

# Deanonymisierbare elektronische Zahlungsmittel

- ein Vergleich zweier Implementationen -



Diplomarbeit von Sylvain Franke  
FB Informatik, TU-Darmstadt

08. November 2002

## **Zusammenfassung**

Thema dieser Diplomarbeit sind deanonymisierbare elektronische Zahlungssysteme. Vorgestellt und miteinander verglichen werden das von *Camenisch/Maurer/Stadler* [4] sowie das von *Jakobsson/Yung* [8] vorgeschlagene Zahlungssystem. Der Vergleich erfolgt sowohl anhand einer theoretischen Betrachtung der Protokolle als auch anhand von Erkenntnissen und Meßwerten, die durch die Implementierung der Systeme erlangt wurden.

# Prolog

”Weit draußen in den unerforschten Einöden eines total aus der Mode gekommenen Ausläufers des westlichen Spiralarms der Galaxis leuchtet unbeachtet eine kleine gelbe Sonne. Um sie herum kreist in einer Entfernung von ungefähr achtundneunzig Millionen Meilen ein absolut unbedeutender, kleiner blaugrüner Planet, dessen vom Affen abstammende Bioformen so erstaunlich primitiv sind, daß sie Digitaluhren noch immer für eine unwahrscheinlich tolle Erfindung halten. Dieser Planet hat - oder besser gesagt, hatte - ein Problem: die meisten seiner Bewohner waren fast immer unglücklich. Zur Lösung dieses Problems wurden viele Vorschläge gemacht, aber die drehten sich meistens um das Hin und Her kleiner bedruckter Papierscheinchen, und das ist einfach drollig, weil es im großen und ganzen ja nicht die kleinen bedruckten Papierscheinchen waren, die sich unglücklich fühlten. Und so blieb das Problem bestehen. Vielen ging es schlecht, den meisten sogar miserabel, selbst denen mit Digitaluhren.” (Douglas Adams, Per Anhalter durch die Galaxis)

Dies ist nicht die Geschichte dieser Papierscheinchen.

Mein Dank gilt allen Mitarbeitern des Fachgebiets Theoretische Informatik, bei denen ich jederzeit für Fragen oder Probleme ein offenes Ohr fand. Insbesondere möchte ich mich bei Dennis Kügler für die geduldige Betreuung bedanken.

# Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 08. November 2002

Sylvain Franke

Die in diesem Dokument erwähnten Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen.

# Inhaltsverzeichnis

Prolog

Inhaltsverzeichnis

Einleitung	1
<b>1 Anonyme elektronische Zahlungssysteme</b>	<b>3</b>
1.1 Elektronische Zahlungssysteme . . . . .	4
1.1.1 Einfache Zahlungssysteme . . . . .	4
1.1.2 Anonyme Zahlungssysteme . . . . .	5
Anonymität . . . . .	5
Unverkettbarkeit . . . . .	5
1.1.3 Deanonymisierbare Zahlungssysteme . . . . .	7
1.1.4 aktiver/passiver Trustee . . . . .	8
1.1.5 online-/offline-Zahlungssysteme . . . . .	8
1.1.6 Münzgenerationen . . . . .	9
1.1.7 Münzstruktur . . . . .	9
1.1.8 Kaufvertrag: Angebot & Annahme . . . . .	10
1.2 Anwendungsfälle . . . . .	11
1.3 Implementierung . . . . .	12
1.4 Blinde Signaturen . . . . .	13
1.5 Anforderungen . . . . .	15
<b>2 Camenisch/Maurer/Stadler</b>	<b>17</b>
2.1 Einführung . . . . .	18
2.1.1 PKLOG . . . . .	18
2.1.2 PLOGEQ . . . . .	18
2.1.3 BlindPLOGEQ . . . . .	19
2.1.4 Grundidee . . . . .	20
2.1.5 Sicherheit . . . . .	20
DL-Problem . . . . .	21
DDH-Problem . . . . .	21

2.1.6	Änderungen . . . . .	21
2.2	Protokolle . . . . .	22
	Setup/Initialisierung . . . . .	22
	Abheben . . . . .	22
	Bezahlen . . . . .	25
	Rückgabe . . . . .	27
	Münzverfolgung . . . . .	29
	Kundenverfolgung . . . . .	30
2.3	Umsetzung der Anforderungen . . . . .	31
2.4	Implementierung . . . . .	34
<b>3</b>	<b>Jakobsson/Yung</b>	<b>37</b>
3.1	Einführung . . . . .	38
	3.1.1 Grundidee . . . . .	38
	3.1.2 Sicherheit . . . . .	39
	3.1.3 Änderungen . . . . .	39
3.2	Protokolle . . . . .	41
	Setup/Initialisierung . . . . .	41
	Abheben . . . . .	41
	Bezahlen . . . . .	44
	Rückgabe . . . . .	45
	Münzverfolgung . . . . .	47
	Kundenverfolgung . . . . .	48
3.3	Umsetzung der Anforderungen . . . . .	50
3.4	Implementierung . . . . .	53
<b>4</b>	<b>Vergleich</b>	<b>54</b>
4.1	Anforderungen & Sicherheit . . . . .	55
4.2	Effizienz . . . . .	59
	4.2.1 Engagement . . . . .	59
	4.2.2 Theoretische Betrachtung . . . . .	59
	Rechenaufwand der Anwendungsfälle . . . . .	59
	Speicherplatzbedarf . . . . .	62
	4.2.3 Praktische Betrachtung/Messung . . . . .	64
	Abheben . . . . .	65
	Bezahlen . . . . .	65
	Rückgabe . . . . .	66
4.3	Fazit . . . . .	67
	<b>Literaturverzeichnis</b>	<b>69</b>

<b>A</b>	<b>Implementierung</b>	<b>71</b>
<b>A.1</b>	<b>Infrastruktur</b> . . . . .	<b>72</b>
<b>A.1.1</b>	<b>Bank</b> . . . . .	<b>72</b>
<b>A.1.2</b>	<b>Kunde</b> . . . . .	<b>73</b>
<b>A.1.3</b>	<b>Händler</b> . . . . .	<b>74</b>
<b>A.1.4</b>	<b>Trustee</b> . . . . .	<b>75</b>
<b>A.1.5</b>	<b>CA</b> . . . . .	<b>76</b>
<b>A.1.6</b>	<b>Cryptomanager &amp; JCA/JCE</b> . . . . .	<b>76</b>
<b>A.1.7</b>	<b>Kommunikation</b> . . . . .	<b>77</b>
<b>A.2</b>	<b>Realisierung der Anwendungsfälle</b> . . . . .	<b>78</b>
<b>A.2.1</b>	<b>Abhebevorgang</b> . . . . .	<b>78</b>
<b>A.2.2</b>	<b>Bezahlvorgang</b> . . . . .	<b>78</b>
<b>A.2.3</b>	<b>Rückgabevorgang</b> . . . . .	<b>79</b>
<b>A.2.4</b>	<b>Deanonymisierung</b> . . . . .	<b>80</b>

# Einleitung

Der E-Commerce hat sich in den vergangenen Jahren rasant entwickelt. Einer Studie des Beratungsunternehmens McKinsey & Company zufolge, konnte alleine der Internet-Einzelhandel im Jahr 2001 ein Wachstum zwischen 60 und 80 % verzeichnen [9]. Diese neuen Märkte im Internet sind auf geeignete, sichere Zahlungsformen angewiesen. Bisher werden für den Zahlungsvorgang im Internet vor allem traditionelle Zahlungsformen verwendet, wie z.B. die Bestellung per Nachnahme, Scheck, Kreditkarte, Lastschrift oder auf Rechnung.

Betrachtet man die, gegenüber dem traditionellen Handel, im E-Commerce veränderten Geschäftsbedingungen, erscheint es sinnvoll, neue Arten elektronischer Zahlungsverfahren einzuführen, die stärker an den Bedürfnissen des neuen Mediums Internet ausgerichtet sind.

Wichtige Voraussetzung für die Akzeptanz elektronischer Zahlungssysteme ist die Berücksichtigung der Anforderungen möglichst aller Teilnehmer. Neben Sicherheit und Effizienz spielt der Schutz der Privatsphäre des Zahlenden eine bedeutende Rolle.

Anonyme elektronische Zahlungssysteme bieten diesen Schutz, aber sie schaffen auch neue Probleme: Im Schutz der Anonymität kann elektronisches Geld in großen Mengen unbeobachtbar transferiert werden, wodurch es sich besonders gut für kriminelle Aktivitäten wie Erpressung oder Geldwäsche eignet. Diese Zwickmühle führt zur Entwicklung deanonymisierbarer elektronischer Zahlungssysteme.

In dieser Diplomarbeit werden elektronische Zahlungssysteme untersucht, die eine durch eine vertrauenswürdige dritte Partei kontrollierte Deanonymisierung ermöglichen.

Diese Untersuchung erfolgt stellvertretend anhand von zwei ausgewählten Zahlungssystemen, die unterschiedliche Strategien der Einbindung der ver-

trauenswürdigen Partei in den Geldfluß repräsentieren. Bei dem einen handelt es sich um das 1996 von Jan Camenisch, Ueli Maurer und Markus Stadler vorgestellte Zahlungssystem [4]. Ihm gegenübergestellt wird das ebenfalls 1996 vorgestellte, von Markus Jakobsson und Moti Yung entworfene System [8].

Um die Realisierbarkeit der beiden Systeme zu überprüfen, wurden sie als Prototypen implementiert. Als Infrastruktur diente die am Lehrstuhl Theoretische Informatik an der TU-Darmstadt entwickelte JCash-Architektur (s. Anhang A).

Um diese Infrastruktur nutzen zu können, mußten die Zahlungssysteme an einigen Stellen angepaßt werden. Dabei wurde allerdings darauf geachtet, die von Camenisch/Maurer/Stadler bzw. Jakobsson/Yung entwickelten Grundkonzepte nicht zu verändern.

In Kapitel 1 werden zunächst, ausgehend von einfachen elektronischen Zahlungssystemen, schrittweise die Grundlagen und Mechanismen deanonymisierbarer elektronischer Zahlungssysteme erklärt. Im Anschluß folgt eine Charakterisierung der an einem solchen System beteiligten Teilnehmer, der zu realisierenden Vorgänge sowie der wesentlichen zahlungssystemunabhängigen Implementierungsgrundlagen. Abschließend erfolgt eine Zusammenstellung von Anforderungen, die später als Grundlage für die Bewertung der Systeme dienen soll.

In Kapitel 2 und 3 werden die beiden Zahlungssysteme jeweils in ihrer angepaßten Form detailliert vorgestellt und bezüglich der im vorangehenden Kapitel formulierten Anforderungen geprüft.

Im abschließenden 4. Kapitel werden die beiden Zahlungssysteme bezüglich ihrer Sicherheit und ihrer Effizienz einander gegenübergestellt. Der Vergleich der Effizienz erfolgt zunächst auf theoretischer Ebene über eine Komplexitätsbetrachtung der einzelnen Anwendungsfälle und über die Betrachtung des Speicherplatzbedarfs. Anschließend werden die Laufzeiten der Anwendungsfälle der beiden Systeme verglichen. Zum Schluß erfolgt eine Zusammenfassung der gewonnenen Erkenntnisse.

# Kapitel 1

## Anonyme elektronische Zahlungssysteme

## 1.1 Elektronische Zahlungssysteme

Ein elektronisches Zahlungssystem besteht aus Protokollen, in denen die teilnehmenden Parteien (Bank/Kunde/Händler) miteinander interaktiv agieren. Ein Protokoll ist ein verteilter Algorithmus, an dem mehrere Parteien in einer Weise teilhaben, so daß das Ergebnis von allen Beteiligten abhängt.

Teilnehmer an einem elektronischen Zahlungssystem sind mindestens eine Bank, ein Kunde sowie ein Händler. Die Bank ist ein Kreditinstitut, das ein elektronisches Zahlungssystem als Service anbietet. Ein Kunde ist eine Person, die bei der Bank ein Konto hält. Ein Händler ist ein Kaufmann, der das von der Bank betriebene elektronische Zahlungssystem als Zahlungsmittel akzeptiert und dazu ebenfalls ein Konto bei der Bank hält.

Das Ziel eines Zahlungssystems ist der sichere Transfer von Kapital, als Vergütung für Dienstleistungen oder zur Bezahlung von Waren, von einem Kundenkonto auf das Konto eines Händlers. Bei diesem Geldfluß lassen sich in der Regel folgende Vorgänge unterscheiden:

Der Abhebevorgang, in dem der Kunde Geld von seinem Konto bei der Bank abhebt, der Bezahlvorgang, in dem der Kunde das abgehobene Geld an einen Händler zahlt, sowie der Einreichvorgang, in dem der Händler das erhaltene Geld auf sein Konto bei der Bank einzahlt.

Zusätzlich muß der Kunde die Möglichkeit haben, in einem Rückgabevorgang durch ihn abgehobenes Geld wieder auf sein Konto bei der Bank einzuzahlen.

In den folgenden Abschnitten wird schrittweise zu deanonymisierbaren Zahlungssystemen hingeführt. Ausgehend von einfachen elektronischen Zahlungssystemen werden nach und nach Eigenschaften hinzugefügt bzw. Alternativen diskutiert. Dabei wird an verschiedenen Stellen auf die in den folgenden Kapiteln näher beschriebenen und einander gegenübergestellten Zahlungssysteme von *Camenisch/Maurer/Stadler* [4] und von *Jakobsson/Yung* [8] verwiesen.

### 1.1.1 Einfache Zahlungssysteme

Einfache elektronische Zahlungssysteme basieren im wesentlichen auf einer Authentifizierung des Kunden, d.h. seine Identität und die von ihm gewünschte Transaktion müssen sowohl dem Händler als auch der Bank offengelegt werden.

Dieses einfache System wird z.B. bei Zahlungen mit Kreditkarte genutzt. Der Kunde authentifiziert sich durch Präsentation seiner Kreditkarte (evtl. in Kombination mit seinem Personalausweis). Daraufhin läßt der Händler die Liquidität des Kunden durch die Bank prüfen. Um sicher zu sein, daß der Kunde mit der Belastung seines Kontos einverstanden ist, muß dieser eine

PIN eingeben bzw. eine Unterschrift abgeben.

Anstatt einer Authentifizierung mittels Karte plus PIN reicht bei vielen Systemen nur eine PIN bzw. nur auf der Karte notierte Informationen.

Solche einfachen Systeme haben den Nachteil, daß die Bank genau nachvollziehen kann, wann und wo ein Kunde sein Geld ausgibt. Auch ein Händler erhält Informationen über den Kunden und kann dessen Kaufverhalten bei ihm registrieren.

### 1.1.2 Anonyme Zahlungssysteme

Um die bei den einfachen elektronischen Zahlungssystemen auftretenden Nachteile zu umgehen, muß das Zahlungssystem Anonymität und Unverkettbarkeit (unlinkability) bieten.

#### Anonymität

Anonymität [*gr.-nlat.*]: das Nichtbekanntsein, Nichtgenanntsein; Namenlosigkeit.

Diese Definition nach Duden liefert Anhaltspunkte dafür, was Anonymität ausmacht: Die Identität einer oder mehrerer an einem anonymen Vorgang beteiligten Instanzen ist nicht bestimmbar, weil sie entweder den anderen beteiligten Instanzen nicht bekannt ist (Nichtbekanntsein), gegenüber den anderen beteiligten Instanzen nicht in Erscheinung tritt (Nichtgenanntsein) oder innerhalb des anonymen Vorgangs ohne erkennbaren Namen agiert (Namenlosigkeit).

A. Pfitzmann und M. Köhntopp definieren in [15] Anonymität als den Zustand der Nicht-Identifizierbarkeit einer beteiligten Instanz innerhalb der Menge aller möglichen Instanzen.

Bezogen auf ein Zahlungssystem bedeutet Anonymität, daß es nicht möglich ist, ausgehend von einem Zahlungsvorgang, die Identität eines Kunden aus der Menge aller Kunden zu identifizieren.

#### Unverkettbarkeit

In Bezug auf ein System, für das wir diese Eigenschaft beschreiben möchten, bedeutet Unverkettbarkeit von zwei oder mehr Objekten, daß -innerhalb des Systems- diese Objekte nicht mehr und nicht weniger unterscheidbar sind, als sie dies anhand des a-priori Wissens (d.h. unabhängig von jeglichen wahrgenommenen Eigenschaften) sind (Vgl. [15]).

Unverkettbarkeit stellt sicher, daß ein Benutzer mehrfachen Gebrauch von

den Betriebsmitteln oder von den Services des Systems machen kann, ohne daß andere in der Lage sind, diese Nutzungen in Verbindung zu bringen. Unverkettbarkeit erfordert, daß Benutzer und/oder andere Personen nicht imstande sind festzustellen, ob der gleiche Benutzer bestimmte spezifische Betriebe im System verursachte (Vgl. [15]).

Im Falle eines Zahlungssystems sind die oben erwähnten Objekte Zahlungsvorgänge bzw. Zahlungsmittel.

Als technische Lösung für anonymes Bezahlen bieten sich münzbasierte Zahlungen auf Basis von blinden Signaturen an, wie es z.B. in [18] vorgeschlagen wird. Diese Systeme bilden die Funktionalität von Bargeld in elektronischer Form nach. Eine Bank erzeugt digitale Münzen durch blindes Signieren einer zufällig und kollisionsfrei vom Kunden erzeugten Zahl, welche als Seriennummer der Münze bezeichnet wird. Der Kunde kann nun die Münze zum Bezahlen bei einem Händler verwenden, ohne daß der Händler oder die Bank die Identität des Zahlenden ermitteln können, da durch die blinde Signatur sichergestellt ist, daß die Bank die Seriennummer bei einer Zahlung keinem Abhebevorgang zuordnen kann.

Anstatt münzbasierter Systeme können auch anonyme Konten, wie sie z.B. in [6] beschrieben sind, realisiert werden, um anonymes Zahlen zu ermöglichen.

Bei den in dieser Arbeit untersuchten Systemen ([4], [8]) handelt es sich um münzbasierte Zahlungssysteme. Aus diesem Grund wird im folgenden vereinfachend von Münzen die Rede sein. Dies schließt nicht aus, daß die vorgestellten Mechanismen auch für andere (z.B. account-basierte) Systeme realisierbar sind.

Ein Problem von anonymen Zahlungssystemen ist, daß sie für kriminelle Zwecke mißbraucht werden können. So kann es zur Geldwäsche (d.h. zum Einschleusen von Vermögenswerten aus organisierter Kriminalität in den legalen Finanz- und Wirtschaftskreislauf mit dem Ziel der Verschleierung der wahren Herkunft des Geldes) mißbraucht werden.

Außerdem könnte es als ideales Zahlungsmittel für Erpressungen verwendet werden, da es auf elektronischem Weg ohne physischen Kontakt übergeben werden kann und danach nicht mehr einfrierbar oder identifizierbar ist. Auch Steuerhinterziehung sowie Schmiergeldzahlung werden mit einem anonymen Zahlungssystem erleichtert (Vgl. [19]).

Um derartigem Mißbrauch vorzubeugen, existieren unterschiedliche Lösungsansätze. So kann eine Limitierung des maximal abhebbaren Betrags das Sy-

stem für die beschriebenen kriminellen Zweckentfremdungen unattraktiv machen (Vgl. [16]). Eine effizientere Form der Vorbeugung wird durch Deanonymisierbarkeit erreicht.

### 1.1.3 Deanonymisierbare Zahlungssysteme

Um kriminelle Aktivitäten zu verhindern, werden Mechanismen benötigt, mit denen die Anonymität von Zahlungen gezielt aufgehoben werden kann.

Zwei Arten der Deanonymisierung sind notwendig:

- Münzverfolgung: Die Möglichkeit, Münzen wiederzuerkennen, die während eines auffälligen Abhebevorgangs erzeugt wurden. Dies macht das System für Raub und Erpressungen unattraktiv (Vgl. [4] S. 4).
- Kundenverfolgung: Die Möglichkeit, von einer erhaltenen Münze auf die Identität des Kunden zu schließen, der sie abgehoben hat. Damit kann bei Verdacht der Geldwäsche bzw. von Steuerhinterziehung der Geldfluß aufgedeckt werden (Vgl. [4] S. 4).

Um die Privatsphäre eines ordentlichen Kunden zu schützen und eine Zweckentfremdung der Münz- oder Kundenverfolgung zu verhindern, muß die Deanonymisierung gesetzlich geregelt und möglichst durch geeignete Mechanismen gegen Mißbrauch gesichert werden.

Deanonymisierung kann auf folgende Weise sicher realisiert werden:

Analog zu dem bei Geldscheinen verwendeten Prinzip, die Seriennummern der Scheine aufzuschreiben, können beim Abhebevorgang der elektronischen Münzen Parameter gespeichert werden, die eine Wiedererkennung erlauben. Um dabei Anonymität gegenüber der Bank zu erreichen, ist eine vertrauenswürdige dritte Partei notwendig, die im folgenden (nach [4]) als Trustee<sup>1</sup> bezeichnet wird. Die Aufgabe des Trustee ist es, aktiv oder passiv (siehe Kap. 1.1.4) beim Abhebevorgang mitzuwirken, so daß eine Deanonymisierung nur durch ihn und die Bank gemeinsam möglich ist (Vgl. [8], [4]).

Alternativ sind andere Vorgehensweisen möglich (Vgl. z.B [11]), die allerdings nicht Thema dieser Arbeit sein sollen.

---

<sup>1</sup>Statt Trustee wird auch Ombudsmann (Vgl. [8]) und Judge (Vgl. [5]) als Bezeichnung verwendet.

### 1.1.4 aktiver/passiver Trustee

Sollen Münz- und Kundenverfolgung auch für die Bank ausschließlich bei begründetem Verdacht möglich sein, so ist ein Trustee notwendig. Er kontrolliert die von der Bank und den Ermittlungsbehörden gestellten Deanonymisierungsanfragen. Um die Anonymität aufheben zu können, muß er in die Vorgänge (z.B. Abhebevorgang) aktiv oder passiv involviert sein. Dementsprechend können folgende Vorgehensweisen unterschieden werden:

- Der Trustee ist an jedem Abhebevorgang aktiv beteiligt. Er spielt dabei die Rolle eines Zwischenhändlers zwischen Kunden und Bank. Das in Kap. 3 beschriebene Zahlungssystem nach [8] folgt dieser Vorgehensweise.
- Der Trustee ist nur bei der Eröffnung eines Kundenkontos aktiv beteiligt. Bei allen anderen Vorgängen (Abheben, Bezahlen, Einreichen, Zurückgeben) ist er passiv (Vgl.[5]).
- Der Trustee ist bei keinem regulären Vorgang aktiv beteiligt. Er wird nur zur Deanonymisierung benötigt. Das in Kap. 2 beschriebene Zahlungssystem nach [4] folgt dieser Vorgehensweise.

Vor- bzw. Nachteile von aktivem und passivem Trustee werden in Kap. 4 am Beispiel der untersuchten Zahlungssysteme herausgearbeitet.

### 1.1.5 online-/offline-Zahlungssysteme

Es ist möglich, online- und offline-Zahlungssysteme zu realisieren. Bei einem offline-System können Abhebevorgang, Bezahlvorgang und Einreichvorgang zeitlich verzögert nacheinander ablaufen. Beim Bezahlvorgang und beim Einreichen handelt es sich um Protokolle, bei denen jeweils nur zwei Parteien (Kunde und Händler bzw. Händler und Bank) gleichzeitig online kommunizieren. Bei einem online-System sind der Bezahlvorgang und das Einreichen in einer Phase zusammengefaßt und es müssen alle beteiligten Parteien (Kunde, Händler und Bank) online miteinander verbunden sein.

Sei  $t_w$  der Zeitpunkt des Abhebevorgangs,  $t_p$  der Zeitpunkt des Bezahlvorgangs und  $t_d$  der Einreichzeitpunkt des für die Bezahlung verwendeten Geldes. Dann läßt sich der Unterschied zwischen online- und offline-Zahlungssystemen wie folgt darstellen:

- $t_w < t_p = t_d$  - online-Zahlungssystem,
- $t_w < t_p < t_d$  - offline-Zahlungssystem.

Ein Problem von offline-Systemen ist, daß das mehrfache Ausgeben der abgehobenen Münzen ohne zusätzliche Hardware nicht verhindert werden kann, sondern erst im nachhinein von der Bank aufgedeckt wird. Demgegenüber steht ein Effizienzvorteil, da es während eines Bezahlvorgangs nicht notwendig ist, mit der Bank verbunden zu sein und mit ihr zu kommunizieren.

Die in Kap. 2 und Kap. 3 untersuchten Zahlungssysteme nach [4] und [8] können durch einfache Modifikationen sowohl als online- als auch als offline-Zahlungssysteme realisiert werden. Da es theoretisch über das Internet kein Problem darstellt, online mit der Bank zu kommunizieren, wird im folgenden nur auf die Realisierung der online-Varianten eingegangen.

### 1.1.6 Münzgenerationen

Um die Sicherheit der Bankschlüssel besser garantieren zu können, bietet es sich an, diese in regelmäßigen Abständen gegen neue auszutauschen. Diese Vorgehensweise impliziert, daß jede Münze einen zeitlichen Gültigkeitsbereich erhält. Alle während eines bestimmten Zeitraums abgehobenen Münzen werden durch die Bank mit der gleichen Schlüsselgeneration signiert. Die so erzeugten Münzen einer Generation können dann bis zu einem bestimmten Tag zum Zahlen verwendet werden (Akzeptanzphase). Im Anschluß ist es bis zu einem endgültigen Verfallsdatum möglich, die Münze an die Bank zurückzugeben (Rückgabephase).

Ein weiterer erheblicher Vorteil dieser Vorgehensweise ist die Reduzierung des Speicheraufwandes auf Bank- bzw. Trusteseite. So können abgelaufene Münzen aus der Liste der zum Zahlen verwendeten Münzen gelöscht werden, wenn ihre Gültigkeit abgelaufen ist. Analog können auch die Parameter der Münzen, die zur Ermöglichung der Deanonymisierung gespeichert wurden, behandelt werden.

### 1.1.7 Münzstruktur

Die bis hier besprochenen Eigenschaften münzbasierter deanonymisierbarer Zahlungssysteme ermöglicht es, eine allgemeine Beschreibung der Münzstruktur zu geben. Eine Münze besteht demnach aus

- einer Seriennummer,
- einer von der Bank blind erteilten Signatur der Seriennummer,
- einer Nennung oder Referenzierung ihres Wertes sowie
- eines Verweises auf ihre Generation.

Art und Form der Seriennummer und der Signatur sind dabei allerdings zahlungssystemabhängig.

### 1.1.8 Kaufvertrag: Angebot & Annahme

Das BGB legt die vertragstypischen Pflichten beim Kaufvertrag in §433 allgemein wie folgt fest:

1. Durch den Kaufvertrag wird der Verkäufer einer Sache verpflichtet, dem Käufer die Sache zu übergeben und das Eigentum an der Sache zu verschaffen. Der Verkäufer hat dem Käufer die Sache frei von Sach- und Rechtsmängeln zu verschaffen.
2. Der Käufer ist verpflichtet, dem Verkäufer den vereinbarten Kaufpreis zu zahlen und die gekaufte Sache abzunehmen.

Verträge bestehen dabei aus übereinstimmenden Willenserklärungen. Sie kommen zustande durch das Angebot (Vertragsantrag, Offerte) einer Seite und die Annahme (Vertragsannahme, Akzept) der anderen. Mit der Annahme gilt der Vertrag als geschlossen und bindend; beide Seiten sind ab diesem Zeitpunkt Vertragspartner. In dem Angebot sind die wesentlichen Vertragsbestandteile enthalten (welche Ware, welche Dienstleistung, zu welchem Preis etc.), damit dies nur noch durch ein "Ja" angenommen oder durch ein "Nein" abgelehnt werden braucht. Die Annahme muss demjenigen, der das Angebot gemacht hat, gegenüber erklärt werden.

## 1.2 Anwendungsfälle

Die in dieser Arbeit untersuchten deanonymisierbaren online-Zahlungssysteme erfordern die Partizipation von Bank, Kunde, Händler und Trustee.

- Die Bank führt die Konten ihrer Kunden, gibt Münzen heraus und nimmt Münzen an bzw. zurück.
- Ein Kunde hält ein Konto bei der Bank, hebt Münzen ab und bezahlt damit bei einem Händler oder gibt sie an die Bank zurück.
- Ein Händler hält ebenfalls ein Konto bei der Bank, nimmt Münzen als Vergütung für Dienstleistungen oder zur Bezahlung von Waren entgegen und reicht die erhaltenen Münzen bei der Bank ein.
- Die Rolle des Trustee im regulären Zahlungsverkehr ergibt sich je nachdem, ob es sich um ein System mit aktivem oder passivem Trustee handelt (Vgl. Kap. 1.1.4).

Diese Teilnehmer interagieren im Rahmen der Vorgänge „Abheben“, „Bezahlen“ und „Rückgabe“, die sich folgendermaßen darstellen:

- Beim *Abhebevorgang* erstellen Kunde und Bank gemeinsam Münzen, wobei das Konto des Kunden mit dem abgehobenen Betrag belastet wird.
- Beim *Bezahlvorgang* übergibt der Kunde Münzen an den Händler. Dieser reicht die Münzen bei der Bank ein und bekommt den Betrag auf seinem Konto gutgeschrieben. Im Gegenzug erbringt er die von ihm geforderte Dienstleistung bzw. liefert die bestellte Ware.
- Beim *Rückgabevorgang* reicht der Kunde vorher von ihm abgehobene Münzen bei der Bank ein und bekommt den Betrag auf seinem Konto gutgeschrieben.

Zusätzlich bieten die Zahlungssysteme die Möglichkeit zur Münz- und Kundenverfolgung, wie sie in Kap. 1.1.3 beschrieben wurde.

## 1.3 Implementierung

Im folgenden Abschnitt werden kurz die wesentlichen zahlungssystemunabhängigen Implementierungsgrundlagen beschrieben. Eine detailliertere Darstellung ist im Anhang zu finden (s. A.1).

Die Teilnehmer Bank, Händler und Trustee sind als Server implementiert. Der Kunde besitzt eine elektronische Geldbörse (`Wallet`) mit einem Münzspeicher (`CoinStore`), in dem er die bei einem Abhebevorgang erzeugten Münzen speichert.

Als Schnittstelle zu den Implementierungen der konkreten Protokolle benutzen die Teilnehmer abstrakte Zahlungssysteme (`PaymentSystem`) (s. Anhang A.1 & Kap. 2.4 & 3.4).

Voraussetzung für die Protokolle ist eine Möglichkeit, Parameter senden und empfangen zu können. Dazu müssen insbesondere komplexe Objekte (z.B. Münzen, Schlüssel, Signaturen) reversibel in `byte`-Folgen übersetzt werden. Diese Codierung geschieht gemäß ASN.1-Definition (Vgl. Anhang A.1.7).

Die Bank betreibt eine Datenbank zur Verwaltung ihrer Kunden und der Konten sowie zur Speicherung von Münzparametern zu allen abgehobenen und eingereichten Münzen.

Händler und Trustee betreiben ebenfalls Datenbanken.

Die von den Zahlungssystemen benötigten Standard-Kryptoverfahren (z.B. Signaturen) werden über installierte Kryptoprovider gemäß der Java Cryptography Architecture (JCA/JCE) bereitgestellt.

Die Verwaltung der öffentlichen Schlüssel der Teilnehmer erfolgt über eine Certification-Authority (CA, s. A.1.5). Für seinen geheimen Schlüssel besitzt jeder Teilnehmer eine persönliche Sicherheitsumgebung (PSE).

## 1.4 Blinde Signaturen

Angenommen Alice möchte, daß ein Dokument von einer Person oder Institution signiert wird, ohne daß diese Einsicht in das Dokument erhält oder die resultierende Signatur selbst sieht. Dies kann mit Blinden Signaturen - die von Chaum in [7] eingeführt wurden - realisiert werden. "Blind" heißt in diesem Zusammenhang, daß die Sicht, die der Unterzeichner beim Signieren hat, statistisch unabhängig von der zu signierenden Botschaft  $m$  und der resultierenden Signatur  $S(m)$  ist.

Zu zahlreichen Signaturverfahren existieren Varianten zur Erstellung blinder Signaturen (z.B. RSA, Schnorr). Im Folgenden wollen wir die Blinde Schnorr Signatur genauer betrachten, da sie im weiteren Verlauf der Arbeit des öfteren Erwähnung findet.

### Schnorr Signatur

Sei  $g \in G$  ein Erzeuger einer endliche Gruppe  $G$  mit primärer Ordnung  $q$ , in der es schwierig ist, diskrete Logarithmen zu berechnen. Für das Schlüsselpaar  $(x, y)$  gelte  $y = g^x$ , für ein zufällig gewähltes  $x \in \mathbb{Z}_q$ .

$\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ , sei eine kollisionsresistente Hashfunktion.

Eine Schnorr Signatur zu einer Botschaft  $m$  ist ein Paar  $(c, s) \in (\mathbb{Z}_q, \mathbb{Z}_q)$ , für das die Verifikationsformel  $c = \mathcal{H}(m || g^s y^c)$  gilt.

Diese Signatur kann nur mit Hilfe des geheimen Schlüssels  $x$  wie folgt gebildet werden:

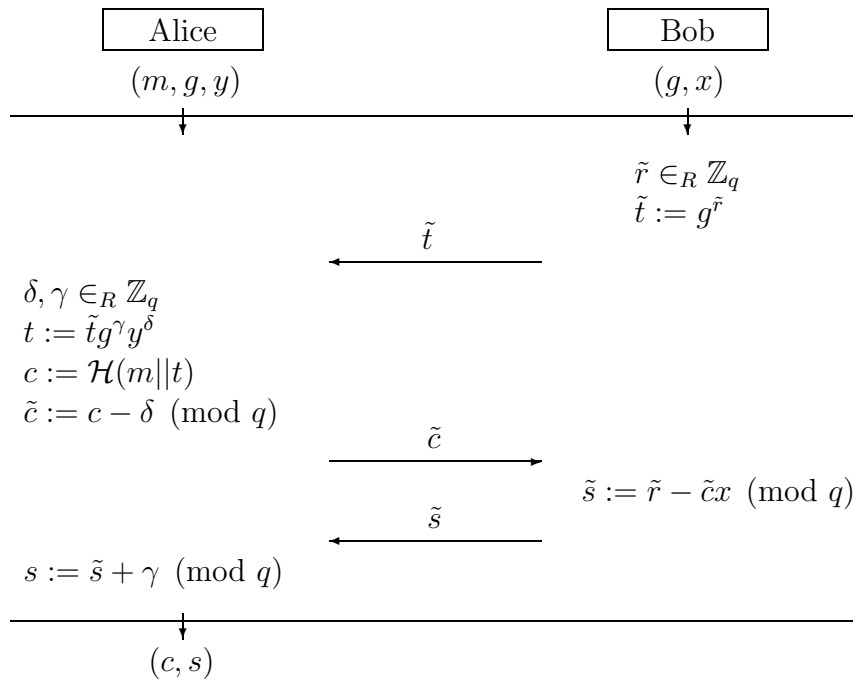
$$c = \mathcal{H}(m || g^r) \text{ für ein zufällig gewähltes } r \in \mathbb{Z}_q \text{ sowie}$$

$$s = r - cx \pmod{q}.$$

Es gilt:  $\mathcal{H}(m || g^s y^c) = \mathcal{H}(m || g^{s+xc}) = \mathcal{H}(m || g^{r-cx+xc}) = \mathcal{H}(m || g^r) = c$ .

### Blinde Schnorr Signatur

Für eine Blinde Schnorr Signatur ergibt sich folgendes Protokoll:



Falls beide Teilnehmer sich an das Protokoll halten, ist das Paar  $(c, s)$  eine gültige Schnorr Signatur für die von Alice gewählte Botschaft  $m$ . Es gilt:

$$g^s y^c = g^{\tilde{s} + \gamma} y^{\tilde{c} + \delta} = g^{\tilde{r} - \tilde{c}x + \gamma + \tilde{c}x} y^\delta = \tilde{t} g^\gamma y^\delta = t$$

und somit zur Verifikation die Bedingung

$$c = \mathcal{H}(m || g^s y^c).$$

Der Beweis, daß Bob in diesem Protokoll "blind" ist, wird u.a. in [4] geführt.

## 1.5 Anforderungen

Anonyme elektronische Zahlungsmittel müssen einer Reihe von Anforderungen gerecht werden. In der Literatur (z.B. [4], [5], [6], [8], [12]) werden Anforderungen formuliert. Im folgenden werden diese, als eine Grundlage für die Bewertung der in Kap. 3 und Kap. 4 vorgestellten online-Zahlungssysteme, zusammengefaßt:

- **Fälschungssicherheit**  
Es darf nicht möglich sein, Falschgeld herzustellen.  
Als Falschgeld gelten Münzen, die ohne Beteiligung der Bank erzeugt wurden, aber von dieser als gültiges Zahlungsmittel akzeptiert werden.
- **Mehrfachausgabesicherheit**  
Es darf desweiteren nicht möglich sein, eine Münze doppelt auszugeben bzw. mit Münzen im Wert  $V$  von der Bank akzeptierte Zahlungen vorzunehmen, deren Wert  $V$  übersteigt (*overspending*).
- **Akzeptanzgarantie**  
Ein Kunde muß sicher sein, daß regulär abgehobene Münzen innerhalb eines definierten, ihm bekannten Zeitraums für Zahlungen akzeptiert werden oder an die Bank zurückgegeben werden können.  
Insbesondere muß der Kunde davor sicher sein, nicht durch eine Falschaussage der Bank des *overspending* beschuldigt zu werden.
- **Abhebesicherheit**  
Ein Kunde muß sicher sein, daß niemand außer ihm unter seinem Namen von seinem Konto Münzen abheben kann.
- **Ausgabesicherheit**  
Außerdem muß er sicher sein, daß niemandem außer ihm die von ihm abgehobenen Münzen ausgeben oder zurückgeben kann.
- **Rückgabesicherheit**  
Ein Kunde darf nur die von ihm selbst abgehobenen Münzen zurückgeben.
- **Bankraubsicherheit**  
Es sollte einem Angreifer oder einer Gruppe von Angreifern nicht möglich sein, die Bank zu zwingen, Münzen herauszugeben, die später als gültiges Zahlungsmittel akzeptiert werden. Später bezieht sich im Zusammenhang mit elektronischen Zahlungsmitteln u.U. auf den relativ

kurzen Zeitraum ( $\Delta$ ), den ein Bezahlvorgang einschließlich der elektronischen Warenauslieferung benötigt.

Ebenso sollte es nicht möglich sein, durch Raub der geheimen Schlüssel der Bank, Münzen zu erzeugen, die später als gültiges Zahlungsmittel akzeptiert werden.

- **Anonymität**

Es darf nicht möglich sein, ausgehend von einem Zahlungsvorgang, die Identität eines Kunden aus der Menge aller Kunden zu identifizieren.

- **Unverkettbarkeit**

Anhand von in verschiedenen Zahlungsvorgängen benutzten Münzen, darf es unter regulären Umständen weder der Bank noch irgendjemand anderem möglich sein zu erkennen, ob sie vom selben Kunden verwendet wurden.

- **Kundenverfolgung**

Es muß möglich sein, anhand einer während eines Zahlungsvorgangs benutzten Münze, eine Kundenverfolgung vorzunehmen.

Diese darf nur bei begründetem Verdacht, unter Berücksichtigung gesetzlicher Auflagen und nur durch legitimierte Teilnehmer (z.B. Trustee) möglich sein.

- **Münzverfolgung**

Desweiteren muß es möglich sein, anhand einer während eines Abhebevorgangs erzeugten Münze, eine Münzverfolgung vorzunehmen und die betroffene Münze einzufrieren.

Dies darf ebenfalls nur bei begründetem Verdacht, unter Berücksichtigung gesetzlicher Auflagen und nur durch legitimierte Teilnehmer möglich sein.

# Kapitel 2

Camenisch/Maurer/Stadler

## 2.1 Einführung

Bei dem von Camenisch, Maurer und Stadler in [4] vorgestellten System ist der Trustee nur bei der Initialisierung sowie der Deanonymisierung aktiv beteiligt (passiver Trustee, vgl. Kap. 1.1.4). Das System basiert auf zwei Arten nichtinteraktiver Beweise, die auf dem DL-Problem auf endlichen Gruppen primer Ordnung basieren. Diese nichtinteraktiven Beweise sind einerseits Signaturen, mit denen Nachrichten (z.B. Seriennummern von Münzen) signiert werden können. Andererseits wird mit ihnen die Kenntnis oder die Gleichheit diskreter Logarithmen bewiesen.

### 2.1.1 PKLOG

Sei  $G$  eine endliche Gruppe mit primer Ordnung  $q$ , in der es schwierig ist, diskrete Logarithmen zu berechnen.  $g \in G$  sei ein Erzeuger von  $G$  und für  $x, y \in G$  gelte  $y = g^x$ .  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  sei eine kollisionsresistente Hashfunktion.

Der nicht-interaktive Beweis  $PKLOG(m, g, y)$  (*engl.: Proof of knowledge of a discrete logarithm*) beweist die Kenntnis des diskreten Logarithmus  $\log_g y$  und besteht aus einer Schnorr Signatur  $(c, s)$  zu der Botschaft  $m||g||y$ .

Es gilt:

$$PKLOG(m, g, y) = (c, s) \text{ mit}$$

$$c = \mathcal{H}(m||g||y||g^s y^c)$$

Solch ein Beweis kann nur gebildet werden, wenn man den diskreten Logarithmus  $\log_g y = x$  kennt. Dann ergibt sich  $(c, s)$  wie folgt:

$$c = \mathcal{H}(m||g||y||g^r) \text{ für ein zufällig gewähltes } r \in \mathbb{Z}_q \text{ und}$$

$$s = r - cx \pmod{q}.$$

Die Botschaft  $m$  kann auch der leere String  $\epsilon$  sein.

### 2.1.2 PLOGEQ

Der nicht-interaktive Beweis  $PLOGEQ(m, g_1, y_1, g_2, y_2)$  (*engl.: Proof of equality of the discrete logarithm*) beweist die Gleichheit der diskreten Logarithmen  $\log_{g_1} y_1 = \log_{g_2} y_2$  für  $g_1, g_2, y_1, y_2 \in G$ . Gleichzeitig dient er dazu, die Kenntnis des diskreten Logarithmus zu beweisen. Er besteht aus einem Paar  $(c, s)$ , für das gilt:

$$PLOGEQ(m, g_1, y_1, g_2, y_2) = (c, s) \text{ mit}$$

$$c = \mathcal{H}(m||g_1||g_2||y_1||y_2||g_1^s y_1^c ||g_2^s y_2^c)$$

Ein solcher Beweis kann nur gebildet werden, wenn man die diskreten Logarithmen  $\log_{g_1} y_1$  und  $\log_{g_2} y_2$  kennt und beide den selben Wert  $x$  haben. Dann ergibt sich das Paar  $(c, s)$  wie folgt:

$$c = \mathcal{H}(m||g_1||g_2||y_1||y_2||g_1^r ||g_2^r) \text{ f\"ur ein zuf\"allig gew\"ahltes } r \in \mathbb{Z}_q \text{ sowie}$$

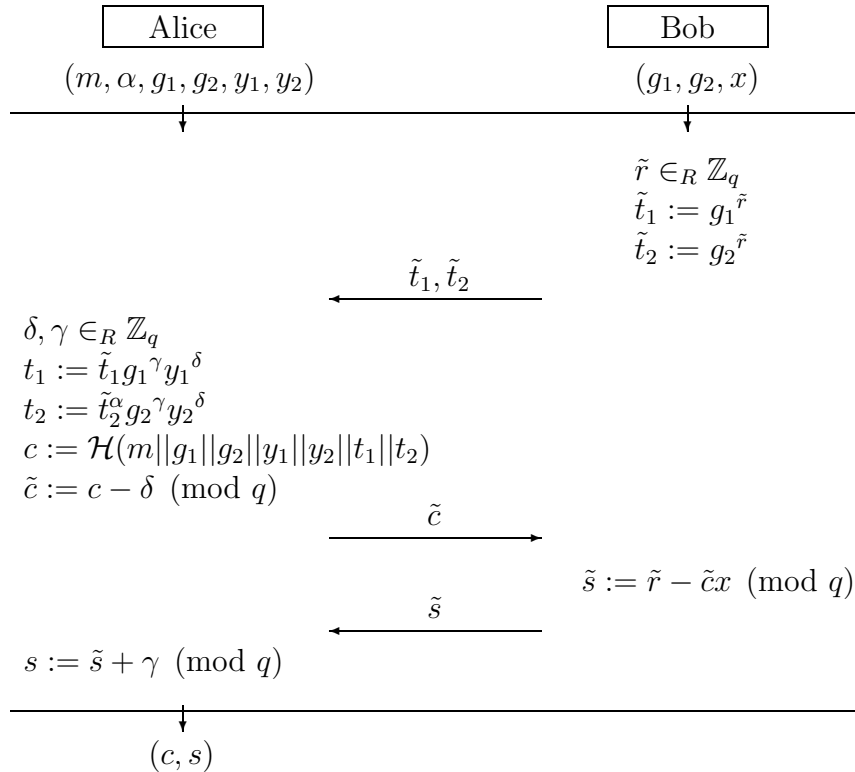
$$s = r - cx \pmod{q}.$$

Auch hier kann  $m$  der leere String  $\epsilon$  sein.

### 2.1.3 BlindPLOGEQ

Angelehnt an die Blinde Schnorr Signatur l\"a\ss t sich der Beweis PLOGEQ derart modifizieren, da\ss nunmehr Bob den Beweis leistet, ohne die gleichzeitig signierte Botschaft  $m$  oder das resultierende Paar  $(c, s)$  zu sehen.  $\text{BlindPLOGEQ}(m, \alpha, g_1, y_1, g_2, y_2) = (c, s)$  dient also zum einen als Beweis f\"ur die Gleichheit der Diskreten Logarithmen  $\log_{g_1} y_1$  und  $\log_{g_2} y_2$ , zum anderen ist es eine Blinde Signatur f\"ur die Botschaft  $m||g_1||y_1||g_2||y_2$ .

Ein solcher Beweis kann nur gebildet werden, wenn Bob  $\log_{g_1} y_1 = \log_{g_2} y_2 = x$  kennt. Dann ergibt sich das Paar  $(c, s)$  wie folgt:



Es gilt analog zur Blinden Schnorr Signatur:

$$g_i^s y_i^c = g_i^{\bar{s}+\gamma} y_i^{\bar{c}+\delta} = g_i^{\bar{r}-\bar{c}x+\gamma+\bar{c}x} y_i^\delta = \tilde{t}_i g_i^\gamma y_i^\delta = t_i, i \in \{1, 2\}$$

und somit zur Verifikation die Bedingung:

$$c := \mathcal{H}(m || g_1 || g_2 || y_1 || y_2 || g_1^s y_1^c || g_2^s y_2^c).$$

Der Beweis, daß Bob in diesem Protokoll "blind" ist, erfolgt analog zu dem in [4] für Blinde Schnorr Signaturen geführten Beweis.

### 2.1.4 Grundidee

Für jede Münze wählt der Kunden einen geheimen Wert  $\alpha$ , der zum einen als Münzschlüssel zum Signieren des Kaufvertrages verwendet wird, zum anderen aber auch zum Blenden der Münze. Die Münze selbst besteht aus  $c\#$ ,  $h_p = g_1 g_2^\alpha$ ,  $z_p = g_1^x g_2^{x\alpha}$  und einem von der Bank blind erbrachten Beweis, daß  $\log_{h_p} z_p = \log_g y$  gilt (d.h. beide Logarithmen entsprechen dem privaten Schlüssel der Bank).

Aus Sicht der Bank wird der Beweis jedoch für  $h_w = g_1^{\alpha^{-1}} g_2$  erbracht und es gilt  $h_p = h_w^\alpha$ . Ohne Kenntnis von  $\alpha$  kann die Bank die "Abhebesicht"  $h_w$  und die "Zahlungssicht"  $h_p$  einer Münze nicht verknüpfen. Damit der Trustee genau diese Verknüpfung zur Münz- bzw. Kundenverfolgung durchführen kann, hinterlegt der Kunde während des Abhebevorgangs zu jeder abgehobenen Münze den Wert  $d = y_\tau^\alpha$  bei der Bank. Wegen  $y_\tau = g_2^\tau$ , gilt  $h_p = g_1 d^{\tau^{-1}}$  (Münzverfolgung) bzw.  $d = (h_p/g_1)^\tau$  (Kundenverfolgung). Damit der Kunde bei der Bank keinen falschen Wert  $d$  hinterlegen kann, muß er beweisen, daß der Wert korrekt gebildet wurde (d.h. daß  $\log_{g_1}(h_w/g_2) = \log_d y_\tau$  gilt).

Um Mehrfachausgaben verhindern zu können, speichert die Bank  $(c\#, h_p)$ , der im Rahmen eines Bezahl- oder Rückgabevorgangs eingereichten Münzen.

### 2.1.5 Sicherheit

Die Sicherheit des von Camenisch, Maurer und Stadler in [4] vorgestellten Systems hängt von der Schwierigkeit ab, in der gewählten Gruppe das DL-Problem und das Decision-Diffie-Hellman-Problem (DDH) zu berechnen.

#### DL-Problem

Die in dem hier untersuchten Zahlungssystem verwendeten nichtinteraktiven Beweise PKLOG, PLOGEQ bzw. BlindPLOGEQ basieren auf dem DL-Problem auf endlichen Gruppen primär Ordnung  $q$ . D.h., es muß in der

gewählten Gruppe  $G$  schwierig sein, zu gegebenem  $y$  und  $g$  einen Exponenten  $x \in \mathbb{Z}_q^*$  zu finden, mit dem  $y = g^x$  gilt.

Mögliche Gruppen sind z.B.  $q$ -elementige Untergruppen der primen Restklassengruppe oder  $q$ -elementige Untergruppen der Punktgruppe elliptischer Kurven über Primkörpern.

### Decision-Diffie-Hellman-Problem (DDH)

Ebenso muß es in der gewählten Gruppe schwierig sein,  $g^{(a \cdot b)}$  und  $g^c$  für gegebene  $g, g^a$  und  $g^b$  mit zufälligen Exponenten  $a, b, c \in \mathbb{Z}_q^*$  zu unterscheiden. Im Falle des hier untersuchten Zahlungssystems muß es schwierig sein, zu den der Bank bekannten  $h_p/g_1 = g_2^\alpha$  und  $y_\tau = g_2^\tau$  zu entscheiden, ob  $d = g_2^{(\alpha \cdot \tau)}$  gilt. Ansonsten wäre der Bank eine Deanonymisierung der Münzen ohne Partizipation des Trustee möglich.

Mögliche Gruppen sind ebenfalls z.B.  $q$ -elementige Untergruppen der primen Restklassengruppe oder  $q$ -elementige Untergruppen der Punktgruppe elliptischer Kurven über Primkörpern. In diesen ist der beste bekannte Algorithmus DDH zu lösen die Berechnung der diskreten Logarithmen (s. [2]).

## 2.1.6 Änderungen

Die in [4] vorgestellten Protokolle wurden wie folgt modifiziert:

- Statt eines öffentlichen Schlüssels  $(g, y)$  und Erzeugern  $g_1, g_2 \in G$  veröffentlicht die Bank zu ihrem privaten Schlüssel  $x$  öffentliche Schlüssel  $(g, y), (g_1, y_1), (g_2, y_2)$ . Somit kann  $z_w = h_w^x = (g_1^{\alpha^{-1}} g_2)^x = y_1^{\alpha^{-1}} y_2$  vom Kunden eigenständig berechnet werden.
- Statt  $V$  mit einem leeren String  $\epsilon$  zu berechnen, nutzen wir PKLOG, um beim Bezahlvorgang die Annahme des Angebots zu signieren.
- Ein Rückgabevorgang wurde in [4] nicht vorgeschlagen.

## 2.2 Protokolle

Im folgenden Abschnitt werden die Protokolle des Zahlungssystems im Detail vorgestellt.

### Setup/Initialisierung

Die Bank wählt zunächst für jede neue Münzgeneration eine endliche Gruppe  $G$  mit primitiver Ordnung  $q$ . Dabei muß sie darauf achten, daß es in der gewählten Gruppe schwierig ist, diskrete Logarithmen zu berechnen. In dieser Gruppe werden über eine verifizierbare Pseudo-Zufallsfunktion zufällige Elemente  $g, g_1, g_2 \in G$  gewählt, ohne daß die Bank  $\log_j k$  für  $j, k \in \{g, g_1, g_2\}$ ,  $j \neq k$  kennt <sup>1</sup>. Anschließend wählt die Bank pro Münzwert  $i$  einen privaten Schlüssel  $x_i$  und berechnet auf der gewählten Gruppe  $G$  die öffentlichen Schlüssel:

$$y_i := g^{x_i}, \quad y_{1_i} := g_1^{x_i}, \quad y_{2_i} := g_2^{x_i}.$$

Damit ergibt sich für einen Münzwert  $i$  das Schlüsselpaar:

$$\mathcal{K}_i = ((x_i), (G, g, g_1, g_2, y_i, y_{1_i}, y_{2_i})).$$

Die Bank publiziert die öffentlichen Schlüssel und veranlaßt den Trustee seinerseits, ein zu der neuen Generation gehöriges Schlüsselpaar zu generieren. Dazu verwendet dieser die von der Bank gewählten Systemparameter  $(q, g_2, G)$  und wählt als seinen privaten Schlüssel ein Element  $\tau \in \mathbb{Z}_q^*$ . Mit diesem berechnet er  $y_\tau = g_2^\tau$  auf  $G$ . Damit ergibt sich für den Trustee das Schlüsselpaar:

$$\mathcal{K}_\tau = ((\tau), (G, g_2, y_\tau)).$$

Abschließend publiziert er den erzeugten öffentlichen Schlüssel.

### Abheben

Um Münzen abzuheben, muß sich der Kunde zunächst mit der Bank verbinden. Dabei findet eine Authentifizierung statt. Anschließend erzeugen Kunde und Bank gemeinsam die Münzen mit der gewünschten Wertigkeit  $i$  wie folgt:

1. Der Kunde sucht die öffentlichen Schlüssel  $(G, g, g_1, g_2, y_i, y_{1_i}, y_{2_i})$  sowie  $(G, g_2, y_\tau)$ , und die Bank sucht ebenfalls den öffentlichen Schlüssel  $(G, g_2, y_\tau)$  sowie ihr Schlüsselpaar  $\mathcal{K}_i = ((x_i), (G, g, g_1, g_2, y_i, y_{1_i}, y_{2_i}))$ .

---

<sup>1</sup>Verifizierbare Pseudo-Zufallsfunktionen werden z.B. in [13] behandelt.

2. Der Kunde wählt eine zufällige Schlüsselnummer  $c\#$  und ein Element  $\alpha \in \mathbb{Z}_q^*$ . Anschließend berechnet er auf  $G$

- $h_w = g_1^{\alpha^{-1}} g_2$
- $d = y_\tau^\alpha$

sowie den Beweis (s. Kap. 2.1, PLOGEQ)

- $U = PLOGEQ(\epsilon, g_1, h_w/g_2, d, y_\tau) = (c_u, s_u)$ .

*U beweist, daß  $h_w, d$  regulär mit gleichem DL  $\alpha$  gebildet wurde.*

Die so berechneten  $h_w, d$  und  $U$  sendet er an die Bank.

3. Die Bank vergibt zunächst eine eindeutige Sitzungsnummer  $session\#$  und verifiziert  $U \stackrel{?}{=} PLOGEQ(\epsilon, g_1, h_w/g_2, d, y_\tau)$  (s. Kap. 2.1).  
Schlägt dieser Test fehl, so wird der Abhebevorgang abgebrochen.

4. Der Kunde bildet nun

- $h_p = h_w^\alpha$
- $z_p = y_1 y_2^\alpha$

und berechnet gemeinsam mit der Bank den Beweis (s. Kap. 2.1, BlindPLOGEQ)

- $W = BlindPLOGEQ(c\#, \alpha, g, y_i, h_p, z_p) = (c_w, s_w)$ .

*Die Bank signiert die Münze damit blind und beweist gleichzeitig, daß  $z_p = h_p^{x_i}$  und  $y_i = g^{x_i}$  mit gleichem DL  $x_i$  gilt. Aus ihrer Sicht beweist sie dies allerdings für das Paar  $(z_w, h_w)$ , wobei der geheime Münzschlüssel  $\alpha$  dem Kunden als Blendparameter dient ( $z_p = z_w^\alpha, h_p = h_w^\alpha$ ).*

Während dieses Protokolls speichert die Bank die Sitzungsnummer  $session\#, d, i$ , die bei der Authentifizierung festgestellte  $id$  des Kunden sowie einen Verweis auf die aktuelle Generation  $gen\#$  in ihrer Datenbank und belastet das Kundenkonto mit dem Wert der gerade erzeugten Münze der Sorte  $i$ .

5. Der Kunde verifiziert nun  $W \stackrel{?}{=} BlindPLOGEQ(c\#, g, y_i, h_p, z_p)$  (s. Kap. 2.1).

Schlägt dieser Test fehl, so gibt er alle während dieser Session abgehobenen Münzen an die Bank zurück.

Der Kunde speichert nun die erzeugte Münze, bestehend aus  $c\#$ ,  $\alpha$ ,  $h_p$ ,  $z_p$ ,  $W$ , ihrem Wert  $i$  und einem Verweis auf ihre Generation  $gen\#$ , ab.

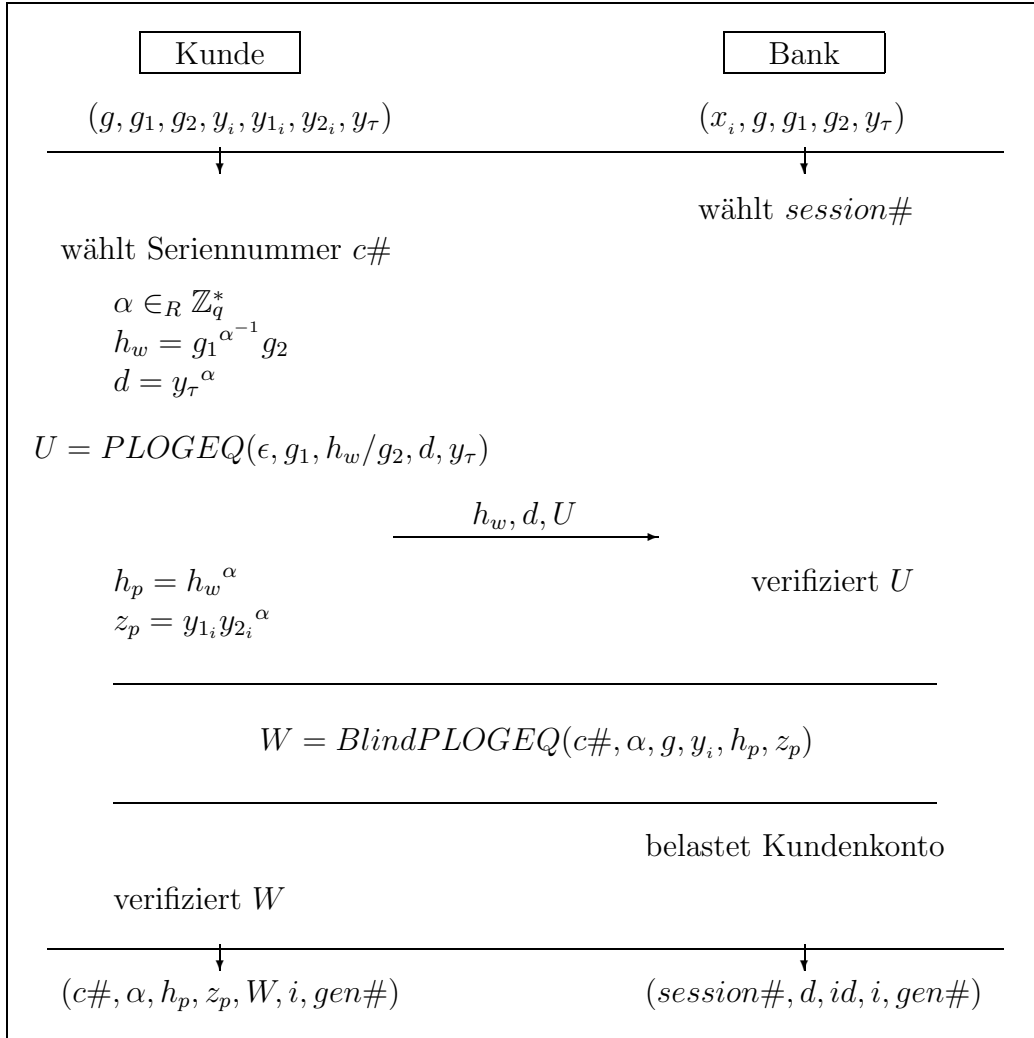


Abbildung 2.1: Abhebeprotokoll

**Bezahlen**

Der Bezahlvorgang besteht aus einem Protokoll zwischen Kunden, Händler und Bank. Vor dem Protokoll sendet der Händler zunächst sein Angebot  $o$  an den Kunden.

Zum besseren Verständnis des Protokolls nehmen wir im Folgenden zunächst an, daß nur mit einer einzigen Münze bezahlt wird.

1. Der Kunde berechnet zunächst den Hashwert  $\tilde{o} = H(o)$  und holt die benötigte Münze (bestehend aus  $c\#, \alpha, h_p, z_p, W$ , ihrem Wert  $i$  und einem Verweis auf ihre Generation  $gen\#$ ) aus seinem Münzspeicher. Dann besorgt er sich den zu dem jeweiligen Münzwert  $i$  und der jeweiligen Münzgeneration  $gen\#$  passenden Bankschlüssel  $(G, g, g_1, g_2, y_i, y_{1_i}, y_{2_i})$ . Nun berechnet er den Beweis (s. Kap. 2.1, PKLOG)

- $V = PKLOG(\tilde{o}, g_2, h_p/g_1) = (c_v, s_v)$ .

*Er signiert so die Annahme des Händlerangebots ( $\tilde{o}$ ) und beweist gleichzeitig, daß die Münze ihm gehört, sowie, daß  $h_p = h_w^\alpha = (g_1^{\alpha^{-1}} g_2)^\alpha = g_1 g_2^\alpha$  gilt. Dadurch wird verhindert, daß der Kunde beim Abheben  $h_p = h_w^\alpha g^\beta$  und  $z_p = z_w^\alpha y^\beta$  für ein  $\beta \neq 0$  wählt und so eine spätere Deanonymisierung unmöglich macht.*

*$W$  und  $V$  bilden zusammen den hinreichenden Beweis, daß die verwendete Münze gültig und deanonymisierbar ist.*

Anschließend sendet er  $c\#, h_p, z_p, W, V, i$  und  $gen\#$  unter Angabe des Kaufvertrags  $o$  an den Händler.

2. Der Händler berechnet ebenfalls den Hashwert  $\tilde{o} = H(o)$  und reicht die erhaltene Münze zusammen mit  $\tilde{o}$  an die Bank weiter.
3. Die Bank verifiziert die erhaltenen Beweise

$$W \stackrel{?}{=} BlindPLOGEQ(c\#, \alpha, g, y_i, h_p, z_p) \text{ und}$$

$$V \stackrel{?}{=} PKLOG(\tilde{o}, g_2, h_p/g_1) \text{ (s. Kap. 2.1)}$$

und prüft, ob

- die Münze mit der Nummer  $c\#$  nicht bereits früher zum Zahlen verwendet wurde und
- die Münze mit dem Parameter  $h_p$  nicht durch die Bank blockiert wurde (*siehe Münzverfolgung*) bzw. zurückgegeben wurde (*siehe Rückgabe*) und

- der Einreichzeitpunkt in der Akzeptanzphase der Münze liegt.

Schlägt einer dieser Tests fehl, so wird die Münze zurückgewiesen und der Bezahlvorgang abgebrochen.

Sonst schreibt die Bank dem Händler den Wert der Münze auf seinem Konto gut und speichert  $c\#, h_p, \tilde{o}$  und  $V$  in ihrer Datenbank. Abschließend sendet sie eine Bestätigung der Gutschrift an den Händler.

4. Nach Eingang der Bestätigung vergleicht der Händler den gutgeschriebenen Betrag mit dem im Kaufvertrag vereinbarten Kaufpreis. Stimmen die Beträge überein, erfolgt die Auslieferung der Ware. Ansonsten erfolgt eine Benachrichtigung des Kunden.

In der Regel werden mehrere Münzen benötigt, um einen Kauf zu tätigen. Dann kann das obige Protokoll effizient parallelisiert werden.

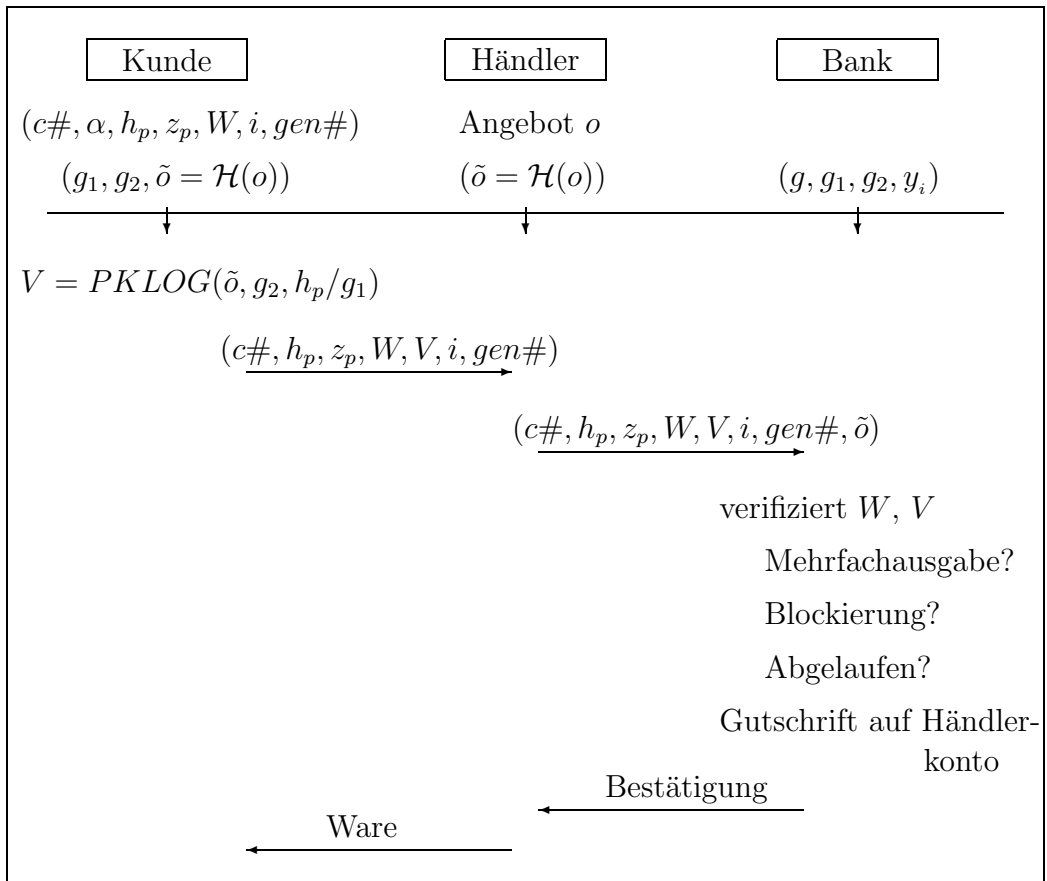


Abbildung 2.2: Bezahlprotokoll

## Rückgabe

Der Rückgabevorgang besteht aus einem Protokoll zwischen Kunden und Bank. Zum besseren Verständnis des Protokolls nehmen wir im Folgenden an, daß nur eine einzige Münze zurückgegeben wird.

1. Der Kunde holt zunächst die zurückzugebende Münze (bestehend aus  $c\#, \alpha, h_p, z_p, W$ , ihrem Wert  $i$  und einem Verweis auf ihre Generation  $gen\#$ ) aus seinem Münzspeicher. Anschließend besorgt er sich den öffentlichen Schlüssel des Trustee  $y_\tau$  sowie den zu dem jeweiligen Münzwert  $i$  und der jeweiligen Münzgeneration  $gen\#$  passenden Bank-schlüssel  $(G, g, g_1, g_2, y_i, y_{1_i}, y_{2_i})$  und wählt eine zufällige Zahl  $\rho$ .

Nun berechnet er:

- $d = y_\tau^\alpha$

und den Beweis (s. Kap. 2.1, PLOGEQ)

- $V = PLOGEQ(\rho, y_\tau, d, g_2, h_p/g_1) = (c_v, s_v)$

*Er signiert so die Rückgabe ( $\rho$ ) und beweist gleichzeitig, daß  $d$  zu der eingereichten Münze gehört. Dies zeigt er, indem er darlegt, daß  $d = y_\tau^\alpha$  und  $h_p/g_1 = g_2^\alpha$  den gleichen DL  $\alpha$  besitzen.*

Anschließend sendet er  $c\#, d, h_p, z_p, W, \rho, V$ , den Wert  $i$  und  $gen\#$  an die Bank.

2. Die Bank verifiziert das erhaltene  $V \stackrel{?}{=} PLOGEQ(m, y_\tau, d, g_2, h_p/g_1)$  (s. Kap. 2.1) und prüft, ob
  - das Paar  $(id, d)$  in ihrem Speicher der regulär abgehobenen Münzen gespeichert ist und
  - die Münze mit dem Parameter  $h_p$  nicht bereits früher zum Zahlen verwendet wurde sowie
  - der Rückgabezeitpunkt innerhalb der Rückgabephase der Münze liegt.

Schlägt einer dieser Tests fehl, so wird die Münze zurückgewiesen und der Rückgabevorgang abgebrochen.

Sonst schreibt die Bank dem Kunden den Wert der Münze auf seinem Konto gut und speichert  $c\#, h_p, \rho$  und  $V$  in ihrer Datenbank. Abschließend sendet sie eine Bestätigung der Gutschrift an den Kunden.

3. Der Kunde überprüft abschließend den gutgeschriebenen Betrag.

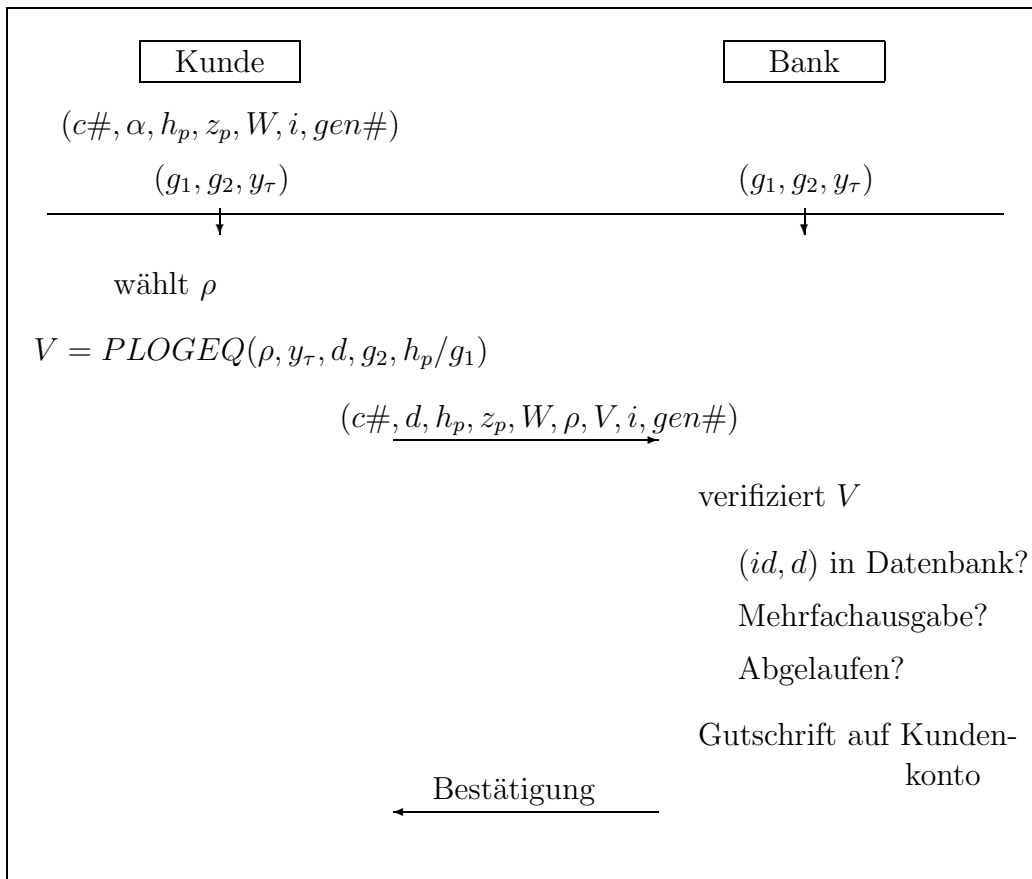


Abbildung 2.3: Rückgabeprotokoll

### Münzverfolgung

Die Münzverfolgung besteht aus einem Protokoll zwischen Bank und Trustee. Sie kann von den während des Abhebevorgangs beobachteten Münzparametern ausgehend erfolgen.

Als Voraussetzung muß eine richterliche Legitimation vorliegen.

Während des Abhebevorgangs speichert die Bank  $session\#$ ,  $d$  sowie die  $id$  des Kunden. Es gilt:

- $g_1 d^{\tau^{-1}} = g_1 (y_\tau^\alpha)^{\tau^{-1}} = g_1 g_2^\alpha = (g_1^{\alpha^{-1}} g_2)^\alpha = h_w^\alpha = h_p$

Um alle nicht auf legalem Wege abgehobenen Münzen eines Abhebevorgangs während eines Zahlungsvorgangs oder Rückgabevorgangs identifizieren zu können, wird folgendes Protokoll durchlaufen:

1. Die Bank sucht zunächst, für alle während des Abhebevorgangs  $session\#$  erzeugten Münzen, die von ihr gespeicherten Parameter  $(d, id)$  und sendet  $d$  zusammen mit der richterlichen Legitimation an den Trustee.
2. Der Trustee kontrolliert die richterliche Legitimation und berechnet mit seinem privaten Schlüssel  $\tau$

- $d^{\tau^{-1}}$

und sendet das Ergebnis an die Bank.

3. Die Bank berechnet nun

- $h_p = g_1 d^{\tau^{-1}}$

und setzt  $h_p$  auf eine Liste nicht zu akzeptierender Münzen. Bei jedem Zahlungsvorgang und jedem Rückgabevorgang wird das  $h_p$  der eingereichten Münze dahingehend überprüft, ob es in dieser Liste vorkommt. Sollte dies der Fall sein, so wird die Münze abgewiesen.

### Kundenverfolgung

Die Kundenverfolgung besteht aus einem Protokoll zwischen Bank und Trustee. Sie kann von den während des Bezahlvorgangs beobachteten Münzparametern ausgehend erfolgen.

Während des Bezahlvorgangs sieht die Bank  $h_p$ . Es gilt:

- $(h_p/g_1)^\tau = (h_w^\alpha/g_1)^\tau = ((g_1^{\alpha^{-1}} g_2)^\alpha/g_1)^\tau = ((g_1 g_2^\alpha)/g_1)^\tau = (g_2^\alpha)^\tau = (g_2^\tau)^\alpha = y_\tau^\alpha = d$

Um die Identität des Kunden offenzulegen, der die Münze abgehoben hat, wird folgendes Protokoll durchlaufen:

1. Die Bank berechnet zunächst

- $h_p/g_1$

und sendet das Ergebnis an den Trustee.

2. Der Trustee berechnet nun

- $d = (h_p/g_1)^\tau$

und sendet  $d$  an die Bank.

3. Anschließend sucht die Bank  $d$  in den während allen Abhebevorgängen gespeicherten Tupeln  $(d, id, session\#)$  und findet so die gesuchte  $id$  des Kunden.

## 2.3 Umsetzung der Anforderungen

Im folgenden wird das von Camenisch/Maurer/Stadler vorgeschlagene Zahlungssystem bezüglich der in Kap. 1.5 formulierten Anforderungen untersucht. Dabei werden die verwendeten Mechanismen nochmals hervorgehoben.

- **Fälschungssicherheit**

Die Bank akzeptiert Münzen nur dann, wenn  $W = (c_w, s_w)$  ein gültiger Beweis ist, d.h. wenn  $H(c\#||g||h_p||y_i||z_p||g^{s_w}y_i^{c_w}||h_p^{s_w}z_p^{c_w}) = c_w$  gilt. Da der nichtinteraktive "blinde" Beweis (blindPLOGEQ)  $W$  nur mit dem geheimen Münzschlüssel der Bank  $x_i$  berechnet werden kann, ist niemand ohne Mithilfe der Bank in der Lage, gültige Münzen zu erzeugen. Die Sicherheit von  $W$  entspricht der einer blinden Schnorr-Signatur.

- **Mehrfachausgabesicherheit**

Bezahlt ein Kunde mit einer Münze  $(c\#, \alpha, h_p, z_p, s_w, c_w, i, gen\#)$  (bzw. gibt er die Münze zurück), so setzt die Bank die Seriennummer  $c\#$  (bzw.  $h_p$ ) auf die Liste aller eingereichten Münzen. Versucht der Kunde die Münze ein weiteres Mal zum Zahlen zu verwenden bzw. zurückzugeben, stellt die Bank dies anhand der gespeicherten Parameter  $c\#$  oder  $h_p$  fest und lehnt die Münze ab. Da die Seriennummer  $c\#$  zu  $W$  signiert wurde, kann der Kunde diese auch nicht für eine Zahlung manipulieren, ohne daß die Bank dies feststellt. Dasselbe gilt für  $h_p$ , das sowohl in den Beweis  $W$  als auch in einen Beweis  $V$  einfließt.

- **Akzeptanzgarantie**

Der Kunde kann über den nichtinteraktiven Beweis blindPLOGEQ  $W$  der Bank beweisen, daß diese an dem Abhebevorgang beteiligt war und die Münze  $(c\#, \alpha, h_p, z_p, s_w, c_w, i, gen\#)$  regulär abgehoben wurde. Behauptet die Bank, daß die eingereichte Münze mit der Seriennummer  $c\#$  bereits zu einem früheren Zeitpunkt eingereicht wurde, so muß sie den vom Kunden während des besagten Bezahlvorgangs bzw. Rückgabevorgangs gegebenen Beweis  $V' = (s'_v, c'_v)$  vorlegen. Ist  $V'$  ein korrekter Beweis, so hat die Bank ihre Behauptung bewiesen, da  $V'$  nur mit dem privaten Münzschlüssel  $\alpha$  berechnet werden kann, den nur der Kunde kennt. Kann die Bank keinen korrekten Beweis vorlegen, muß sie die eingereichte Münze akzeptieren.

- **Abhebesicherheit**

Durch eine Authentifizierung (z.B. über eine public-key Signatur) kann sichergestellt werden, daß niemand außer dem Kunden selbst Münzen in dessen Namen abheben kann.

- **Ausgabesicherheit**

Sowohl beim Bezahlvorgang als auch beim Rückgabevorgang berechnet der Kunde zu jeder Münze ( $c\#, \alpha, h_p, z_p, s_w, c_w, i, gen\#$ ) einen Beweis  $V = (s_v, c_v)$ . Da dies jeweils nur mit dem privaten Münzschlüssel  $\alpha$  möglich ist, den nur der Kunde kennt und der zu keinem Zeitpunkt des Abhebevorgangs für jemand anderen sichtbar war, kann niemand außer dem Kunden die von ihm abgehobenen Münzen verwenden.

Werden dem Kunden Münzen einschließlich der geheimen Münzschlüssel  $\alpha$  gestohlen, so kann er den Diebstahl melden und eine Münzverfolgung beantragen.

- **Rückgabesicherheit**

Der Kunde sendet beim Rückgabevorgang nach einer Authentifizierung  $c\#, d, h_p, z_p, W, \rho, V$ , den Wert  $i$  und  $gen\#$  an die Bank. Diese prüft wie oben den Beweis  $V$ , der auch die Zusammengehörigkeit der beim Abheben erzeugten Parameter  $d$  und  $h_p$  beweist. Außerdem prüft sie, ob die vorliegende Münze durch den verbundenen Kunden ( $id$ ) abgehoben wurde (d.h. das Tupel  $(id, d)$  existiert im Speicher der regulär abgehobenen Münzen).

- **Bankraubsicherheit**

Zum Schutz vor Raub müssen auf Bankseite - zahlungssystemunabhängig - Sicherheitsmaßnahmen getroffen werden, die den Bankserver vor unbefugten Zugriffen schützen.

Ist es einem Angreifer gelungen die Bankschlüssel zu rauben oder von der Bank Münzen zu erzwingen, so hilft nur eine Rückrufaktion aller sich im Umlauf befindlicher legal erzeugten Münzen der betroffenen Münzgeneration. Dazu stoppt die Bank die Annahme der Münzen dieser Generation im Rahmen von Bezahlvorgängen und fordert die Kunden auf, die betreffenden Münzen zurückzugeben (s. Rückgabesicherheit) und zum Bezahlen neue Münzen der nächsten Generation abzuheben. Diese Vorgehensweise stellt allerdings einen erheblichen Eingriff in den regulären Zahlungsverkehr dar und ist für den Kunden mit einem Mehraufwand verbunden.

- **Anonymität**

Sowohl die Bank als auch niemand anderes kann ohne Beteiligung des Trustee (bzw. ohne dessen privaten Schlüssel  $\tau$ ) eine zum Zahlen verwendete Münze mit einem Abhebevorgang bzw. mit einem Kunden in Verbindung bringen.

Keiner der von der Bank während eines Bezahlvorgangs empfangenen

Münzparameter  $c\#, h_p, z_p, s_w, c_w, \tilde{o}, s_v$  und  $c_v$  war während des Abhebevorgangs sichtbar oder läßt auf die Identität des Kunden schließen. Um die Verbindung zwischen den beim Abheben gesehenen Parametern  $m\#, d, id, h_w, z_w$  zu der eingereichten Münze herzustellen, müßte ein Angreifer die diskreten Logarithmen von  $d = y_\tau^\alpha$  und  $h_p = h_w^\alpha$  berechnen oder DDH lösen, d.h. entscheiden, ob zu  $h_p/g_1 = g_2^\alpha$  und  $y_\tau = g_2^\tau$  gilt:  $d \stackrel{?}{=} g_2^{(\alpha \cdot \tau)}$ .

- **Unverkettbarkeit**

Anhand zweier beliebiger, vom selben Kunden zur Zahlung verwendeter Münzen  $(c\#, \alpha, h_p, z_p, s_w, c_w)$  und  $(c\#', \alpha', h_p', z_p', s_w', c_w')$ , kann nicht erkannt werden, daß sie vom selben Kunden abgehoben und zur Zahlung verwendet wurden, da alle Parameter verschiedener Münzen voneinander unabhängig gewählt  $(c\#, \alpha)$  bzw. gebildet  $(h_p, z_p, s_w, c_w)$  werden.

Um eine Verbindung herzustellen, müßte ein Angreifer die eingereichten Münzen mit den beim Abheben gesehenen Parametern verknüpfen können. Dazu müßte er wiederum DL oder DDH lösen können (Vgl. Anonymität).

- **Kundenverfolgung**

Eine Kundenverfolgung ist über den in Kap. 2.2 (siehe Kundenverfolgung) beschriebenen Mechanismus möglich.

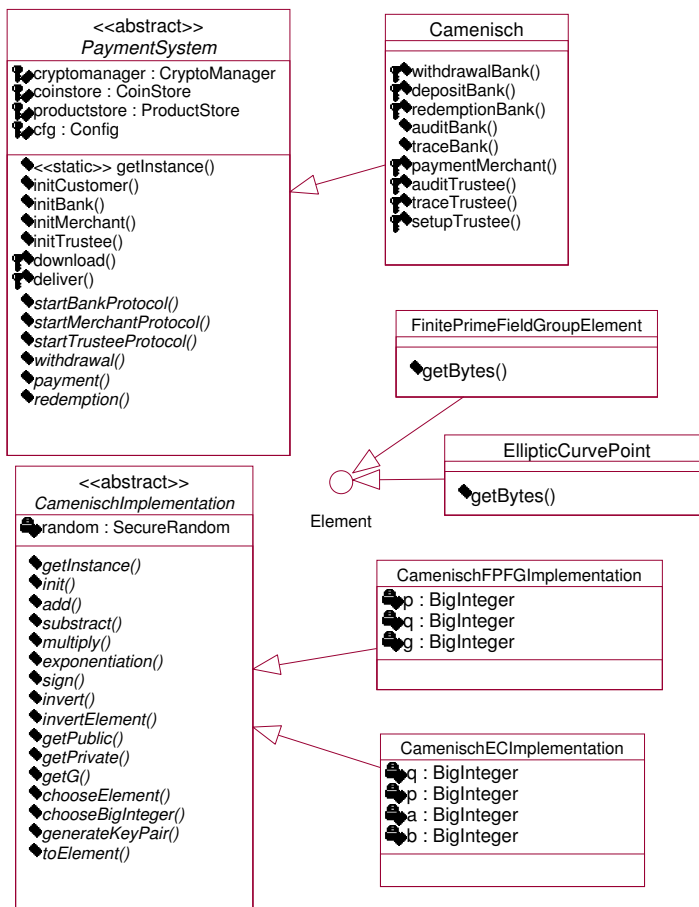
- **Münzverfolgung**

Eine Münzverfolgung ist über den in Kap. 2.2 (siehe Münzverfolgung) beschriebenen Mechanismus möglich.

## 2.4 Implementierung

Wie in den Protokollen (Kap. 2.2) gesehen, wird bei der Wahl der Gruppe  $G$  nur eine prime Ordnung  $q$  gefordert. Außerdem soll es schwierig sein, in  $G$  diskrete Logarithmen zu berechnen (DL) und das DDH-Problem zu lösen. Dies läßt viele Möglichkeiten zur Auswahl offen (z.B.  $q$ -elementige Untergruppe der primen Restklassengruppe oder  $q$ -elementige Untergruppe der Punktegruppe elliptischer Kurven über Primkörpern).

Ziel der Implementierung war es daher, die Protokolle gruppenunabhängig zu implementieren und die eigentlichen Gruppenoperationen über eine abstrakte Klasse bereitzustellen. Damit ergab sich folgende Klassenstruktur:



`class Camenisch extends PaymentSystem`

In der Klasse `Camenisch` sind alle oben beschriebenen Protokolle gruppenunabhängig implementiert. Die Methode `withdrawal` setzt die Kundenseite und `withdrawalBank` die Bankseite des Abhebeprotokolls um. Der Bezahlvorgang

wird auf Kundenseite durch die Methode `payment`, auf Händlerseite durch `paymentMerchant` und auf Bankseite durch `depositBank` implementiert. Die Münzrückgabe wird analog durch `redemption` und `redemptionBank` umgesetzt. Die Protokolle zur Anonymitätsauflösung werden durch `auditBank` und `auditTrustee` (vom Abhebevorgang ausgehende Verfolgung) sowie durch `traceBank` und `traceTrustee` (vom Bezahlvorgang ausgehende Verfolgung) implementiert.

Der Zugriff auf die benötigten Gruppenelemente erfolgt über das Interface `Element`. Auf Gruppenoperationen und anderen gruppenabhängigen Funktionalitäten wird über die abstrakte Klasse `CamenischImplementation` zugegriffen.

Die benötigten Standard-Kryptoverfahren (symmetrische und asymmetrische Verschlüsselung, Hashfunktion, Schlüsselverwaltung) werden über ein `CryptoManager`-Objekt zur Verfügung gestellt. Es dient als Schnittstelle zu den installierten Kryptoprovidern der Java Cryptography Architecture (JCA/JCE)(Vgl. 1.3 & A.1.6).

### **abstract class CamenischImplementation**

Die abstrakte Klasse `CamenischImplementation` dient als Schnittstelle zwischen den in der Klasse `Camenisch` gruppenunabhängig implementierten Protokollen und den darin benötigten konkreten Operationen auf der jeweils gewählten Gruppe, die z.B. von der Klasse `CamenischFPFGImplementation` bereitgestellt werden.

Um ein Objekt einer ihrer Erben zu generieren, bietet die Klasse `CamenischImplementation` eine `getInstance` Methode.

### **class CamenischFPFGImplementation**

**extends CamenischImplementation**

Die Klasse `CamenischFPFGImplementation` implementiert die Gruppenoperationen und andere gruppenabhängige Funktionalitäten, die in den Protokollen des Zahlungssystems benötigt werden, für q-elementige Untergruppen primärer Restklassengruppen.

Neben den klassischen Gruppenoperationen (Addition, Multiplikation, Exponentiation, Invertierung, ...) bietet sie Methoden zur zufälligen Wahl von Gruppenelementen (`chooseElement`) bzw. passender Exponenten (`chooseBigInteger`). Außerdem stellt sie Methoden zur Schlüsselgenerierung (`generateKeyPair`) und Extraktion der Schlüsselparameter (`getPublic`, `getPrivate`, `getG`) zur Verfügung.

Um ein Objekt der Klasse `CamenischFPFGImplementation` für eine konkrete Gruppe zu initialisieren, muß die Methode `init` ausgeführt werden.

**class CamenischECImplementation**

**extends** CamenischImplementation

Die Klasse `CamenischECImplementation` bietet die gleiche Funktionalität wie `CamenischFPFGImplementation` für  $q$ -elementige Untergruppen der Punktgruppe elliptischer Kurven über Primkörpern.

**interface Element**

Das Interface `Element` dient als Schnittstelle zwischen den in der Klasse `Camenisch` gruppenunabhängig implementierten Protokollen und den konkreten Repräsentationen von Gruppenelementen.

**class FinitePrimeFieldGroupElement**

**extends** BigInteger **implements** Element

Die Klasse `FinitePrimeFieldGroupElement` implementiert ein Element einer beliebigen primen Restklassengruppe.

Um das Gruppenelement senden und empfangen zu können, bietet die Klasse die Methode `getBytes`, die eine `byte[]`-Repräsentation des Elements erzeugt.

**class EllipticCurvePoint**

**extends** cdc.ec.java.PointGFP **implements** Element

Die Klasse `EllipticCurvePoint` implementiert einen Punkt einer beliebigen elliptischen Kurve über einem Primkörper.

Um den Punkt senden und empfangen zu können, bietet die Klasse die Methode `getBytes`, die eine `byte[]`-Repräsentation des Punkts erzeugt.

Die Klassen `CamenischFPFGImplementation` und `CamenischECImplementation` (bzw. `FinitePrimeFieldGroupElement` und `EllipticCurvePoint`) sind hier nur zwei von vielen denkbaren Möglichkeiten gruppenspezifischer Implementierungen.

# Kapitel 3

Jakobsson/Yung

## 3.1 Einführung

Bei dem von Jakobsson und Yung in [8] vorgestellten Zahlungssystem ist der Trustee sowohl bei der Initialisierung und Deanonymisierung als auch beim Abhebevorgang aktiv beteiligt (aktiver Trustee, vgl. Kap. 1.1.4).

### 3.1.1 Grundidee

Der Abhebevorgang besteht aus einem Protokoll zwischen Kunden, Bank und Trustee, wobei die Kommunikation zwischen Kunden und Trustee, unter Verwendung symmetrischer Verschlüsselung, über die Bank geschleust wird, ohne daß diese die übermittelten Daten einsehen kann und ohne daß der Trustee die Identität des Kunden sieht. Mit anderen Worten anonymisiert die Bank also die Kommunikation zwischen Kunden und Trustee und der Trustee hebt die gewünschten Münzen für den Kunden bei der Bank ab. Zur Realisierung dieser Kommunikation zwischen Kunden, Bank und Trustee werden folgende Standard-Kryptoverfahren benötigt:

- Ein symmetrisches Verschlüsselungsverfahren  $(E_k, D_k)$ , wobei  $k$  der symmetrische Schlüssel ist.

*Das verwendete Verfahren muß folgende Eigenschaft erfüllen:*

*Für Schlüssel  $K_1, K_2$ , Botschaften  $m_1, m_2$  mit  $(m_1 \neq m_2)$  und die resultierende Ciphertexte  $c_1 = E_{K_1}(m_1)$  und  $c_2 = E_{K_2}(m_2)$  darf es nicht möglich sein, einen Schlüssel  $K$  zu finden, für den  $c_2 = E_K(m_1)$  gilt.*

- Ein asymmetrisches Verschlüsselungsverfahren für die Bank  $(E_{bank}, D_{bank})$  und für den Trustee  $(E_{trustee}, D_{trustee})$ .

*Das verwendete Verfahren muß Semantische Sicherheit bieten. D.h. Ciphertexte müssen unter einer Chosen Plaintext Attacke ununterscheidbar sein. Einem Angreifer (dem Trustee) darf es nicht möglich sein zu raten, ob eine bestimmte Botschaft ( $id$ ) verschlüsselt wurde und dies zu überprüfen, indem er diese selber verschlüsselt und die Ciphertexte vergleicht.*

Daneben werden folgende Signaturverfahren benötigt:

- Ein blindes Signaturverfahren  $(S_i, V_i)$ , mit dem die Bank die Münze blind signiert und damit beweist, daß sie an der Erzeugung der Münze beteiligt war.
- Ein Signaturverfahren  $(S_{trustee}, V_{trustee})$ , mit dem der Trustee die erzeugte Münze ebenfalls signiert und damit beweist, daß er an der Erzeugung der Münze beteiligt war.

*Das verwendete Verfahren darf nicht transparent blendbar sein. D.h., der Trustee muß sicher sein, daß er nicht blind signiert (not transparently blindable).*

- Ein Signaturverfahren  $(S_{coin}, V_{coin})$ , mit dem der Kunde beim Bezahlvorgang die Annahme des Händlerangebots (bzw. beim Rückgabevorgang die Rückgabe) signiert sowie beweist, daß die verwendeten Münzen ihm gehören.

Eine Münze besteht bei diesem Zahlungssystem im Wesentlichen aus dem Münzschlüsselpaar  $(x_{coin}, y_{coin})$  für das Signaturverfahren  $(S_{coin}, V_{coin})$ , dem Münzschlüssel  $K_{coin}$  für das symmetrische Verschlüsselungsverfahren  $(E_{K_{coin}}, D_{K_{coin}})$  sowie aus den Signaturen  $s$  und  $\sigma$ , für die  $V_i(s) = \sigma$  bzw.  $V_{trustee}(\sigma) = y_{coin}$  gilt.

Während des Abhebevorgangs speichert der Trustee die von der Bank vergebene Sitzungsnummer  $m\#$  sowie  $K_{trustee}, y_{coin}, K_{coin}$  und das mit  $E_{bank}$  verschlüsselte  $\bar{id}$ . Die Bank speichert ihrerseits u.a.  $m\#, id$  und das mit  $E_{K_{coin}}$  verschlüsselte  $\bar{s}$ . Zur Deanonymisierung können dann im Falle eines begründeten Verdachtes passende  $m\#$  bzw.  $y_{coin}$  in den jeweiligen Datenbanken gesucht werden.

Um Mehrfachausgabe verhindern zu können, speichert die Bank das Paar  $(y_{coin}, s)$  der im Rahmen eines Bezahl- oder Rückgabevorgangs eingereichten Münzen.

### 3.1.2 Sicherheit

Die Sicherheit des von Jakobsson und Yung in [8] vorgestellten Zahlungssystems hängt von der Sicherheit der verwendeten Verschlüsselungs- und Signaturverfahren ab.

Mögliche symmetrische Verschlüsselungsverfahren sind z.B. DESede oder RC6. Als asymmetrische Verschlüsselungsverfahren sind z.B. RSA oder ElGamal und als Signaturverfahren bzw. blindes Signaturverfahren sind z.B. Schnorr- bzw. blinde Schnorr Signatur einsetzbar.

### 3.1.3 Änderungen

Die in [8] vorgestellten Protokolle wurden wie folgt modifiziert:

- Statt eines "message recovery" Signaturverfahrens, das bei der Verifikation die signierte Botschaft aus der Signatur rekonstruiert, benutzen wir für  $(S_i, V_i)$  ein Standard-Signaturverfahren. D.h., die signierte Botschaft  $(\sigma)$  muß mit in der Münze gespeichert werden.

- Um einerseits Rückgabesicherheit zu garantieren, aber im Gegenzug nicht bei Rückgabe einer Münze der Bank die Möglichkeit zu geben, alle Münzen dieser Abhebesitzung wiederzuerkennen, verwenden wir beim Abhebevorgang für jede Münze einen eigenen Schlüssel  $K_{coin}$ , um  $\bar{s}$  und  $\bar{\sigma}$  zu bilden, statt für alle Münzen den Sitzungsschlüssel  $K_{trustee}$  zu verwenden.
- Da im Rahmen dieser Arbeit eine online-Variante untersucht wird, ist der Bezahlvorgang gegenüber dem Original-Protokoll dahingehend verändert, daß der Händler die empfangenen Münzen nicht zu prüfen braucht, da dies unmittelbar durch die Bank geschieht.
- Statt eines zufälligen "challenges"  $c$ , signieren wir beim Bezahlvorgang mit  $S_{coin}$  die Annahme des Angebots.
- Ein Rückgabevorgang im hier verwendeten Sinne wurde in [8] nicht vorgeschlagen.

## 3.2 Protokolle

Im folgenden Abschnitt werden die Protokolle des Zahlungssystems im Detail vorgestellt.

### Setup/Initialisierung

Die Bank wählt zunächst für jede neue Münzgeneration pro Münzwert  $i$  ein Schlüsselpaar für blinde Signaturen  $(S_i, V_i)$ . Außerdem benötigt die Bank ein Schlüsselpaar zur Ver- und Entschlüsselung  $(E_{bank}, D_{bank})$ .

Der Trustee benötigt ebenfalls ein Schlüsselpaar zur Ver- und Entschlüsselung  $(E_{trustee}, D_{trustee})$  sowie ein Schlüsselpaar für das Signaturverfahren  $(S_{trustee}, V_{trustee})$ . Sowohl Bank als auch Trustee publizieren ihre öffentlichen Schlüssel. Die verwendeten Signatur- bzw. Verschlüsselungsverfahren können beliebig gewählt und kombiniert werden.

### Abheben

Um Münzen abzuheben, muß sich der Kunde zunächst mit der Bank verbinden. Dabei findet eine Authentifizierung statt. Die Bank ihrerseits muß sich mit dem Trustee verbinden (bzw. in Verbindung stehen). Anschließend erzeugen Kunde, Bank und Trustee gemeinsam die Münzen der Sorte  $i$  wie folgt:

1. Der Kunde wählt einen Sitzungsschlüssel  $K_{trustee}$  zur symmetrischen Verschlüsselung und verschlüsselt diesen wie folgt:

- $\bar{K}_{trustee} = E_{trustee}(K_{trustee})$

Zusätzlich wählt er für jede Münze einen Schlüssel zur symmetrischen Verschlüsselung  $K_{coin}$  sowie ein Schlüsselpaar  $(x_{coin}, y_{coin})$  für das Signaturverfahren  $(S_{coin}, V_{coin})$ . (Aus Effizienzgründen bietet es sich hier an, dasselbe Signaturverfahren und dieselben Systemparameter zu benutzen, die die Bank für ihre Münzschlüssel verwendet.)

$K_{coin}$  und den öffentlichen Münzschlüssel  $y_{coin}$  verschlüsselt er anschließend mit dem Sitzungsschlüssel  $(K_{trustee})$  zu  $\bar{K}_{coin} = E_{K_{trustee}}(K_{coin})$  und  $\bar{y}_{coin} = E_{K_{trustee}}(y_{coin})$ .

Die ihn repräsentierende  $id$  verschlüsselt er im Anschluß zweifach wie folgt:

- $\bar{id} = E_{bank}(id)$
- $\overline{\bar{id}} = E_{K_{trustee}}(\bar{id})$

Nun sendet der Kunde  $\overline{K}_{trustee}$ ,  $\overline{K}_{coin}$ ,  $\overline{y}_{coin}$  und  $\overline{id}$  an die Bank.

2. Die Bank wählt zunächst eine eindeutige Sitzungsnummer  $m\#$  und sendet das erhaltene  $\overline{K}_{trustee}$ ,  $\overline{K}_{coin}$ ,  $\overline{y}_{coin}$ ,  $\overline{id}$  zusammen mit  $m\#$  an den Trustee.
3. Der Trustee entschlüsselt zuerst  $K_{trustee} = D_{trustee}(\overline{K}_{trustee})$ . Mit dem so erhaltenen Sitzungsschlüssel entschlüsselt er:
  - $y_{coin} = D_{K_{trustee}}(\overline{y}_{coin})$ ,
  - $K_{coin} = D_{K_{trustee}}(\overline{K}_{coin})$ ,
  - $\overline{id} = D_{K_{trustee}}(\overline{id})$ .

Die so erhaltene  $\overline{id}$  sendet er an die Bank.

4. Die Bank überprüft, ob  $id = D_{bank}(\overline{id})$  mit der während der Authentifizierung festgestellten  $id$  übereinstimmt. Ist dies nicht der Fall, so wird der Abhebevorgang abgebrochen.
5. Der Trustee signiert den öffentlichen Münzschlüssel  $y_{coin}$  und läßt die erhaltene Signatur  $\sigma = S_{trustee}(y_{coin})$  blind von der Bank signieren. *Der Trustee und die Bank beweisen durch diese Signaturen, daß sie an der Erzeugung der Münze beteiligt waren.* Anschließend überprüft er, ob für die erhaltene Signatur  $s = S_i(\sigma)$  gilt:
  - $V_i(s) = \sigma^1$

Schlägt die Verifikation fehl, so wird der Abhebevorgang abgebrochen. Sonst speichert der Trustee  $m\#$ ,  $K_{trustee}$ ,  $y_{coin}$ ,  $K_{coin}$  sowie  $\overline{id}$  in seine Datenbank und verschlüsselt  $s$  und  $\sigma$  zu:

- $\overline{s} = E_{K_{coin}}(s)$
- $\overline{\sigma} = E_{K_{coin}}(\sigma)$

Zum Schluß sendet er  $\overline{s}$  und  $\overline{\sigma}$  an die Bank.

6. Die Bank speichert  $m\#$ ,  $id$ ,  $i$ ,  $\overline{s}$ ,  $\overline{\sigma}$ ,  $\overline{id}$  und einen Verweis auf ihre Generation  $gen\#$  in ihrer Datenbank. Anschließend belastet sie das Konto des Kunden mit dem Wert der gerade erzeugten Münze der Sorte  $i$ . Zum Abschluß sendet sie  $\overline{s}$ ,  $\overline{\sigma}$  an den Kunden.

---

<sup>1</sup>Auf diese Prüfung kann an dieser Stelle auch verzichtet werden, da sie vom Kunden ebenfalls durchgeführt wird.

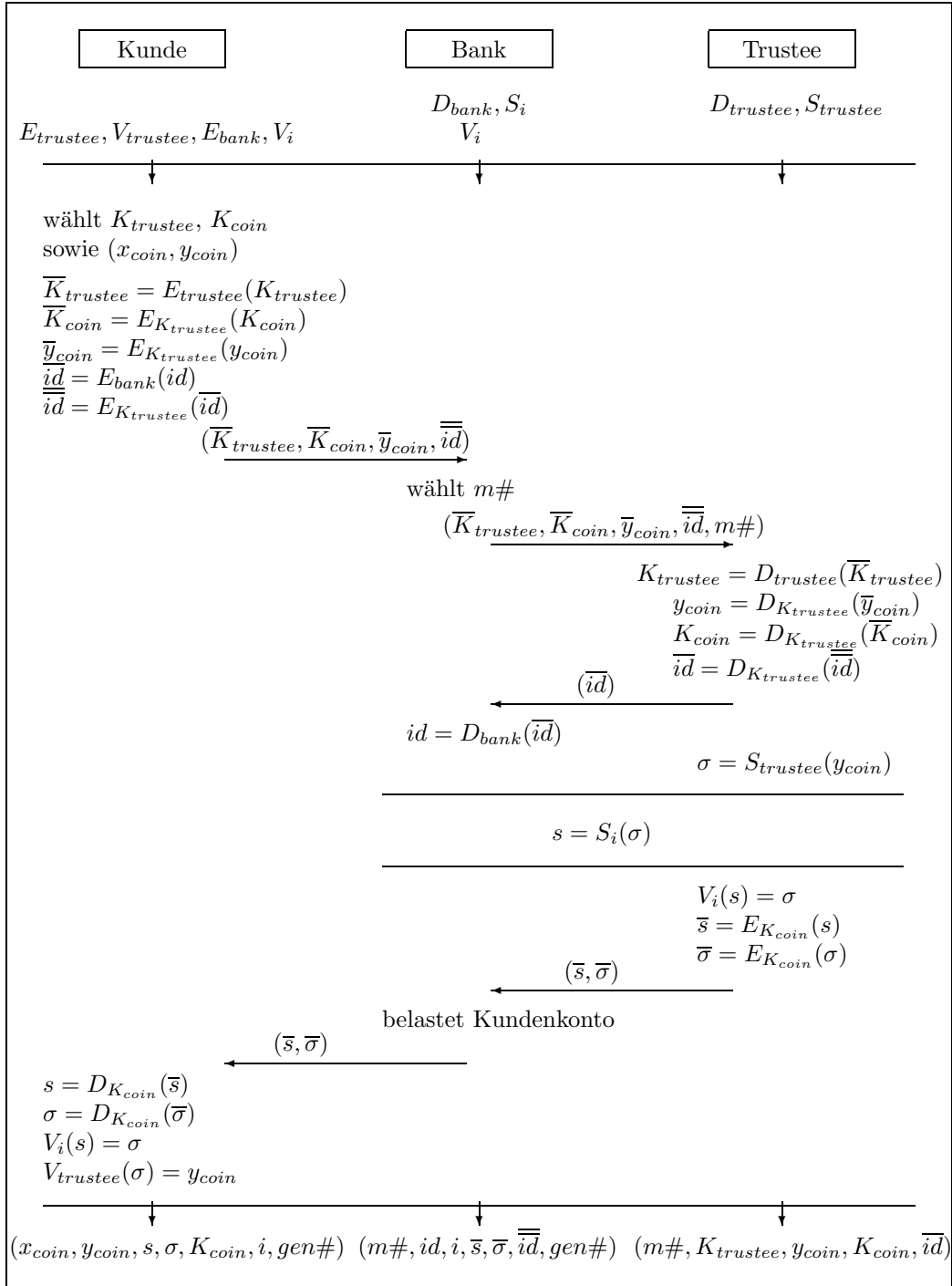


Abbildung 3.1: Abhebeprotokoll

7. Der Kunde empfängt  $\bar{s}$  und  $\bar{\sigma}$  von der Bank und entschlüsselt  $s = D_{K_{coin}}(\bar{s})$  sowie  $\sigma = D_{K_{coin}}(\bar{\sigma})$  und überprüft die Signaturen:

- $V_i(s) = \sigma$
- $V_{trustee}(\sigma) = y_{coin}$

Schlägt einer dieser Tests fehl, so sendet der Kunde alle während dieser Session abgehobenen Münzen durch Offenlegen von  $s$ ,  $\sigma$ ,  $K_{coin}$  und  $K_{trustee}$  an die Bank zurück<sup>2</sup>.

Sonst speichert er die erzeugte Münze, bestehend aus  $x_{coin}$ ,  $y_{coin}$ ,  $s$ ,  $\sigma$ ,  $K_{coin}$  sowie dem Wert der Münze und einem Verweis auf ihre Generation  $gen\#$ .

### Bezahlen

Der Bezahlvorgang besteht aus einem Protokoll zwischen Kunden, Händler und Bank. Vor dem Protokoll sendet der Händler zunächst sein Angebot  $o$  an den Kunden.

Zum besseren Verständnis des Protokolls nehmen wir im Folgenden zunächst an, daß nur mit einer einzigen Münze bezahlt wird.

1. Der Kunde berechnet zunächst den Hashwert  $h(o)$  und holt die benötigte Münze (bestehend aus  $x_{coin}$ ,  $y_{coin}$ ,  $s$ ,  $\sigma$ ,  $K_{coin}$  sowie dem Wert  $i$  der Münze und einem Verweis auf ihre Generation  $gen\#$ ) aus seinem Münzspeicher. Anschließend berechnet er die Signatur  $a = S_{coin}(h(o))$  und sendet  $y_{coin}$ ,  $s$ ,  $\sigma$ ,  $a$ ,  $i$  und  $gen\#$  unter Angabe des Kaufvertrags  $o$  an den Händler.

*Der Kunde signiert durch  $a$  die Annahme des Händlerangebots  $o$  und beweist gleichzeitig, daß die verwendete Münze ihm gehört.*

2. Der Händler bildet ebenfalls den Hashwert  $h(o)$  und reicht die erhaltene Münze (bestehend aus  $y_{coin}$ ,  $s$ ,  $\sigma$ ,  $a$ ,  $i$  und  $gen\#$ ) zusammen mit  $h(o)$  an die Bank weiter.
3. Die Bank verifiziert die Signaturen

- $V_i(s) = \sigma$

---

<sup>2</sup>Die Bank prüft dann, ob  $D_{K_{trustee}}(\bar{K}_{coin}) = K_{coin}$ ,  $D_{K_{trustee}}(\bar{y}_{coin}) = y_{coin}$ ,  $D_{K_{coin}}(\bar{s}) = s$  und  $D_{K_{coin}}(\bar{\sigma}) = \sigma$  gelten. Ist dies der Fall und  $s$  oder  $\sigma$  keine korrekte Signatur, so schreibt sie dem Kunden den abgehobenen Betrag wieder auf seinem Konto gut. Dieses zusätzliche Rückgabeprotokoll ist nur für die Rückgabe aller Münzen einer Abhebesession sinnvoll anwendbar, da durch das Offenlegen des Sessionkeys  $K_{trustee}$  die Anonymität aller Münzen der Abhebesession aufgehoben wird.

- $V_{trustee}(\sigma) = y_{coin}$
- $V_{coin}(a) = h(o)$

Außerdem prüft sie, ob

- die Münze mit  $y_{coin}$ ,  $s$  nicht bereits früher zum Zahlen verwendet wurde und
- die Münze mit dem Tupel  $(y_{coin}, s)$  nicht durch die Bank blockiert wurde (*siehe Münzverfolgung*) und
- der Einreichzeitpunkt in der Akzeptanzphase der Münze liegt.

Schlägt einer dieser Tests fehl, wird der Bezahlvorgang abgebrochen. Sonst schreibt die Bank dem Händler den Wert der Münze auf seinem Konto gut und speichert  $y_{coin}$ ,  $s$ ,  $h(o)$  und  $a$  in ihrer Datenbank. Abschließend sendet sie eine Bestätigung der Gutschrift an den Händler.

4. Nach Eingang der Bestätigung vergleicht der Händler den gutgeschriebenen Betrag mit dem im Kaufvertrag vereinbarten Kaufpreis. Stimmen die Beträge überein, erfolgt die Auslieferung der Ware. Sonst erfolgt eine Benachrichtigung des Kunden.

In der Regel werden mehrere Münzen benötigt, um einen Kauf zu tätigen. Das obige Protokoll kann dann effizient parallelisiert werden.

### Rückgabe

Der Rückgabevorgang besteht aus einem Protokoll zwischen Kunden und Bank. Zum besseren Verständnis des Protokolls nehmen wir im Folgenden an, daß nur eine einzige Münze zurückgegeben wird.

1. Der Kunde holt zunächst die zurückzugebende Münze (bestehend aus  $x_{coin}$ ,  $y_{coin}$ ,  $s$ ,  $\sigma$ ,  $K_{coin}$  sowie dem Wert  $i$  der Münze und einem Verweis auf ihre Generation  $gen\#$ ) aus seinem Münzspeicher. Anschließend wählt er eine zufällige Zahl  $\rho$  und berechnet
  - $a = S_{coin}(\rho)$ .

Nun sendet er  $y_{coin}$ ,  $\sigma$ ,  $s$ ,  $K_{coin}$ ,  $\rho$ ,  $a$ , den Wert  $i$  und  $gen\#$  an die Bank. *Der Kunde signiert durch  $a$  die Rückgabe. Durch Offenlegen von  $K_{coin}$  wird es der Bank ermöglicht, die Verbindung zwischen der eingereichten Münze und den von der Bank während des Abhebevorgangs gespeicherten Münzparametern herzustellen.*

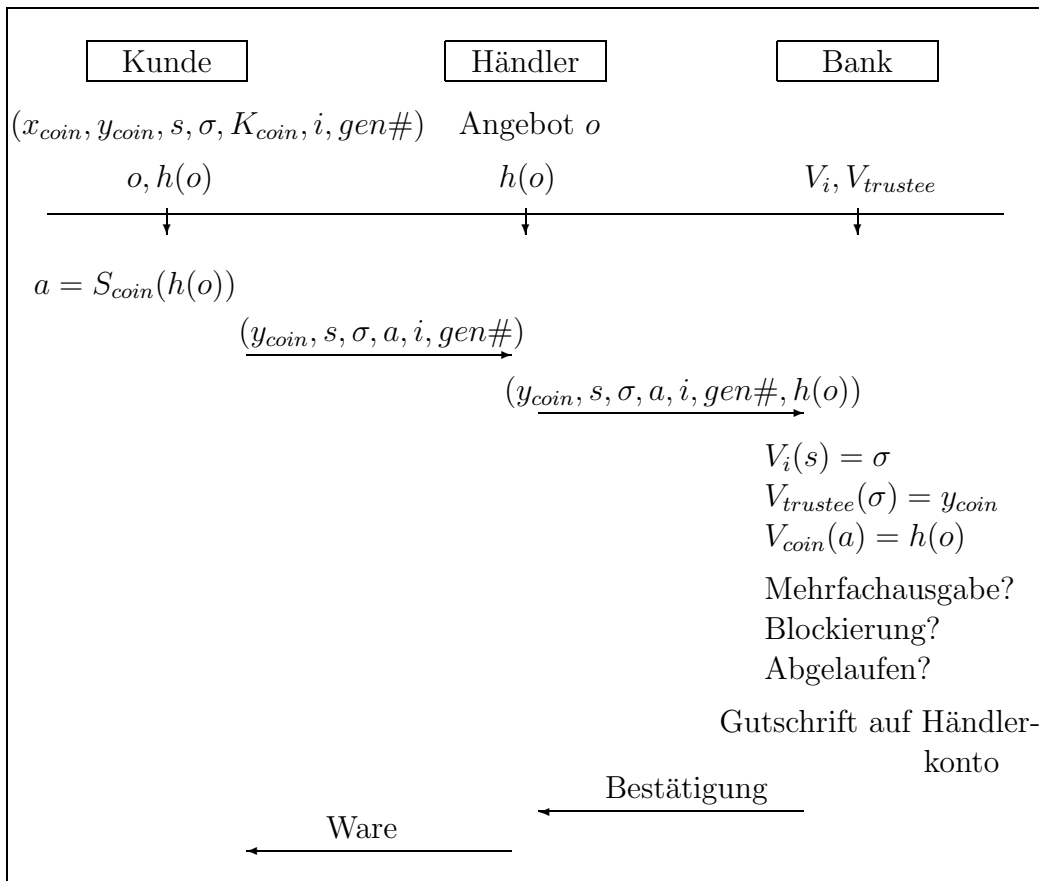


Abbildung 3.2: Bezahlprotokoll

2. Die Bank verifiziert die erhaltene Münze, indem sie prüft, ob

- $V_i(s) = \sigma$  und
- $V_{trustee}(\sigma) = y_{coin}$  sowie
- $V_{coin}(a) = \rho$  gelten.

Außerdem berechnet sie

- $E_{K_{coin}}(s) = \bar{s}$

und prüft, ob

- das Paar  $(id, \bar{s})$  in ihrem Speicher der regulär abgehobenen Münzen gespeichert ist und
- die durch das Tupel  $(y_{coin}, s)$  repräsentierte Münze nicht bereits früher zum Zahlen verwendet wurde sowie

- der Rückgabezeitpunkt innerhalb der Rückgabephase der Münze liegt.

Schlägt einer dieser Tests fehl, so wird die Münze zurückgewiesen und der Rückgabevorgang abgebrochen.

Sonst schreibt die Bank dem Kunden den Wert der Münze auf seinem Konto gut und speichert  $y_{coin}$ ,  $s$ ,  $\rho$  und  $a$  in ihrer Datenbank. Abschließend sendet sie eine Bestätigung der Gutschrift an den Kunden.

3. Der Kunde überprüft abschließend den gutgeschriebenen Betrag.

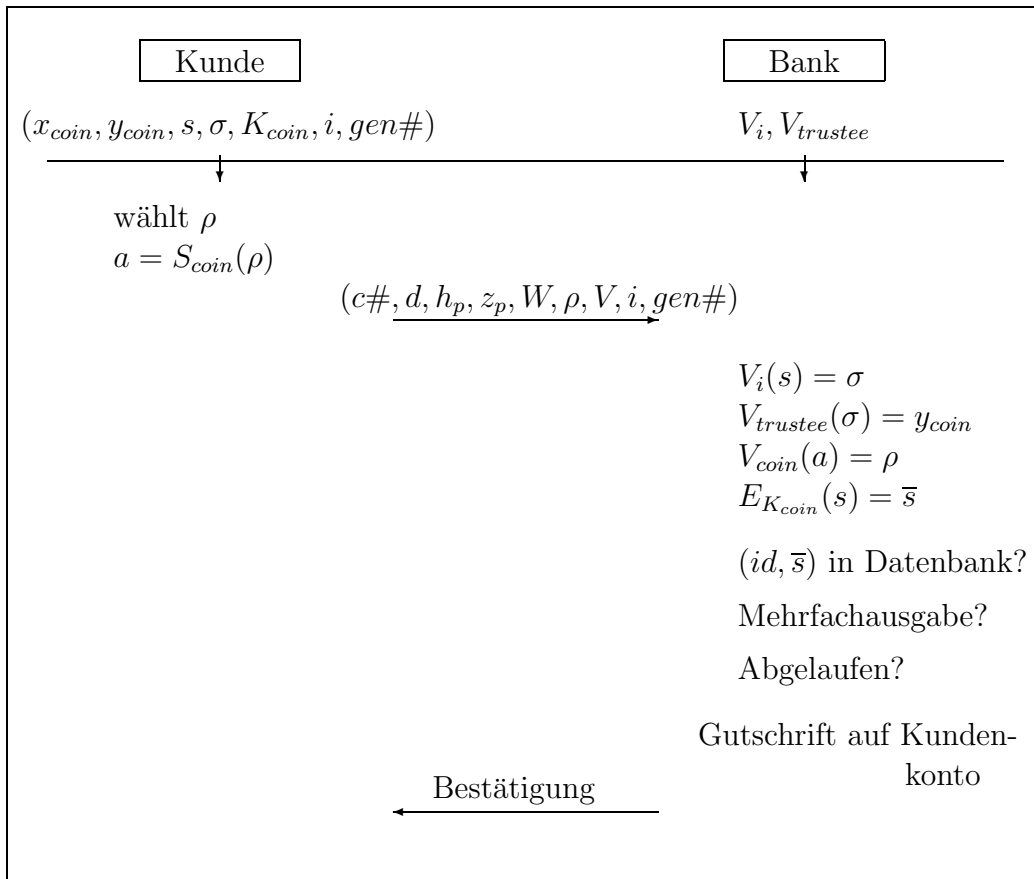


Abbildung 3.3: Rückgabeprotokoll

### Münzverfolgung

Die Münzverfolgung besteht aus einem Protokoll zwischen Bank und Trustee. Sie kann von den während des Abhebevorgangs von Bank und Trustee

beobachteten Münzparametern ausgehend erfolgen.  
Als Voraussetzung muß eine richterliche Legitimation vorliegen.

Während des Abhebevorgangs speichert die Bank die Sitzungsnummer des Abhebevorgangs  $m\#$  und die  $id$  des Kunden sowie  $\bar{s}$ ,  $\bar{\sigma}$ ,  $\bar{id}$ ,  $gen$ ,  $i$ . Um alle nicht auf legalem Wege abgehobenen Münzen eines Abhebevorgangs während eines Zahlungsvorgangs oder Rückgabevorgangs identifizieren zu können, wird folgendes Protokoll durchlaufen:

1. Die Bank sendet die richterliche Legitimation sowie  $m\#$  an den Trustee.
2. Der Trustee prüft die richterliche Legitimation. Anschließend sucht er zum empfangenen  $m\#$  in seiner Datenbank die während des Abhebevorgangs gespeicherten Tupel  $(m\#, K_{trustee}, y_{coin}, K_{coin}, \bar{id})$  und sendet  $K_{coin}$  an die Bank.
3. Die Bank sucht nun in ihrer Datenbank die passenden Tupel  $(m\#, id, \bar{s}, \bar{\sigma}, \bar{id})$  und berechnet:

- $s = D_{K_{coin}}(\bar{s})$

Abschließend setzt die Bank die Tupel  $(y_{coin}, s)$  auf eine Liste nicht zu akzeptierender Münzen. Bei jedem Zahlungsvorgang und jedem Rückgabevorgang wird das Tupel  $(y_{coin}, s)$  der eingereichten Münze dahingehend überprüft, ob es in dieser Liste vorkommt. Sollte dies der Fall sein, so wird die Münze abgewiesen.

### Kundenverfolgung

Die Kundenverfolgung besteht aus einem Protokoll zwischen Bank und Trustee. Sie kann von den während des Bezahlvorgangs von der Bank beobachteten Münzparametern ausgehend erfolgen.

Während des Bezahlvorgangs sieht die Bank  $y_{coin}$ ,  $s$ ,  $\sigma$ , den Wert der Münze  $i$ ,  $gen\#$  sowie das Paar  $(h(o), a)$ . Um die Identität des Kunden offenzulegen, der die Münze abgehoben hat, wird folgendes Protokoll durchlaufen:

1. Die Bank sendet die richterliche Legitimation sowie  $y_{coin}$  an den Trustee.
2. Der Trustee prüft die richterliche Legitimation. Anschließend sucht er zum empfangenen  $y_{coin}$  in seiner Datenbank das während des Abhebevorgangs gespeicherte Tupel  $(m\#, K_{trustee}, y_{coin}, K_{coin}, \bar{id})$  und sendet  $m\#, K_{trustee}$  und  $\bar{id}$  an die Bank.

3. Die Bank sucht nun wie oben in ihrer Datenbank die passenden Tupel  $(m\#, id, \bar{s}, \bar{\sigma}, \overline{id})$  und prüft, ob

- $\overline{id} = D_{K_{trustee}}(\overline{id})$
- $id = D_{bank}(\overline{id})$  gilt.

Somit hat die Bank die gesuchte  $id$  des Kunden.

### 3.3 Umsetzung der Anforderungen

Im folgenden wird das von Jakobsson/Yung vorgeschlagene Zahlungssystem bezüglich der in Kap. 1.5 formulierten Anforderungen untersucht. Dabei werden wie im vorangehenden Kapitel (2.3) die zur Erfüllung der Anforderungen verwendeten Mechanismen nochmals hervorgehoben.

- **Fälschungssicherheit**

Die Bank akzeptiert eine Münze  $(y_{coin}, s, \sigma, a, i, gen\#)$  nur dann, wenn  $V_i(s) = \sigma$  und  $V_{trustee}(\sigma) = y_{coin}$  gültige Signaturen sind. Da die Signatur  $s = S_i(\sigma)$  nur mit dem privaten Schlüssel der Bank und die Signatur  $\sigma = S_{trustee}(y_{coin})$  nur mit dem privaten Schlüssel des Trustee berechnet werden kann, ist niemand ohne die Mithilfe von Bank und Trustee in der Lage, gültige Münzen zu erzeugen.

- **Mehrfachausgabesicherheit**

Bezahlt ein Kunde mit einer Münze  $(y_{coin}, s, \sigma, a, i, gen\#)$  so setzt die Bank das Tupel  $(y_{coin}, s)$  auf die Liste aller eingereichten Münzen.

Versucht der Kunde die Münze ein weiteres Mal zum Zahlen zu verwenden bzw. zurückzugeben, stellt die Bank dies anhand des gespeicherten Tupels fest und lehnt die Münze ab. Da das Tupel  $(y_{coin}, s)$  über die Signaturen  $V_i(s) = \sigma$  und  $V_{trustee}(\sigma) = y_{coin}$  geprüft wird, kann der Kunde diese auch nicht für eine Zahlung bzw. Rückgabe manipulieren, ohne daß die Bank dies feststellt.

- **Akzeptanzgarantie**

Der Kunde kann über die blinde Signatur der Bank  $V_i(s) = \sigma$  beweisen, daß die Bank an dem Abhebevorgang beteiligt war und die Münze  $(x_{coin}, y_{coin}, s, \sigma, a, i, gen\#)$  regulär abgehoben wurde.

Behauptet die Bank, daß die durch das Tupel  $(y_{coin}, s)$  repräsentierte Münze bereits zu einem früheren Zeitpunkt eingereicht wurde, so muß sie die vom Kunden während des besagten Bezahlvorgangs bzw. Rückgabevorgangs gegebene Signatur  $a'$  vorlegen. Ist  $a'$  eine korrekte Signatur, so hat die Bank ihre Behauptung bewiesen, da  $a'$  nur mit dem privaten Münzschlüssel  $x_{coin}$  berechnet werden kann, den nur der Kunde kennt. Kann die Bank keine korrekte Signatur vorlegen, muß sie die eingereichte Münze akzeptieren.

- **Abhebesicherheit**

Durch eine Authentifizierung (z.B. über eine public-key Signatur) kann sichergestellt werden, daß niemand außer dem Kunden selbst Münzen in dessen Namen abheben kann.

- **Ausgabesicherheit**

Beim Bezahlvorgang berechnet der Kunde zu jeder Münze die Signatur  $a = S_{coin}(h(o))$ . Da dies nur mit dem privaten Münzschlüssel  $x_{coin}$  möglich ist, den nur der Kunde kennt und der zu keinem Zeitpunkt des Abhebevorgangs für jemand anderen sichtbar war, kann niemand außer dem Kunden die von ihm abgehobenen Münzen zum Bezahlen verwenden.

Beim Rückgabevorgang beweist der Kunde nach einer Authentifizierung durch Offenlegen des symmetrischen Münzschlüssels  $K_{coin}$  den Zusammenhang zwischen der eingereichten Münze ( $y_{coin}, \sigma, s, i, gen\#$ ) und der von ihm abgehobenen Münze. Die Bank prüft den Zusammenhang, indem sie  $\bar{s} = E_{K_{coin}}(s)$  berechnet und mit den während des Abhebevorgangs gespeicherten Münzparametern ( $m\#, id, i, \bar{s}, \bar{\sigma}, \bar{id}$ ) vergleicht.

Werden dem Kunden Münzen einschließlich der geheimen Münzschlüssel  $x_{coin}$  gestohlen, so kann er den Diebstahl melden und eine Münzverfolgung beantragen.

- **Rückgabesicherheit**

Der Kunde sendet beim Rückgabevorgang nach einer Authentifizierung  $y_{coin}, \sigma, s, K_{coin}, \rho, a$ , den Wert  $i$  und  $gen\#$  an die Bank. Diese prüft, wie oben beschrieben, den Zusammenhang zwischen der eingereichten Münze und der vom selben Kunden abgehobenen Münze, indem sie  $\bar{s} = E_{K_{coin}}(s)$  berechnet und mit den während des Abhebevorgangs gespeicherten Münzparametern ( $m\#, id, i, \bar{s}, \bar{\sigma}, \bar{id}$ ) vergleicht.

- **Bankraubsicherheit**

Zum Schutz vor Raub müssen auf Bankseite - zahlungssystemunabhängig - Sicherheitsmaßnahmen getroffen werden, die den Bankserver vor unbefugten Zugriffen schützen.

Ist es einem Angreifer gelungen die Bankschlüssel zu rauben oder von der Bank Münzen zu erzwingen, so muß die Bank alle nach diesem Vorfall durchgeführten Bezahlvorgänge unter Einbezug des Trustee vollziehen. Der Trustee überprüft in diesem Fall die Signatur  $V_{trustee}(\sigma) = y_{coin}$  und sieht in seiner Datenbank nach, ob  $y_{coin}$  existiert, d.h., ob die eingereichte Münze ( $y_{coin}, s, \sigma, a, i$  und  $gen\#$ ) in einem legalen Abhebevorgang erzeugt wurde. Ist dies nicht der Fall, so wird die Münze abgelehnt und der Bezahlvorgang abgebrochen.

- **Anonymität**

Sowohl die Bank als auch niemand anderes kann ohne Beteiligung des

Trustee (bzw. ohne Zugriff auf die vom Trustee während des Abhebevorgangs gespeicherten Parameter) eine zum Zahlen verwendete Münze mit einem Abhebevorgang bzw. mit einem Kunden in Verbindung bringen. Alle Münzparameter, die dies ermöglichen würden, waren für die Bank während des Abhebevorgangs nur mit  $K_{coin}$  bzw.  $K_{trustee}$  verschlüsselt sichtbar bzw. wurden von ihr blind signiert ( $S_i(\sigma) = s$ ).

Auch der Trustee kann ohne Beteiligung der Bank eine zum Zahlen verwendete Münze nicht mit einem Abhebevorgang bzw. mit einem Kunden in Verbindung bringen, da er während des Abhebevorgangs die  $id$  des Kunden nur verschlüsselt ( $E_{bank}(id)$ ) sehen konnte.

- **Unverkettbarkeit**

Anhand zweier beliebiger, vom selben Kunden für verschiedene Bezahlvorgänge verwendeter Münzen  $(y_{coin}, s, \sigma, a)$  und  $(y'_{coin}, s', \sigma', a')$ , kann nicht erkannt werden, daß sie von demselben Kunden abgehoben und zur Zahlung verwendet wurden, da alle Parameter verschiedener Münzen voneinander unabhängig gewählt ( $y_{coin}$ ) bzw. gebildet ( $s, \sigma, a$ ) werden.

Um eine Verbindung herzustellen, müßte ein Angreifer die eingereichten Münzen mit den beim Abheben gesehenen Parametern verknüpfen können. Dazu müßte er wiederum die symmetrischen ( $K_{coin}$  bzw.  $K_{trustee}$ ) oder die asymmetrischen Verschlüsselungen ( $E_{bank}$ ) brechen können (Vgl. Anonymität).

- **Kundenverfolgung**

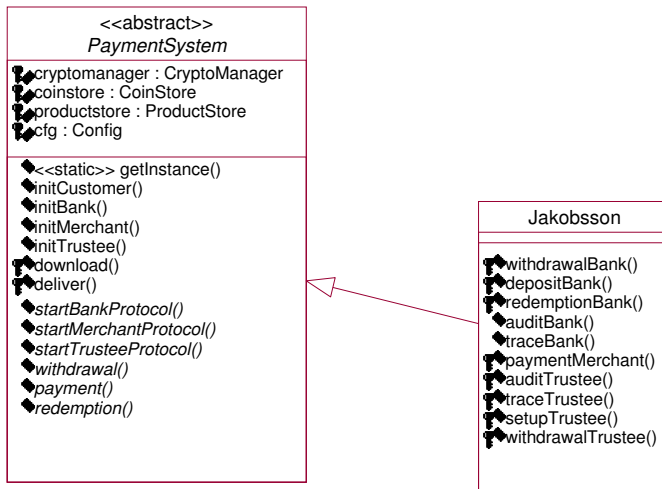
Eine Kundenverfolgung ist über den in Kap. 3.2 (siehe Kundenverfolgung) beschriebenen Mechanismus möglich.

- **Münzverfolgung**

Eine Münzverfolgung ist über den in Kap. 3.2 (siehe Münzverfolgung) beschriebenen Mechanismus möglich.

### 3.4 Implementierung

Wie in der Beschreibung der Protokolle (Kap. 3.2) gesehen, verwendet Jakobsson lediglich Standard-Kryptoverfahren (symmetrische und asymmetrische Verschlüsselung, Signatur, BlindeSignatur). Da diese über den CryptoManager bereitgestellt werden (Vgl. 1.3 & A.1.6), ergibt sich für das Design des Zahlungssystems folgende einfache Klassenstruktur:



`class Jakobsson extends PaymentSystem`

In der Klasse `Jakobsson` sind alle in Kap. 4 beschriebenen Protokolle implementiert. Die Methode `withdrawal` setzt die Kundenseite und `withdrawalBank` die Bankseite des Abhebeprotokolls um. Der Bezahlvorgang wird auf Kundenseite durch die Methode `payment`, auf Händlerseite durch `paymentMerchant` und auf Bankseite durch `depositBank` implementiert. Die Münzurückgabe wird analog durch `redemption` und `redemptionBank` umgesetzt. Die Protokolle zur Anonymitätsauflösung werden durch `auditBank` und `auditTrustee` (vom Abhebevorgang ausgehende Verfolgung) sowie durch `traceBank` und `traceTrustee` (vom Bezahlvorgang ausgehende Verfolgung) implementiert.

Die benötigten Kryptoverfahren (symmetrische und asymmetrische Verschlüsselung, Signatur, BlindeSignatur, Hashfunktion, Schlüsselverwaltung) werden über ein `CryptoManager`-Objekt zur Verfügung gestellt. Es dient als Schnittstelle zu den installierten Kryptoprovidern der Java Cryptography Architecture (JCA/JCE)(Vgl. 1.3).

# Kapitel 4

## Vergleich

## 4.1 Anforderungen & Sicherheit

Ein Zahlungssystem muß, um als sicher zu gelten, als notwendige Bedingung, die in Kap. 1.5 formulierten Anforderungen erfüllen. In Kap. 2.3 und Kap. 3.3 wurden die einzelnen Zahlungssysteme bezüglich dieser Anforderungen untersucht. Dabei wurden die zur Erfüllung der einzelnen Anforderungen verwendeten Mechanismen herausgearbeitet. In der folgenden Tabelle werden die Ergebnisse aus Kap. 2.3 und Kap. 3.3 zusammengefaßt und einander gegenübergestellt.

<b>Camenisch/Maurer/Stadler</b>	<b>Jakobsson/Yung</b>
<b>Fälschungssicherheit</b>	
Signatur der Münze mittels blind-PLOGEQ durch die Bank	Blinde Signatur der Münze durch die Bank und Signatur der Münze durch den Trustee
<b>Mehrfachausgabesicherheit</b>	
Speicherung charakteristischer Münzparameter aller eingereichten Münzen durch die Bank	
<b>Akzeptanzgarantie</b>	
Gültigkeitsbeweis einer Münze mittels blindPLOGEQ durch die Bank Schutz des Kunden vor falscher overspending-Beschuldigung durch die Bank mittels PKLOG (bzw. PLOGEQ) durch den Kunden	Gültigkeitsbeweis einer Münze mittels Blinde Signatur durch die Bank Schutz des Kunden vor falscher overspending-Beschuldigung durch die Bank mittels Signatur durch den Kunden
<b>Abhebesicherheit</b>	
Authentisierung des Kunden über public-key Signatur vor dem Abhebevorgang	
<b>Ausgabesicherheit</b>	
Beweis des Besitzes bzw. des Eigentums der Münze mittels PKLOG durch den Kunden bei Bezahlvorgang Beweis für das persönliche Abheben der eingereichten Münze mittels PLOGEQ durch den Kunden bei Rückgabevorgang	Beweis des Besitzes bzw. des Eigentums der Münze mittels Signatur durch den Kunden bei Bezahlvorgang Beweis für das persönliche Abheben der eingereichten Münze durch Offenlegen des symmetrischen Münzschlüssels durch den Kunden beim Rückgabevorgang.

<b>Rückgabesicherheit</b>	
Authentifizierung und PLOGEQ durch den Kunden sowie Vergleich des Tupels $(id, d)$ mit den beim Abheben gespeicherten Parametern durch die Bank	Authentifizierung und Offenlegen des symmetrischen Münzschlüssels $K_{coin}$ durch den Kunden sowie Vergleich von $K_{coin}(s)$ mit dem beim Abheben gespeicherten Parameter $\bar{s}$ durch die Bank
<b>Bankraubsicherheit</b>	
Möglichkeit alle Münzen der betroffenen Münzgeneration für Bezahlvorgänge zu sperren und nur noch im Rahmen von Rückgabevorgang zu akzeptieren.	Möglichkeit, bei Raub unrechtmäßig erzeugte Münzen über Trustee als ungültig zu erkennen
<b>Anonymität</b>	
Gewährleistung der Anonymität durch blindPLOGEQ, DL- und DDH-Problem (s.u.)	Gewährleistung der Anonymität durch blinde Signatur, symmetrische und asymmetrische Verschlüsselung
<b>Unverkettbarkeit</b>	
Unverkettbarkeit der Münzen durch Unabhängigkeit aller Münzparameter verschiedener Münzen und durch Anonymität (s.u.)	
<b>Kundenverfolgung</b>	
Möglichkeit der Kundenverfolgung durch Kooperation von Bank und Trustee mittels Kenntnis des DL $\tau$ durch den Trustee und Speicherung von $(id, d)$ während des Abhebevorgangs durch die Bank	Möglichkeit der Kundenverfolgung durch Kooperation von Bank und Trustee mittels Speicherung von $(m\#, K_{trustee}, y_{coin}, \overline{id})$ durch den Trustee und $(m\#, id, \overline{id})$ durch die Bank während des Abhebevorgangs
<b>Münzverfolgung</b>	
Möglichkeit der Münzverfolgung durch Kooperation von Bank und Trustee mittels Kenntnis des DL $\tau$ durch den Trustee und Speicherung von $(session\#, d)$ während des Abhebevorgangs durch die Bank	Möglichkeit der Münzverfolgung durch Kooperation von Bank und Trustee mittels Speicherung von $(m\#, K_{coin})$ durch den Trustee und $(m\#, \bar{s}, \bar{\sigma})$ durch die Bank während des Abhebevorgangs

Die Gegenüberstellung in der Tabelle zeigt, daß sowohl das von Camenisch/Maurer/Stadler als auch das von Jakobsson/Yung vorgeschlagene Zahlungssystem die Anforderungen erfüllt.

Als bemerkenswerter Unterschied ist festzuhalten, daß bei Jakobsson/Yung die Möglichkeit besteht, die nach einem Raub unrechtmäßig erzeugten Münzen über den Trustee als ungültig zu erkennen. Bei Camenisch/Maurer/Stadler bleibt nach einem Raub nur die Möglichkeit, alle sich im Umlauf befindlichen Münzen nur noch im Rahmen von Rückgabevorgängen zu akzeptieren. Diese Vorgehensweise stellt allerdings einen erheblichen Eingriff in den regulären Zahlungsverkehr dar und ist für den Kunden mit einem Mehraufwand verbunden.

Bei der Umsetzung von Anonymität und Unverkettbarkeit verwenden beide Systeme kryptographische Verfahren. Während bei Camenisch/Maurer/Stadler die Anonymität unmittelbar mit dem DL-Problem und dem DDH-Problem (Vgl. Kap. 2.1.5) zusammenhängt, verwendet Jakobsson/Yung eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung, um die Identität des Kunden zu verbergen, und eine blinde Signatur, um der Verbindung einer Münze mit einem Abhebevorgang vorzubeugen.

Kunden- und Münzverfolgung beruhen bei beiden Systemen auf dem Prinzip, während des Abhebevorgangs Parameter zu speichern, die im Falle eines begründeten Verdachts durch Kooperation von Bank und Trustee zur Deanonymisierung herangezogen werden. Bei Jakobsson/Yung speichern sowohl die Bank als auch der Trustee unabhängig voneinander Parameter. Dagegen genügt es bei Camenisch/Maurer/Stadler, daß die Bank Parameter speichert. Mit Hilfe seines geheimen Schlüssels kann der Trustee anhand dieser Parameter die Kunden- oder Münzverfolgung vornehmen.

Rückgabesicherheit wird bei beiden Systemen dadurch erreicht, daß der Kunde die Verbindung zwischen der eingereichten Münze und dem Abhebevorgang offenlegt. Dies geschieht bei Camenisch/Maurer/Stadler durch einen nichtinteraktiven Beweis (PLOGEQ), der zeigt, daß  $h_p$  und  $d$  den gleichen diskreten Logarithmus besitzen. Bei Jakobsson/Yung wird die Rückgabesicherheit durch das verwendete symmetrische Verschlüsselungsverfahren (Vgl. Kap. 3.1) garantiert. Der Kunde deckt bei Rückgabe den zur Münze gehörigen symmetrischen Münzschlüssel auf.

Fälschungssicherheit, Akzeptanzgarantie und Ausgabesicherheit beruhen bei Jakobsson/Yung auf Signaturen bzw. blinden Signaturen. Camenisch/Maurer/Stadler verwendet analog die Beweise PKLOG, PLOGEQ bzw. blindPLOGEQ, die jeweils als Signatur dienen.

Die Sicherheit des von Camenisch/Maurer/Stadler vorgeschlagenen Zahlungssystems hängt direkt von der Sicherheit der verwendeten Beweise (PKLOG, PLOGEQ und blindPLOGEQ) ab. Sollte sich diese eines Tages durch die Entwicklung neuer Algorithmen bzw. durch neue Hardware als unsicher herausstellen, bleibt als Ausweg nur das Ausweichen auf eine andere Gruppe  $G$ , auf der die Beweise noch als sicher gelten.

Da bei Jakobsson/Yung die verwendeten Krypto-Algorithmen zur Verschlüsselung und Signatur beliebig austauschbar und kombinierbar sind, kann hier flexibler auf die Entwicklung neuer Algorithmen bzw. neuer Hardware reagiert werden.

## 4.2 Effizienz

In diesem Abschnitt werden die beiden Zahlungssysteme zunächst hinsichtlich ihres theoretischen Rechenaufwands und ihres Speicherbedarfs hin untersucht und einander gegenübergestellt. Anschließend erfolgt ein Vergleich der anhand der Implementierung tatsächlich festgestellten Laufzeiten.

### 4.2.1 Engagement

Der theoretischen und praktischen Betrachtung der Protokolle liegen folgende Festlegungen zu Grunde:

#### **Camenisch/Maurer/Stadler**

Als den Protokollen zugrundeliegende Gruppe wurde eine  $q$ -elementige Untergruppe der primen Restklassengruppe  $(\mathbb{Z}/p\mathbb{Z})^*$  mit  $|p| = 1024$  bit und  $|q| = 160$  bit gewählt. Als Hashverfahren wurde SHA1 verwendet.

#### **Jakobsson/Yung**

Folgende Standard-Kryptoverfahren wurden gewählt:

- symmetrische Verschlüsselung: RC6
- asymmetrische Verschlüsselung: RSA ( $|n| = 1024$  bit)
- Signatur: Schnorr-Signatur
- Blinde Signatur: blinde Schnorr-Signatur

Sowohl für die Schnorr-Signatur als auch für die blinde Schnorr-Signatur wurden DSA-Schlüssel der Stärke 1024 sowie das Hashverfahren SHA1 verwendet.

### 4.2.2 Theoretische Betrachtung

#### **Rechenaufwand der Anwendungsfälle**

Im folgenden Abschnitt werden die Vorgänge Abheben, Bezahlen und Rückgabe auf ihre Effizienz hin untersucht und einander gegenübergestellt. Dazu werden die Protokolle hinsichtlich der rechenaufwandintensiven Gruppenoperationen analysiert.

Die Authentifizierung zwischen Kunden und Bank wurde nicht in die theoretische Betrachtung einbezogen, da sie vom eigentlichen Zahlungssystem unabhängig ist und von beiden Systemen benötigt wird und somit für einen Vergleich nicht relevant ist.

Auf Kundenseite ergibt sich für  $n$  Münzen folgender Rechenaufwand:

	Camenisch/Maurer/Stadler			Jakobsson/Yung		
	Abheben	Bezahlen	Rückgabe	Abheben	Bezahlen	Rückgabe
multi $exp \bmod p$ <sup>1</sup>	4n	-	-	2n	-	-
$exp \bmod p$	6n	n	3n	3n	n	n
$inv \bmod p$	-	n	n	-	-	-
$mult \bmod p$	3n	2n	2n	-	n	n
$add \bmod q$	3n	n	n	-	n	n
sym Cipher <sup>2</sup>	-	-	-	23n	-	-
hashing <sup>2</sup>	65n	15n	34n	22n	8n	8n

Tabelle 4.1: Rechenaufwand für die Anwendungsfälle auf Kundenseite

Aus Tabelle 4.1 wird deutlich, daß Camenisch/Maurer/Stadler auf Kundenseite beim Abhebe- und Rückgabevorgang gegenüber Jakobsson/Yung deutliche Nachteile bezüglich des Rechenaufwands hat. Der Bezahlvorgang ist auf Kundenseite bei Camenisch/Maurer/Stadler geringfügig aufwendiger.

Entscheidend für die Effizienz eines Zahlungssystems ist auch der Rechenaufwand für die Bank (bzw. den Trustee), da diese(r) in der Regel Anfragen mehrerer Kunden gleichzeitig bearbeiten muß.

	Camenisch/Maurer/Stadler			Jakobsson/Yung			
	Abheben	Bezahlen	Rückgabe	Abheben <sup>3</sup>	Bezahlen	Rückgabe	
multi $exp \bmod p$	2n	3n	2n	3n	-	3n	3n
$exp \bmod p$	2n	-	-	2n	3n	-	-
$inv \bmod p$	-	n	n	-	-	-	-
$mult \bmod p$	n	2n	2n	3n	n	-	-
$add \bmod q$	n	-	-	3n	n	-	-
sym Cipher <sup>2</sup>	-	-	-	23n	-	-	3n
hashing <sup>2</sup>	34n	49n	34n	31n	-	30n	30n
DB r/w	n+1	n+1	n+1	n	n+1	n+1	n+1
DB r	-	n	n	-	-	n	n

Tabelle 4.2: Rechenaufwand für die Anwendungsfälle auf Bank-/Trusteesseite

Aus Tabelle 4.2 ist zu entnehmen, daß der Rechenaufwand für den Abhebevorgang auf Bankseite (bzw. Trusteeseite) bei Jakobsson/Yung deutliche Nachteile aufweist. Der Rückgabevorgang ist bei Jakobsson/Yung auf Bankseite ebenfalls etwas aufwendiger. Beim Bezahlvorgang hat Jakobsson/Yung

dagegen auf Bankseite leichte Vorteile.

Insgesamt ergibt sich für die Anwendungsfälle folgender Rechenaufwand:

	Camenisch/Maurer/Stadler			Jakobsson/Yung		
	Abheben	Bezahlen	Rückgabe	Abheben	Bezahlen	Rückgabe
multi $exp \bmod p$ <sup>1</sup>	6n	3n	2n	5n	3n	3n
$exp \bmod p$	8n	n	3n	8n	n	n
$inv \bmod p$	-	2n	2n	-	-	-
$mult \bmod p$	4n	4n	4n	4n	n	n
$add \bmod q$	4n	n	n	4n	n	n
sym Cipher <sup>2</sup>	-	-	-	46n	-	3n
hashing <sup>2</sup>	99n	64n	68n	53n	38n	38n
DB r/w	n+1	n+1	n+1	2n+1	n+1	n+1
DB r	-	n	n	-	n	n

Tabelle 4.3: Rechenaufwand für die Anwendungsfälle

Aus Tabelle 4.3 ist zu entnehmen, daß der Rechenaufwand für den Abhebevorgang insgesamt bei Jakobsson/Yung deutliche Vorteile aufweist. Beim Bezahlvorgang hat Jakobsson/Yung bezüglich des Rechenaufwands insgesamt leichte Vorteile. Beim Rückgabevorgang ist dagegen Camenisch/Maurer/Stadler im Vorteil.

---

<sup>1</sup>Multi-Exponentiationen der Art  $x^a y^b \bmod p$  bzw. bei Camenisch/Maurer/Stadler auf Kundenseite eine der Art  $x^a y^b z^c \bmod p$  mit  $a, b, c \in \mathbb{Z}_q^*$  (Vgl. [10])

<sup>2</sup>Die Angabe bezieht sich auf die Anzahl der zu ver-/entschlüsselnden bzw. zu hashenden Blöcke.

<sup>3</sup>Die linke Spalte bezeichnet den Rechenaufwand des Trustee, die rechte den der Bank.

### Speicherplatzbedarf

Im folgenden wird der von den Zahlungssystemen benötigte Speicherplatz untersucht. Es wird zunächst der auf Kundenseite zur Speicherung der abgehobenen Münzen benötigte Platz bestimmt. Anschließend wird der Speicherplatz betrachtet, der für die beim Abhebevorgang, beim Bezahlvorgang und beim Rückgabevorgang von der Bank (bzw. vom Trustee) erhobenen Münzparameter benötigt wird.

Der Speicherbedarf für vom Kunden abgehobene Münzen ist in der folgenden Tabelle aufgeführt:

Camenisch/Maurer/Stadler			Jakobsson/Yung		
$c\#$		64 bit	$K_{coin}$		128 bit
$\alpha$	$\ln q$	160 bit	$x_{coin}$	$\ln q$	160 bit
$W$	$2 \ln q$	320 bit	$s$	$2 \ln q$	320 bit
			$\sigma$	$2 \ln q$	320 bit
$h_p$	$\ln p$	1024 bit	$y_{coin}$	$\ln p$	1024 bit
$z_p$	$\ln q$	160 bit			
$i$		32 bit	$i$		32 bit
$gen\#$		32 bit	$gen\#$		32 bit
$\sum$	$\ln p + 4 \ln q + 128$	1792 bit	$\sum$	$\ln p + 5 \ln q + 192$	2016 bit

Tabelle 4.4: Münzgrößen auf Kundenseite

Die von der Bank während des Bezahlvorgangs bzw. Rückgabevorgangs erhobenen Münzparameter benötigen folgenden Speicher:

Camenisch/Maurer/Stadler			Jakobsson/Yung		
$c\#$		64 bit	$y_{coin}$	$\ln q$	1024 bit
$h_p$	$\ln p$	1024 bit	$s$	$2 \ln q$	320 bit
$\tilde{o}$	$\ln q$	160 bit	$h(o)$	$\ln q$	160 bit
$V$	$2 \ln q$	320 bit	$a$	$2 \ln q$	320 bit
$\sum$	$\ln p + 3 \ln q + 64$	1568 bit	$\sum$	$\ln p + 5 \ln q$	1824 bit

Tabelle 4.5: Speicherbedarf pro eingereichter Münze auf Bankseite

Die von der Bank (bzw. dem Trustee)<sup>4</sup>während des Abhebevorgangs erhobenen Münzparameter benötigen folgenden Speicher:

Camenisch/Maurer/Stadler			Jakobsson/Yung <sup>4</sup>		
$session\#$		64 bit	$m\#$		64 bit
$d$	$\ln p$	1024 bit	$\overline{id}$	$\ln p$	1024 bit
$id$		32 bit	$id$		32 bit
$i$		32 bit	$i$		32 bit
$gen\#$		32 bit	$gen\#$		32 bit
			$\bar{s}, \bar{\sigma}$		768 bit
			$(m\#)$		(64 bit)
			$(y_{coin})$	$(\ln p)$	(1024 bit)
			$(id)$	$(\ln p)$	(1024 bit)
			$(K_{coin})$		(128 bit)
			$(K_{Trustee})$		(128 bit)
$\Sigma$	$\ln p + 160$	1184 bit	$\Sigma$	$\ln p + 928$ $(2 \ln q + 320)$	1952 bit (2368 bit)

Tabelle 4.6: Speicherbedarf pro eingereichter Münze auf Bankseite (bzw. Trusteeseite)

Bei der Gegenüberstellung fällt auf, daß Camenisch/Maurer/Stadler gegenüber Jakobsson/Yung bezüglich des Speicherplatzbedarfs Vorteile hat. Sowohl die auf Kundenseite gespeicherte Münze als auch die von der Bank während des Bezahlvorgangs bzw. Rückgabevorgangs erhobenen Parameter benötigen bei Jakobsson/Yung geringfügig mehr Speicherplatz. Auch für die beim Abhebevorgang gespeicherten Parameter benötigen Jakobsson/Yung deutlich mehr Speicher. Neben den von der Bank erhobenen Parametern kommt bei Jakobsson/Yung im Gegensatz zu Camenisch/Maurer/Stadler die nötige Speicherung von Münzparametern auf Trusteeseite hinzu.

<sup>4</sup>Die vom Trustee erhobenen Münzparameter sind in Klammern dargestellt.

### 4.2.3 Praktische Betrachtung/Messung

Im folgenden werden die Ergebnisse der Laufzeitanalyse für die Vorgänge Abheben, Bezahlen und Rückgabe der beiden Zahlungssysteme einander gegenübergestellt. Die Zeiten wurden jeweils aus Java heraus mittels der Funktion `System.currentTimeMillis` gemessen. Als Testinfrastruktur dienten folgende zu einem LAN verbundenen Rechner:

- Bank: UltraSPARC-IIe CPU, 500MHZ, 128 MB, Sun Solaris 8
- Trustee: UltraSPARC-IIe CPU, 500MHZ, 128 MB, Sun Solaris 8
- Händler: UltraSPARC-IIe CPU, 500MHZ, 128 MB, Sun Solaris 8
- Kunde: UltraSPARC-IIe CPU, 500MHZ, 128 MB, Sun Solaris 8

Den Messungen liegen ebenfalls, wie im voranstehenden Abschnitt, die in Kap. 4.2.1 gemachten Festlegungen zugrunde.

Für die einzelnen Münzsorten wurden folgende Werte definiert:

1 cent	32 cent	1024 cent	32768 cent
2 cent	64 cent	2048 cent	65536 cent
4 cent	128 cent	4096 cent	...
8 cent	256 cent	8192 cent	...
16 cent	512 cent	16384 cent	...

Damit lassen sich selbst hohe Beträge mit relativ wenigen Münzen zusammenstellen. Den Messungen lagen folgende Splittungen zugrunde:

- 1 - 100 Münzen :  $1 - 100 * 1 \text{ cent}$
- 1 Euro :  $2 * 1 + 1 * 2 + 2 * 4 + 1 * 8 + 1 * 16 + 2 * 32 \text{ cent} \hat{=} 9 \text{ Münzen}$
- 10 Euro :  $2 * 1 + 1 * 2 + 1 * 4 + 2 * 8 + 1 * 16 + 2 * 32 + 2 * 64 + 2 * 128 + 2 * 256 \text{ cent} \hat{=} 15 \text{ Münzen}$
- 100 Euro :  $2 * 1 + 1 * 2 + 1 * 4 + 1 * 8 + 2 * 16 + 1 * 32 + 1 * 64 + 1 * 128 + 2 * 256 + 2 * 512 + 2 * 1024 + 1 * 2048 + 1 * 4096 \text{ cent} \hat{=} 18 \text{ Münzen}$
- 1000 Euro :  $2 * 1 + 1 * 2 + 1 * 4 + 1 * 8 + 1 * 16 + 2 * 32 + 1 * 64 + 2 * 128 + 1 * 256 + 2 * 512 + 2 * 1024 + 1 * 2048 + 1 * 4096 + 1 * 8192 + 1 * 16384 + 2 * 32768 \text{ cent} \hat{=} 22 \text{ Münzen}$

Die nachfolgend aufgeführten Zeiten für Abhebe-, Bezahl- und Rückgabevorgänge ergaben sich jeweils als Durchschnitt über 5 Messungen.

### Abheben

	<b>Camenisch/Maurer/Stadler</b>	<b>Jakobsson/Yung</b>
1 Münze	2019 ms	1683 ms
10 Münzen	13036 ms	10872 ms
30 Münzen	36577 ms	30998 ms
100 Münzen	119028 ms	101734 ms
1 Euro	12875 ms	10805 ms
10 Euro	19363 ms	16410 ms
100 Euro	22751 ms	18951 ms
1000 Euro	27311 ms	23145 ms

Tabelle 4.7: gemessene Zeiten pro Abhebevorgang

Aus der Tabelle ist für den Abhebevorgang ein deutlicher Geschwindigkeitsvorteil bei Jakobsson/Yung zu erkennen. Dieser läßt sich durch die in Kap. 4.2.2 aufgeführte geringere Komplexität von Jakobsson/Yung erklären.

### Bezahlen

	<b>Camenisch/Maurer/Stadler</b>	<b>Jakobsson/Yung</b>
1 Münze	1082 ms	1092 ms
10 Münzen	4623 ms	4421 ms
30 Münzen	12741 ms	12112 ms
100 Münzen	40879 ms	39119 ms
1 Euro	4032 ms	3838 ms
10 Euro	6573 ms	6258 ms
100 Euro	7736 ms	7299 ms
1000 Euro	9260 ms	8845 ms

Tabelle 4.8: gemessene Zeiten pro Bezahlvorgang

Beim Bezahlvorgang ist Jakobsson/Yung leicht im Vorteil gegenüber Camenisch/Maurer/Stadler. Dieser Vorteil läßt sich über die in Kap. 4.2.1 festgestellte tendenziell geringere Komplexität des Bezahlvorgangs von Jakobsson/Yung erklären.

**Rückgabe**

	<b>Camenisch/Maurer/Stadler</b>	<b>Jakobsson/Yung</b>
1 Münze	703 ms	827 ms
10 Münzen	3960 ms	4659 ms
30 Münzen	11265 ms	13099 ms
100 Münzen	35821 ms	42644 ms
1 Euro	3683 ms	4385 ms
10 Euro	5926 ms	6971 ms
100 Euro	6982 ms	8120 ms
1000 Euro	8308 ms	9832 ms

Tabelle 4.9: gemessene Zeiten pro Rückgabevorgang

Beim Rückgabevorgang ist ein Unterschied zugunsten von Camenisch/Maurer/Stadler erkennbar. Dieser läßt sich durch die in Kap. 4.2.1 festgestellte geringere Komplexität des Rückgabevorgangs von Camenisch/Maurer/Stadler erklären.

### 4.3 Fazit

Der Vergleich der beiden untersuchten und implementierten Zahlungssysteme führte zu folgenden Erkenntnissen:

Sowohl das von Camenisch/Maurer/Stadler als auch das von Jakobsson/Yung vorgeschlagene Zahlungssystem erfüllen die in Kap. 1.5 aufgeführten Anforderungen. Ein wesentlicher Unterschied liegt allerdings in der Möglichkeit, bei Jakobsson/Yung die nach einem Raub unrechtmäßig erzeugten Münzen über den Trustee als ungültig zu erkennen. Bei Camenisch/Maurer/Stadler bleibt in diesem Fall nur die Möglichkeit, für Bezahlvorgänge die Akzeptanz aller im Umlauf befindlicher Münzen der betroffenen Generation zu beenden und die Kunden zu bitten, diese Münzen zurückzugeben. Dies bedeutet eine erhebliche Beeinträchtigung des Zahlungsverkehrs.

Die Sicherheit des von Camenisch/Maurer/Stadler vorgeschlagenen Zahlungssystems hängt direkt von der Sicherheit der verwendeten Beweise PKLOG, PLOGEQ und blindPLOGEQ ab. Sollte sich diese in Zukunft durch die Entwicklung neuer Angriffe bzw. durch neue Hardware als unsicher herausstellen, bleibt als Ausweg nur das Ausweichen auf eine andere Gruppe  $G$ , auf der diese Beweise noch als sicher gelten.

Da bei Jakobsson/Yung die verwendeten Krypto-Algorithmen zur Verschlüsselung und Signatur beliebig austauschbar und kombinierbar sind, kann hier flexibler auf die Entwicklung neuer Angriffe bzw. neuer Hardware reagiert werden.

Das von Jakobsson/Yung vorgestellte Zahlungssystem hat beim Abhebevorgang gegenüber Camenisch/Maurer/Stadler deutliche Vorteile und beim Bezahlvorgang leichte Vorteile bezüglich des Rechenaufwands. Beim Rückgabevorgang ist dagegen Camenisch/Maurer/Stadler bezüglich des Rechenaufwands im Vorteil.

Bei einer gesonderten Betrachtung des Rechenaufwands auf Bank- bzw. Trusteseite, wo in der Regel Anfragen mehrerer Kunden gleichzeitig bearbeitet werden müssen, fällt auf, daß der Rechenaufwand für den Abhebevorgang auf Bankseite (bzw. Trusteseite) bei Jakobsson/Yung deutliche Nachteile aufweist. Der Rückgabevorgang ist bei Jakobsson/Yung auf Bankseite ebenfalls etwas aufwendiger. Beim Bezahlvorgang hat Jakobsson/Yung dagegen auf Bankseite leichte Vorteile.

Der Speicherbedarf pro abgehobener Münze ist auf Kundenseite bei dem von Camenisch/Maurer/Stadler vorgestellten Zahlungssystem geringer als bei dem von Jakobsson/Yung vorgestellten System.

Auch für die auf Bankseite während des Abhebevorgangs und des Bezahl- bzw. Rückgabevorgangs gespeicherten Münzparameter benötigt das von Jakobsson/Yung vorgestellte Zahlungssystem mehr Speicherplatz. Hinzu kommt die nötige Speicherung von Münzparametern auf Trusteseite.

Die Laufzeitanalyse zeigt beim Abhebevorgang einen deutlichen Geschwindigkeitsvorteil bei dem von Jakobsson/Yung vorgestellten Zahlungssystem. Beim Bezahlvorgang zeigt das von Jakobsson/Yung vorgestellte Zahlungssystem leichte Vorteile. Beim Rückgabevorgang ist dagegen bei dem von Camenisch/Maurer/Stadler vorgestellte System ein besseres Laufzeitverhalten zu beobachten.

Das von Jakobsson/Yung vorgestellte Zahlungssystem stellt erheblich höhere Anforderungen an die Trustee-Infrastruktur. Um den aktiven Trustee zu realisieren, ist ein leistungsfähiger Server und eine Datenbank notwendig. Dies verursacht zusätzliche Betriebskosten. Die Verfügbarkeit des Trustee muß kontinuierlich gewährleistet sein, da bei einem Ausfall kein Abhebevorgang möglich ist. Wird ein Trustee zentral von mehreren Banken genutzt, so kann dieser schnell zum Flaschenhals des gesamten Systems werden. Demgegenüber steht der oben erwähnte Vorteil, die nach einem Raub unrechtmäßig erzeugten Münzen über den Trustee als ungültig erkennen zu können. Zudem übernimmt der Trustee während des Abhebevorgangs einige Funktionen, die bei Systemen mit passivem Trustee (z.B Camenisch/Maurer/Stadler) von der Bank geleistet werden müssen. Damit kann sich die Bank verstärkt um die Bearbeitung von Bezahl- und Rückgabevorgängen kümmern.

Beide untersuchten Zahlungssysteme lassen sich sicher und effizient realisieren. Für das von Camenisch/Maurer/Stadler vorgestellte System spricht der geringere Speicherplatzbedarf und die geringeren Anforderungen an die Trustee-Infrastruktur.

Dagegen bietet das von Jakobsson/Yung vorgestellte Zahlungssystem einen bezüglich der Laufzeit effizienteren Abhebevorgang sowie ein höheres Maß an Flexibilität, um bei Entwicklung neuer Algorithmen bzw. neuer Hardware weiterhin Sicherheit garantieren zu können.

# Literaturverzeichnis

- [1] Wolfgang Arnold, E-Payment-Systeme: Geld für den elektronischen Markt, Die Bank 8/2001, S.571-581
- [2] Dan Boneh, The Decision Diffie-Hellman Problem, Lecture Notes in Computer Science 1423, S. 48-63, 1998
- [3] Johannes Buchmann, Einführung in die Kryptographie, Springer, 1999
- [4] Jan Camenisch, Ueli Maurer, Markus Stadler, Digital Payment Systems with Passive Anonymity-Revoking Trustee, in ESORICS: European Symposium on Research in Computer Security. LNCS, Springer-Verlag, 1996
- [5] Jan Camenisch, Jean-Marc Piveteau, Markus Stadler, An Efficient Fair Payment System, in proceedings of 3rd ACM Computer and Communication Security '96, S. 88-94, ACM press, 1996
- [6] Jan Camenisch, Jean-Marc Piveteau, Markus Stadler, An Efficient Electronic Payment System Protecting Privacy, in Computer Security - ESORICS 94, volume 875 of Lecture Notes in Computer Science, S. 207-215, Springer Verlag, 1994
- [7] D. Chaum, Blind signatures for untraceable payments, in D. Chaum, R. L. Rivest, and A. T. Sherman, editors, Advances in Cryptology: Proceedings of Crypto 82, pages 199-203, Plenum Press, 23-25 August 1983
- [8] Markus Jakobsson, Moti Yung, Revokable and Versatile Electronic Money (Extended Abstract), in 3rd ACM Conference on Computer and Communications Security CCS 96, S. 76-87, ACM Press, 1996
- [9] Michael Kliger und Dr. Joachim Gripp, McKinsey & Company, Internet-Einzelhandel bringt richtig Kohle, Computerzeitung 18.2.2002, 2002
- [10] Bodo Möller, Algorithms for multi-exponentiation, Technical Report TI-8/01, TU Darmstadt, Fachbereich Informatik, 2001

- [11] Dennis Kügler, Holger Vogt, Unsichtbare Markierungen in elektronischem Geld, Kommunikationssicherheit Schwerpunkt Internet (KSI 2001), Vieweg Verlag, S. 262-271, 2001
- [12] Lennart Meier, Special Aspects of Escrow-based E-Cash Systems, Master's Thesis, Universität des Saarlandes, Department of Computer Science. Chair of Prof. Dr. Birgit Pfitzmann, 2000
- [13] Silvio Micali, Michael Rabin und Salil Vadhan, Verifiable random functions, in 40th Annual Symposium on Foundations of Computer Science, New York, S. 120130, 1999.
- [14] Holger Petersen, Faires elektronisches Geld, erschienen in Proc. 5. Deutscher IT-Sicherheitskongreß, SecuMedia Verlag, 1997, S. 427-444
- [15] A. Pfitzmann, Marit Köhntopp: Anonymity, Unobservability and Pseudonymity - A Proposal for Terminology, in Hannes Federrath (Hg.): Designing Privacy Enhancing Technologies; Proc. Workshop on Design Issues in Anonymity and Unobservability, S. 1-9, 2001
- [16] T. Sander, A. Ta-Shma, Flow control: A new approach for anonymity control in electronic cash systems, in Financial cryptography: Third International Conference, FC 99, Anguilla, British West Indies, February 22-25, in Band 1648 of Lecture Notes in Computer Science, S. 46-61, Springer-Verlag, 1999
- [17] C.P. Schnorr, Efficient signature generation for smart cards, Journal of Cryptology, 4(3), S. 239-252, 1991
- [18] Berry Schoenmakers, Basic Security of the ecash Payment System, in B. Preneel, V. Rijmen (eds), State of the Art in Applied Cryptography, Leuven, Belgien, 1997
- [19] S. von Solms, D. Naccache, On blind signatures and perfect crimes, in Computer & Security, 11(6), S. 581-583, 1992
- [20] Stadler M., Piveteau J.-M. and Camenisch J., Fair Blind Signatures, in Proc. EUROCRYPT '95, LNCS 921, S. 209-219, Springer-Verlag, 1995.
- [21] Stadler M., Cryptographic Protocols for Revocable Privacy, Dissertation submitted to the Swiss Federal Institute of Technology Zürich, Diss. ETH No.11651, Zürich, 1996.

Anhang A

Implementierung

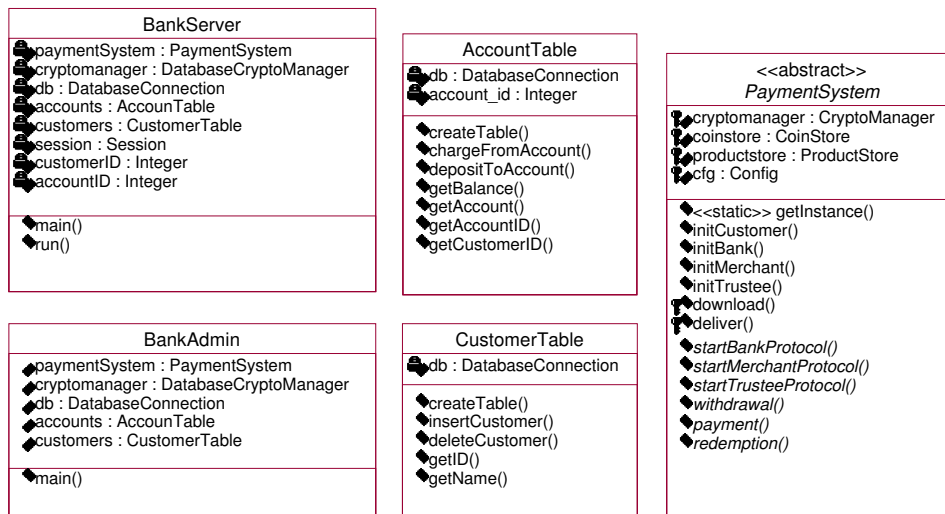
## A.1 Infrastruktur

Im folgenden werden die für die Implementierung der Zahlungssysteme benötigten Infrastrukturen dargestellt. Neben den in den Protokollen involvierten Parteien - Bank, Kunde, Händler, Trustee (Vgl. Kap. 2.2) und CA, werden auch deren Schnittstellen zur JCA/JCE und die zur Kommunikation benötigten Strukturen beschrieben.

### A.1.1 Bank

Die Bank ist ein Kreditinstitut, das ein anonymes elektronisches Zahlungssystem als Service anbietet. Dazu benötigt sie eine Datenbank zur Kunden- und Kontoverwaltung sowie zur Speicherung notwendiger Münzparameter. Zur automatisierten Bearbeitung eingehender Kundenaufträge muß sie einen Server betreiben, der die Aufträge erkennt und die dazugehörigen Protokolle auf Bankseite startet.

Folgende Klassen implementieren die notwendigen Anforderungen unabhängig vom konkreten Zahlungssystem:



Die Klasse **BankServer** implementiert einen Server, der eingehende Kundenaufträge bearbeitet. Für jeden Kunden, der sich mit dem Server verbindet, wird eine neue Instanz des **BankServers** als Thread gestartet.

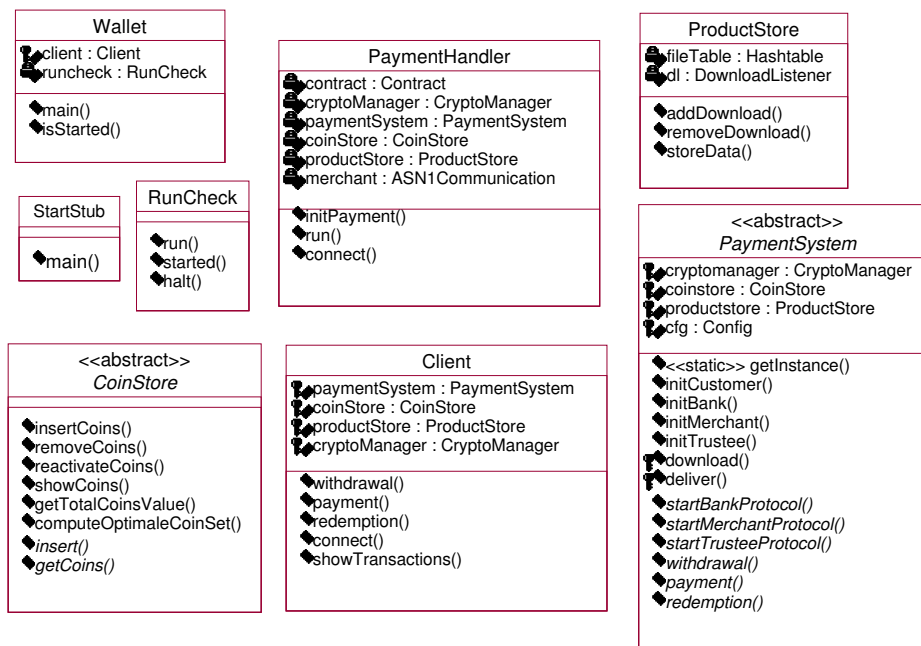
Die Verwaltung der Kunden und ihrer Konten in einer Datenbank erfolgt über die Klassen **CustomerTable** und **AccountTable**. Als Schnittstelle zu den Protokollen der Zahlungssysteme dient die Klasse **PaymentSystem** (Vgl.

1.3). Die Klasse `BankAdmin` implementiert ein Werkzeug, um die Bank administrieren zu können. Sie bietet Funktionen zur Schlüsselerzeugung und Beantragung von Zertifikaten, zur Datenbankadministration und zur Initialisierung des Zahlungssystems.

### A.1.2 Kunde

Ein Kunde ist eine natürliche Person, die bei der Bank ein Konto hält. Um die Zahlungssysteme nutzen zu können, benötigt er ein zertifiziertes Schlüssel-paar zur Authentifizierung sowie Software, die die Kundenseite der Protokolle, die Speicherung der abgehobenen Münzen und ein UI implementiert. Diese Software muß desweiteren die Möglichkeit bieten, den Zahlungsvorgang für über einen Browser getätigte Einkäufe einzuleiten.

Folgende Klassen implementieren diese Anforderungen unabhängig vom konkreten Zahlungssystem und dem tatsächlich verwendeten Münzspeicher:



Die Klasse `Wallet` implementiert ein zahlungssystemunabhängiges UI. Neben den Funktionen “Abheben” und “Rückgabe” bietet es die Möglichkeit, eine `String`-Repräsentation der abgehobenen Münzen zu betrachten sowie alle getätigten Transaktionen einzusehen. Desweiteren dient es als Zugang zum Bankkonto des Kunden. Außerdem bietet es dem Kunden die Möglichkeit, ein Schlüsselpaar zu erzeugen und von der CA (Vgl. A.1.5) zertifizieren

zu lassen.

Die Klasse `Client` implementiert zahlungssystemunabhängig alle vom Kunden ausführbaren Anwendungsfälle (Bezahlen, Abheben, Rückgabe)(Vgl. A.2).

Die Klasse `CoinStore` bietet die Möglichkeit, bei einem Abhebevorgang erzeugte Münzen zu speichern. Konkret verwendete Speichermedien (z.B. Datei, Chipkarte) können durch Erben von `CoinStore` unterstützt werden.

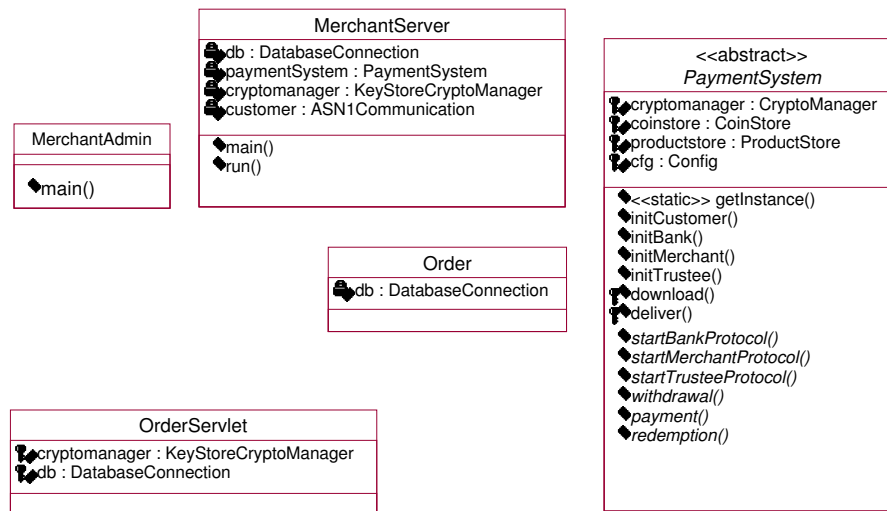
Die Klasse `PaymentHandler` dient dazu, die Zahlung akzeptierter Kaufverträge abzuwickeln (Vgl. A.2.2). Auf elektronischem Weg ausgelieferte Produkte werden über die Klasse `ProductStore` gespeichert.

Als Schnittstelle zum konkreten Zahlungssystem dient die abstrakte Klasse `PaymentSystem`.

### A.1.3 Händler

Ein Händler ist ein Kaufmann, der ein von der Bank betriebenes anonymes elektronisches Zahlungssystem als Zahlungsmittel akzeptiert. Er hält ebenfalls ein Konto bei der Bank. Der Händler benötigt eine Möglichkeit zur Verwaltung der angebotenen Güter und der Bestellungen sowie zur Verbreitung seiner Angebote. Zur automatisierten Bearbeitung eingehender Bestellungen muß er einen Server betreiben, der die Bestellungen entgegennimmt und die dazugehörigen Protokolle auf Händlerseite startet.

Folgende Klassen implementieren die notwendigen Anforderungen unabhängig



Die Klasse `MerchantServer` implementiert einen Server, der eingehende Be-

stellungen bearbeitet. Für jeden sich verbindenden Kunden wird eine neue Instanz des `MerchantServers` als Thread gestartet.

Als Schnittstelle zum konkreten Zahlungssystem dient die abstrakte Klasse `PaymentSystem`.

Die Klasse `OrderServlet` ist dafür verantwortlich, aus in der Datenbank gespeicherten Informationen zum Angebot Bestellformulare zu generieren. Sie generiert zudem zu ausgefüllten Bestellungen, die durch ein Objekt der Klasse `Order` repräsentiert werden, ein Angebot, das vom Kunden über einen Browser empfangen werden kann.

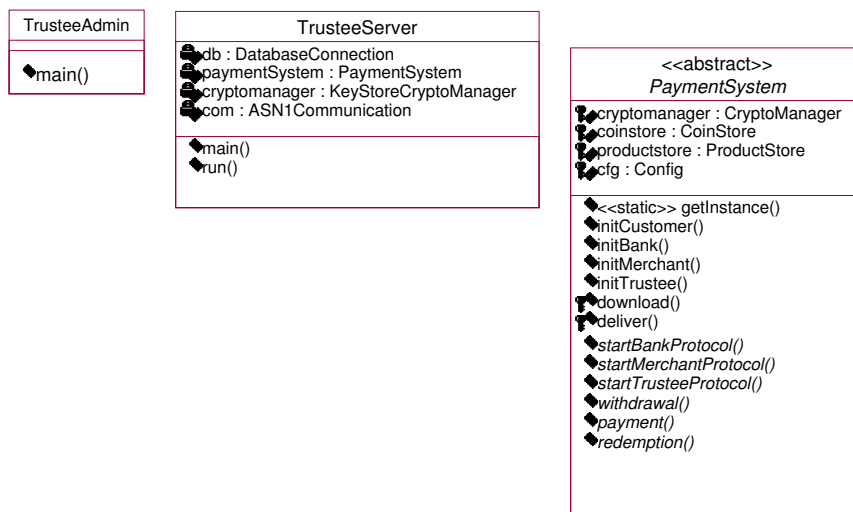
Die Klasse `MerchantAdmin` implementiert ein Werkzeug, mit dem der Händler seine Angebote konfigurieren und alle eingegangenen Bestellungen überwachen kann.

### A.1.4 Trustee

Der Trustee ist eine anerkannte vertrauenswürdige juristische Person. Seine Aufgabe ist es, den von Regierung und Bank für Zahlungsmittel geforderten Mechanismus zur Verfolgung und Einfrierung von Kapital im Sinne des Datenschutzes zu kontrollieren.

Zur automatisierten Bearbeitung dieser Aufgabe muß der Trustee einen Server betreiben, der eingehende Anfragen prüft und die notwendigen Protokolle auf Trusteeseite startet.

Folgende Klassen implementieren die an den Trustee gestellten Anforderungen unabhängig vom konkreten Zahlungssystem:



Die Klasse `TrusteeServer` implementiert einen Server. Seine Aufgabe ist

es, auf Anfragen der Bank zu reagieren und die entsprechenden Protokolle auf Trusteeseite zu starten. Für jede neue Anfrage wird eine neue Instanz des `TrusteeServers` als Thread gestartet.

Als Schnittstelle zum konkreten Zahlungssystem dient die abstrakte Klasse `PaymentSystem`.

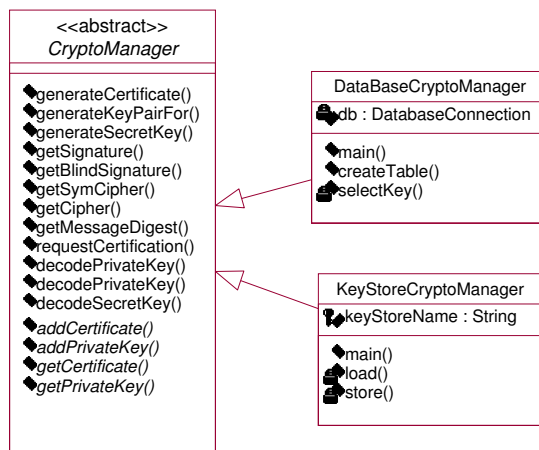
Die Klasse `TrusteeAdmin` implementiert ein Werkzeug, um den Trustee zu konfigurieren.

### A.1.5 CA

Die Certification-Authority (CA) bürgt mit ihrer Signatur für die Korrektheit und Gültigkeit der öffentlichen Schlüssel der im Zahlungssystem involvierten Parteien (Bank, Kunde, Händler und Trustee). Sie verwaltet ein Verzeichnis (Directory) der von ihr erzeugten Zertifikate mit den dazugehörigen Teilnehmernamen. Ein zu einem Namen gespeichertes Zertifikat wird auf Anfrage herausgegeben. Statt einer CA sind in der Regel mehrere CAs involviert, die über die transitive Relation 'bürgt für' untereinander in Beziehung stehen.

### A.1.6 Cryptomanager & JCA/JCE

Der Cryptomanager dient als Schnittstelle zwischen den Zahlungssystemen und den installierten Kryptoprovidern der Java Cryptography Architecture (JCA/JCE). Außerdem dient er als Schnittstelle zur CA und zu den jeweiligen privaten Speichern, in denen die privaten Schlüssel abgelegt werden. Da diese Speicher unterschiedlicher Form sein können (z.B. Datenbank, Datei, Chipkarte), ergibt sich folgende Klassenstruktur:



Die Klasse `CryptoManager` implementiert die oben beschriebenen Schnittstellen. Die für das Ablegen und Holen der Schlüssel aus dem privaten Speicher verantwortlichen Methoden (`addPrivateKey`, `getPrivateKey`, `addCertificate`, `getCertificate`) werden von der Klasse `DataBaseCryptoManager` für Datenbanken und von der Klasse `KeyStoreCryptoManager` für Dateien implementiert.

### A.1.7 Kommunikation

Die Kommunikation zwischen den Teilnehmern erfolgt über ASN.1 kodierte Nachrichten (`JCashMessage`) gemäß folgender Definition:

```
JCashMessage ::= SEQUENCE {
    protocolOID [0] ObjectIdentifier,
    parameter [1] ANY OPTIONAL
    errorCode INTEGER OPTIONAL,
    errorDescription PRINTABLE STRING OPTIONAL,
}
```

Der Parameter vom Typ `ANY OPTIONAL` beinhaltet die zu sendende Botschaft. Die Parameter `errorCode` und `errorDescription` können verwendet werden, um über evtl. aufgetretene Fehler zu informieren. `protocolOID` spezifiziert einen optionalen Befehlscode, um z.B. einen Abhebevorgang einzuleiten.

## A.2 Realisierung der Anwendungsfälle

Im folgenden Abschnitt werden die Programmabläufe, der durch die oben beschriebenen Klassen realisierten Vorgänge “Abheben”, “Bezahlen”, “Rückgabe”, “Deanonymisierung” (Vgl. Kap. 1.2), zahlungssystemunabhängig beschrieben. Die Protokolle zu den implementierten Zahlungssystemen wurden bereits in Kap. 2.2 und Kap. 3.2 detailliert dargestellt, so daß hier lediglich der Verlauf bis zu deren Start aufgezeigt wird.

### A.2.1 Abhebevorgang

Um Münzen abzuheben, muß der Kunde zunächst die Klasse `Wallet` starten. Dies geschieht über eine `main`-Methode. Nun kann der Kunde die Funktion “Abheben” wählen und den gewünschten Betrag eingeben. Die Klasse `Wallet` ruft die Methode `withdrawal` aus der Klasse `Client` auf, die wiederum über die Methode `withdrawal` von `PaymentSystem` die Kundenseite des Abhebeprotokolls startet.

Die Bankseite des Abhebeprotokolls wird über die Methode `startBankProtocol` von `PaymentSystem` durch die Klasse `BankServer` gestartet, sobald sich die Kundenseite mit dem Bankserver verbunden hat. `startBankProtocol` interpretiert den vom Kunden empfangenen Befehlscode (`protocolOID`, Vgl. A.1.7) und startet die Methode `withdrawalBank` der konkreten Zahlungssystem-Implementierung.

Bei Systemen mit aktivem Trustee (z.B. Jakobsson) muß auf Trusteeseite ebenfalls das Abhebeprotokoll gestartet werden. Analog zum Vorgehen auf Bankseite startet der Trusteeserver die Methode `startTrusteeProtocol`, die den empfangenen Befehlscode interpretiert und die Methode `withdrawal-Trustee` der konkreten Zahlungssystem-Implementierung aufruft.

Während des Abhebevorgangs erzeugte Münzen werden über die Methode `insertCoin` von `CoinStore` im jeweiligen Münzspeicher gespeichert.

### A.2.2 Bezahlvorgang

Wünscht ein Kunde ein Produkt des Händlers zu kaufen, so muß er zunächst auf der Internetseite des Händlers über eine Eingabemaske eine Bestellung aufgeben. Der Händler generiert über die Klasse `OrderServlet` zu der ausgefüllten Bestellung einen Kaufvertrag, der vom Kunden über einen Webbrowser empfangen wird.

Über die Klasse `StartStub` wird das `Wallet` vom Browser aus gestartet, sobald der Vertrag (`contract`) vom Händler empfangen wurde. Akzeptiert der Kunde den Vertrag, so wird ein `PaymentHandler`-Objekt generiert, das den

Bezahlvorgang als Thread abwickelt. Dazu wird die Methode `payment` von `PaymentSystem` aufgerufen, die das Bezahlprotokoll auf Kundenseite startet. Die Methode `payment` greift über ein `CoinStore`-Objekt auf die während des Abhebevorgangs gespeicherten Münzen zu und baut eine Verbindung zum Server des Händlers auf.

Auf Händlerseite wartet der `MerchantServer` auf Kunden. Pro verbundenem Kunden wird die Methode `startMerchantProtocol` von `PaymentSystem` aufgerufen, die den vom Kunden empfangenen Befehlscode (`protocolOID`) interpretiert und die Methode `paymentMerchant` der konkreten Zahlungssystem-Implementierung aufruft, und so die für den Bezahlvorgang notwendigen Protokolle auf Händlerseite startet. Innerhalb der Methode `paymentMerchant` wird desweiteren eine Verbindung zum Server der Bank aufgebaut, um die vom Kunden im Laufe des Bezahlprotokolls erhaltenen Münzen online einzureichen.

Auf Bankseite wird die Methode `startBankProtocol` von `PaymentSystem` durch die Klasse `BankServer` gestartet, sobald sich der Händler mit dem Bankserver verbunden hat. `startBankProtocol` interpretiert den vom Händler empfangenen Befehlscode (`protocolOID`, Vgl. A.1.7) und startet die Methode `depositBank` der konkreten Zahlungssystem-Implementierung. Dadurch werden die für den Einreichvorgang notwendigen Protokolle auf Bankseite gestartet.

Nach erfolgreichem Abarbeiten der Protokolle liefert der Händler die Ware aus. Dies geschieht, indem am Ende von `paymentMerchant` die Methode `deliver` aufgerufen wird.

Auf Kundenseite wird durch den Aufruf der Methode `download` von `PaymentSystem` am Ende von `payment` der Empfang der Ware gestartet und über ein `ProductStore`-Objekt die empfangene Ware auf dem Kundenrechner gespeichert.

### A.2.3 Rückgabevorgang

Um Münzen zurückzugeben, wählt der Kunde im `Wallet` die Funktion "Rückgabe" und gibt den gewünschten Betrag ein. Die Klasse `Wallet` ruft die Methode `redemption` aus der Klasse `Client` auf, die wiederum über die Methode `redemption` von `PaymentSystem` die Kundenseite des Rückgabeprotokolls startet. Die Methode `redemption` greift über ein `CoinStore`-Objekt auf die während des Abhebevorgangs gespeicherten Münzen zu und baut eine Verbindung zum Server der Bank auf.

Die Bankseite des Rückgabeprotokolls wird über die Methode `startBankProtocol` von `PaymentSystem` durch die Klasse `BankServer` gestartet, sobald sich die Kundenseite mit dem Bankserver verbunden hat. `startBankProtocol`

interpretiert den vom Kunden empfangenen Befehlscode (`protocolID`) und startet die Methode `redemptionBank` der konkreten Zahlungssystem-Implementierung. Dadurch werden die für den Rückgabevorgang notwendigen Protokolle auf Bankseite gestartet.

#### A.2.4 Deanonymisierung

Für eine Münzverfolgung startet die Bank über ein UI die Methode `auditBank` der konkreten Zahlungssystem-Implementierung. Dabei muß sie den Abhebevorgang spezifizieren, während dem die zu verfolgenden Münzen erzeugt wurden. `auditBank` startet die Bankseite des Münzverfolgungs-Protokolls. Die Trusteeseite des Protokolls wird vom `TrusteeServer` über die Methode `startTrusteeProtocol` gestartet, sobald eine legitimierte Anfrage der Bank vorliegt. `startTrusteeProtocol` ruft die Methode `auditTrustee` der konkreten Zahlungssystem-Implementierung auf.

Für eine Kundenverfolgung startet die Bank die Methode `traceBank` der konkreten Zahlungssystem-Implementierung. Dabei muß sie die Münze spezifizieren, die deanonymisiert werden soll. `traceBank` startet die Bankseite des Kundenverfolgungs-Protokolls. Die Trusteeseite des Protokolls wird vom `TrusteeServer` über die Methode `startTrusteeProtocol` gestartet, sobald eine legitimierte Anfrage der Bank vorliegt. `startTrusteeProtocol` ruft die Methode `traceTrustee` der konkreten Zahlungssystem-Implementierung auf.