A_2', c', r, d, d_1, d_2)$ die Identität des signierenden Mitglieds korrekt aufdecken.

Beweis. Ist s eine gültige Gruppensignatur, so folgt aus den Gleichungen

$$g_1^{d_1} g_2^{d_2} = A_1^{\mathrm{H}(A_1,A_2,A_1',A_2',c',r)} A_1',$$
$$f_2^{d_2} = A_2^{\mathrm{H}(A_1,A_2,A_1',A_2',c',r)} A_2',$$

dass (A_1, A_2) eine ElGamal-Verschlüsselung unter dem öffentlichen Schlüssel f_2 von $\mathcal{D}$ ist. Da $\mathcal{M}$ unter Annahme 7.2 nur mit Hilfe von $\mathcal{O}$ signieren kann, muss $\mathcal{M}$ seine Identität I verschlüsseln, denn andernfalls würde $\mathcal{O}$ die Kommunikation mit $\mathcal{M}$ abbrechen. Das heißt, durch den Observer $\mathcal{O}$ wird sichergestellt, dass $\mathcal{M}$ die richtige Identität I verschlüsselt. □

Insgesamt folgt aus den Lemmata 7.3.1 bis 7.3.8 der folgende Satz.

Satz 7.3.9. Das in diesem Abschnitt vorgestellte Gruppensignaturverfahren mit Observern ist im Random-Oracle-Modell unter Annahme 7.2 beweisbar sicher im Sinne von Definition 7.2.2.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

In fairen elektronischen Geldsystemen begünstigen Sicherheitslücken in den Systemen der Bank und des Deanonymisierers die Kompromittierung der privaten Schlüssel beider Parteien, woraus sich weitreichende Konsequenzen für die Sicherheit der betroffenen Geldsysteme ergeben. Die vorliegende Arbeit stellt einen neuen Sicherheitsbegriff für elektronische bzw. faire elektronische Geldsysteme vor: Die langfristig sicheren elektronischen bzw. langfristig sicheren fairen elektronischen Geldsysteme. Im Gegensatz zu klassischen Geldsystemen werden die privaten Schlüssel der Bank und des Deanonymisierers in langfristig sicheren Geldsytemen in regelmäßigen Intervallen aktualisiert. Ein Angreifer darf in t der insgesamt N Intervalle die privaten Schlüssel der Bank bzw. des Deanonymisierers kompromittieren, ohne dass sich dies negativ auf die Sicherheit des Systems in den restlichen $N - t$ Intervallen auswirkt.
Die kryptografischen Grundlagen und Bausteine, der in dieser Arbeit entwickelten langfristig sicheren Geldsysteme, wurden in Kapitel 2 dargestellt und konstruiert. Unter anderem wurden mehrere Zero-Knowledge-Beweise als Baustein für die in den Abschnitten 6.2 und 6.3 entwickelten langfristig sicheren Geldsysteme entworfen.
Ein Überblick über den aktuellen Stand der Forschung im Bereich elektronischer Geldsysteme und eine formale Definition dieser wurde in Kapitel 3 gegeben. Die bereits angesprochenen Konsequenzen einer Kompromittierung der Bank bzw. des Deanonymisierers wurden in Kapitel 4 ausführlich diskutiert. Bestehende Lösungsansätze aus anderen Bereichen der Kryptografie wurden hinsichtlich ihrer Integrierbarkeit in das von Frankel *et al.* vorgeschlagene faire Geldsystem (*FOLC*) [FTY98] geprüft. Dabei wurde deutlich, dass sich das Key-Insulation-Modell auf Grund der in [DKXY03] vorgeschlagenen Methoden ausgezeichnet eignet.
In Kapitel 5 wurde zunächst das Sicherheitsmodell langfristig sicherer blinder Signaturen vorgestellt, um darauf aufbauend in den Abschnitten 5.3 und 5.4 ein formales Sicherheitsmodell langfristig sicherer (fairer) elektronischer Geldsysteme zu definieren. Gleichzeitig konnte in Abschnitt 5.3.2 durch die Konstruktion eines generischen langfristig sicheren Geldsystems gezeigt werden, dass der Entwurf langfristig sicherer Geldsysteme auf Ba-

sis sicherer Geldsysteme möglich ist. Verglichen mit dem Basissystem verdoppelt sich allerdings der zur Verifikation einer Münze benötigte Rechenaufwand. Auf Grund ihrer Intervallaufteilung haben langfristig sichere Geldsysteme generell den Vorteil, dass sie es der Bank auf einfachste Art und Weise ermöglichen kurzlebige (zeitlich eingeschränkt gültige) Münzen auszugeben, was bisher nur mit partiell blinden Signaturen und zusätzlichem Rechenaufwand möglich war. Kurzlebige Münzen sind nicht nur als Zahlungsmittel denkbar, sie können auch in weiteren Anwendungen wie beispielsweise anonymisierten Prepaid-Services dazu eingesetzt werden, Kunden den Erwerb eines zeitlich begrenzten Zugriffs auf Inhalte oder Services des Anbieters zu ermöglichen.
Aufbauend auf *FOLC* wurde in Abschnitt 6.2 ein langfristig sicheres faires elektronisches Geldsystem entwickelt, das Bank, Kunden und Händler somit nicht nur optimal vor einer Kompromittierung der Bank, sondern auch vor der Kompromittierung des Deanonymisierers schützt. Damit Double-Spending von der Bank bereits im Voraus verhindert werden kann, wurde in Abschnitt 6.3 gezeigt, mit welchen Methoden man das System zu einer langfristig sicheren fairen elektronischen Geldbörse erweitern kann.
Die beiden in Kapitel 6 konstruierten langfristig sicheren Systeme sind nahezu genauso effizient wie *FOLC*. Lediglich die zur Gewährleistung der Fairness benötigten Zero-Knowledge-Beweise implizieren einen geringfügig höheren Rechenaufwand. Da beiden Systemen letzlich das von S. Brands vorgeschlagene Geldsystem [Bra94] zugrunde liegt, lassen sich mit den eingesetzten Techniken alle auf [Bra94] basierenden Geldsysteme zu langfristig sicheren Geldsystemen transformieren.
In Kapitel 7 wurde die Fragestellung analysiert, unter welchen Voraussetzungen in fairen elektronischen Geldsystemen die Observer vom Deanonymisierer und nicht von der Bank ausgegeben werden können, und ob sich dadurch ggf. effizientere Protokolle entwickeln lassen. Dazu wurde in Abschnitt 7.1 ein generisches faires elektronisches Geldsystem entwickelt, in dem die Observer vom Deanonymisierer verteilt und Kunden- und Münztracing unter Verwendung der Observer realisiert werden. Dieses System wurde von uns in [NS06] veröffentlicht. Unter der Annahme, dass die vom Deanonymisierer ausgehändigten Observer nicht von der Bank ausgelesen werden können, konnten in Abhängigkeit vom Anonymitätsgrad der Kunden (Anonymität oder Pseudonymität) unterschiedlich effiziente Ansätze angegeben werden. Auf diese Weise lassen sich, verglichen mit anderen fairen Geldsystemen, durch den Einsatz von Observern effizientere Abhebe- und Bezahl-Protkolle konstruieren. Allerdings ist die Effizienzsteigerung mit einem hohen Maß an Vertrauen in die Sicherheit der Observer verbunden, da von ihr letztlich die Sicherheit des Systems abhängt. Ein Vorteil des Systems ist aber, dass es auf einem beliebigen, sicheren blinden Signaturverfahren aufgebaut werden kann, so dass das System durch den Einsatz eines langfristig sicheren blinden Signaturverfahrens zu einem langfristig sicheren elektronischen Geldsystem erweitert werden kann. Verwendet man für Kunden- und Münztracing zusätzlich ein geeignetes Verschlüsselungsverfahren wie beispielsweise [DKXY02], so kann das System außerdem in ein langfristig sicheres faires elektronisches Geldsystem transformiert werden.
Einer der in Abschnitt 7.1 präsentierten Vorschläge zur Realisierung des Kundentracing entspricht den von Kim *et al.* dargestellten Gruppensignaturen mit Observern [KPW97] und wurde als Baustein des generischen Systems in Abschnitt 7.2 genauer untersucht. Es

wurden Schwachstellen von [KPW97] identifiziert und ein verbessertes Sicherheitsmodell für Gruppensignaturen mit Observern vorgeschlagen. Abschließend wurde im darauffolgenden Abschnitt 7.3 ein neues Gruppensignaturverfahren mit Observern konstruiert, das als Baustein in dem in Abschnitt 7.1 vorgeschlagenen System eingesetzt werden kann.

8.2 Ausblick

Mit den in den Abschnitten 6.2 und 6.3 vorgestellten Systemen konnten erste langfristig sichere faire elektronische Geldsysteme entwickelt werden. Leider wächst in beiden Systemen mit der Anzahl t der Intervalle, die von einem Angreifer kompromittiert werden dürfen, die Länge des privaten und öffentlichen Schlüssels linear. Auch wenn der Aufwand zur Erstellung und Verifikation einer Münze sicher das entscheidende Effizienzkriterium eines elektronischen Geldsystems ist, sind kurze Schlüssellängen gerade in Systemen mit begrenzter Bandbreite und begrenztem Speicherplatz von Interesse. Daher sollte in Zukunft untersucht werden, ob sich langfristig sichere Geldsysteme entwerfen lassen, in denen die Schlüssellänge unabhängig von der Anzahl t der kompromittierbaren Intervalle ist, wie es zum Beispiel in Signaturverfahren mit Forward-Security möglich ist. Forward-Security ist ein weiterer Ansatz, mit dem elektronische Geldsysteme vor einer Kompromittierung des Signaturschlüssels der Bank geschützt werden können. Prinzipiell lassen sich Blackbox-Konstruktionen von Signaturverfahren mit Forward-Security (z.B. [Kra00]) auf elektronische Geldsysteme übertragen, sind aber auf Grund ihres höheren Rechen- und Speicherbedarfs für elektronische Geldsysteme eher ungeeignet. Im Bereich der digitalen Signaturen konnten durch die Modifikation bestehender Signaturverfahren Verfahren entwickelt werden, die genauso effizient sind, wie die Systeme auf denen sie basieren (z.B. [IR01]). Für elektronische Geldsysteme sind solche Konstruktionen meines Wissens nach nicht bekannt. Vielversprechend wäre es daher, zukünftig bestehende Geldsysteme bzw. die den Systemen zugrunde liegenden blinden Signaturverfahren daraufhin zu untersuchen. In puncto Effizienz sind hier besonderes die auf [Bra94] basierenden Geldsysteme, deren Sicherheit auf der Diskreter-Logarithmus-Annahme beruht, zu erwähnen. Allerdings existiert bis heute noch kein Signaturverfahren mit Forward-Security, dessen Sicherheit auf dem Diskreter-Logarithmus-Problem basiert. Das heißt, es sollte in Zukunft grundsätzlich untersucht werden, ob und wie man solche Systeme konstruieren kann.
Wie in den Kapiteln 1 und 3 beschrieben, haben sich für den Entwurf elektronischer Geldsysteme im Wesentlichen zwei Ansätze etabliert: Systeme die auf blinden Signaturen basieren (z.B. [CFN89], [Bra94], [GT03]) und solche die ohne diesen Baustein auskommen und die Anonymität des Geldsystems allein durch Zero-Knowledge-Beweise realisieren (z.B. [STSY00], [CHL05]). Da in dieser Arbeit ausschließlich elektronische Geldsysteme betrachtet wurden, die auf blinden Signaturverfahren aufbauen, schließt sich im Anschluss an diese Arbeit die Fragestellung an, ob elektronische Geldsysteme, die ohne blinde Signaturen auskommen, mit den hier verwendeten Techniken zu langfristig sicheren elektronischen Geldsystemen erweitert werden können.

Bezeichnungen

Die in dieser Arbeit verwendeten Bezeichnungen werden in den entsprechenden Kapiteln eingeführt bzw. definiert. Zur besseren Übersicht wird an dieser Stelle ein tabellarischer Überblick der wichtigsten Bezeichnungen gegeben.

$\mathbb{Z}_p^*$	multiplikative Gruppe von $\mathbb{Z}_p$
G_q	zyklische Untergruppe von $\mathbb{Z}_p^*$ mit Primzahlordnung q
$g, g_1, \ldots, g_8$	Generatoren von G_q

H	Hashfunktion
Σ	Alphabet
Σ^*	Menge aller endlichen Zeichenfolgen über dem Alphabet Σ
Σ^n	Menge aller Zeichenfolgen der Länge n über dem Alphabet Σ
L	Sprache über dem Alphabet Σ ($L \subseteq \Sigma^*$)
$\mathsf{view}_A^B(P)$	Protokollansicht von Teilnehmer A bei Durchführung des Protokolls P mit Teilnehmer B

pk	öffentlicher Schlüssel
sk	privater Schlüssel

M	Menge aller Klartexte
C	Menge aller Geheimtexte

$\mathcal{A}$	Angreifer
$\mathcal{S}$	Signierer
$\mathcal{R}$	Empfänger
$\mathcal{P}$	Prover
$\mathcal{P}^*$	betrügerischer Prover
$\mathcal{V}$	Verifier
$\mathcal{V}^*$	betrügerischer Verifier

$\mathcal{B}$	Bank
$\mathcal{H}$	Händler
$\mathcal{K}$	Kunde
$\mathcal{O}$	Observer
$\mathcal{D}$	Deanonymisierer
$\mathcal{T}$	Tresor
Coin	elektronische Münze

Literaturverzeichnis

[ACJT00] Ateniese, Giuseppe ; Camenisch, Jan ; Joye, Marc ; Tsudik, Gene: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: *Advances In Cryptology - Proceedings Of Crypto '00* Bd. 1880, Springer, 2000 (Lecture Notes in Computer Science), S. 255–270

[AF96] Abe, Masayuki ; Fujisaki, Eiichiro: How to Date Blind Signatures. In: *Advances In Cryptology - Proceedings Of Asiacrypt '96* Bd. 1163, Springer, 1996 (Lecture Notes in Computer Science), S. 244–251

[And97] Anderson, Ross: *Invited Lecture*. 4th ACM Conference On Computer and Communication Security, 1997

[And02] Anderson, Ross: Two Remarks on Public Key Cryptology / Computer Laboratory, University Of Cambridge. 2002 (UCAM-CL-TR-549). – Technical Report

[AO00] Abe, Masayuki ; Okamoto, Tatsuaki: Provably Secure Partially Blind Signatures. In: *Advances In Cryptology - Proceedings Of Crypto '00* Bd. 1880, Springer, 2000 (Lecture Notes in Computer Science), S. 271–286

[ARD08] *ARD/ZDF Onlinestudie 2008*. `http://www.ard-zdf-onlinestudie.de`. Version: Juli 2008

[BDJR97] Bellare, Mihir ; Desai, Anand ; Jokipii, Eron ; Rogaway, Phillip: A Concrete Security Treatment of Symmetric Encryption. In: *Proceedings Of The 38th Annual Symposium on Foundations of Computer Science FOCS '97*, IEEE Computer Society, 1997, S. 394–403

[BDW05] Bao, Feng ; Deng, Robert H. ; Wang, Shuhong: Cryptoanalysis Of a Forward Secure Blind Signature Scheme with Provable Security. In: *Proceedings Of The 7th International Conference on Information and Communications Security, ICICS '05* Bd. 3783, Springer, 2005 (Lecture Notes in Computer Science), S. 53–60

[BGK95] Brickell, Ernie ; Gemmell, Peter ; Kravitz, David: Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In: *SODA '95: Proceedings Of The Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, 1995, S. 457–466

[BM99] Bellare, Mihir ; Miner, Sara: A Forward-Secure Digital Signature Scheme. In: *Advances In Cryptology - Proceedings Of Crypt '99* Bd. 1666, Springer, 1999 (Lecture Notes in Computer Science), S. 431–448

[BM01] Boyd, Colin ; Maitland, Greg: Fair Electronic Cash Based on a Group Signature Scheme. In: *Proceedings Of ICICS '01* Bd. 2229, Springer, 2001 (Lecture Notes in Computer Science), S. 461–465

[BNS05] Beutelspacher, Albrecht ; Neumann, Heike ; Schwarzpaul, Thomas: *Kryptografie in Theorie und Praxis*. 1. Auflage. Vieweg, 2005

[BR93] Bellare, Mihir ; Rogaway, Phillip: Random Oracles Are Practicle: A Paradigm for Designing Efficient Protocols. In: *Proceedings Of The 1st ACM Conference on Computer and Communications Security, CCS '93*, ACM Press, 1993, S. 62–73

[Bra93] Brands, Stefan: An Efficient Off-Line Electronic Cash System based on the Representation Problem / Centrum voor Wiskunde en Informatica. 1993 (CS-R9323). – Technical Report

[Bra94] Brands, Stefan: Untraceable Offline Cash in Wallets with Observers. In: *Advances In Cryptology - Proceedings Of Crypto '93* Bd. 773, Springer, 1994 (Lecture Notes in Computer Science), S. 302–318

[BSW06] Beutelspacher, Albrecht ; Schwenk, Jörg ; Wolfenstetter, Klaus-Dieter: *Moderne Verfahren der Kryptographie*. 6. Auflage. Vieweg, 2006

[Bun07] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: *Online-Shopping: Die meisten zahlen per Lastschrift.* `http://www.bitkom.org/files/documents/BITKOM-PI_Online-Bezahlen_04.10.2007.pdf`. Version: Oktober 2007

[Bun08a] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahn (Hrsg.): *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung.* : Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahn, Februar 2008. `http://www.bundesnetzagentur.de/media/archive/12198.pdf`

[Bun08b] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: *Deutsche kaufen gerne im Internet ein.* `http://www.bitkom.org/files/documents/BITKOM_Presseinfo_Online-Shopping_24_01_2008.pdf`. Version: Januar 2008

[CEG87] Chaum, David ; Evertse ; Graaf van d.: An improved Protocol for Demonstrating Possesion of Discrete Logarithms and some Generalizations. In: *Advances in Cryptology - Proceedings Of Eurocrypt '87* Bd. 304, Springer, 1987 (Lecture Notes in Computer Science), S. 127–141

[CFMT96] Chan, Agnes ; Frankel, Yair ; MacKenzie, Philip ; Tsiounis, Yiannis: Mis-representation of Identities in E-Cash Schemes and How To Prevent It. In: *Advances in Cryptology - Proceedings of Asiacrypt '96* Bd. 1163, Springer, 1996 (Lecture Notes in Computer Science), S. 276–285

[CFN89] Chaum, David ; Fiat, Amos ; Naor, Moni: Untraceable Electronic Cash. In: *Advances In Cryptology - Proceedings Of Crypto '88* Bd. 403, Springer, 1989 (Lecture Notes in Computer Science), S. 319–327

[CG04] Camenisch, Jan ; Groth, Jens: Group Signatures: Better Efficiency and New Theoretical Aspects. In: *Proceedings of SCN '04* Bd. 3352, Springer, 2004 (Lecture Notes in Computer Science), S. 120–133

[CH91] Chaum, David ; Heijst, Eugène van: Group Signatures. In: *Advances In Cryptology - Proceedings Of Eurocrypt '91* Bd. 547, Springer, 1991 (Lecture Notes in Computer Science), S. 257–265

[Cha83] Chaum, David: Blind Signatures For Untraceable Payments. In: *Advances In Cryptology - Proceedings Of Crypto '82*, Plenum Press, 1983, S. 199–203

[Cha84] Chaum, David: Blind Signature Systems. In: *Advances In Cryptology - Proceedings Of Crypto '83*, Plenum Press, 1984, S. 153

[CHL05] Camenisch, Jan ; Hohenberger, Susan ; Lysyanskaya, Anna: Compact E-Cash. In: *Advances In Cryptology - Proceedings Of Eurocrypt '05* Bd. 3494, Springer, 2005 (Lecture Notes in Computer Science), S. 302–321

[CK06] Camenisch, Jan ; Koprowski, Maciej: Fine-grained Forward-secure Signature Schemes without Random Oracles. In: *Discrete Applied Mathematics* 154 (2006), Nr. 2, S. 175–188

[CMS97] Camenisch, Jan ; Maurer, Ueli ; Stadler, Markus: Digital Payment Systems With Passive Anonymity-Revoking Trustees. In: *Journal Of Computer Security* 5 (1997), Nr. 1, S. 33–43

[CP93] Chaum, David ; Pedersen, Torben P.: Wallet Databases With Observers. In: *Advances In Cryptology - Proceedings Of Crypto '92* Bd. 740, Springer, 1993 (Lecture Notes in Computer Science), S. 89–105

[CP94] Cramer, Ronald ; Pedersen, Torben P.: Improved Privacy In Wallets With Observers. In: *Advances In Cryptology - Proceedings Of Eurocrypt '93* Bd. 765, Springer, 1994 (Lecture Notes in Computer Science), S. 329–343

[CPS94] Camenisch, Jan ; Piveteau, Jean-Marc ; Stadler, Markus: Blind Signatures Based on the Discrete Logarithm Problem. In: *Advances in Cryptology - Proceedings Of Eurocrypt' 94* Bd. 950, Springer, 1994 (Lecture Notes in Computer Science), S. 428–432

[CPS95] Camenisch, Jan ; Piveteau, Jean-Marc ; Stadler, Markus: Fair Blind Signatures. In: *Advances In Cryptology - Proceedings Of Eurocrypt '95* Bd. 921, Springer, 1995 (Lecture Notes in Computer Science), S. 209–219

[CT03] Canard, Sébastien ; Traoré, Jacques: On Fair E-cash Systems Based on Group Signature Schemes. In: *ACISP* Bd. 2727, Springer, 2003 (Lecture Notes in Computer Science), S. 237–248

[DCK03] Duc, Dang N. ; Cheon, Jung H. ; Kim, Kwangjo: A Forward-Secure Blind Signature Scheme Based On The Strong RSA Assumption. In: *Proceedings Of Information and Communications Security ICICS '03* Bd. 2836, Springer, 2003 (Lecture Notes in Computer Science), S. 11–21

[DF89] Desmedt, Yvo ; Frankel, Yair: Threshold Cryptosystems. In: *Advances In Cryptology - Proceedings Of Crypto '00* Bd. 435, Springer, 1989 (Lecture Notes in Computer Science), S. 307–315

[DH76] Diffie, Whitfiled ; Hellman, Martin E.: New Directions in Cryptography. In: *IEEE Transactions on Information Theory* 22 (1976), Nr. 6, S. 644–654

[DKXY02] Dodis, Yevgeniy ; Katz, Jonathan ; Xu, Shouhuai ; Yung, Moti: Key-Insulated Public Key Cryptosystems. In: *Advances In Cryptology - Proceedings Of Eurocrypt '02* Bd. 2332, Springer, 2002 (Lecture Notes in Computer Science), S. 65–82

[DKXY03] Dodis, Yevgeniy ; Katz, Jonathan ; Xu, Shouhuai ; Yung, Moti: Strong Key-Insulated Signature Schemes. In: *Proceedings Of Public Key Cryptography PKC '03* Bd. 2567, Springer, 2003 (Lecture Notes in Computer Science), S. 130–144

[ElG85] ElGamal, Taher: A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In: *IEEE Transactions on Information Theory* 31 (1985), Nr. 4, S. 469–472

[EO94] Eng, Tony ; Okamoto, Tatsuaki: Single-Term Divisible Electronic Coins. In: *Advances In Cryptology - Proceedings Of Eurocrypt '94* Bd. 950, Springer, 1994 (Lecture Notes in Computer Science), S. 306–319

[Fer93] Ferguson, Nils: Single Term Off-Line Coins. In: *Advances In Cryptology - Proceedings Of Eurocrypt '93* Bd. 765, Springer, 1993 (Lecture Notes in Computer Science), S. 318–328

[FH07] Freund, Roland ; Hoppe, Ronald: *Stoer/Bulirsch: Numerische Mathematik 1.* 10. Auflage. Springer, 2007

[FN01] Fremdt, Christine ; Neumann, Heike: Transferability in Coin System with Wallets with Observers. In: *Communications and Multimedia Security Issues Of The New Century CMS '01*, Kluwer Verlag, 2001, S. 255–265

[FS87] Fiat, Amos ; Shamir, Adi: How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In: *Advances In Cryptology - Proceedings Of Crypto '86* Bd. 263, Springer, 1987 (Lecture Notes in Computer Science), S. 186–194

[FTY96] Frankel, Yair ; Tsiounis, Yiannis ; Yung, Moti: Indirecte Discourse Proofs: Achievung Fair Off-Line E-Cash. In: *Proceedings Of Asiacrypt '96* Bd. 1163, Springer, 1996 (Lecture Notes in Computer Science), S. 286–300

[FTY98] Frankel, Yair ; Tsiounis, Yiannis ; Yung, Moti: Fair Electronic Cash Made Easy. In: *Proceedings Of Asiacrypt '98* Bd. 1514, Springer, 1998 (Lecture Notes in Computer Science), S. 257–270

[FY93] Franklin, Matthew K. ; Yung, Moti: Secure and Efficient Off-Line Digital Money (Extended Abstract). In: *Proceedings of ICALP '93* Bd. 700, Springer, 1993 (Lecture Notes in Computer Science), S. 265–276

[Ges08] Gesellschaft für Konsumforschung: *GfK-Studie Webscope.* `http://www.gfk.com/group/press_information/press_releases/002220/index.de.html`. Version: März 2008

[GM84] Goldwasser, Shafi ; Micali, Silvio: Probabilistic Encryption. In: *Journal Of Computer And System Science* 28 (1984), Nr. 2, S. 270–299

[GMR89] Goldwasser, Shafi ; Micali, Silvio ; Rackoff, Charles: The Knowledge Complexity of Interactive Proof Systems. In: *SIAM Journal On Computing* 18 (1989), Nr. 1, S. 186–208

[GT03] Gaud, Matthieu ; Traoré, Jacques: On the Anonymity of Fair Offline E-cash Systems. In: *Proceedings of Financial Cryptography '03* Bd. 2742, Springer, 2003 (Lecture Notes in Computer Science), S. 34–50

[HMP94] Horster, Patrick ; Michels, Markus ; Petersen, Holger: Meta-Message Recovery and Meta-Blind Signature Schemes Based on The Discrete Logarithm Problem and Their Applications. In: *Advances in Cryptology - Proceedings of Asiacrypt' 94* Bd. 917, Springer, 1994 (Lecture Notes in Computer Science), S. 224–237

[IR01] Itkis, Gene ; Reyzin, Leonid: Forward-Secure Signatures with Optimal Signing and Verifying. In: *Advances In Cryptology - Proceedings Of Crypto '01* Bd. 2139, Springer, 2001 (Lecture Notes in Computer Science), S. 332–354

[IR02] Itkis, Gene ; Reyzin, Leonid: SiBIR: Signer-Base Intrusion-Resilient Signatures. In: *Advances In Cryptology - Proceedings Of Crypto '02* Bd. 2442, Springer, 2002 (Lecture Notes in Computer Science), S. 499–514

[Itk01] Itkis, Gene: Intrusion-Resilient Signatures: Generic Constructions, or Defeating Strong Adversary with Minimal Assumptions. In: *Proceeding Of The 3rd Conference on Security in Communication Networks SCN '02* Bd. 2576, Springer, 2001 (Lecture Notes in Computer Science), S. 102–118

[JLO97] Juels, Ari ; Luby, Michael ; Ostrovsky, Rafael: Security of Blind Digital Signatures. In: *Advances In Cryptology - Proceedings Of Crypt '97* Bd. 1294, Springer, 1997 (Lecture Notes in Computer Science), S. 150–164

[JN03] Joux, Antoine ; Nguyen, Kim: Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. In: *Journal Of Cryptology* 16 (2003), Nr. 4, S. 239–247

[KPW97] Kim, Seungjoo ; Park, Sangjoon ; Won, Dongho: Group Signatures with Observers. In: *Proceedings Of CISCS '97 - Conference on Information Security and Cryptology*, 1997, S. 19–25

[KR01] Kozlov, A ; Reyzin, Leonid: Forward-Secure Signatures with Fast Key Update. In: *Proceeding Of The 3rd Conference on Security in Communication Networks SCN '02* Bd. 2576, Springer, 2001 (Lecture Notes in Computer Science), S. 241–256

[Kra00] Krawczyk, Hugo: Simple Forward-Secure Signatures From Any Signature Scheme. In: *Proceedings Of The 7th ACM Conference on Computer and Communications Security CCS '00*, ACM Press, 2000, S. 108–115

[LHL94] Li, Chuan-Ming ; Hwang, Tzonelli ; Lee, Narn-Yih: Threshold-Multisignature Schemes where Suspected Forgery Implies Traceability of Adversarial Shareholders. In: *Advances In Cryptology - Proceedings Of Eurocrypt '94* Bd. 950, Springer, 1994 (Lecture Notes in Computer Science), S. 194–204

[Mao04] Mao, Wenbo: *Modern Cryptography Theory & Practice*. 1. Auflage. Hewlett-Packard Books, 2004

[MRS88] Micali, Silvio ; Rackoff, Charles ; Sloan, Bob: The Notion of Security for Probabilistic Cryptosystems. In: *SIAM Journal On Computing* 17 (1988), Nr. 2, S. 412–426

[NS03] Neumann, Heike ; Schwarzpaul, Thomas: Fairness In Wallets with Observers. In: *Proceeding Of The International Workshop on Trust and Privacy in Digital Business TRUSTBUS '03*, IEEE Computer Society, 2003, S. 364–368

[NS06] Neumann, Heike ; Schwarzpaul, Thomas: Digital Coins: Fairness Implemented By Observers. In: *Journal Of Theoretical and Applied Electronic Commerce Research* 1 (2006), Nr. 1, S. 1–15

[Oka92] Okamoto, Tatsuaki: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: *Advances In Cryptology - Proceedings Of Crypt '92* Bd. 740, Springer, 1992 (Lecture Notes in Computer Science), S. 31–53

[Oka95] Okamoto, Tatsuaki: An Efficient Divisible Electronic Cash Scheme. In: *Advances In Cryptology - Proceedings Of Crypto '95* Bd. 963, Springer, 1995 (Lecture Notes in Computer Science), S. 438–451

[OO91] Okamoto, Tatsuaki ; Ohta, K.: Universal Electronic Cash. In: *Advances In Cryptology - Proceedings Of Crypto '91* Bd. 576, Springer, 1991 (Lecture Notes in Computer Science), S. 324–337

[Pop04] Popescu, Constantin: An Off-line Payment System Based on a Group Blind Signature Scheme. In: *Proceedings Of The 3rd Conference on Security and Network Architectures SAR 04*, 2004, S. 101–110

[PS96] Pointcheval, David ; Stern, Jacques: Provably Secure Blind Signature Schemes. In: *Advances In Cryptology - Proceedings Of Asiacrypt '96* Bd. 1163, Springer, 1996 (Lecture Notes in Computer Science), S. 252–265

[PS00] Pointcheval, David ; Stern, Jacques: Security Arguments for Digital Signatures and Blind Signatures. In: *Journal Of Cryptology* 13 (2000), Nr. 3, S. 361–396

[RSA78] Rivest, Ronald L. ; Shamir, Adi ; Adleman, Leonard M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications Of The ACM* 21 (1978), Nr. 2, S. 120–126

[Sch91] Schnorr, Claus-Peter: Efficient Signature Generation by Smart Cards. In: *Journal Of Cryptology* 4 (1991), Nr. 3, S. 161–174

[Sha79] Shamir, Adi: How to share a secret. In: *Communications Of The ACM* 22 (1979), Nr. 11, S. 612–613

[SN92] Solms, Sebastiaan von ; Naccache, David: On Blind Signatures and Perfect Crimes. In: *Computers and Security* 11 (1992), Nr. 6, S. 581–583

[ST98] Solages, Aymeric de ; Traoré, Jacques: An Efficient Fair Off-Line Electronic Cash System With Extensions to Checks and Wallets with Observers. In: *Proceedings Of Financial Cryptography '98* Bd. 1465, Springer, 1998 (Lecture Notes in Computer Science), S. 275–295

[Sti06] Stinson, Douglas R.: *Cryptography Theory And Practice*. 3. Auflage. Chapman & Hall/CRC, 2006

[STS99] Sander, Tomas ; Ta-Shma, Amnon: Auditable, Anonymous Electronic Cash. In: *Advances In Cryptology - Proceedings Of Crypto '99* Bd. 1666, Springer, 1999 (Lecture Notes in Computer Science), S. 555–572

[STSY00] Sander, Tomas ; Ta-Shma, Amnon ; Yung, Moti: Blind, Auditable Membership Proofs. In: *Proceedings of Financial Cryptography '00* Bd. 1962, Springer, 2000 (Lecture Notes in Computer Science), S. 53–71

[Tra99] Traoré, Jaques: Group Signatures and Their Application to Privacy-Protecting Offline Electronic Cash Systems. In: *Proceedings Of Information Security and Privacy '99* Bd. 1597, Springer, 1999 (Lecture Notes in Computer Science), S. 228–243

[Tsi97] Tsiounis, Yiannis: *Efficient Electronic Cash: New Notions and Techniques*, College Of Computer Science, Northeastern University, Boston, MA, USA, Diss., 1997

[TY98] Tsiounis, Yiannis ; Yung, Moti: On the Security of ElGamal Based Encryption. In: *Proceedings Of Public Key Cryptography PKC '98*, Springer, 1998, S. 117–134

[Una08] Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein: *Datenhandel: „Die sichtbare Spitze des Eisbergs wird größer“*. `http://www.verbraucherzentrale-sh.de/UNIQ122027070824214/link481821A.html`. Version: August 2008

[Ver08] Verbraucherzentrale Schleswig-Holstein: *Callcenter sind im Besitz von Kontodaten*. `http://www.verbraucherzentrale-sh.de/UNIQ122027070824214/link481821A.html`. Version: August 2008

[Wei05] Wei, Victor K.: *More Compact E-Cash with Efficient Coin Tracing*. Cryptology ePrint Archive, Report 2005/411. `http://eprint.iacr.org`. Version: November 2005

[Wer02] Werner, Annette: *Elliptische Kurven in der Kryptographie*. 1. Auflage. Springer, 2002

[XY03] Xu, Shouhuai ; Yung, Moti: Retrofitting Fairness on the Original RSA-Based E-Cash. In: *Proceedings Of Financial Cryptography '03* Bd. 2742, Springer, 2003 (Lecture Notes in Computer Science), S. 51–68

Index