Institut für theoretische Nachrichtentechnik
PROF. DR.-ING. ANTON KUMMERT
Bergische Universität GH Wuppertal

Februar 1998

Studienarbeit

Kryptografie im elektronischen Zahlungsverkehr

—

Recherche und Vergleich von Systemen zum „Electronic Commerce“

CHRISTIAN H. GEUER,
christian.geuer@crypto.gun.de

30. Juni 1998

Hiermit versichere ich, daß ich die Arbeit selbständig verfaßt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und Zitate kenntlich gemacht habe.

Inhaltsverzeichnis

1 Einführung

Die vorliegende Studienarbeit wurde am„Lehrstuhl für theoretische Nachrichtentechnik“ an der „Bergischen Universität GH Wuppertal“ erstellt. Herzlichen Dank an PROF. ANTON KUMMERT vom „Lehrstuhl für theoretische Nachrichtentechnik“ und an PROF. CHRISTOPH RULAND vom „Institut für Nachrichtenübermittlung“ an der„Universität GH Siegen“.

1.1 Thema der Studienarbeit

> Im ersten Teil der Arbeit sollen Kriterien und Anforderungen an elektronische Zahlungssysteme erarbeitet werden unter besonderer Berücksichtigung von Sicherheitsaspekten.
>
> Der zweite Teil vergleicht und analysiert geplante und bereits implementierte Systeme auf Basis der erarbeiteten Kriterien und erörtert Sicherheitsproblematiken aus Sicht von Kunde, Händler und Bank.
>
> Zum Abschluß werden kurz Ausblicke auf geplante Erweiterungen und noch zu entwickelnde Komponenten gegeben.

1.2 Aufbau der Arbeit

Kapitel 1 soll eine kurze Motivation für die vorliegende Arbeit geben, d.h., was ist das Internet, was ist Geld, warum muß man es im Internet übertragen und warum ist die Sicherheit dabei so problematisch.

Kapitel 2 beschreibt kurz grundlegende Prinzipien der Kryptografie und der Sicherheitsprotokolle. Hier wird auf alle im Buch vorkommenden Methoden kurz eingegangen.

Kapitel 3 stellt die Anforderungen an und Eigenschaften von digitalen Zahlungesystemen dar. Neben den Anforderungen wie Sicherheit und einem wirtschaftlichen Einsatz besitzen die Systeme Eigenschaften wie Anonymität oder Teilbarkeit.

Kapitel 4 geht auf die verschiedenen Systeme ein und versucht, Begriffe und Eigenschaften der unterschiedlichen Ansätze zu erklären. Dabei werden die wichtigsten Eigenschaften aus Kapitel 3 im speziellen angewendet. Das Hauptproblem der vorliegenden Arbeit ist die Tatsache, daß nicht auf jeden Punkt eingegangen werden kann, da nicht alle Systeme komplett offengelegt sind. So sind z.B. die Interna vieler SmardCard-basierter Systeme unbekannt, da die SmartCard’s

hauptsächlich zum Schutz der System-Interna eingesetzt werden. Daher fallen speziell in diesem Abschnitt viele Beschreibungen recht mager aus.

Kapitel 5 gibt einen kurzen Überblick über geplante oder wünschenswerte Erweiterungen.

Da diese Arbeit eine Literatur-Recherche ist, wurde großer Wert auf die Ausführlichkeit des *Literaturverzeichnisses* gelegt. Ich hoffe, daß ausreichend viele aktuelle Schriften, Artikel und Web-Ressourcen aufgeführt sind.

1.3 Das Internet

Das Internet ist in aller Munde. Der Firmen–`URL` am Ende der Fernseh-Werbung und das `http://www.blablabla.de/` in den Reklamen der Printmedien sind nicht mehr wegzudenken. Es werden Produkte wie Software oder Dienstleistungen wie Übersetzer-Services angeboten, doch läßt die Nachfrage auf dem Online-Weg i.A. zu wünschen übrig. Die Marktforscher und Propheten des Business erwarten ein Wachstum des Internet-Commerce, welches ähnliche Dimensionen wie das Wachstum des Netzes selbst haben soll. Doch bis dato gibt es wenig Möglichkeiten, auch wirklich online Geschäfte abzuwickeln. Denn wenn man die Angebote der Firmen ansieht, findet man außer Selbstdarstellungen und evtl. der Möglichkeit, sich Werbung (Verbraucherinformation) zuschicken zu lassen, selten etwas, um online eine Bestellung zu tätigen. Häufig wird auf den Webseiten der Händler auf die telefonische oder schriftliche Bestell-Möglichkeiten verwiesen.

Das erklärte Ziel der Wirtschaft und der Banken ist also, gesellschaftliche Strukturen wie den Handel und das Geld in die elektronische Welt abzubilden.

In manchen Fällen jedoch bietet sich dem Kunden die Möglichkeit, doch online einen Kauf zu tätigen. Und hier kommen die Kreditkarten ins Spiel. Der Kunde wird aufgefordert, seine Bestellung zusammen mit seiner Kreditkartennummer via e-mail zum Händler zu transferieren oder er wird aufgefordert, seine Daten via CGI-Skript direkt in die Händler-Datenbank zu befördern.

Nun sind mit dieser Handlungsweise verschiedene Sicherheitsprobleme verbunden.

1.4 Das Internet und seine Sicherheitsprobleme

In der Welt des Internet werden Datenpakete von A nach B befördert, ohne sich um Inhalte, Absender oder Transportwege zu kümmern. Der Sender versieht die Daten einfach mit Adress-Information und liefert sie bei seinem lokalen Netzwerkanschluß ab. Welchen Weg die Daten nun nehmen, durch welche Leitungen sie laufen, in wessen Computer sie zwischengespeichert werden, kann vom Absender nicht kontrolliert werden.

Nun kann in einem Vermittlungsrechner (oder auch auf einer Übertragungsleitung/Funkstrecke) der gesamte Paketinhalt abgehört werden, d.h., der Besitzer des Computers oder auch der Lauscher an der Leitung ist in der Lage, herauszufinden von wem das Paket kommt, an wen es geht und auch was der Inhalt des Paketes ist. Darüberhinaus besteht sogar die Möglichkeit, durchlau-

fende Pakete zu verändern oder sogar neue Pakete mit falscher Absenderinformation zu generieren. Es muß also davon ausgegangen werden, daß der Inhalt der Pakete bekannt wird, daß er verändert wurde oder daß er von jemand ganz anderem kommt, als im Adress-Feld vermerkt ist.

Um die Problematik noch aufzuweiten, kann ein Paket durch extrem viele Vermittlungsrechner laufen, d.h., mit jedem Rechner und jeder Leitung auf dem Weg steigt die Chance, daß irgend eine unbefugte Handlung vorgenommen wird, extrem an. Wenn also in irgendeinem Bestellformular oder in einer e-mail die Kreditkartennummer übertragen wird, ist es durchaus möglich, daß sie in falsche Hände kommt. Man darf diese Möglichkeit allerding nicht dahingehend deuten, daß jede übertragene Information in die Hände von Gaunern kommt. Wenn man in irgendeiner Absteige in Südostasien seine Kreditkarte aus der Hand gibt und der Kellner damit im Hinterzimmer verschwindet, ist das Risiko bedeutend höher als im Internet, aber auch dort ist es nicht zu vernachlässigen.

1.5 Parteien

In den Beschreibungen der einzelnen Protokolle sind verschiedene Parteien erwähnt, die im folgenden kurz erklärt werden sollen.

1.5.1 Der Kunde

Der Kunde möchte beim Händler etwas kaufen. Er möchte eine Ware bekommen und ist bereit, dafür im Gegenzug eine bestimmte Menge Geld an den Händler zu zahlen.

Aus Sicht des Kunden besteht die Möglichkeit, daß der Händler ihn betrügt, d.h., daß der Kunde eine Zahlung an den Händler leistet und sich der Händler danach weigert, die Ware zu liefern. Eine solche Weigerung kann z.B. dadurch begründet sein, daß der Händler sagt, er habe das Geld nicht erhalten. Bei einem Kauf im Ladengeschäft des Händlers besteht diese Möglichkeit nicht, da der Kunde im direkten Kontakt (von Angesicht zu Angesicht) mit dem Händler steht und hier ein fairer Austausch Geld gegen Ware und Quittung erfolgt.

Der Kunde ist an einfach zu bedienenden Systemen interessiert, die von vielen Händlern akzeptiert werden. Nach Möglichkeit sollten alle Anwendungsfälle innerhalb eines Programms angedeckt sein. Der Kunde möchte beim Händler nicht komplizierte Formulare ausfüllen müssen, um eine Transaktion einzuleiten. Vorteilhaft wäre es auch noch, die Geschäfte nicht nur von Zuhause aus erledigen zu können, sondern auf der Arbeit oder von Unterwegs Waren ordern und bezahlen zu können. Darüber hinaus sollte das System keine hohen Kosten verursachen, die letztendlich auf den Verbraucher abgewälzt werden.

1.5.2 Der Händler

Der Händler bietet Ware an. Bei dieser Ware kann es sich sowohl um materielle Güter handeln, die nach der Bezahlung auf dem Versandweg zum Kunden gelangen als auch um digitalisierte

Produkte wie Informationen, Texte, Bilder oder Musik, die der Kunde direkt über das verwendete Netz beziehen kann. Der Händler gibt dem Kunden die Möglichkeit, im Austausch gegen einen vorher festzulegenden Betrag Waren zu erhalten. Der Händler muß sich dagegen schützen, daß der Kunde nach erfolgter Lieferung die Zahlung verweigert.

Der Händler ist bereit, die Lieferung vor der Bezahlung durchzuführen, wenn die Bank als vertrauenswürdiger Dritter bereit ist, für den Kunden geradezustehen.

Der Händler ist an einem Zahlungssystem interessiert, das ihm einen möglichst großen Kundenkreis erschließt. Das Zahlungssystem sollte Zahlungen jedweder Größe annehmen und alle Transaktionen automatisch ohne Eingriffe durch den Händler anwickeln können. Darüber hinaus ist der Händler an einer genauen Aufstellung der getätigten Transaktionen und georderten Artikel interessiert, um Kundenprofile erstellen zu können. Diese Profile erlauben es dem Händler, seine Produktpalette und Werbung noch exakter auf den gewünschten Kundenkreis auszurichten.

1.5.3 Die Bank

Die Bank vertritt die Ansprüche von Kunden und Händlern. Die Bank muß sich sowohl gegen betrügerische Kunden als auch gegen betrügerische Händler schützen. Die Bank muß nicht erfahren, was Handlungsgegenstand zwischen dem Kunden und dem Händler war.

In vielen Zahlungsprotokollen dient die Bank als Clearing-Stelle zwischen Kunde und Händler, d.h., sie garantiert dem Händler die Zahlungsfähigkeit des Kunden. Die Bank agiert somit als Risiko-Manager, d.h., sie teilt das Risiko eines Betruges gerecht zwischen Kunde und Händler auf. In technischer Hinsicht bezieht sich das Risiko-Managment auf die Bereitstellung sicherer Zahlungsverfahren, in finanzieller Hinsicht übernimmt die Bank das Risiko eventueller Zahlungsausfälle oder unerlaubter Belastungen des Kontos.

Die Bank ist an einem Zahlungsverfahren interessiert, das möglichst viele Kunden und Händler an die Bank bindet. Der Ausbau einer Monopolstellung ist der Anreiz, ein proprietäres Verfahren einzusetzen, das nur von der eigenen Bank angeboten wird. Die Kunden und Händler dagegen wünschen sich Verfahren, die eine Verrechnung der Beträge zwischen den jeweiligen Kreditinstituten erlauben.

2 Kryptografie und grundlegende Prinzipien

Bevor im einzelnen die Interna der verschiedenen Implementierungen erörtert werden, sollen in diesem Kapitel kurz die mathematischen und kryptografischen Grundlagen beleuchtet werden, die für das Verständnis unbedingt notwendig sind.

2.1 Symmetrische Verschlüsselung

Bei den symmetrischen Verschlüsselungsverfahren werden binäre Daten mit Block– bzw. Stromchiffren verschlüsselt. Hierbei handelt es sich um Algorithmen, die für die Ver– und die Entschlüsselungsoperation den selben Schlüssel verwenden, daher auch als *single-key* bekannt. Da die symmetrischem Verfahren wesentlich länger bekannt sind, werden sie auch als *konventionelle* Chiffren bezeichnet.

Blockchiffren: Blockchiffren bilden Datenblöcke festgelegter Größe auf ebensogroße Blöcke ab. Es handelt sich hierbei um eine eindeutige Abbildung. Die Abbildungsvorschrift wird durch den Schlüssel bestimmt. Bei einer Schlüssellänge von n bit gibt es somit 2^n unterschiedliche Abbildungsvorschriften. Die Blöcke werden aus zusammnengefaßten Bytes des Klartextes gebildet. Die Größe liegt normalerweise zwischen 8 und 16 Bytes. Bekannte Blockchiffren sind der DES (Data Encryption Standard), Tripel-DES, IDEA (International Data Encryption Algorithm) oder auch Blowfish.

Stromchiffren: Bei Stromchiffren handelt es sich um Generatoren, die einen Bitstrom liefern, der zum Klartextbitstrom hinzuaddiert wird. Sie arbeiten bit-weise, manchmal auch Byte-weise. Die Generatoren erzeugen nach einer Initialisierung mit dem Schlüssel einen bit-Strom mit extrem langer Periodizität. Der erzeugte bit-Strom ist pseudozufällig und darf nicht vorhersagbar sein. Als bekannte Vertreter dieser Gattung sind hier der RC4 von RSA Data Security Inc. und der A5, der im GSM-Netz verwendet wird, zu nennen.

2.2 Asymmetrische Verschlüsselung

Das nach seinen Erfindern Rivest, Shamir und Adleman benannte RSA-Verfahren [RSA78] ist das wohl bekannteste Public-Key-Verfahren. Es basiert auf modularer Arithmetik mit Primzahlen. Sowohl Ver- als auch Entschlüsselung bestehen aus einer Exponentiation modulo einer

Zahl n, die das Produkt zweier geheimer Primzahlen ist. Die Sicherheit des Systems liegt in der Schwierigkeit der Faktorisierung dieser Zahl. Bis heute gibt es keinen Weg, die bei hinreichend großen Zahlen eine Faktorisierung in vernünftiger Zeit durchführt.

Mit RSA können Verschlüsselung/Entschlüsselung und auch el. Signatur bzw. deren Verifikation durchgeführt werden. Die mathematische Relation zwischen dem öffentlichen und dem geheimen Schlüssel ist dergestalt, daß die beiden zusammengehören, aber gleichberechtigt sind, d.h., nach der Generierung der Zahlen kann der Nutzer frei entscheiden, welchen der beiden er veröffentlicht und welcher der Private sein soll. Der private Schlüssel wird zum entschlüsseln und unterschreiben verwendet, der öffentliche dient verschlüsseln bzw. der Verifikation der Unterschrift. Hier soll nun kurz beschrieben werden, wie das Verfahren arbeitet:

1. Zuerst werden zwei hinreichend große Primzahlen p und q erzeugt. Diese sollten heutzutage eine Größe zwischen 500 und 1000 bit aufweisen. Die Zahlen werden zufällig gewählt und dann mit Primzahltests geprüft.

2. Das Produkt $n = pq$ bildet den Modulus in den nachfolgenden Berechnungen. Dieses Produkt hat gewöhnlich eine Größe von 1000 bis 2000 bit, welche für heutige Verhältnisse normal sind. Je größer die Operanden sind, desto länger dauern die Operationen, auf der anderen Seite steigt in gleichem Maße die Sicherheit der Parameter.

3. Danach wird ein geheimer Schlüssel e gewählt. Der größte gemeinsame Teiler von e und dem Produkt $((p-1)(q-1))$ muß 1 sein, d.h., e muß relativ prim zu $((p-1)(q-1))$ sein.

4. Im Anschluß wird der öffentliche Schlüssel d gewählt. Dieser ist die Inverse zu e mod $((p-1)(q-1))$. Die Inverse wird i.A. unter Verwendung von Euklids Algorithmus bestimmt. Es muß also gelten:

   $$ed \equiv 1 \mod ((p-1)(q-1))$$

   bzw.

   $$d = e^{-1} \mod ((p-1)(q-1))$$

5. Die beiden Faktoren p und q werden nun nicht mehr benötigt und können vernichtet werden. Das Wertepaar e und n bildet nun den geheimen Schlüssel, der öffentliche setzt sich aus d und n zusammen.

Nachdem nun die Schlüsselgenerierung abgeschlossen ist, können die Schlüssel verwendet werden. Der Vorgang von Verschlüsselung und Signatur ist relativ einfach. Die Nachricht wird in eine Zahl m ($m \hat{=}$ Message) umgewandelt, wobei $m < n$ gelten muß . Falls diese Konvertierung nicht möglich ist, kann die Nachricht auch in mehrere Blöcke m_i umgewandelt werden.

Nun kann die eigentliche Operation ausgeführt werden:

$$
\begin{aligned}
c &= m^d \bmod n \\
m &= c^e \bmod n \\
&= \left(m^d \bmod n\right)^e \bmod n \\
&= m^{d^e} \bmod n \\
&= m^{de} \bmod n \\
&= m \bmod n
\end{aligned}
$$

2.3 Digitale Signatur

Bei der digitalen Signatur handelt ist sich im Normalfall um eine Mischung von asymmetrischen Verfahren (RSA) und Hash-Funktionen (siehe unten). Die Hash-Funktion bildet eine Prüfsumme der zu signierenden Nachricht; diese Prüfsumme wird mit dem privaten Schlüssel verarbeitet (signiert). Das Ergebnis wird an die Nachricht angehängt. Der Empfänger und Verifizierer der Nachricht berechnet ebenfalls den Hash-Wert der Nachricht, berechnet mit dem öffentlichen Schlüssel des Absenders den vom Absender erzeugten Hash-Wert und überprüft, ob beide identisch sind.

2.4 Hash-Funktionen

Eine Hash-Funktion $H(m)$ bildet eine beliebig lange Nachricht m eindeutig auf einen Wert h fester Länge ab: $h = H(m)$. Eine bekanne Hash-Funktion ist die CRC[1]-Berechnung, welche eine einfache Prüfsumme darstellt. Damit eine Funktion eine„kryptografisch sichere“ Hash-Funktion bildet, müssen weitere Bedingungen gelten:

- $h = H(m)$ muß einfach und schnell zu berechnen sein
- Die Funktion darf nicht umkehrbar sein, d.h., zu gegebenem h ein m_i zu konstruieren. Es darf also keinen Weg geben, um $m = H^{-1}(h)$ zu berechnen.
- Wenn m_1 und $h_1 = H(m_1)$ gegeben sind, darf es nicht möglich sein, ein m_2 zu konstruieren, für das $H(m_2) = H(m_1) = h_1$ gilt.

Häufig wird gefordert, daß die Änderung eines einzelnen bits im Eingangsdatenstrom jedes bit im Hash-Wert mit einer Wahrscheinlichkeit von 50% ändert. Mit Hash-Funktionen werden also Veränderungen, sei es surch Übertragungsfehler oder auch Fälschungen, sichtbar gemacht.

[1] Cyclic Redundancy Check; dient der Überprüfung von störungsfreier Übertragung

2.5 Blinde Signaturen

Bei der „normalen“ Signatur sind der unterschreibenden Partei die zu signierenden Inhalte bekannt. Das Konzept der blinden Signatur wurde erstmals von DAVID CHAUM in [Cha83] vorgestellt. CHAUM bezitzt Patente für verschiedene Varianten blinder Signaturen.

Alice legt Bob ein Dokument vor, welches von Bob unterschrieben werden soll, ohne daß Bob Kenntnis vom Inhalt erhalten soll. Eine einfache Möglichkeit wäre, Bob einfach die Hash-Werte der zu signierenden Daten zu übergeben und diese unterschreiben zu lassen. Bob kann dann aber eine Datenbank mit Hash-Werten aufbauen; wenn ihm die Dokumente dann zu Gesicht kommen, kann nachvollzogen werden, *wer* ihm die Hashwerte *wann* zur Unterschrift vorgelegt hat.

Daher wird bei vollständig blinden Signaturen ein anderer Weg gegangen. Hier wird die Kommutativität des RSA-Verfahrens ausgenutzt:

1. Alice erzeugt ihre Nachricht m
2. Alice erzeugt eine Zufallszahl k, welche zum Modul n teilerfremd sein muß und berechnet dann

   $$r = mk^e \bmod n,$$

   d.h., k wurde mit Bobs öffentlichem Schlüssel e verschlüsselt. Bob kann die beiden Faktoren m und k^e nicht voneinander trennen.
3. Nachdem Bob r vorgelegt bekommen hat, signiert er r mit seinem geheimen Schlüssel d:

   $$r^d \bmod n = (mk^e)^d \bmod n = m^d k^{ed} \bmod n$$

   Im RSA–Verfahren gilt nun $M^{ed} \bmod n = M \bmod n$ für $M < n$. Also gilt

   $$r^d \bmod n = m^d k \bmod n$$
4. Alice kann nun einfach durch die Berechnung

   $$t = r^d k^{-1} \bmod n = m^d \bmod n$$

   den *Blindingfaktor* k herausrechnen und hat mit $t = m^d \bmod n$ eine blind unterschriebene Nachricht.

2.6 Signaturen mit diskreten Logarithmen

Neben RSA können diskrete Logarithmen zur digitalen Signatur herangezogen werden. Die Stärke dieses Systems basiert auf der Schwierigkeit, die Berechnung $g^a \bmod p$ umzukehren, wenn p eine Primzahl ist. g wird der Generator genannt und a ist eine ganze Zahl. g, p und $g^a \bmod p$ sind bekannt, und es ist nicht möglich, a zu bestimmen.

Obwohl es ein schwieriges Thema ist, stehen verschiedene Kryptologen auf dem Standpunkt, daß der diskrete Logarithmus sicherer als RSA ist. RSA kann sowohl durch Erfolge in der Faktorisierung als auch durch die Lösung des DiscreteLog-Problems gebrochen werden. Ein neuer Faktorisierungsalgorithmus würde p und q aufdecken und die Lösung des diskreten Log. würde RSA brechen, wenn die beiden Moduli nicht prim wären, was durchaus möglich ist, da ja i.A. nur mit hinreichender Sicherheit festgestellt wird, ob p und q prim sind.

Um den diskreten Logarithmus zu verstehen, soll im nachfolgenden eine interaktive Session des Beweises und der Überprüfung beschrieben werden. In der Grundform des disk. Log. ist für eine Unterschrift eine Interaktion zwischen dem Beweisenden und dem Prüfenden norwendig, was durch leichte Modifikationen auch obsolet wird. Die nachfolgenden Schritte sind stark an [Way97] angelehnt:

1. Ein Dokument mit dem Hash-Wert m soll signiert werden. Dazu wird $m^x \bmod p$ berechnet, wobei x der geheime Schlüssel ist. Der Beweisende schickt dem Prüfer das Tupel, bestehend aus der Signatur $m^x \bmod p$, dem Generator g, p und $g^x \bmod p$. Die letzten 3 Werte können auch in einem öffentlichen Directory abgelegt werden, wo der Prüfer sich die Daten holt.

2. Wenn nun die Unterschrift geprüft werden soll, wählt der Beweisende eine Zufallszahl w und berechnet $g^w \bmod p$ und $m^w \bmod p$. Der Wert w wird geheim gehalten.

3. Der Prüfer erzeugt eine Zufallszahl c und schickt sie an den Beweisenden. Hier wird sozusagen ein zero-knowledge-proof durchgeführt.

4. Der Beweisende schickt $r = cx + w$ zurück an den Prüfer. Aus den dem Prüfer bekannten Werten r und c können die anderen beiden Teile der Geradengleichung nicht hergeleitet werden.

5. Der Prüfer berechnet jetzt $g^r \bmod p$ und vergleicht das Ergebnis mit $g^{x^c} \cdot g^w \bmod p$. Wenn die beiden Ergebnisse gleich sind, ist die Unterschrift geprüft.

Das System basiert auf der Tatsache, daß nur der Besitzer von x die Geradengleichung so lösen kann, daß $r = cx + w$ sich in $g^r \bmod p = g^{cx} g^w \bmod p$ transformiert. Durch die Verwendung von Hash-Funktionen kann dem System die Notwendigkeit für einen Dialog genommen werden. Die modifizierte Variante läuft wie folgt ab:

1. Der Hash-Wert m wird wie zuvor durch $m^x \bmod p$ gebildet. w ist eine Zufallszahl. $m^w \bmod p$, $g^w \bmod p$, g und $m^x \bmod p$ werden berechnet.

2. In dieser Variante kann der Prüfer ja keine Zufallszahl c mehr schicken, daher wird $c = \text{Hash}\{m^x \bmod p, m^w \bmod p, g^w \bmod p\}$ gebildet, also alle 3 Exponentiationen gehen als Input in eine sichere Hash-Funktion ein. Da das Ergebnis aufgrund der Eigenschaften einer Hash-Funktion nicht vorherbestimmt werden kann, kann der Prüfer sicher sein, daß c ein zufälliger Wert ist, der für jeden Wert von w anders ist.

3. r, $m^x \bmod p$, $m^w \bmod p$ und $g^w \bmod p$ bilden die Unterschrift, wobei $r = cx + w$ gilt.

4. Der Prüfer kann die Unterschrift prüfen, wobei er c selbst durch den Hash berechnen muß und dann wie zuvor vorgeht.

2.7 Bit Commitment

Bit-Commitment bedeutet bit-Festlegung. Hierbei handelt es sich im ein interaktives Protokoll, welches z.B. vor einer Berechnung erfolgt, um gewisse Parameter für die Berechnung festzulegen, an die sich beide Seiten halten können, ohne daß einer von beiden im Laufe des Geschehens seine Meinung revidieren kann und sich für etwas anderes entscheidet. SCHNEIER [Sch96] formuliert es so: *„Alice möchte sich auf eine Voraussage verpflichten (d.h. auf ein bit oder eine Folge von bits), möchte diese aber erst später offenlegen. Bob dagegen möchte sicherstellen, daß Alice ihre Ansicht nicht ändert, nachdem sie sich auf ihre Voraussage festgelegt hat.*“

Bit-Commitment kann beispielsweise mittels Hash-Funktionen geschehen. Alice berechnet $\text{Hash}\{\text{Random} \,||\, \text{Data}\}$ und schickt das Ergebnis an Bob. Hierbei ist Random eine Zufallszahl und Data die festzulegenden Daten. Wenn sie nun ihre Daten offenlegen will, schickt die Bob die beiden Werte. Durch die Eigenschaften der Hash-Funktion wird verhindert, daß Alice ihre Meinung im Nachhinein ändert, durch die Zufallsdaten verhindert Alice, daß Bob ihre Daten schon vorher berechnet, denn wenn es sich nur um ein bit handeln würde, gäbe es ja nur zwei Hash-Werte.

2.8 Zufallszahlen

Zufallszahlen sind oft der Knackpunkt bei vielen kryptografischen Applikationen. Selbst wenn kryptografisch sichere Algorithmen zur Chiffrierung Verwendung finden und auch das Protokoll selbst formal genau spezifiziert und auf Schwachstellen hin analysiert ist, können Implementierungsfehler bei der Zufallszahlenerzeugung die gesamte Arbeit zunichte machen. Ein Beispiel ist eine Version des Netscape Navigator, bei dem der Zufallszahlengenerator zur Erzeugung von Sitzungsschlüsseln für SSL schlechte Eingangswerte hatte und ein Angreifer die Ergebnisse relativ leicht bestimmen konnte.

Was ist nun unter schlechten Eingangswerten zu verstehen? Da im Rechner per Definition erst mal kein Zufall existieren soll (warum hieße er sonst Rechner?), muß ein Zufallszahlengenerator seine Werte aus nicht vorhersagbaren Eingabewerten ableiten. PGP verwendet Hash-Funktionen

als Zufallszahlengenerator. Um ein bißchen Entropie zu sammeln, mißt PGP die Zeitdauer zwischen zwei Tastaturanschlägen des Anwenders. Die Windows-Version des Secure-Shell-Clients leitet aus willkürlichen Mausbewegungen des Anwenders die Eingabe für den Zufallszahlengenerator ab. Wenn man aber Parameter wie die Uhrzeit als einzige Eingabe für die Zufallsfunktion nutzt, wird so der Wertebereich extrem eingeschränkt. Wenn in einer Implementation der Input des Generators nur eine Entropie von 1 bit hat, dann besitzt auch die Ausgabe nur 1 bit Entropie.

2.9 Challenge & Response

Challenge & Response-Systeme sind Frage/Antwort-Systeme. Bob möchte die Identität von Alice prüfen und stellt daher Fragen, die Alice nur mit geheimem Wissen beantworten kann. Bob und Alice verfügen über das gleiche Geheimnis. Wenn Bob Alice aber einfach nach dem Geheimnis fragt und das Geheimnis im Klartext über einen unsicheren Kanal übertragen wird, kann auch ein Lauscher zukünftig als Alice auftreten. Daher stellt Bob Alice eine Rechenaufgabe, die Challenge. Alice und Bob berechnen beide mit Hilfe des beiden bekannten Geheimnisses die Antwort. Alice nennt Bob ihre Antwort (Response). Bob prüft, ob der übertragene Wert mit dem von ihm berechneten identisch ist. Bei Gleichheit ist der Beweis erbracht.

In der Praxis werden Challenge & Response-Systeme mit Zufallszahlen als Challenge, einer Zufallszahl als Geheimnis und einer kryptografisch sicheren Hashfunktion als Rechenaufgabe genutzt. Bob und Alice kennen beide die Zahl S. Bob nennt Alice die Zufallszahl C, und beide berechnen $R = \text{Hash}\{C \,||\, S\}$, d.h., die Hashfunktin wird auf die Konkatenation von C und S angewendet. Alice sendet Bob R zurück und Bob kann die Identität prüfen. C und R werden über den unsicheren Kanal übertragen. Aufgrund der Einweg-Eigenschaft kann ein Angreifer nicht auf S schließen. Wichtig ist nur, daß jeder Wert C nur einmal verwendet wird, da ein Angreifer bei Wiederholungen die richtige Response liefern kann.

2.10 Hybride Systeme

Fast alle kryptografischen Programme und Applikationen sind hybrid, d.h., es werden verschiedene kryptografische Algorithmen gemeinsam verwendet, um die gewünschte Funktionalität zu erreichen.

Das z.Zt. wohl bekannteste Programm zur Verschlüsselung von e-mail ist PGP. Hier werden Nachrichten mit einer symmetrischen Chiffre (IDEA oder 3DES) verschlüsselt, wobei ein zufällig gewählter Session-Key erzeugt wird. Dieser Session-Key wird nun mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Somit kann also die Verschlüsselung der eigentlichen Daten unter Verwendung einer schnellen (symmetrischen) Chiffre erfolgen und nur der Session-Key muß durch das langsame (asymmetrische) Verfahren geschleust werden.

Genau der gleiche Ansatz wird bei der Unterschrift werwendet: Die eigentlichen Daten werden durch eine schnelle Hash-Funktion (MD5) geschickt und nur der (deutlich kleinere) Hash-Wert

muß nun unterschrieben werden.

2.11 Ein einfaches anonymes Online-Zahlungssystem

Um einem Kunden die Möglichkeit zu geben, anonym elektronische Zahlungen zu leisten, eignen sich D. CHAUM's blinde Signaturen hervorragend. Der Kunde erzeugt eine Zufallszahl ausreichender Größe, die als Seriennummer dienen soll. Er blindet sie und läßt sie von der Bank signieren. Die Bank hat für jeden möglichen Münzwert (1 Pfennig, 2 Pfennig, 4 Pfennig, 2^3 Pfennig, etc.) einen individuellen Signaturschlüssel. Die Bank verwendet den gewünschten Signaturschlüssel für die Münze und schreibt sich den entsprechenden Betrag selbst gut.

Der Kunde entfernt nun den Blinding-Faktor und hat die blind signierte Münze. Wenn er die Münze an einen Händler weitergibt, reicht der Händler die Münze bei der ausgebenden Bank ein. Die Bank prüft anhand der Seriennummer, ob die Münze zum ersten Mal eingereicht wurde. Wenn das der Fall ist, trägt sie die Münze in ihre Datenbank für „verbrannte" Münzen ein und gibt dem Händler den Betrag. Anhand der blinden Signatur kann die Bank prüfen, daß die Münze wirklich von ihr signiert wurde.

Da die eingereichte Münze nicht mit der blind signierten Münze in Verbindung gebracht werden kann, ist es eine anonyme Zahlung gewesen. Das Protokoll muß allerdings online abgewickelt werden, damit das Mehrfachausgeben ein- und derselben Münze rechtzeitig aufgedeckt werden kann.

Wenn alle Münzen vom Kunden individuell gewählte Werte haben sollen, ist eine Protokolländerung notwendig. Der Kunde erzeugt n Münzrohlinge, die alle mit unterschiedlichen Faktoren geblindet werden. Ein Rohling besteht jetzt nicht nur aus einer Seriennummer, sondern auch einer Wertangabe. Der Kunde reicht alle Rohlinge bei der Bank ein. Die Bank fordert die Blindingfaktoren zu allen bis auf einen Rohling. Nachdem sie alle Rohlinge decodiert hat und sich überzeugt hat, daß die Wertangaben alle identisch sind (der Kunde also nicht versucht hat, zu betrügen), signiert sie den letzten (nicht geprüften) Rohling mit ihrem Schlüssel und reicht ihn an den Kunden zurück. Der Kunde rechnet den Blinding-Faktor heraus und verfügt dann über eine digital signierte Münze, in dem der gewünschte Wert festgehalten ist.

2.12 Ein einfaches anonymes Offline-Zahlungssystem

In einem Off-Line-System [Way97, Seiten 57–61] kann die Bank die Gültigkeit bzw. die Erstverwendung der Münze nicht sofort nach Verwendung prüfen. Daher wird der Kunde beim Münz-Abheben, d.h., bei der Münzerzeugung, dazu gezwungen, seine Identität fest in die Münze zu codieren. Wenn die Münze ein einziges Mal ausgegeben wird, kann die Bank bei Einrichung der Münze die Identität des Kunden nicht rekonstruieren. Wenn die Münze ein zweites Mal ausgegeben wird, erhält die Bank genug Information, um die Identität zu rekonstruieren. Dies wird durch Cut-and-Choose– bzw. Bit-Commitment–Protokolle beim Zahlvorgang gewährlei-

stet. Beim Online-Zahlungsvorgang hat der Käufer dem Händler die Münze einfach 1:1 herüberkopiert, wie er es bei jedem anderen Händler auch getan hätte, weil die Münze online geprüft werden konnte.

Der Kunde erzeugt k Münzen, von denen die Bank wieder zufällig eine auswählt und die Anderen offenlegen läßt. Zusätzlich zur individuellen Seriennummer muß jetzt die Identität mit in die Münzen eingebracht werden:

Um die Identität in die Münze einzuprägen, wird die Identität zuerst einmal als String I dargestellt:

$$I = (\text{Kundenidentität} \mid\mid \text{Bankenname})$$

Um zwei Hälften (id_1, id_2) der Identität zu generieren, wird die Identität mit eine Zufallsfolge P exkulsiv-OR-verknüpft:

$$id_1 = I \text{ xor } P$$

und

$$id_2 = P$$

Die Identität kann somit aus

$$I = id_1 \text{ xor } id_2 = I \text{ xor } P \text{ xor } P = I$$

wiederhergestellt werden.

Der Kunde erstellt j Identitätspaare, jeweils mit neu gewähltem P. Wenn die Bank also an zwei Hälften $id_{1,i}$ und $id_{2,i}$ gelangt, kann sie die Identität rekonstruieren.

Jede der $2j$ Hälften in jeder der k Münzrohlinge wird jetzt mit einem bit-commitment-bestimmten Schlüssel $Key_{(a,b)}$ symmetrisch verschlüsselt. Beispielsweise werden im Rohling m die beiden Identitäten $id_{1,i}$ und $id_{2,i}$ mit den Schlüsseln $Key_{(2i,m)}$ und $Key_{(2i+1,m)}$ verschlüsselt. Die Schlüssel werden mit in den Münzen festgelegt (Bit-Commitment).

Die k Münzen werden mit individuellen Blindingfaktoren geblindet und der Bank übermittelt. Die Bank wählt jetzt zufällig eine Münze aus, die sie nicht zu untersuchen beabsichtigt und läßt sich für die anderen $(k-1)$ Münzen die Blind-Faktoren liefern. Die Bank entschlüsselt die Identitätshälften mit Hilfe der in den Münzen enthaltenen Schlüssel und prüft, ob alle Identitäten korrekt aufgebaut wurden.

Die Bank signiert die verbliebene Münze und gibt sie dem Kunden, der den Blinding-Faktor herausrechnet und seine digital signierte Münze erhält.

Wenn der Kunde einen Kauf tätigen möchte, erzeugt der Händler einen zufälligen j-bit-Vektor, den er dem Kunden gibt.

Für jedes 0-bit i enthüllt der Kunde $id_{1,i}$, indem er den Schlüssel $Key_{(2i,b)}$ liefert (der ja per Bit-Commitment festgelegt ist). Für jedes 1-bit i enthüllt der Kunde $id_{2,i}$, indem er den Schlüssel $Key_{(2i+1,b)}$ liefert. Am Ende hat der Kunde j unterschiedliche Hälften der Identität enthüllt. Aber keine zwei Hälften, die dem Händler bekannt sind, bilden ein Paar, das die Identität enthüllen könnte.

Der Händler kann prüfen, daß der Kunde die richtigen Schlüssel präsentiert hat, weil der Kunde alle Schlüssel im Bit-Commitment-Schema festgelegt hat. Darüber hinaus kann der Händler die Signatur der Bank prüfen. Der Händler reicht die Daten und den verwendeten j-bit-Vektor an die Bank weiter, die alles speichert.

Wenn eine Münze bei der Bank eingereicht wird, die die gleiche Seriennummer hat, aber einen neuen j-bit-Vektor hat, kann die Bank die Kundenidentität offenlegen. Da der Händler den j-bit-Vektor zufällig erzeugt, ist die Chance, daß zwei Händler den gleichen Vektor generieren, 2^{-j}.

Wenn die Seriennummer und der j-bit-Vektor der neu eingereichten Münze mit einer bereits eingereichten Münze identisch sind oder die entschlüsselten Hälften kein sinnvolles Ergebnis liefern, weiß die Bank, daß der Händler einen Betrug versucht hat.

Auch diese Münz-Variante muß nach einer Transaktion bei der Bank eingereicht werden. Da aber die Identität in den Münzen festgehalten ist, kann der Händler mit der Einreichung warten und die Ware schon ausliefern, bevor die Münzen geprüft sind. Das Problem mit diesen Münzen liegt in der Größe einer einzelnen Münze. Durch die vielen Daten, die in der Münze gespeichert werden müssen, werden die Münzen sehr groß. Darüber hinaus ist auch die Münzerzeugung relativ teuer, weil viele aufwendige Protokollschritte notwendig sind, um die Integrität einer Münze zu sichern.

3 Kriterien und Anforderungen an elektronische Zahlungssysteme

Im folgenden werden Bewertungskriterien und Anforderungen für elektronische Zahlungssysteme eingeführt, welche als Grundlage für den nachfolgenden Vergleich der unterschiedlichen Systeme dienen sollen.

Eine exakte Abgrenzung zwischen den verschiedenen Unterpunkten ist nicht möglich, da sich vieles überschneidet. Dennoch sind verschiedene Punkte zur Verdeutlichung explizit erwähnt.

3.1 Systemsicherheit

Die Systemsicherheit dient dem Schutz aller im Zahlungsvorgang beteiligten Parteien vor Betrügern.

Der Kunde muß sicher sein, daß sein Geld ihm nicht bei der Übertragung zum Händler durch abhören der Leitung abhanden kommt, d.h., es wird gefordert, daß auch bei ungesicherten Übertragungsleitungen ausreichender Schutz gewährleistet wird. Dies kann z.B. durch Verschlüsselung der Daten geschehen.

3.2 Transaktionskosten und Effizienz

Die Transaktionskosten sind ein wichtiger Faktor bei der Bewertung der Systeme. Bei Online-Systemen entstehen Kosten durch Nutzung von Telefon- oder Datenleitungen, bei Offline-Systemen werden die Kosten im wesentlichen durch die Kosten für die eingesetzte Hardware bestimmt. Wenn bei einem System zusätlich zum automatisierten Ablauf noch manuelle Schritte notwendig werden, schnellen die Kosten so schnell in die Höhe, daß nur noch große Transaktionen effizient sind.

3.3 Anonymität

Bei anonymen Zahlungssystemen kann die Bank eine eingenommene Münze nicht mit dem Abhebevorgang der Münze verknüpfen. Wenn die Bank alle Seriennummern und die Namen der abhebenden Personen notieren würde und ein Geldschein nur einmal verwendet werden könnte,

bevor er zur Bank zurück muß, könnte die Bank exakt feststellen, wer mit wem Geschäftsbeziehungen unterhält. An irgendeiner Stelle muß die Spur unterbrochen werden, wenn ein anonymes Zahlungssystem gefordert ist.

3.3.1 Rückverfolgbarkeit

Rückverfolgbarkeit bedeutet die Möglichkeit, im nachhinein den Weg des Geldes zu verfolgen. Das Geld wird von einer ausgebenden Stelle erzeugt, vom Kunden weitergereicht, evtl. über mehrere Stationen und wird am Ende vom Händler bei der Bank eingereicht. Wenn nun eine Möglichkeit besteht, den kompletten Weg des Geldes zu bestimmen, existiert ein Instrumentarium zur Offenlegung aller Handelsbeziehungen.

Dies ist zum Schutz der Privatsphäre des Einzelnen zu verhindern.

3.3.2 Mißbrauch der Anonymität

Problematisch wird der Schutz der Privatsphäre, wenn dieses Feature ausgenutzt wird, beispielsweise um digitalen Bankraub, Erpressung oder auch digitale Geldwäsche zu betreiben. Daher können Algorithmen zum Einsatz kommen, die im Verdachtsfall eine Offenlegung der Geldwege zulassen. Die einfache und bequeme Nutzung digitalen Geldes kann laut [FW96] auch zu Steuerflucht größeren Ausmaßes führen.

3.4 Rekonstruierbarkeit

Die Eigenschaft **Rekonstruierbarkeit** ist gegeben, wenn die Bank den Weg einer bei ihr eingereichten Münze rekonstruieren kann. Diese Eigenschaft ist eng mit der Anonymität verwandt. Bargeld hat diese Eigenschaft nicht. Ein Geldschein trägt bei seiner Erzeugung eine Seriennummer. Wenn die Bundesbank ihn ausgibt, liegt ihr das Ausgabedatum vor. Der Zeitpunkt, wann er wieder in Erscheinung tritt, kann Jahre später sein. Der Weg, den er in der Zwischenzeit genommen hat, kann nicht rekonstruiert werden.

Um die Rekonstruierbarkeit zu ermöglichen, muß bei jedem Vorgang die Identität der beiden Parteien mit an die Münze gekettet werden. Das Problem ist, daß die Größe der Münze immer weiter wächst und somit die Gesamt-Performance des Zahlungssystems stark sinkt.

3.5 Online / Offline

Um eine Möglichkeit zu haben, begangenen Betrug schnell aufzudecken, gibt es verschiedene Möglichleiten. Die naheliegendste ist, das gesamte verwendete Protokoll **online** auszulegen, d.h., jeder Schritt läuft in Echtzeit über ein Netzwerk ab. Wenn der Kunde einen Auftrag gibt, erkundigt sich der Händler sofort bei der Bank des Kunden, ob dieser zahlungsfähig ist. Auch

eine daraufhin initiierte Transaktion wird sofort zwecks Verrechnung an die beteiligten Banken weitergereicht.

Diese Verfahren hat den großen Vorteil, daß nach Abschluß der Transaktion alle Beteiligten die Sicherheit haben, daß alles korrekt abgelaufen ist. Der Händler hat u.U. sogar schon das Geld auf seinem Konto und die Bank weiß genau, daß kein Betrüger am Werk war. Diese Vorteile erkauft man sich aber durch hohe Transaktionskosten. Bei Online-Systemen wird eine existierende und verfügbare Netzinfrastruktur als gegeben und notwendig vorausgesetzt. Daher sind Online-Protokolle nur in Netzwerkumgebungen wie dem Internet oder bei proprietären Anbietern wie T-Online, AOL oder CompuServe denkbar.

Die andere Möglichkeit im Protokoll-Design ist die Annahme, daß Transaktionen **offline** durchgeführt werden können. Um hier Betrug wirkungsvoll zu verhindern, muß der Rechner, der die Transaktionen durchführt, in regelmäßigen Intervallen Kontakt zur Bank aufnehmen. Dies kann via Telefon oder Internet bei Computern bzw. während des Aufladevorgangs der SmartCard am Bankautomaten erreicht werden. Darüber hinaus muß in die Transaktiondaten die Identität der Teilnehmer eincodiert werden. Bei einem elektronischen Scheck ist der Absender (ähnlich dem Papierscheck) ja sowieso bekannt, da der Scheck ja nur eine Referenz auf existierende Konten ist.

Bei digitalem (und anonymem) Bargeld wird es schon schwieriger. Der dort verwendete Ansatz ist grundlegend anders: Wenn eine digitale Münze ihren Besitzer wechselt, wird zwischen beiden Parteien ein Protokoll gefahren, welches teils durch Zufallswerte bestimmt wird, welche *beide* Parteien mit einfließen lassen[1]. Dadurch kann eine Münze nicht durch bloßes Abhören des Protokolls wieder eingespielt werden. Nun wird in die Münze die Identität der ausgebenden Partei eincodiert. Um die Identität wirklich offenzulegen, muß die Münze (mindestens) zweimal ausgegeben werden, d.h., es muß ein Betrugsfall vorliegen.

Offline-Protokolle haben den Vorteil, niedrigere Kommunikationskosten als Online-Protokolle zu haben, was sich durch Zeitverzögerungen beim Aufdecken von Betrügern auswirkt.

3.6 Akzeptanz

Das schönste digitale Geld nützt nichts, wenn niemand daran interessiert ist. Hier müssen sowohl Kunden als auch Händler dazu gebracht werden, die Protokolle zu unterstützen. In [FW96, Seite 23] heißt es:

> Wir definieren Akzeptanzfähigkeit als die Eigenschaft eines Zahlungssystems, überall angenommen zu werden. Das heißt, daß die Annahmebereitschaft des Systems nicht auf die Bank beschränkt bleibt, die es betreibt. ...
> Elektronische Beträge, die eine bestimmte Bank herausgibt, sollten auch von anderen Banken angenommen werden.

[1] siehe bit commitment

Wenn ein Kunde sich dazu entscheidet, das Angebot für digitale Zahlungsmöglichkeiten zu nutzen, muß es auch genügend Händler geben, die diese Nachfrage befriedigen können.

Zu bevorzugen sind hier Verfahren und Protokolle, die offen spezifiziert sind, öffentlich begutachtet werden können, einfach in der Anwendung sind und breit unterstützt werden.

3.7 Bedienbarkeit

Der Umgang mit Bargeld bereitet den Menschen verständnismäßig keine Schwierigkeiten. Die Handlungen (Protokolle) sind plausibel, einfach zu kontrollieren und nachzuvollziehen. Man weiß, was man dem Händler aus der eigenen Geldbörse gibt und sieht die Ware über den Tisch wandern. Wenn man in die Welt der Computer wechselt, ist alles nicht mehr ganz so offensichtlich. Man sieht nicht, welche Daten über die Telefonleitung in den Computer kommen, was im Gehäuse der Smartcard passiert und was auf der Festplatte gelöscht wird.

Um die **Bedienbarkeit** zu gewährleisten, muß für den Benutzer transparent sein, was mit seiner elektronischen Geldbörse geschieht. Ihm muß klar sein, wann er einen Zahlungsvorgang einleitet oder sich sonstige Änderungen ergeben.

Hier sind also die GUI[2]- und Ergonomie-Experten gefragt, intuitiv bedienbare und konsistente Oberflächen zu generieren.

3.8 Kleinhändler

Es sollte keine riesige Infrastruktur notwendig sein, um digitales Geld einzunehmen. Wenn die Anforderungen an Online-Anbindung, Rechnerleistung etc. gering sind, können auch Kleinhändler [3] die neuen Möglichkeiten nutzen.

3.9 Übertragbarkeit

Eng mit der Kleinhändler-Problematik verwoben ist die **Übertragbarkeit**. Darunter ist die Möglichkeit zu verstehen, daß Kunden untereinander Guthaben austauschen können; in diese Transaktionen soll die Bank nicht zwecks Verrechnung involviert sein.

Bei Bargeld ist diese Übertragbarkeit gegeben; der Empfänger kann an Ort und Stelle Echtheit und Menge verifizieren. Bei elektronischen Münzen wird die Umsetzung schon schwieriger. Elektronische Münzen, die beliebig oft innerhalb der Kundengruppe ohne Bankkontakt weitergereicht werden können, werden i.A. durch Kompromisse bei der Systemsicherheit erreicht. Die Verfügungsrechte, für die Geld symbolhaft steht, sollten schnell und einfach übertragbar sein. Siehe hierzu [FW96, Seite 23]:

[2] Graphical User Interface — Grafische Benuzeroberfläche

[3] micromerchants

In der Praxis erscheint die Umsetzung eines Systems, das den Benutzern Guthabentransfers ohne Verrechnung erlaubt und gleichzeitig absolute Sicherheit garantiert, unmöglich.

3.10 Wechselfähigkeit

Die Wechselfähigkeit bezieht sich auf die Wechselstuben der Währung, auf die Gateways zwischen den Protokollen. Wenn ein Kunde beim System A eingetragen ist, der Händler aber nur Währung B akzeptiert, sind hier Währungsgateways notwendig, die (u.U. gegen Gebühr) eine Konvertierung zwischen verschiedenen Formaten ermöglichen.

Neben den verschiedenen Zahlungsverfahren müßen hier natürlich auch Möglichkeiten existieren, verschiedene Landeswährungen zu tauschen, u.U. auch innerhalb eines Protokolls.

3.11 Skalierbarkeit

Wenn ein Zahlungssystem über das Versuchsstadium herauswächst und auf breiter Ebene eingesetzt werden soll, besteht u.U. das Problem, daß im Systementwurf Flaschenhälse existieren, die nur eine begrenzte Nutzerzahl zulassen. Als Beispiel wären hier zentrale Elemente wie Authentifikations-Server denkbar, die nur eine endliche Zahl von Operationen pro Zeitabschnitt zulassen, aber aufgrund der Spezifikation auch nicht parallelisiert werden können, um größere Volumina zu verarbeiten.

Es ist also wünschenswert, nicht schon im Vorfeld Bedingungen festzulegen, die das Wachstum des Systems hinterher einschränken. Dezentrale Strukturen, Parallelisierung und schnelle Algorithmen sind hier die entscheidenden Faktoren.

3.12 Teilbarkeit

Um Beträge beliebiger Höhe darstellen zu können, sind verschiedene Ansätze möglich:

- Münzen können genau den gewünschten Wert darstellen, d.h., eine Münze kann „krumme“ Beträge annehmen.
- Münzen haben feste Werte, u.U. auch unterschiedlich große wie beim normalen Bargeld. Um krumme Beträge zu zahlen, muß man mehrere Münzen liefern (wie z.B. bei DigiCash).
- Das Geld-System gibt die Möglichkeit, Wechselgeld zu erhalten (wie bei Millicent).
- Das Geld ist teilbar.

Die Teilbarkeit drückt aus, daß ein Wert in beliebig viele Stücke geteilt werden kann. Beim Gold war diese Eigenschaft gegeben. Man konnte von einem Gold-Nugget etwas abschaben, hatte eine sehr kleines Stück Gold, und der Nugget war nun eben die Differenz weniger Wert. Das heutige Bargeld hat diese Eigenschaft nicht. Ein Pfennig ist die kleinste Einheit. Nun gibt es verschiedene Anwendungen, wo es wünschenswert wäre, noch kleinere Stückelungen zu erzeugen. [FW96, Seite 24] sagen hierzu:

> Unter Teilbarkeit versteht man die Wahlmöglichkeit, eine Banknote des Wertes *x* in eine beliebige Anzahl kleinerer Banknoten zu tauschen, die auf Beträge jedweder Höhe lauten und deren Gesamtwert wiederum *x* entspricht.

Nun sollte man unter einer „*beliebigen Anzahl*“ trotzdem noch eine endliche Menge sehen. Man stellt einfach die Frage, was die kleinste sinnvolle Einheit ist und definiert das als Basiseinheit. Wenn nun nur noch Basiseinheiten existieren, kommt man zu einem neuen Problem. Wenn die Basiseinheit z.B. $1/100$ Pfennig ist, muß man für die Zahlung eines Betrages von 10 DM 100.000 Basiseinheiten verarbeiten. OKAMOTO und OHTA fordern in [OO92, Oka95] von einem universellen Zahlungssystem Teilbarkeit. Alle Systeme, die Teilbarkeit realisieren, basieren auf binären Bäumen (siehe Kapitel 2).

In [Sto97] differenziert der Autor die Systeme in **Macro-Payments**, **Mikro-Payments** und **Pico-Payments**.

Macropayments: Hierbei geht es um Beträge deutlich größer als 10,- DM. Aufgrund der hohen Auftragswerte können hier durchaus Kreditkartenbasierte Systeme wie z.B. SET Verwendung finden, da die Transaktionskosten in Relation zum Gesamtwert gering ausfallen. In diesen Bereichen lohnt es sich, mit starken und auch rechenaufwendigen Algorithmen jede Transaktion abzuschirmen, da es hier um hohe Beträge geht. Hier ist die Anonymität häufig nicht so wichtig.

Mikropayments: Beträge unter 10,- DM. In diesem Bereich sind elektronische Geldbörsen wie z.B. eCash denkbar, da hier die Kosten für Kreditkartentransaktionen ins Gewicht fallen würden. Hier lohnt sich der Einsatz von starker Kryptografie, um die einzelnen Buchungen abzusichern.

Picopayments: Pico-Payments liegen im Pfennigbereich und darunter. Auch hier gibt es Systeme wie z.B. MilliCent, die vom Wirkungsgrad geeignet sind. Hier lohnt sich der Einsatz von Kontroll-Instanzen, die jede Transaktion überprüfen, **nicht**. Aufgrund des geringen Wertes der Transaktionen bringt es beispielsweise nichts, jede Operation mit RSA im 2048-bit-Bereich abzusichern, weil es viel zu aufwendig ist. Dafür muß hier verhindert werden, daß in großem Stil Falschgeld erzeugt wird.

3.13 Software / Hardware

Die große Frage in diesem Abschnitt ist: Gießt man die Algorithmen in Silizium und packt alles in ein stabiles Gehäuse oder soll es auf allen Plattformen laufen?

Software-Produkte können auf allen Computern eingesetzt werden. Die Entwicklungszyklen sind meist kürzer als bei Hardware-Implementationen, weil ein Compilerdurchlauf schneller zu bewerkstelligen ist als Portierung auf Hardware-Sprache, programmieren der ASIC's[4] und Integration in die Schaltung bzw. im schlimmsten Fall ein Redesign der Schaltung mit der Notwendigkeit, neue Platinen zu erstellen. Darüber hinaus kann man ein schon im Betrieb befindliches System einfacher updaten bzw. patchen als ausgegebene Smartcards. Software kann (relativ) schnell auf neue Plattformen portiert werden und neue Kundenkreise erschließen; man hat nicht die Einschränkungen, die bei Hardware existieren.

Die große Schwierigkeit kommt dann, wenn die Programme in ungesicherten Umgebungen wie Windows, MacOS oder DOS laufen. Die Programme sind ein leichtes Ziel für Viren, trojanische Pferde, etc. Bösartige Angreifer können Programme schreiben, die auf dem Client-Rechner Modifikationen in der Geldbörsen– und Verschlüsselungssoftware vornehmen. Parallel laufende Prozesse (Viren) können Speicherabbilder von der Geldsoftware erzeugen und so evtl. im RAM befindliche Passwörter oder andere geheime Daten in Erfahrung bringen.

Zusätzlich ist bei reinen Software-Produkten das „*reverse-engeneering*“ für erfahrene Assembler-Programmierer ein Kinderspiel, d.h., die genaue Funktionsweise der Programme muß als bekannt angenommen werden. In Software-Systemen darf weder der Anwender noch ein Angreifer Vorteile aus der Fälschung der Daten ziehen können.

Bei **Hardware**, z.B. SmartCards, werden große Anstrengungen unternommen, dem reverse-engeneering und der Fälschung einen Riegel vorzuschieben. Diese Chips werden auch *tamper-proof* genannt. Beim Chip-Design werden Leiterbahnen übereinander gestapelt, um die Funktion der Schaltung zu verschleiern, Passivierungsschichten eingebaut, die die Schaltung bei Sauerstoffkontakt unbrauchbar machen. Optische Sensoren prüfen, ob das Gehäuse geöffnet bzw. aufgefräst wurde. Thermische Sensoren überwachen die Betriebstemperatur, Spannungssensoren schalten die Karte aus, wenn die Versorgungsspannung die spezifizierten Grenzen verläßt und Frequenzzähler sorgen dafür, daß die Schaltung ihren Dienst verweigert, wenn findige Naturen die Schaltung über- oder untertakten wollen.

All diese Bedingungen treten nämlich ein, wenn ein physikalischer Angriff auf die Karte unternommen wird. BIHAM nennt das *differentielle Fehleranalyse*. Diese Angriffsform basiert auf der Annahme, daß Schaltungen, die ansonsten korrekt arbeiten, außerhalb ihrer Spezifikation betrieben zu unberechenbarem Verhalten neigen, d.h. im Klartext, ihre Geheimnisse preisgeben [Wob97, Seiten 132-133].

Hardware-Produkte haben den großen Vorteil, daß sie unempfindlich gegen Viren und weit entfernte Angreifer (Hacker) sind. Ein Hardware-Token kann sich der Anwender einfach in die Tasche stecken und mitnehmen. Wenn beispielsweise kryptografische Schlüssel oder digitale

[4] Application Specific Integrated Cirquit / Anwenderspezifische IC's

Münzen auf diesem Token gespeichert sind, liegen sie dort aufgrund der Portabilität sicherer als auf der heimischen Festplatte. Darüber hinaus sind die Daten auch vor dem Anwender geschützt, d.h., es wird die Annahme gemacht, daß auch der legitime Anwender die auf der Karte enthaltenen Daten nicht einsehen, entschlüsseln oder nach seinem Belieben verändern kann.

3.14 Plattformen

Um eine breite Akzeptanz zu erreichen ist es notwendig, nicht auf ein Computersystem eingeschränkt zu sein. Das Programm sollte auf vielen Plattformen lauffähig sein, d.h., Windows 3.1x/95/NT sollten ebenso vertreten sein wie MacOS oder Unix. Bei reinen Software-Produkten wäre es wünschenswert, Quellcodes zu haben, um Portierungen auf neue Plattformen vornehmen zu können. Andererseits hätte dies den Nachteil, daß trojanische Pferde mit ähnlicher Oberfläche auftauchen könnten, die vom Kunden nicht von der Originalversion unterscheidbar wären; wobei ein nicht vorliegender Quellcode für einen motivierten Angreifer sicherlich kein unlösbares Problem ist.

Das Bereitstellen von Bibliotheken mit einfach aufzurufenden Transaktionsfunktionen wäre hier sicherlich ein Schritt in die richtige Richtung.

4 Kreditkartenzahlungen

4.1 *i*KP — Keyed Payment Protocols

iKP [BGH+95][1] wurde von IBM entwickelt und ist eine Gruppe der drei Protokolle 1KP, 2KP und 3KP. Die Zahl *i* steht für die Anzahl der Parteien, die RSA-Schlüssel besitzen.

Hier wurde die POS[2]-Struktur für das Internet adaptiert. Die involvierten Parteien sind Kunde, Händler und Händlerbank.

4.1.1 Eine Transaktion

1. Der Kunde übergibt dem Händler einen Auftrag.
2. Der Händler sendet dem Kunden die Rechnung zu.
3. Der Kunde schickt dem Händler die Zahlungsinformation.
4. Der Händler leitet die Information zwecks Verifikation als Autorisierungsanfrage an die eigene Bank weiter.
5. Die Händlerbank leitet die Anfrage an die Kundenbank weiter.
6. Die Autorisierungsantwort der Kundenbank wird an den Händler weitergeleitet.
7. Der Händler bestätigt dem Kunden den Auftrag und beginnt die Warenlieferung.

Das das Sicherheitsniveau kennzeichnende *i* besagt nun, wer über RSA-Schlüssel für die ihm zugesandten Informationen verfügt:

Protokoll	1KP	2KP	3KP
Bank	ja	ja	ja
Händler	nein	ja	ja
Kunde	nein	nein	ja

[1] `http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html`
[2] Point Of Sale

4.1.2 Kryptografie

MD5 und RSA mit Schlüssellängen zwischen 768 und 1024 bit bilden die kryptografische Grundlage von *i*KP. Dazu werden den Nachrichten Zeitstempel hinzugefügt und es besteht die Möglichkeit, „*salted messages*“ zu generieren, die ähnlich dem `crypt()`-Befehl in der Unix-Systembibliothek Zufallszahlen in die Nachrichten mit einbaut, um verschiedene Nachrichten gleichen Inhalts unterschiedlich aussehen zu lassen.

4.1.3 Eigenschaften

Letzlich ist das *i*KP-System nur eine Transposition der guten alten Überweisung für das Internet, zwar mit sicheren Methoden, aber trotzdem bleibt es eine Überweisung.

Anonymität existiert nicht, genau wie bei der schriftlichen Überweisung. Alle drei Parteien sind eindeutig bestimmbar, die Zahlungen sind jederzeit **rückverfolgbar**. Bezüglich der Zahlungsgröße ist zu sagen, daß es sich hier nur um einen Transportmechanismus zum Transfer der Zahlungsinformation handelt, wie letztlich bei allen Konten– bzw. Kreditkartenbasierten Systemen. Es ist also eher im **Macropayment**-Bereich angesiedelt. **Teilbarkeit, Übertragbarkeit** und **Wechselfähigkeit** sind bei keinem der kontenbasierten Systeme gegeben, also auch nicht bei *i*KP. *i*KP ist ein **Online-System**, da die Zahlung online mit den Banken autorisiert wird.

4.1.4 Perspektiven

*i*KP wurde mit dem Focus entwickelt, bestehende Infrastrukturen der Kreditwirtschaft zu nutzen und nicht alles neu aufzubauen. *i*KP ist für den Guthabentransfer gedacht, und somit ist die untere Schranke für **Transaktionskosten** mit den Kosten für den Guthabentransfer selbst angegeben.

[FW96, Seite 43] meinen zum Thema *i*KP:

> Als weiteren Vorteil führen die Autoren an, daß IBM für *i*KP keine Eigentumsrechte beansprucht, die Protokolle also allgemein zugänglich und nutzbar sind. Doch abgesehen von speziell entwickelter, auf diesem Protokoll basierender Software ist es schwer vorstellbar, was an *i*KP überhaupt als Eigentumsrecht betrachtet und evtl. geschützt werden sollte. *i*KP kombiniert nämlich nur den Einsatz diverser, bereits vorhandener kryptografischer Techniken.
>
> Durch seinen Aufbau auf dem Überweisungssystem erbt das *i*KP-Protokoll dessen Schwächen, insbesondere die uneingeschränkte Rückverfolgbarkeit von Zahlungen. In ihren Veröffentlichungen behaupten sogar die Autoren von *i*KP selbst, daß die aktuell kryptografisch am weitesten fortgeschrittenen Zahlungssysteme viel Wert auf Nichtverfolgbarkeit und auf Anonymität legen. Damit liegt der Schluß nahe, daß sie ihr eigenes *i*KP-System für eines der weniger weit entwickelten Systeme halten.

4.1.5 Literatur

[BGH+95], [FW96, Seiten 41–43], [GZ96] und `http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html` .

4.2 STT — Secure Transaction Technology

Im September 1995 stellten VISA und Microsoft *STT*, die „sichere Übertragungstechnik“ vor. STT beschreibt verschiedene Protokolle, von der eigentlichen Zahlung bis zu aufwendigen Zertifizierungshierarchien. Integraler Bestandteil von STT ist die *Gesellschaft*, eine zentrale Einrichtung, die die gesamte Zertifizierung von RSA-Schlüsseln übernimmt. Jeder Teilnehmer hat bei STT zwei asymmetrische Schlüsselpaare, das eine zum Entschlüsseln empfangener Nachrichten, das andere zum Signieren zu versendender Nachrichten.

Ähnlich wie bei PEM wird eine Hierarchie für STT-Zertifikate aufgebaut, wobei unter der Gesellschaft (Association) die Banken (Issuer und Aquirer) angesiedelt sind. Die Banken wiederum zertifizieren die Händler und Kunden. Diese Struktur entspricht dem wirklichen Modell, daß jeder Kunde von der Bank zertifiziert wird, von der ihm die Kreditkarte ausgegeben wurde und jeder Händler von der Bank zertifiziert wird, bei der er sein Konto hat.

Mit dieser Zertifizierungsstruktur wird die Grundlage geschaffen, daß Nachrichten vertraulich zwischen beliebigen Partnern ausgetauscht werden können und daß die Authenzität der Nachrichten gewährleistet ist.

Hier gelten die gleichen Eigenschaften wie bei *i*KP, d.H., uneingeschränkte Rückverfolbarkeit, keine Anonymität, hohe Transaktionskosten, Macropayments, keine Teilbarkeit, keine Übertragbarkeit und auch keine Wechselfähigkeit.

4.2.1 Eine Transaktion

Bei jeder Transaktion müssen die Zertifikate, bei STT auch Beglaubigungen oder auch *credentials* genannt, zusammen mit der verschlüsselten Nachricht übertragen werden. Dadurch entstehen relativ große Datenpakete, die übertragen werden müssen.

Bei der Transaktion werden nun die Zahlungsdaten mittels Hashfunktion und eigenem Signaturschlüssel signiert, dann werden die Zahlungsdaten und die Signatur mit dem Chiffrierschlüssel des Empfängers verschlüsselt und das entstandene Paket zusammen mit den Zertifikaten an den Empfänger geschickt.

STT führt die von SET übernommenen „dualen Signaturen“ ein, bei denen die Überweisungsdaten vom Kunden direkt mit dem Schlüssel der Kundenbank verschlüsselt werden, damit der Händler die Transaktionsdaten selbst nicht einsehen kann. Auf die duale Signatur wird bei SET noch ausführlich eingegangen.

4.2.2 Perspektive

STT wird nicht mehr durch VISA unterstützt, sondern wurde durch SET ersetzt.

4.3 SEPP — Secure Electronic Payment Protocol

Das *SEPP*, das „sichere elektronische Zahlungsprotokoll“, wurde im November 1995 von Mastercard, IBM, Netscape, GTE und CyberCash veröffentlicht.

Als erster großer Unterschied zu STT ist anzumerken, daß die Zertifizierungshierarchie viel flacher ist, nämlich nur noch eine einzige Ebene hat. Es gibt **das** Trust-Center. Hiermit wird auch schon das erste Problem aufgezeigt. Durch diese Zentralisierung kann es zu schlechter Performance führen, daß nur noch eine Institution die Zertifikate ausstellt. Gleichzeitig ist das aber auch der großen Vorteil von SEPP. Während bei STT jede Kunden– und Händlerbank im Netz erreichbar sein muß, um Zertifikate auszustellen, reicht es bei SEPP, wenn die betreffende Bank dem Trustcenter die Genehmigung zur Zertifizierung eines Teilnehmers gibt.

Der zweite Unterschied ist die Tatsache, daß nicht mehr alles im Internet ablaufen muß, sondern auch bestehende Infrastrukturen wie das bankeninterne SWIFT-Netz[3] mitbenutzt werden. Beispielsweise muß die Kundenbank nicht über eine Internetanbindung verfügen. Die Kundenbank wird nicht benötigt, um beantragte Zertifizierungen von Kunden zu genehmigen und um Autorisierungsanfragen der Händlerbank zu bestätigen. Beides kann auch über interne Netze abgewickelt werden.

Vom Protokoll her ist SEPP wenig interessant. Nach der Bestellung durch den Kunden, welche die Identität des Kunden und eine vom Kunden vergebene Bestellungsnummer enthält, schickt der Händler eine Rechnung an den Kunden. Auf der Rechnung ist die Bestellnummer des Kunden, eine Händlerbestellnummer und die Bestelldaten wie Preise etc. vermerkt. Der Kunde übermittelt dem Händler jetzt eine Nachricht, in der die beiden Bestellidentifikationen und die Zahlungsanweisung für die Kundenbank enthalten sind. Diese Nachricht signiert der Kunde. Nachdem die Händlerbank die Nachricht durch den Händler weitergereicht bekommen hat, fragt sie bei der Kundenbank um Autorisierung nach und teilt dem Händler das Ergebnis der Anfrage mit. Zum Abschluß muß nur noch der Händler die Lieferung initiieren.

4.3.1 Perspektive

SEPP wurde 3 Monate nach Veröffentlichung durch SET abgelöst.

[3] S.W.I.F.T.: Society for Worldwide Inter-Bank Financial Telecommunication

4.4 CyberCash

Die Firma CyberCash Inc. wurde 1994 von WILLIAM MELTON und DANIEL LYNCH in Reston/VA/USA gegründet.

CyberCash (`http://www.cybercash.com/`) ist ein Kreditkartenbasiertes System, bei dem der Kunde einen Account bei der CyberCash GmbH eröffnet. CyberCash verwendet kryptografische Methoden zur Absicherung der Transaktionen und arbeitet teilanonym, d.h, der Händler erhält keine Information über den Kunden. Das System ist auf das Internet ausgerichtet und wird schon eingesetzt, unter anderem von der Sachsen LB (`http://www.sachsenlb.de/`) und der Dresdner Bank (`http://www.dresdner-bank.de/`).

Seit der Einigung der goßen Kreditkartenunternemen auf den künftigen Standard SET arbeitet CyberCash Inc. intensiv an der Normung von SET mit und integriert SET in seine eigenen Produkte. Aufgrund der starken Ähnlichkeiten zwischen beiden Verfahren wird CyberCash als einer der Vorläufer von SET angesehen. Generell kann zu CyberCash gesagt werden, daß es sich eher um den sicheren Transport von Verrechnungsinformationen denn um ein eigenständiges Zahlungssystem handelt. Auch die Sachsen LB und die Dresdner Bank wollen später auf SET umsteigen.

4.4.1 Der Kunde

Registrierung

Wenn ein Kunde CyberCash nutzen will, muß er sich bei seiner Bank registrieren lassen. Dabei authentisiert er sich mit seinem Personalausweis und seiner Kreditkarte oder ec-Karte. Von der Bank erhält er dann seine Wallet-Software. Der Begriff Wallet ist in diesem Zusammenhang ein wenig widersprüchlich, denn es handelt sich hier weniger um eine Geldbörse zum Speichern digitaler Münzen sondern um die Software zum Verarbeiten der Kreditkarteninformationen. Wenn im folgenden von Kreditkarteninformationen die Rede ist, kann es sich um Kontoverbindungen, ec-Kartendaten oder auch Kreditkartendaten handeln. CyberCash ist nicht auf Kreditkarten festgelegt, sondern kann auch über Giro-Konten laufen. Diese Begriffe sollen hier synonym verwandt werden.

Beim Start der Wallet-Software werden die Wallet-ID, die Adresse des Kunden, seine e-mail-Adresse, eine sog. Verifikations-ID, ein Paßwort und die Kreditkarteninformationen festgelegt. Das Paßwort dient der Verschlüsselung der Stammdaten und der Transaktionslogbuches auf der Festplatte.

Die Verifikations-ID wird bei der Registrierung verschlüsselt an CyberCash übertragen. Sie dient als Kennung, um im Notfall seinen Account bei CyberCash sperren zu können. Nach einem

Festplatten-Crash, bei Kompromittierung der Daten oder wenn das Passwort vergessen wurde, kann der Kunde seinen Account telefonisch (Out-Band-Signalling) sperren lassen. Nach dieser Sperrung ist das Wallet nicht mehr einsatzfähig und der Kunde muß zusammen mit der Bank klären, was passiert ist.

Protokollierung

Das eben erwähnte Transaktionslogbuch wird auf dem Kunden-PC angelegt. Hier werden alle Transaktionen festgehalten, damit der Kunde alles nachvollziehen kann. Darüber hinaus wird auch bei Händler und auf dem Transaktions-Gateway protokolliert.

4.4.2 Kryptografie

CyberCash hat von der amerikanischen Regierung eine Ausfuhrerlaubnis für ihr System bekommen. Der Hauptgrund dafür ist die Tatsache, daß es nicht möglich ist, mit CyberCash geheime Nachrichten im großen Stil zu verschlüsseln, sondern daß das Wallet nur Finanzdaten absichert.

Bei der symmetrischen Chiffre setzt CyberCash auf den DES (Data Encryption Standard). Für CyberCash war es wohl eine Abwägung zwischen Sicherheit und Absatz-Chancen. Da die USA Exportbeschränkungen für starke Kryptografie haben, verwenden viele Firmen schwache Chiffren, um ihre Systeme exportieren zu können. Der DES ist mit seinen 56 bit effektiver Schlüsselbreite zwar heutzutage in Regionen, die für Geheimdienste und Verbrechersyndikate gleichermaßen per Brute-Force brechbar sind. Da der verwendetete Schlüssel aber nur ein Session-Key für eine einzige Transaktion ist, welche darüber hinaus nur ca. 20 Sekunden dauert, der Schlüssel danach also nicht mehr verwendet werden kann, ist damit zu rechnen, daß hier nicht in das System eingebrochen wird.

Die asymmetrische Seite wird mit 1024 bit großen RSA-Schlüsseln erledigt. Als Hash-Funktion wird der MD5 verwendet. CyberCash verwendet sowohl für die Clearing-Stelle (Den CyberCash-Server) als auch für Händler und Kunden RSA-Schlüsselpaare. CyberCash agiert darüber hinaus als Zertifizierungsinstanz für diese Schlüssel. Dadurch wird der Kommunikationsoverhead kleiner, weil nicht bei jeder Transaktion Schlüssel und Unmengen diverser Zertifikate übertragen werden müssen. Darüber hinaus wird das System nach außen hin angeschottet und proprietär. Je nach dem, ob der Markt und die Kunden ein offenes System haben möchten, liegt hier u.U. ein Wettbewerbsnachteil für CyberCash, denn der Aufwand, in das bestehende System verschiedene Clearing-Instanzen einzubauen, kann schnell zu Inkonsistenzen führen.

4.4.3 Das Zahlungsprotokoll

1. Der Kunde und der Händler werden sich handelseinig bezüglich eines Produktes. Der Händler sendet dem Kunden die Daten bezüglich der Preises und der Abrechnung.

2. Die Wallet-Software präsentiert dem Kunden die Transaktionsdaten und Zahlungsoptio-

nen. Die Zahlungsinformation, bestehend aus Preis und Kreditkarteninformation, werden mit dem öffentlichen Schlüssel des CyberCash-Servers verschlüsselt und mit dem privaten Schlüssel des Kunden signiert. Danach wird die verschlüsselte und signierte Information zum Händler geschickt.

3. Der Händler fügt an die erhaltenen Daten seine eigene Identifikation und den Preis an. Das entstandene Paket wird vom Händler signiert und an den CyberCash-Server geschickt.

4. Der CyberCash-Server entpackt die Informationen, prüft die Signaturen von Kunde und Händler und verifiziert, daß die beiden Preise identisch sind. Dann wird die Information über das Banken-Netz an die Bank des Händlers weitergeleitet.

5. Bei Erfolg wird das Konto des Kunden belastet und der CyberCash-Server generiert zwei Bestätigungen, eine für den Händler und die andere für den Kunden. Beide werden mit dem jeweiligen öffentlichen Schlüssel verschlüsselt und an den Händler geschickt.

6. Diesem steht es nun frei, die Meldung für den Kunden an diesen weiterzuleiten. In jedem Fall leitet das Händlersystem jetzt den Warentransport bzw. die Möglichkeit zum Download der Ware ein.

4.4.4 Kleinhändler

CyberCash bietet auch Kleinhändlern die Möglichkeit, Kreditkartenzahlungen anzunehmen. *CyberCashs angekündigte Scheckzahlung soll auch Privatpersonen ermöglichen, Zahlungen anzunehmen.* [SFE97, Seite 47]

4.4.5 Anonymität

Das System bietet nur eine schwache Teilanonymität, denn bei Notwendigkeit teilt CyberCash dem Händler die Identität des Kunden mit. Bei einer Standard-Abwicklung einer Transaktion erfährt der Händler nur die e-mail-Adresse und die IP-Adresse des Kunden bzw. die IP eines evtl. zwischengeschalteten Application-Proxy.

Die CyberCash als Clearing-Instanz erfährt genau wie die beteiligten Banken sowohl die Identität des Kunden als auch die des Händlers und den Preis der Ware. Bezüglich Anonymität sind hier also wie bei allen auf Kreditkarten basierenden Systemen keine Wunder zu erwarten.

4.4.6 Teilbarkeit

Wie bei allen Kreditkarten basierten Systemen gibt es die Eigenschaft Teilbarkeit hier nicht.

4.4.7 Transaktionskosten

Wie bei allen Kreditkarten basierten sind die Transaktionskosten relativ hoch, denn die Kosten für die Kreditkarte bilden die untere Schranke, wodurch sich das System nur für Macropayments eignet.

4.4.8 Skalierbarkeit

Der Händler reicht seine Transaktionsdaten an CyberCash weiter, von wo aus die Daten auf den jeweiligen Server-Core-Processor wandern. Hier hat das System die Möglichkeit, eine Parallelisierung vorzunehmen und somit große Datenmengen verarbeiten zu können. Das Nadelör ist der einzelne Gateway-Server, der allerdings von Banken lizensiert werden kann.

4.4.9 Bedienbarkeit

Die Bedienbarkeit ist Sache der jeweiligen Bank, welche das Wallet an den Kunden ausliefert. Hier kann es durchaus Unterschiede zwischen verschiedenen Realisierungen geben.

4.4.10 Software/Hardware

Das System ist komplett in Software realisiert. Die Angriffsmöglichkeiten durch Viren sind minimal, da die Daten durch das Kundenpaßwort gechützt verschlüsselt auf der Platte liegen. Während die Wallet-Software aktiv ist, kann ein Virus unter Windows 3.1x oder Windows 95 evtl. Fragmente der geheimen Daten aus dem Speicher kopieren, wenn diese nicht sofort nach Gebrauch vom Wallet zurückverschlüsselt werden. Es bleibt zu hoffen, daß unter Windows NT hier die Speicherschutzmechanismen ansetzen. Problematisch ist die Ersetzung der Wallet-Software durch ein trojanisches Pferd, welches dem Kunden bei der Registrier-Prozedur die Anmeldedaten abnötigt.

4.4.11 Architektur des Gateway

Auf dem CyberCash-Gateway-Server laufen verschiedene Dienste. Eine Standard-Firewall (Advanced Routers und Firewalls) filtert nach IP-Adressen (intern/extern) und Zielport (80/HTTP). Auf der Firewall erfolgt ein vollständiges Logging (sowohl *deny* als auch *access* werden protokolliert). Die Applikationsebene wird auf dem *Preprocessing Computer* abgewickelt. Hier werden erste Konsistenz-Checks gemacht. Dann werden die Daten für den zuständigen *Server Core Processor* aufbereitet.

Auf dem Server Core Processor existieren verschiedene Bereiche für Kunden und Händler, die zur Authentisierung von Wallet-ID, der Einbindung eines neuen Zahlungsmittels in eine Wallet oder die Belastung einer Wallet zuständig sind. Darüber hinaus sind hier verschiedene Daten-

banken angebunden. Nachdem alle Schritte abgearbeitet sind, wird der Prozess an den *Back-End-Processor* weitergeleitet.

Der Back-End-Processor leitet die Anfragen in die jeweiligen Banken-Netzwerke weiter. Er bildet das Gateway für Clearing-Operationen zu den Banken.

4.4.12 Plattformen

Der Kunde

Auf dem Rechner des Kunden kommt ein Web-Browser und die Wallet-Software zum Einsatz. Es existieren Implementaionen für Windows 3.1x, Windows 95 und Windows NT. Versionen für MacOS und UNIX sind geplant.

Der Händler

Auf dem Händler-Rechner läuft das CashRegister, das *Secure Merchant Payment System* . Zur Zeit sind Versionen für Windows NT und Solaris v2.5 verfügbar; es ist geplant, 1998 Versionen für diverse UNIX-Varianten (AIX 4.2, HP-UX 10.0, Red-Hat-Linux 4.2 und SCO 5.4.1) zu veröffentlichen.

4.4.13 Perspektive

Zur Zeit ist CyberCash ein sehr konkurrenzfähiges System, das sichere Transaktionen via Internet ermöglicht. Dennoch ist absehbar, daß mit einer breiten Einführung von SET CyberCash unnötig wird, weil CyberCash nur kompatibel mit sich selbst ist und SET ein offener Standard ist, der breite Unterstützung findet.

4.4.14 Literatur

Literatur zu CyberCash findet sich unter [Sto97, Seite 66–70], unter [Way97, Seite 141–154], und im Internet bei `http://www.cybercash.com/`, `http://www.sachsenlb.de/` und `http://www.dresdner-bank.de/`.

4.5 SET — Secure Electronic Transaction

SET ist ein Übertragungsprotokoll zur sicheren Durchführung von Kreditkartentransaktionen via Internet. Visa (`http://www.visa.com`) und Mastercard (`http://www.mastercard.com`) waren die treibenden Kräfte bei der Normung und Standardtisierung.

SET ist aus verschiedenen Protokollen zusammengewachsen [RK97]: Zuerst hatte CyberCash sein System für Kreditkartenzahlungen auf dem Markt. Die Großen der Branche wollten nach-

ziehen und IBM veröffentlichte *i*KP (Keyed Payment Scheme) zur Bezahlung mit Kreditkarten im Netz. Hewlett-Packard brachte HPPS (HP Payment Scheme) auf den Markt. IBM versuchte, durch die Gründung einer IETF-WG (Internet Engeneering Task Force – Working Group) sein *i*KP zum Standard zu erheben, doch da die großen Kreditkartenfirmen nicht nachzogen, blieb der Erfolg aus. Am 27. Oktober 1995 veröffentlichten Visa und Microsoft eine Spezifikation namens STT (Secure Transaction Technology). Kurze Zeit später brachten MasterCard, IBM, Netscape und CyberCash das SEPP (Secure Electronic Payment Protocol) ins Licht derÖffentlichkeit. SEPP war eine Mischung aus *i*KP, CyberCash und Secure Courier, einer Spezifikation von Netscape. Da die Banken nicht bereit waren, zwei konkurrierende Standards zu unterstützen, einigten sich die beiden Gruppen Anfang 1996 auf SET als gemeinsamen Standard, der auch von American Express unterstützt wird.

Das vielversprechende an SET ist die jedermann zugängliche Spezifikation[4]. Die Detaills von SET sind jedermann zugänglich, das Ziel von SET war (gezwungenermaßen) nicht die Schaffung eines proprietären Produktes, sondern ein von allen akzeptierter Standard. Die Kreditkartenfirmen versprechen sich von SET einen Standard, den alle Software-Hersteller sicher in ihre Produkte einbauen können, um den Handel über Kreditkarten weiter anzuheizen.

Auch SET sollte weniger als Zahlungssystem verstanden werden, sondern als Sicherheits-Layer, welches eine vertrauliche und authentische Übertragung von Kreditkarteninformation erlaubt. Tokenbasierte Zahlungen sind mit SET nicht möglich, es geht um die Ausprägung einer Struktur für die einfache und sichere Nutzung von Kreditkarten. Problematisch für den Kunden ist die notwendige Präsenz der beiden beteiligten Banken im Internet, wenn die Banken selbst Nutzer-Zertifikate ausstellen wollen. Der andere Weg, der u.U. von den Kreditkartenorganisationen auch gewollt ist, fordert von Banken, die nicht aktiv im Internet agieren wollen, das outsourcen ihrer Zertifizierungsinstanzen zu anderen Dienstleistern. Hier bieten sich wieder die Kreditkartenorganisationen an, welche dadurch ihre Position in der Infrastruktur stärken und ihre Märkte absichern.

> [FW96]: Es wäre allerdings wirklich einfacher, wenn eine zentrale Autorisierungsstelle sämtliche Zertifikate für das System ausgäbe. [...] Diesen Fall könnte man allerdings als Verlagerung des »Machtgleichgewichtes« im Kreditkartensystem in Richtung Kreditkartengesellschaften und zu Lasten des Einflusses der Banken interpretieren.

SET bietet Vertraulichkeit der Daten zwischen allen Parteien sowie Authenzität und Integrität der Daten. Bei der Vertraulichkeit geht SET den Weg der *dualen Signatur*, was einfach ausgedrückt bedeutet: Jeder bekommt nur die Information, die er auch wirklich benötigt! Der Händler benötigt die Information, welche Produkte der Kunde bestellt. Die Kontoverbindung oder Kreditinformation des Kunden ist für den Händler nur soweit notwendig, daß er Sicherheit braucht, ob der Kunde zahlt. Die Bank hingegen benötigt die Daten für den Geldtransfer, d.h., Sender,

[4] `http://www.visa.com/cgi-bin/vee/nt/ecomm/set/downloads.shtml`

Empfänger und Menge. Was der Kunde beim Händler bestellt hat, geht die Bank nichts an; das einzige, was von der Bestellung notwendig ist, ist der Betrag des zu überweisenden Geldes.

Hier bietet SET einen Weg, die Information so zu spalten, daß Händler und Bank den jeweils für sie selbst bestimmten Anteil entschlüsseln können und den unbekannten Anteil mit dem „digitalen Fingerabdruck“ auf Intergrität hin überprüfen können. Bestell– und Zahlungsdaten sind also sehr eng miteinander verknüpft. Diese Eigenschaft wird auch *Teilanonymität* genannt.

4.5.1 Das Zahlungsprotokoll

Um eine SET-Transaktion zu verstehen, werden die finanziellen Schritte hier kurz in Prosa formuliert [Way97]:

Kunde und Händler bestimmen den Preis der Ware. Danach bittet der Händler die Händlerbank um Autorisierung der Transaktion. Die Händlerbank bittet nun die Kundenbank, einen dem Preis entsprechenden Betrag beiseite zu legen (ohne ihn schon zu überweisen), um die Transaktion zu decken. Wenn die Deckung von der Kundenbank bestätigt wird, bekommt der Händler über die Händlerbank die Bestätigung. Zu einem späteren Zeitpunkt fordert der Händler die Zahlung offiziell an. Zwischen der Bitte um Festlegung und der eigentlichen Forderung können mehrere Tage oder sogar Monate liegen, da u.U. für Hotels oder gemietete Autos gezahlt werden muß, deren endgültiger Preis sich nicht zum Zeitpunkt der ersten Anfrage festlegen läßt. Wärend dieser Zeit ist der festgelegte Betrag blockiert und steht dem Kunden trotz Erscheinen auf dem Kontostand nicht zur Verfügung.

Wenn die endgültige Geldforderung bei der Händlerbank eingeht, reicht diese sie an die Kundenbank weiter. Die entgültige Geldforderung kann u.U. größer oder auch kleiner als der blockierte Betrag sein. Dann taucht der Betrag auf der Kundenrechnung auf, die Kundenbank schickt den Betrag an die Händlerbank und nach einer (von Bank zu Bank variierenden) Zeitspanne wird der Betrag dem Händler gutgeschrieben.

SET wurde definiert, um die verschiedensten Ansprüche und Forderungen zu erfüllen. Um all das zu erreichen, wird eine Mischung aus digitalen Signaturen und Verschlüsselung verwendet.

1. Bevor die eigentlichen SET-Protokollschritte beginnen, wählt der Kunde beim Händler die zu kaufenden Waren aus. Die geschieht i.A. via Web-Browser. Nachdem dem Kunden die Komplette Bestell-Liste präsentiert wurde, wählt der Kunde die Zahlungsart.

2. Initiate Request:

   Nachdem das Shopping beendet ist, sendet die SET-Software des Kunden die erste Anfrage (`PInitReq`) um Zertifikate an den Händler. Diese Anfrage beinhaltet die zu verwendende Kreditkarte und die Hashwerte (auch *thumbprint* genannt) der beim Kunden lokal vorliegenden Zertifikate, damit der Händler zwecks Aktualisierung evtl. aktuellere Zertifikate schicken kann.

Zusätzlich wird noch ein Zufallswert (`Chall_C`)[5] mitgeschickt, um Replay-Angriffe zu unterbinden, die Nachricht also einzigartig zu machen.

3. Initiate Response:

   Der Händler antwortet mit `PInitRes`, der ZahlungsInitialisierungsAntwort. Der Händler muß zuerst prüfen, ob die Hashwerte der beim Kunden vorliegenden Zertifikate noch‚up to date' sind. Falls einige der Zertifikate nicht auf dem neuesten Stand sind, werden die aktuellen Zertifikate mit eingebunden. Zu diesen Zertifikaten zählen die der CA's selbst, das der Händlerbank und des Händlers[6].

   Bevor der Händler die gesamte Nachricht mit dem geheimen Signaturschlüssel des Händlers signiert, wird der vom Kunden erzeugte Zufallswert (`Chall_C`) mit in die Nachricht eingebunden, um dem Kunden die Gewähr zu geben, daß die Nachricht aktuell ist.

4. Kunde Purchase Request:

   Nach Erhalt von `PInitRes` überprüft der Kunde evtl. vom Händler mitgesendete Zertifikate auf ihre Gültigkeit und fügt sie seinem „Schlüsselbund"[7] hinzu. Ebenfalls wird natürlich die Signatur von `PInitRes` und `Chall_C` überprüft.

   Jetzt erzeugt der Kunde die eigentliche Bestellnachricht `PReq`. Dazu werden zwei verschiedene Subnachrichten erzeugt: die „Order Information" (OI) und die „Payment Instruction" (PI). Es ist zu beachten, daß die Bestellinformation OI **nicht** die Liste der bestellten Waren mit Artikel und Anzahl oder Einzelpreisen enthält. Diese Information wird zwischen Kunde und Händler vor der ersten SET-Nachricht vereinbart („out-of-band-signalling"). Die Hauptinhalt von OI ist ein Hash über die gemeinsam vereinbarten Bestelldaten. Dieser Hash-Wert verhindert, daß hinterher eine der beiden Parteien von der vereinbarten Warenliste zurücktreten kann. Wenn der Händler die OI erhält, berechnet er seinerseits den Hashwert der bei ihm vorliegenden Warenliste. Die beiden Hash-Werte müssen natürlich identisch sein. Wenn der Händler später die Kundendaten an die Händlerbank weiterreicht, fügt er seinen Hash-Wert der Bestellliste mit an, um der Bank die Möglichkeit zu bieten, zu prüfen, ob Kunde und Händler handelseinig sind. Die PI enthält auch den Hashwert.

   Neben dem Hash-Wert enthält die OI mehrere Zufallszahlen, die im weiteren Protokoll-Verlauf als „Salt" gegen Wiedereinspielung von Nachrichten durch Angreifer[8] dienen. Wenn die gesamte Transaktion mit der Nachricht `PInitReq` begann, wird auch `Chall_C` eingebunden. Wenn die erste Nachricht die gerade zu erzeugende `PReq` ist, muß erst ein `Chall_C`-Wert erzeugt werden.

   In das PI-Paket werden Informationen für die Händlerbank eingebunden. Dazu zählen:

[5] Die *Chall*enge; siehe dazu 2.9, Seite 16

[6] Die Aufgabe der Zertifikat-Aktualisierung wird dem Händler übertragen, da davon auszugehen ist, daß er häufiger mit den Banken kommuniziert.

[7] seine Zertifikatsdatenbank

[8] Replay-Attack

- die ID-Nummer des Händlers
- der zu zahlende Betrag
- eine verschleierte Version der Kontonummer/Kreditkartennummer des Kunden
- und der Hash über die Bestell-Liste
- evtl. ein Session-Key, falls später Nachrichten direkt zwischen Kunde und Händlerbank verschlüsselt ausgetauscht werden sollen.

Jetzt wird die „duale Signatur“ von OI und PI erzeugt. Dazu werden die beiden Hash-Werte $h_{OI} = Hash(\text{OI})$ und $h_{PI} = Hash(\text{PI})$ berechnet. Die beiden Hashwerte werden einfach aneinander gehängt und der Gesamthash $h = Hash(h_{OI} \mid\mid h_{PI})$ berechnet. h wird vom Kunden digital signiert und zusammen mit h_{OI} und h_{PI} in `PReq` eingetragen.

Jetzt wird PI symmetrisch mit einem zufällig gewählten DES-Schlüssel $K1$ verschlüsselt ($DES_{K1}\{PI\}$). $K1$ wird zusammen mit der Kreditkartennummer des Kunden mit dem öffentlichen Schlüssel der Händlerbank verschlüsselt.

`PReq` enthält somit folgendes:

- das Kundenzertifikat
- die Bestellinformation OI
  - $Hash\{\text{echte Bestellung}\}$
- $DES_{K1}\{\text{PI}\}$ mit PI =
  - die ID-Nummer des Händlers
  - der zu zahlende Betrag
  - eine verschleierte Version der Kontonummer/Kreditkartennummer des Kunden
  - und der Hash über die Bestell-Liste
  - evtl. ein Session-Key, falls später Nachrichten direkt zwischen Kunde und Händlerbank verschlüsselt ausgetauscht werden sollen.
- $RSA_{PK_{\text{Händlerbank}}}\{K1, \text{Kreditkartennr.}\}$
- die duale Signatur des Kunden: $RSA_{SK_{\text{Kunde}}}\{h_{OI} \mid\mid h_{PI}\}$
- sowie h_{OI} und h_{PI}.

5. Purchase Response:

   `PRes` ist die Antwort des Händlers an den Kunden. Bevor `PRes` gesendet wird, muß der Händler verschiedene Überprüfungen mit `PReq` durchführen. Zuerst wird das Kundenzertifikat auf Gültigkeit geprüft. Der Wert `Chall_C` muß mit dem aus `PInitReq` identisch sein (falls `PInitReq`/`PInitRes` beteiligt war). Das vom Händler berechnete h_{OI} muß identisch mit dem (dual) digital signierten h_{OI} des Kunden. Damit prüft der Händler, daß der Kunde nicht Parameter in der Warenliste geändert hat.

PRes muß mindestens eine Transaktions-ID, Chall_C und einen Status-Code bezüglich des Transaktionsstatus beinhalten. Die Nachricht wird vom Händler digital signiert und zusammen mit dem Händlerzertifikat an den Kunden geschickt.

Die SET-Software des Kunden prüft nun die Signatur und Chall_C, aktualisiert die interne Datenbank mit der Transaktions-ID und das Programm zeigt dem Kunden eine Meldung bezüglich des Transaktionsstatus an.

Die Nachricht PRes an den Kunden ist nicht vorgeschrieben.

6. Händler Authorization Request:

   Der Händler fordert mit AuthReq Authorisierung der Zahlung durch die Händlerbank. Der Hashwert von AuthReq wird vom Händler digital signiert. Dann wird AuthReq symmetrisch mit einem zufällig gewählten Session-Key $K2$ verschlüsselt. $K2$ wird mit dem öffentlichen Schlüssel der Händlerbank verschlüsselt.

   Zusätzlich zum signierten und verschlüsselten AuthReq werden noch die signierten und verschlüsslten Payment-Instructions PI des Kunden aus der Nachricht PReq mitgesendet.

   Damit die Händlerbank die Signaturen prüfen kann, werden die 3 Zertifikate[9] von Kunde und Händler mit übersendet.

7. Wenn die Händlerbank AuthReq empfängt, entschlüsselt sie den symmetrischen Schlüssel $K2$. Mit $K2$ wird AuthReq entschlüsselt. Dann wird das Händlerzertifikat auf Gültigkeit geprüft und die Händlersignatur unter AuthReq verifiziert.

   Die Bank entschlüsselt den digitalen Briefumschlag des Kunden, in dem $K1$ und die Account-Information verborgen sind. Mit $K1$ wird PI entschlüsselt. Dann prüft sie das Kundenzertifikat auf Gültigkeit und verifiziert die duale Kundensignatur, um Integrität von OI und PI zu sichern.

   Danach wird geprüft, ob die in den PI des Kunden und in AuthReq enthaltenen Transaktions-Identifier identisch sind, um zu sichern, daß PI und AuthReq zusammengehören. AuthReq wird umkonvertiert und über sichere Bankennetze an die Kundenbank gesendet.

8. Händlerbank Authorization Response:

   Nachdem die Kundenbank die Anfrage beantwortet hat, erzeugt die Händlerbank eine Antwort AuthRes für den Händler. AuthRes enthält die Antwort der Händlerbank und das Signaturzertifikat der Händlerbank. Die Händlerbank signiert AuthRes.

   Zusätzlich wird CapReq mit eingebunden. Hierbei handelt es sich um ein Token, mit dem der Händler bei der Kundenbank den Betrag einfrieren kann, falls er dieÜberweisung der Beträge nicht sofort mit der Authorisierung in die Wege leitet. Mit Hilfe von CapReq werden bei der Händlerbank die Authorisation und der Einfriervorgang als zusammengehörig markiert.

[9] Signaturschlüssel Kunde, Signaturschlüssel Händler, Key-Exchange-Schlüssel Händler

`AuthRes` und `CapReq` werden symmetrisch verschlüsselt und die beiden Schlüssel $K3$ und $K4$ mit dem RSA-Schlüssel des Händlers verschlüsselt. Zusätzlich werden noch die Händlerbank-Zertifikate mitgeschickt.

9. Der Händler verifiziert die Zertifikate. Er entschlüsselt den symmetrischen Schlüssel $K3$, um `AuthRes` entschlüsseln zu können. Er überprüft die Signatur der Händlerbank unter `AuthRes` und kann nun abhängig von der Bestätigung der Kundenbank, die in `AuthRes` eingebunden ist, mit der Warenauslieferung beginnen.

   Das verschlüsselte „Capture-Token“ wird direkt (un-entschlüsselt) in der Händlerdatenbank gespeichert, um evtl. später darauf zurückzugreifen.

4.5.2 Zertifikate

Instanzen:

SET beschreibt eine komplizierte Hierarchie für digitale Zertifikate, in der außer der obligatorischen Root CA[10] jede Interessengruppe das eingene Trust-Center betreibt:

- ROOT CA, die Wurzel-Instanz, an der das gesamte Vertrauen hängt
- CARDHOLDER CA (CCA), die CA der die Kreditkarten ausgebenden Banken
- MERCHANT CA (MCA), die CA der SET akzeptierenden Händler
- BRAND CA, der Zusammenschluß aller Kreditkartengesellschaften
- AQUIRER PAYMENT GATEWAY CA, also die CA der Händlerbanken, die das Gateway ins SWIFT-Netz bilden
- Die GEOPOLITICAL CA, welche offizielle Zertifikate ausstellt, z.B. das Einwohnermeldeamt oder eine andere Regierungsbehörde

Ungültige Zertifikate:

Wenn ein Kunde sein Zertifikat entzogen bekommt, muß das Zertifikat ungültig werden. Um nicht jedem Händler dieser weiter Erde eine Nachricht schicken zu müssen, sobald ein Kunde als nicht vertrauenswürdig eingestuft wird, werden die Zertifikate an Gültigkeitszeiträume gebunden, d.h., nach Ablauf des „expiration date“ nimmt kein Händler das Zertifikat mehr an und der Kunde muß sich ein neues besorgen. Um Zertifikate, deren Gültigkeitszeitraum noch nicht erreicht ist, zurückzuziehen, erzeugt die zentrale Stelle (CCA) von der aktuellen Datenbank mit zurückgezogenen Zertifikaten einen digitalen Fingerabdruck (Digital signierter Hash-Wert). Ein Händler kann sich bei Zweifeln den Fingerabdruck bei der CCA holen, erzeugt einen Hash der

[10] CA: Certificate Authority, Zertifikate ausgebende Stelle

lokal gespeicherten Datenbank und überträgt erst dann die neue Liste auf seinen Rechner, wenn die Abdrücke nicht übereinstimmen.

Inhalte in Zertifikaten:

Um zu verhindern, daß kriminelle Händler die Daten aus Zertifikaten ausspähen, aber trotzdem die Möglichkeit haben, die Gültigkeit eines Zertifikates prüfen, werden die Zertifikate auf spezielle Weise geschützt: Kundenzertifikate enthalten nicht den Namen des Kunden. Die Kontonummer des Kunden wird mittels „*keyed hash functions*“ verschleiert. Der Kunde und die CCA teilen sich einen geheimen Schlüssel, mit welchem die Kontonummer verschlüsselt wird, bevor die Hash-Funktion angewendet wird. Das Ergebnis ist ein eindeutiges Pseudonym für den Kunden, welches ihn eindeutig kennzeichnet, seine Identität oder Kontonummer aber nicht bloßlegt.

4.5.3 Kryptografie

Asymmetrische Chiffre:

Als **asymmetrisches** Verfahren für Signaturen und die sichere Schlüsselübertragung kommt RSA zum Einsatz, wobei alle Schlüssel Modulus-Längen vom 1024 bit aufweisen. Eine Ausnahme hiervon bildet der Signaturschlüssel der ROOT CA, welcher mit 2048 bit festgelegt ist. Lt. Protokollversion 1.0 vom 31. Mai 1997 wird aber angestrebt, diese Größen zu ändern.

Entität	Signatur	Schlüssel-tausch	Zertifikats-signaturen	CRL
Kunde	1024			
Händler	1024	1024		
Payment Gateway	1024	1024		
Kunden CA	1024	1024	1024	
Händler CA	1024	1024	1024	
Payment Gateway CA	1024	1024	1024	1024
Brand geopolitical CA			1024	1024
Brand CA			1024	1024
Root CA			2048	2048

Blockchiffren:

Als **symmetrisches** Verfahren wurde der DES[11] mit 56 bit Schlüsselbreite gewählt, um sensitive Information wie z.B. die Zahlungsanweisungen zu verschlüsseln. Neben dem DES findet noch der CDMF[12] [JMLW93] Verwendung. Der CDMF ist ein symmetrischer Algorithmus, der für

[11] Data Encryption Standard
[12] Commercial Data Masking Facility

die Kommunikation zwischen Händlerbank und Kunden Verwendung findet. Hierbei handelt es sich weniger um einen Verschlüsselungs– denn um einen Verschleierungsalgorithmus. Beim Design von CDMF lag der DES zugrunde, bei dem die effektive Schlüsselbreite von 56 bit auf 40 bit reduziert wurde. Einem Brute-Force-Angriff kann der CDMF somit nicht wiederstehen.

Hashfunktionen:

Die verwendete **Hash-Funktion** ist der SHA-1[13], der in FIPS 180-1 definiert wird. Auch für die *keyed-hash Funktion* sollte der SHA anstelle von MD5 Verwendung finden.

Zufallszahlenerzeugung:

SET schreibt den Implementierern nicht vor, wie Zufallszahlen erzeugt werden. In [VIS97b] empfehlen die Autoren einen Blick in [ECS94], eine gute Einführung in Computergenerierten Zufall.

Algorithmenunabhängigkeit:

Obwohl SET v1.0 die zu verwendenden und unterstützenden Algorithmen explizit vorschreibt, ist die Spezifikation so ausgelegt, daß Entwickler fest definierte Schnittstellen vorfinden, unter denen sie andere Algorithmen einbauen.

4.5.4 Anonymität

Der Kunde tritt beim Händler unter einem (eindeutigen) Pseudonym auf, d.h., aus den Protokolldaten kann der Händler die Identität des Kunden nicht bestimmen. Wenn der Kunde seine Identität allerdings während des Shopping (vor dem eigentlichen SET-Protokoll) bekannt gibt, liegt das außerhalb des Einflußbereiches von SET. Darüber hinaus erhält der Händler keine Einblicke in die Kontoverbindung oder die Kreditkartendaten des Kunden.

Der Kunde leitet keine Bestell-Liste an die Bank weiter. Die Bank bekommt nur den Hash-Wert der Bestell-Liste und die Preis-Summe zu sehen. Somit kann sie zwar bestimmen, daß der Kunde mit dem Händler in Geschäftsbeziehungen steht, kann aber nicht sagen, was das Handelsobjekt war.

4.5.5 Teilbarkeit

Wie bei allen Kreditkarten basierenden Verfahren kann eine Zahlung nicht (im mathematischen Sinn) geteilt werden. Da der Händler das Geld des Kunden auf dem Kundenkonto einfrieren kann, ist es möglich, nur einen Teil des Geldes überweisen zu lassen.

[13] Secure Hash Algorithm Revision 1

4.5.6 Transaktionskosten

Auch bei den Transaktionskosten muß gesagt werden, daß die eigentliche Kreditkartentransaktion die untere Schranke für die Zahlung ergibt. Aufgrund des aufwendigen Entwurfs mit vielen Trustcentern, Payment-Gateways und Banken sind auch viele Stellen präsent, die bezahlt werden müssen und somit hohe Reibungsverluste generieren.

Aufgrund der hohen Transaktionskosten eignet sich SET nur für Macropayments.

4.5.7 Skalierbarkeit

Wenn sich viele Banken entscheiden, im Internet präsent zu sein und es viele Trustcenter zur Zertifizierung gibt, entsteht eine dezentrale Struktur. In einem breit angelegten System bilden die Händlerbanken bzw. deren Payment-Gateways das Nadelör. An jeder Stelle im Protokoll sind (zumindest in Version v1.0) rechenaufwendige RSA-Operationen notwendig, die den gesamten Prozess verlangsamen.

4.5.8 Bedienbarkeit

Bezüglich der Bedienbarkeit läßt sich überhaupt keine Aussage treffen, da SET keine Applikation „zum Anfassen“ ist, sondern nur eine Protokoll-Spezifikation.

4.5.9 Hardware / Software

SET ist nur eine Protokoll-Spezifikation. Ob eine Institution sich für reine Software-Lösungen entscheidet oder ob Teilbereiche des Protokolls in Silizium gegossen werden sollen, bleibt dem Implementierer überlassen. Hardware wird speziell zum Speichern der privaten Schlüssel empfohlen. Besonders an wichtigen Stellen wie z.B. den CA’s ist der Einsatz von Hardware-Tokens empfohlen.

Durch den Einsatz von Software auf Client-Seite muß auch hier gesagt werden, daß das SET durch Viren und trojanische Pferde angreifbar ist, d.h., Viren können das Kundenzertifikat auslesen, während gerade eine Signatur-Operation im Gange ist oder Kreditkartendaten auslesen, während der Kunde sie gerade ins SET-Programm einträgt. Ebenso könnten Sniffer-Programme die Zufallszahlenerzeugung abhören und so die verwendeten Session-Keys an einen Angreifer transferieren. Neben der Ansicherung des Transfers der Daten über unsichere Netze muß auch die Absicherung der Daten auf dem lokalen Rechner berücksichtigt werden.

Aufgrund der Gefahren für den Client-Rechner wäre es wünschenswert, Zufallszahlenerzeugung, die Signaturerzeugung und auch die Speicherung des privaten Schlüssels auf externe, für Viren unangreifbare Komponenten wie z.B. SmartCards auszulagern. Die Protokoll-Spezifikation läßt sich auch nicht über die Speicherung der privaten Schlüssel aus. Richtlinien bezüglich Verschlüsselung privater Schlüssel sind unbedingt notwendig, wenn verhindet werden

soll, daß unerfahrene Implementierer private Schlüssel ungesichert auf der Kunden-Festplatte speichern.

4.5.10 Perspektive

Bei SET handelt es sich um ein System mit gut durchdachten Ansätzen, das viele Anwendungsfälle abdecken soll. Durch den Einsatz von 56-bit-DES und 40-bit-CDMF wurden schwache symmetrische Algorithmen ins Protokoll eingebaut, was US-amerikanischen Firmen wohl die Ausfuhrerlaubnis sichern soll. Wünschenswert ist der Einbau von starken symmetrischen Algorithmen in kommende Protokoll-Versionen. Darüber hinaus sind auch Alternativen zu RSA denkbar (elliptische Kurven, etc.). Durch den intensiven Einsatz von Hash-Funktionen wird großer Wert auf die Integrität der Daten gelegt, was den Sicherheitslevel festigt.

Der intensive Einsatz von Zertifikaten wirkt sich ebenfalls positiv auf die Gesamtsicherheit aus. Problematisch sind hier die weiter oben angedeuteten politischen Unwägbarkeiten. Zur Zeit existieren noch keine Erfahrungen mit der Verteilung von Zertifikaten für so große Personengruppen. In SET v1.0 ist es noch möglich, daß ein Kunde ohne eigenes Signaturzertifikat Käufe tätigt, obwohl der Standard Kunden-Zertifikate favorisiert. Auf diesem Sektor müßen sich erst praktische Erfahrungen herausbilden.

Grundsätzlich läßt sich sagen, daß SET der aussichtsreichste Kandidat ist, wenn es im Kreditkartenzahlungen im Internet geht. Der große Vorteil von SET im Gegensatz zu einfachen Transport-Protokollen wie SSL[14] liegt in der Rechtsverbindlichkeit. Alle Parteien können die empfangenen Nachrichten speichern und später den Handlungsablauf anhand der Signaturen rechtsverbindlich nachweisen.

4.5.11 Literatur

[RK97], [SFE97, Seiten 39–44], [Luc97e], [Lan97], [Chr96a], [KW97], [Way97, Seiten 159–173]. Die Original-Spezifikationen sind unter `http://www.visa.com/cgi-bin/vee/nt/ecomm/set/downloads.html?2+0/` zu finden, eine Analyse zur Protokoll-Spezifikation unter `http://www.ozemail.com.au/~netsafe/set.html`

[14] Secure Sockets Layer

5 Digitales Bargeld

5.1 Ecash

Ecash ist ein Produkt der Firma DigiCash `http://www.digicash.nl/` in Amsterdam. DigiCash wurde von DAVID CHAUM gegründet. CHAUM gilt als der Erfinder der blinden Signatur und besitzt eine Vielzahl von Patenten zu digitalen Signaturen jeglicher Geschmacksrichtung („Blind unanticipated signature systems“, „One-show blind signature systems“, „Undeniable signature systems“, „Returned-value blind signature systems“, „Unpredictable blind signature systems“ und „Designated-confirmer signature systems“, um nur die Signatur-Patente zu nennen).

5.1.1 Das Prinzip

Ecash ist ein Münz-basiertes, anonymes System. In Zusammenarbeit mit der Bank prägt der Kunde eine Münze, die auf seiner Festplatte gespeichert wird und die er bei Bedarf an den Händler weiterreichen kann. Wenn dieser die Münze bei der Bank einlöst, kann die Bank nicht nachvollziehen, wer der Kunde war, da die Bank die Münzen blind signiert hat. Die Bank kann nur nachvollziehen, ob die Münze echt ist, d.h., ob die Bank die Münze unterschrieben hat und ob sie zum ersten Mal eingereicht wurde[1].

5.1.2 Das Münzerzeugungs–Protokoll

Um eine Münze zu erzeugen, müssen sowohl Bank als auch der Kunde mitwirken, d.h., die Münzen werden nicht einfach von der Bank ausgegeben, sondern werden in einem kooperativen Prozess zwischen Bank und Kunden erzeugt. Das ist nötig, um die Eigenschaft *Anonymität* zu ermöglichen.

- Der Kunde mit der Identität `ID` möchte eine Münze erzeugen, die den Wert i hat. Das Wallet-Programm des Kunden erzeugt eine Zufallszahl m, die als Seriennummern für die Münze dient. Dann wird diese Seriennummer mit dem Faktor k^{e_i} verdunkelt (siehe Abschnitt 2.5, Seite 13) und das Produkt $r_i = m k^{e_i}$ gebildet. Zusätzlich nennt der Kunde der

[1] double-spending verhindern

Bank den gewünschten Wert i der zukünftigen Münze. Jetzt wird das Tupel $\{\texttt{ID}, i, r\}$ mit dem privaten Schlüssel des Kunden signiert und mit dem öffentlichen Schlüssel der Bank verschlüsselt und an die Bank geschickt. Durch die Verschlüsselung kann nur der Bankserver die Nachricht entschlüsseln, und durch die Signatur hat die Bank die Gewähr, daß die Münze wirklich vom Kunden kommt.

- Die Bank entschlüsselt mit ihrem privaten Kommunikationsschlüssel die Nachricht und prüft die Signatur des Kunden. Dann wählt die Bank den mit dem Münz-Wert i korrespondierenden geheimen Schlüssel d_i, signiert die Münze mit dem Schlüssel und hebt den entsprechenden Betrag vom Konto des Kunden ab. Die signierte Münze $t_i^{d_i} = m^{d_i} k^{d_i e_i}$ wird dann an den Kunden geschickt.

- Der Kunde dividiert den Blinding-Faktor k durch die Berechnung $m_i = r_i^{d_i} k^{-1}$ heraus und erhält die blind signierte Münze m^{d_i}, die die Wallet-Software abspeichert.

Die Bank weiß jetzt, daß der Kunde eine Münze mit dem Wert i besitzt, hat aber keine Ahnung von der Seriennummer. Der ganze Prozeß wird von der Wallet-Software des Kunden durchgeführt. Um Double-Spending zu verhindern, speichert die Bank die Seriennummern aller von Händlern eingereichten Münzen ab. Daher muß die Seriennummer m so groß gewählt sein ($\geq$ 100 Dezimalstellen), daß es praktisch nicht möglich ist, daß zwei Münzen mit der gleichen Seriennummer erzeugt werden, da sonst der zweite Kunde des Double-Spending bezichtigt werden würde.

5.1.3 Das Zahlungsprotokoll

Nach erfolgreicher Münzerzeugung liegen die Münzen auf der Festplatte des Kunden bereit. Im nachfolgenden wird das eigentliche Zahlungsprotokoll beschrieben.

1. Bei einem Kauf von Waren via Internet laufen auf dem Kundenrechner die Wallet-Software und der Web-Browser. Im ersten Schritt bestellt der Kunde beim Händler-Server die Waren. Händlerseitig wird die Anfrage von einem CGI-Programm verarbeitet.

2. Das CGI-Programm berechnet aus den Bestelldaten den Warenwert und ruft die Wallet-Software des Händlers auf. Der Händler-Wallet wird der entsprechende Betrag übermittelt.

3. Die Händler-Wallet baut via TCP/IP eine Verbindung mit der Wallet des Kunden auf und übergibt ihr eine Zahlungsaufforderung.

4. Die Kunden-Wallet zeigt einen Benutzer-Dialog mit den Transaktionsdaten (Zahlungshöhe, Shop-ID, Zahlungsbeschreibung) an.

5. Der Kunde entscheidet über die Zahlung. Im Falle einer Negativ-Entscheidung ist das Zahlungsprotokoll an dieser Stelle zuende. Falls der Kunde die Zahlung akzeptiert, prüft

die Wallet, ob genug passende Münzen auf der Festplatte gespeichert sind. Stehen nicht ausreichend oder passende Münzen zur Verfügung, bricht die Wallet die Transaktion ab und gibt eine Fehlermeldung aus.

6. Die Kunden-Wallet überträgt die Münzen an die Händler-Wallet.
7. Die Händler-Wallet reicht die Münzen an den Bank-Server des Kunden zwecks Online-Verifikation weiter.
8. Der Bank-Server prüft die Gültigkeit der Unterschrift d und fragt die Datenbank für verwendete Seriennummern ab, um Double-Spending zu verhindern. Im Erfolgsfall wird das Geld dem Händler gutgeschrieben. Das Ergebnis der Aktionen wird dem Händler mitgeteilt.
9. Wenn die Zahlung akzeptiert wurde, schickt der Händler dem Kunden eine Bestätigung und beginnt mit der Warenlieferung.

5.1.4 Teilbarkeit

Die ecash-Münzen sind nicht teilbar. Dafür können aber Münzen mit verschiedenen Größen (Nominationen, Nominal-Werte) erstellt werden. Basierend auf der Grundeinheit (beispielsweise 1 Pfennig) werden die Münzen in 2er-Potenzen größer, sodaß jeder Betrag darstellbar ist. Die Kundensoftware speichert also zu jedem Münzwert i n_i Münzen auf der Platte, sodaß sich das Guthaben einfach aus $\sum_{i=0}^{N} in_i$ ergibt. Wenn ein kleinerer Betrag gefordert ist als die kleinste Münze im Wallet, so muß vorher bei der Bank eine Münze gegen kleinere gewechselt werden, oder zusammen mit der Bank eine neue erzeugt werden (natürlich gegen Verrechnung).

5.1.5 Hardware / Software

Ecash ist ein Software-System. Die Speicherung der Münzen in SmartCards würde die Speichermöglichkeiten heutiger SmartCards extrem beanspruchen.

5.1.6 Online / Offline

Aufgrund der vielen Protokollschritte, die eine Interaktion mit der Bank erfordern und der Notwendigkeit, erhaltene Münzen sofort einzureichen, kann die unter Ecash verstandene Münze nur Online realisiert werden.

5.1.7 Kleinhändler

Durch die Token-Eigenschaft von Ecash ist es vollkommen egal, ob ein Händler mit Kreditkartenfirmen-Kontakten die Münze einreicht oder eine Privatperson. Jeder kann Ecash

empfangen, weil es bei der Bank sofort in neues Ecash umgewandelt werden kann.

5.1.8 Verlust der Daten

Bei der Erstinstallation der Ecash-Wallet wird dem Kunden ein sog. Geheimschlüssel angezeigt, den er sich in nichtelektronischer Form notieren soll. Bei diesem Geheimschlüssel handelt es sich um die Berechnungsgrundlage für die Blinding-Faktoren, mit denen die Münzen anonymisiert werden. Wenn das Schicksal es böse mit dem Kunden meint und die Platte einen Head-Crash erleidet, muß der Kunde so schnell wie möglich seine Wallet-Software neu einrichten und in der Installation die Option einer „Erneuten Einrichtung“ wählen. Er wird dann aufgefordert, den Geheimschlüssel der Erstinitialisierung einzugeben. Die Wallet-Software nimmt dann Kontakt mit der Bank auf und hebt die Anonymität aller (noch nicht ausgegebenen) Münzen auf.

Die Bank kennt alle vom Anwender generierten Münzen, wenn auch nur in ihrer geblindeten Form. Nachdem alle Münzen entblindet wurden, kann die Bank über die Double-Spending-Datenbank nachvollziehen, welche der Münzen noch nicht eingereicht wurden und dem Kunden den Betrag erstatten.

Dadurch kann dem Kunden das verbliebene Guthaben rekonstruiert werden, wenn nicht ein Angreifer begonnen hat, ausgespähte Münzen auszugeben. Münzen, die von einem Angreifer *nach* Deanonymisierung ausgegeben werden, fallen dem Bank-Server als Fälschungen auf.

5.1.9 Plattformen

Der Kunde

Auf dem Rechner des Kunden kommt ein beliebiger Web-Browser zum Einsatz. Für die Zahlungen gibt es eine Client-Applikation mit grafischer Oberfläche, die „*ecash Purse*“. Dieses Programm gibt es z.Zt. für die gesamte MS-Windows-Welt, d.h., Windows 3.x, Windows 95, Windows NT 3.5x und Windows NT 4.0. Darüber hinaus gibt es text-basierte Clients für Windows NT und die UNIX-Varianten FreeBSD, AIX, HP, Linux, SunOS und Solaris.

Der Händler

Die Händler-Software läuft mit allen gängigen Web-Servern, d.h. in der Unix-Welt unter NCSA httpd ($\geq$ 1.3), W3/CERN httpd ($\geq$ v3.0), Apache httpd, Netscape Commerce Server und dem Netscape Communications Server. Unter Windows 95 und NT stehen der Netscape Commerce Server, Netscape Communications Server, Netscape FastTrack Server, Microsoft Internet Information Server, EMWACS, HTTPS, WebSite, WebHub, u.v.a.[2] zur Verfügung. Unter Unix werden nur ein C-Compiler und eine lauffähige Perl5-Distribution benötigt.

[2] `http://www.digicash.nl/ecash/docs/ShopIntro(23G).pdf`

5.1.10 Bedienbarkeit

Die Bedienbarkeit wird durch die Wallet-Software bestimmt. Trotz der komplexen Protokoll-Schritte ist die Bedienung für den Anwender intuitiv und transparent. Durch die Möglichkeit, Ecash nicht nur an Händler, sondern auch an Privatpersonen zu schicken, werden die Bargeldstrukturen sehr gut modelliert.

Die aktuelle Version der Cyberwallet-Software erlaubt auch automatische Zahlungen (wie übrigens andere Wallets auch). Diese automatischen Zahlungen müssen dann vom Anwender nicht jedes Mal bestätigt werden. Bei solchen Automatismen muß sich noch herausstellen, wie die Anwender diese Möglichkeit nutzen und wie der Mißbrauch verhindert wird (was nur durch Limitierung der Einzelbeträge, durch maximale Summen und durch gute Aufkärung der Anwender möglich ist).

5.1.11 Perspektive

Ecash stellt ein leistungsfähiges System zur Verfügung, das sich sehr gut für unkomplizierte Micropayments oder auch größere Zahlungen eignet. Aufgrund der komplexen Online-Vorgänge und der teuren RSA-Operationen eignet sich Ecash nicht für Picopayments, da unter einer gewissen Schranke selbst Ecash unrentabel wird. Bezüglich der Anonymität und der recht guten Nachbildung der Bargeld-Eigenschaften ist Ecash einer der interessantesten Kandidaten für einen universellen Bargeld-Ersatz.

Durch die Token-Eigenschaft, daß eine Ecash-Münze tatsächlich ihren Wert repräsentiert, sind evtl. rechtliche Probleme mit Bankengesetzen zu erwarten, da Ecash ja eine Münzprägung darstellt, die u.U. staatlich reglementiert wird.

5.1.12 Literatur

[DD97], [Way97, Seiten 189–198], [FW96, Seiten 63–66], [Sie97a, Seiten 125–143], [SFE97, Seiten 59–62]

5.2 NetCash

NetCash[3] ist ein Token-basiertes Zahlungssystem, d.h., es werden elektronische Münzen in Form von Bitfolgen übertragen. NetCash wurde am Institut für Informationswissenschaften an der Universität von Südkalifornien von GENNADY MEDVINSKY und B. CLIFFORD NEUMANN entwickelt und in [MN93] beschrieben.

Ziel der Entwicklung von NetCash war die Entwicklung eines sicheren, skalierbaren, billigen, zuverlässigen und anonymen Zahlungssystems.

[3] `http://nii-server.isi.edu/info/netcash/`

NetCash besteht aus einem Verbund von sog. „*Currency Servern*“, bei denen die Kunden und Händler Münzen abheben und einzahlen oder eintauschen können. Ein Currency Server hat die Funktion einer Bank und Wechselstube. Eine Münze ist eine Bitfolge, in der der erzeugende Currency-Server, im nachfolgenden nur noch CS genannt, die IP-Adresse des CS, das Verfallsdatum der Münze, sowie der Wert der Münze und eine eindeutige Seriennummer festgehalten sind. Diese Bitfolge wird vom CS digital signiert.

5.2.1 Eine Transaktion

Wenn der Kunde einen Kauf tätigen will, stellt er zuerst eine sichere Verbindung zum Händler her. Diese sichere Verbindung kann über RSA und ein symmetrisches Verschlüsselungsverfahren realisiert werden, durch Diffie-Hellman-Keyexchange oder auch durch SSL.Über diese sichere Verbindung überträgt er die Münzen an den Händler. Der Händler baut eine Verbindung zum Currency-Server auf, der die Münzen erzeugt hat. Der Händler überträgt die Münzen an den CS, welcher dann die Gültigkeit prüft. Je nach Händlerwunsch wird das Geld auf dem Konto des Händlers eingezahlt oder es werden dem Händler neue Münzen zurückgeliefert.

1. Wenn der Kunde Münzen von seinem Konto abhebt, erzeugt der CS Münzen in der gewünschten Stückelung, hebt den Betrag vom Kundenkonto ab und sendet die Münzen an den Kunden.

2. Kunde und Händler bauen eine Verbindung auf. Diese Verbindung sollte gesichert sein, z.B. durch SSL (Secure Socket Layers), durch RSA-Verschlüsselung, wenn der Kunde den Schlüsel des Händlers kennt oder durch Diffie-Hellman-Keyexchange.

3. Der Kunde packt die Münzen, einen Session-Key K_{AN} und eine Session-ID zu einem Paket zusammen und schickt es über den sicheren Kanal an den Händler.

4. Der Händler prüft mit den Zertifikaten die Unterschriften der Münzen und leitet die Münzen an den CS weiter. Die Münzen werden dazu mit einem symmetrischen Schlüssel K_{BN} zusammengepackt und mit dem öffentlichen Schlüssel des CS verschlüsselt, bevor sie an den CS weitergeleitet werden.

5. Der Currency-Server prüft, ob die Seriennummern in der Datenbank vorkommen, d.h., ob sie zum ersten Mal eingereicht werden. Falls das der Fall ist, werden dem Händler entweder neue Münzen zugeschickt oder der Betrag dem Händlerguthaben gutgeschrieben. Die neuen Münzen bzw. die Gutschriftsbestätigung wird mit K_{BN} verschlüsselt an den Händler gesendet.

6. Der Händler entschlüsselt das Paket mit K_{BN}. Jetzt ist voll bezahlt. Der Händler beginnt jetzt mit der Warenlieferung.

5.2.2 Ein Münztausch

1. Um erhaltene Münzen gegen andere zu tauschen, packt der Kunde die Münzen und Anweisungen für den CS zusammen. Mit in das Paket kommt ein symmetrischer Schlüssel K. Das gesamte Paket wird mit dem öffentlichen Schlüssel der Bank verschlüsselt.

2. Die Bank (der CS) decodiert das Paket, prüft die Gültigkeit und Echtheit der Münzen, löscht die alten Seriennummern aus der Datenbank und erzeugt neue Münzen. Die neuen Münzen werden symmetrisch mit dem Schlüssel K verschlüsselt und an den Kunden geschickt.

3. Mit Hilfe von K entschlüsselt der Kunde das Paket und hat die neuen Münzen. Da der Schlüssel K nur dem Kunden und der Bank bekannt war, ist das Paket durch die symmetrische Verschlüsselung implizit signiert.

5.2.3 Sicherheit

Zertifikate:

Um die Echtheit von CS-Schlüsseln zu gewährleisten, existiert ein zentraler Signaturschlüssel, der in den USA der FIC (Federal insurance corporation) gehört. Dieser zentrale Schlüssel erstellt Zertifikate für lokale Currency Server.

Double-Spending:

In Systemen, bei denen die Bank die Seriennummer einer Münze erst bei deren Einzahlung erfährt, muß die Bank eine Double-Spending-Datenbank führen, in der alle jemals verwendeten und noch nicht abgelaufenen Seriennummern gelistet sind. Bei NetCash sind die Seriennummern der Bank bei Erzeugung der Münze bekannt, daher speichert der CS die Seriennummern aller im Unlauf befindlichen Münzen und löscht den korrsepondierenden Eintrag, sobald die Münze eingezahlt oder umgetauscht wurde.

Werden Münzen eingelöst, deren Seriennummer nicht in der Datenbank vorliegt, ist das als versuchtes Double-Spending zu werten.

5.2.4 Anonymität

Die ausgegebenen Münzen sind eindeutig an den Kunden gebunden, da der CS weiß, wem er welche Münze gegeben hat. Wenn der Kunde seine Münze also direkt nach Erhalt an einen Händler weiterreicht, kann die Bank nach Einzahlung durch den Händler genau bestimmen, bei welchem Händler der Kunde gekauft hat. Dieser Problematik versucht NetCash auf verschiedene Art und Weisen zu begegnen:

- Die Bank kann sich vertraglich verpflichten, keine Einträge vorzunehmen. Daher wird dem Kunden garantiert, daß er bei Käufen anonym bleibt.
- Der Aufwand, den eine Bank treiben muß, um Kundenprofile zu erstellen, ist bedeutend größer als der normale Verwaltungsaufwand; [Way97, Seite 135] stellt folgende Kalkulation auf:

  Um ausstehende Münzen nachzuhalten, sind pro Münze ca. 8 Byte Festplattenkapazität notwendig. Um die Kundenidentität und das Datum der Ausgabe mit zu speichern, steigt der benötigte Platz auf das Doppelte an. Daher lohnt es sich für Banken nicht, die Information zu speichern.

  Trotzdem bleibt der Aufwand dieser Deanonymisierung endlich (im Gegensatz zu DigiCash, wo die Bank mathematisch keine Möglichkeit hat, die Münzen zu tracen). Es bleibt nur die Frage, ob die Banken für ein Feature zahlen möchten, das die Kunden garnicht haben wollen.
- Es existiert ein ganzes Netz von Currency Servern. Wenn der Kunde seine Münze bei einem beliebigen Currency-Server eintauscht, reicht der beauftragte CS die Münze beim Original-CS ein. Der bestätigt die Echtheit und verrechnet das Geld mit dem beauftragten CS. Nach der Authorisierung gibt der beauftragte CS dem Kunden eine neue Münze. Da bei der Authorisierung der Original-CS die Identität des Kunden nicht weitergibt, hat der Kunde nach dem Tausch eine anonyme Münze.

  Dieses Verfahren setzt voraus, daß die Identität wirklich nicht weitergeleitet wird. Wenn der Kunde sehr mißtrauisch ist, kann er seine Münze bei vielen CS's nacheinander eintauschen, in der Hoffnung, daß die Münze irgendwann nicht mehr weiterverfolgbar wird.
- Wenn die Strafverfolgungsbehörden die Transaktionen eines Kunden überwachen möchten, kann auch selektiv für diesen Kunden die Anonymität aufgehoben werden, d.h, alle Münzen, die der Kunde in Zukunft abhebt, werden notiert. Problematisch wird auch hier das eintauschen über andere CS's, weil dann beim jeweils letzten CS ebenfalls richterliche Genehmigungen vorgelegt werden müssen, die eine Protokollierung rechtfertigen.

5.2.5 Skalierbarkeit

NetCash beschreibt ein Netzwerk von vielen Currency-Servern, die unabhängig voneinander arbeiten können. In jeder Münze sind die IP-Adresse des CS sowie der Name enthalten. Zuerst kann die Münz-empfangende Partei versuchen, die Münze beim CS abzuliefern, der unter der gegebenen IP-Adresse erreichbar ist. Wenn das nicht funktioniert, kann der Händler über Directory-Services mit Hilfe des Namens an andere IP-Adressen von Currency Servern des gleichen Anbieters kommen, die die Münze auch annehmen können und die Zugriff auf die gleiche Datenbank haben.

Die Last kann innerhalb eines CS aufgeteilt werden, da für ganze Seriennummernbereiche unterschiedliche Rechner mit individuellen Datenbanken zuständig sind. Aus Sicht des Gesamtsy-

stems ist die Last ja schon gesplittet, da jeder CS unanhängig von den anderen läuft und seinen Service anbietet. Die einzige zentrale Instanz ist das Trust-Center der FIC, und diese Anwendung ist nicht zeitkritisch, da die Zertifikate über lange Zeiten ihre Gültigkeit haben.

5.2.6 Transaktionskosten

Wenn die Münzen häufig hin- und hergetauscht werden, um beispielsweise die Anonymität zu steigern, fallen viele Verwaltungsvorgänge an. Jede Münzerzeugung ist mit einer RSA-Signatur verbunden. Daher ist NetCash nicht für Picopayments geeignet. Es bleibt die Frage, ob sich Currency-Server die Dienstleistung des Tausches bezahlen lassen, wenn Kunden viele Tauschaktionen initiieren.

5.2.7 Risiken

Durch das eingebaute Ablaufdatum für NetCash-Münzen steht auch hier der jeweilige Besitzer in der Pflicht, Münzen vor Ablauf des Expiration-Date erneuern zu lassen. Wenn die Banken keine Protokollierung der Transaktionen fahren, wird sonst der jeweilige Eintrag aus der Datenbank gelöscht, wenn die Zeit abgelaufen ist.

5.2.8 Online / Offline

NetCash ist als Online-System konzipiert, da die Verhinderung von Double-Spending immer ein Datenbank-Lookup erfordert.

5.2.9 Literatur

[MN93], [SFE97, Seiten 62–64], [Way97, Seiten 131–139], [FW96, Seiten 71–72]

5.3 Millicent

Millicent [GMA+] wurde von der Firma Digital Equipment (DEC, `http://www.digital.com` und `http://www.millicent.com`) im Rahmen des Millicent-Projektes entwickelt. Ziel war die Entwicklung eines Systems, welches sich für Micro- bzw. sogar Picopayments eignet. Unter Picopayments seien hier Zahlungen im Zehntelpfennigsbereich genannt. Solche extrem niedrigen Zahlungen sind beispielsweise beim Abruf von WWW-Seiten wie Zeitschriftenartikeln denkbar.

Der Weg, den Millicent bezüglich der Sicherheitsphilosophie einschlägt, unterscheidet sich stark von den Sicherheitsbedürfnissen von Systemen wie SET. Millicent soll Kleinstbeträge modellieren, d.h., im Gegensatz zu Systemen, wo der Kunde Beträge um die DM 5,- – DM 50,- auf der Festplatte hält, soll der bei Millicent vorgehaltene Gesamtsumme nur DM 1,- – DM 2,- betragen. Aus der niedrigen Summe einer einzelnen Transaktion ergeben sich andere Sicherheitsbedürfnisse:

Als Beispiel sei hier ein Kaugummi-Automat genannt. Wenn ein Kunde einen Kaugummi für 2 Pfennig kaufen möchte und das Gerät sich nach Einwurf des Geldes stur stellt (sowohl bezüglich der Lieferung des Kaugummis als auch der Rückgabe des Geldes) hat der Kunde zwar einen Verlust erlitten, der sich aber in einem verschmerzbaren Rahmen hält. Der Kunde verzichtet auf eine aufwendige Aufklärung des Geschäftsvorfalls und beendet seine Geschäftsbeziehung zu dem Kaugummiautomaten.

Der Händler kann ebenfalls kleinere Verluste verkraften. Wenn (beim Kaugummi-Beispiel) die Ware herausgegeben wird, obwohl eine falsche Münze eingeworfen wurde, geht der Verlust eines einzelnen Vorfalls im Rauschen unter. Was aus Händlersicht allerdings wirkungsvoll verhindert werden muß, ist Betrug im großen Stil, bei dem sich extrem viele Vorfälle mit geringen Geldwerten zu einem großen Schaden summieren.

Im Gegensatz zu DigiCash, wo die Münzen relativ große Werte annehmen können und somit aufwendig generiert und durch sehr sichere Mechanismen geschützt werden müssen, stellt Millicent nur die Forderung, daß die Kosten, die einem Fälscher entstehen, den Gegenwert der Münze bei weitem überschreiten müssen. Bei den Kosten sind nicht die Kosten für Hardware-Anschaffungen gemeint sondern der notwendige Rechenaufwand pro Münze. Das Brechen des Protokolls muß teuerer als der Geldwert werden.

Im folgenden lehne ich mich stark an die exzellente Beschreibung in [GMA+] an:

5.3.1 Millicent

Wenn man von Millicent spricht, kommt man um zwei Begriffe nicht herum: *Broker* und *Scrip*[4].

[4] Nicht Script mit t am Ende, sondern ohne t

Scrip stellt ein Kundenkonto beim Händler dar. Scrip ist das Token, mit dem der Kunde bezahlt. Der Kunde sendet sein Scrip mit der Bestellung an den Händler. Der Händler prüft das Scrip auf Gültigkeit und liefert dem Kunden die bestellte Ware, zusammen mit dem um den Bestell-Wert dekrementierten, neuen Scrip als Wechselgeld. Wenn der Kunde keine Bestellungen mehr tätigen möchte, kann er das verbliebene Scrip einlösen und den Account auflösen.

Ein Kunde, der mit vielen Händlern Geschäftsbeziehungen unterhält, muß für jeden Händler das passende Scrip verfügbar haben. Darüber hinaus muß bei jedem Händler die monatliche Rechnung beglichen werden, denn das Scrip ist ja nur der Geld-Ersatz. Für den Händler entsteht ebenfalls die Problematik von extrem vielen Buchungen. Um das Handling für alle Parteien zu vereinfachen, führt Millicent Broker ein. Ein Broker ist ein Vermittler zwischen Kunde und Händler. Die Kunden und die Händler gehen eine langandauernde Geschäftsbeziehung mit dem Broker ein. Der Broker versorgt die Kunden mit gewünschten Scrip, wenn diese bei einem speziellen Händler etwas kaufen möchten und nehmen nicht mehr benötigtes Scrip zurück, um es beim Händler einzulösen.

Millicent vereinfacht die Account-Verwaltung auf verschiedene Arten:

- Es werden keine rechenaufwendigen und teuren RSA-Signaturen verwendet, sondern nur Hash-Funktionen und Blockchiffren.
- Jeder Händler gibt sein eigenes Scrip heraus. Der Händler ist die prüfende Instanz, wenn es um die Echtheit geht; der Händler verwaltet die Double-Spending-Datenbank, daher muß er nicht bei einer Münzen ausgebenden Bank die Gültigkeit bestätigen lassen, weil er alle Daten lokal vorhalten kann.
- Broker tragen weiter zur Vereinfachung bei. Der Händler muß nur noch mit den Brokern abrechnen. Durch die Broker zwischen Kunde und Händler existieren zwar zwei statt einem Konto, aber die absolute Accountanzahl im gesamten System mit vielen Händlern, Brokern und Kunden, sinkt.
- Bei den meisten Account-basierten Systemen behält die Bank dieÜbersicht über die Konten, d.h., die Bank macht die Buchführung. Bei Millicent muß die Buchführung über den aktuellen Stand der Kundenkonten **nicht** der Händler erledigen, sondern der Kunde. Da der Wert des Scrip implizit im Scrip eincodiert ist, hat nur der Kunde dieÜbersicht über das verbliebene Geld. Da das Scrip durch die digitale Signatur des Händlers geschützt ist, kann der Kunde das Scrip nicht modifizieren.
- Der Buchhaltungsaufwand in der Double-Spending-Datenbank des Händlers wird durch ein Expiration-Date im Scrip noch weiter reduziert. Scrip ist nur eine endliche Zeitspanne gültig, daher kann der Händler alte Einträge aus seiner Datenbank löschen.

5.3.2 Sicherheitsaspekte und Vertrauensstellungen

Alles basiert auf dem begrenzten Wert von Scrip und dem anderen Verhalten der Menschen bei kleinsten Beträgen. Niemand benötigt für eine einzelne Erdnuss eine Quittung. Wenn der Kunde die gekaufte Ware nicht erhält, läßt er es eben. Wenn der Kunden dann und wann mal Bruchteile eines Pfennig verliert, wird er schon nicht allzu verärgert sein.

Das Vertrauensmodell

Die drei Parteien haben ganz unterschiedliche Ansprüche und Stellungen, daher ist das Trust-Model sehr asymmetrisch. Die Broker sind am vertrauenswürdigsten, danach kommen die Händler und am wenigsten wird dem Kunden vertraut. Dem Kunden muß nur vertraut werden, wenn er sich über einen Händler beschwert.

Broker sind große Unternehmen wie Banken oder ISP's[5]. Aus drei Gründen lohnt es sich für Broker nicht, zu betrügen:

1. Kunde und Händler können ohne Interaktion mit dem Broker die Gültigkeit von Scrip prüfen. Wenn hier Differenzen auftauchen, ist es wahrscheinlich, daß der Broker Schuld war.
2. Da der einzelne Kunde nicht viel Scrip hat, muß ein Broker schon extrem viele Kunden betrügen, damit sich die Sache wirklich lohnt, was die Wahrscheinlichkeit, daß es auffliegt, extrem in die Höhe schnellen läßt.
3. Der Gesichtsverlust, den renomierte Unternehmen erleiden, wenn auch nur der Schatten von Unehrlichkeit auf die Firma fällt, wiegt schwerer als minimale Gewinne, die durch Betrug entstehen.

Händler können nur betrügen, indem sie die Ware nach Zahlung nicht liefern. Wenn sich beim Broker die Berichte über solche Vorgänge mehren, weil sich viele Kunden beschweren, beendet der Broker seine Geschäfte mit dem Händler.

Als Ergebnis liegt das Hauptaugenmerk von Millicent auf der Verhinderung von Betrug durch Kunden. Betrügerische Broker und Händler werden sukzessive aus dem System entfernt, wenn sie aufgrund von extrem vielen Beschwerden auffallen.

Das Sicherheitsmodell

Alle Transaktionen werden durch digitale Signaturen gesichert. Wenn im Zusammenhang mit Millicent von digitalen Signaturen die Rede ist, so sind damit immer Hash-Werte über die Konkatenation der zu signierenden Daten und ein von beiden Partnern geteiltes Geheimnis gemeint.

[5] Internet Service Provider

Digitale Signaturen im Sinne von RSA gibt es bei Millicent nicht. Das Geheimnis wird nie im Klartext über unsichere Kanäle gesendet, also kann es nicht abgehört werden. Scrip kann nicht mehrmals verwendet werden. Daher können Replay-Angriffe keinen finanziellen Gewinn liefern. Durch Replay-Angriffe können Kunden oder Broker allerdings in Mißkredit gebracht werden, da der Händler annehmen muß, daß ein legitimer Nutzer Scrip mehrfach ausgeben möchte.

Durch den geringen Wert von Scrip wird die Attraktivität für Angriffe gesenkt. Teure Number-Cruncher werden nicht damit beschäftigt, billiges Scrip zu fälschen. Darüber hinaus würden im Falle eines gelungenen Angriffes viele Transaktionen notwendig sein, was wiederum die Gefahr für den Angreifer steigert.

5.3.3 Scrip

Scrip hat verschiedene Eigenschaften:

- Scrip kann nur bei einem einzigen Händler eingelöst werden.
- Scrip kann nur vom rechtmäßigen Eigentümer ausgegeben werden.
- Scrip kann nur einmal verwendet werden.
- Scrip kann nicht einfach modifiziert oder gefälscht werden.
- Scrip kann einfach durch den Händler erzeugt und geprüft werden.

Daraus hat DEC folgenden Katalog zuammengestellt:

- Im Scrip sind sowohl Wert als auch Händler eindeutig gekennzeichnet.
- Scrip enthält eine Seriennummer, um Double-Spending aufzudecken.
- Die digitale Signatur verhindert die Modifikation oder Fälschung.
- Der Kunde signiert auszugebendes Scrip mit einer Signatur, die der Händler aus dem Scrip selbst ableiten kann.
- Die Signaturen werden durch Hash-Funktionen wie MD5 oder SHA modelliert.

5.3.4 Geheimnisse

Für die Erzeugung, Nutzung und Prüfung von Scrip werden drei geheime Zahlen definiert:

1. Der Kunde bekommt das `customer_secret`, mit dem er die Scrip-Nutzung autorisieren kann, auf einem sicheren Kanal übermittelt.

2. Der Händler hat das master_customer_secret, aus dem er mit dem Scrip zusammen das customer_secret berechnen kann.

3. Das master_scrip_secret verwendet der Händler, um Scrip zu signieren und unfälschbar zu machen.

5.3.5 Scrip-Struktur

Im Scrip sind verschiedene Felder vorhanden:

1. Vendor kennzeichnet eindeutig den Händler, für den das Scrip gültig ist.

2. Der Wert des Scrip ist in Value definiert.

3. Die Seriennummer heißt ID#. Der Händler kann für Gruppen von ID's verschiedene master_scrip_secret's verwenden. Über die ID# kann der Händler herausbekommen, welches master_scrip_secret er verwenden muß.

4. Die Kundennummer Cust_ID# kennzeichnet eindeutig den Kunden, der das Scrip nutzen kann. Aus Cust_ID# und dem master_customer_secret kann der Händler das Kundengeheimnis customer_secret berechnen. Daher muß er es auch nicht speichern, da der Kunde es sozusagen im Scrip mitliefert.

5. Das Ablaufdatum Expires des Scrip kennzeichnet den Tag, nachdem das Scrip seine Gültigkeit verliert.

6. In den Kundeneigenschaften Props kann der Händler Vermerke über den Kunden eintragen, z.B. das Alter, um gewisse Artikel zu reglementieren.

7. Die digitale Signatur ist Certificate. Sie berechnet sich aus den obigen Werten und dem master_scrip_secret.

5.3.6 Scrip-Erzeugung

Das Scrip selbst hat die Form

$$\texttt{Scrip} = (\texttt{Scrip\_Body} \,||\, \texttt{Certificate})$$

, wobei

$$\texttt{Scrip\_Body} = (\texttt{Vendor} \,||\, \texttt{Value} \,||\, \texttt{ID\#} \,||\, \texttt{Cust\_ID\#} \,||\, \texttt{Expires} \,||\, \texttt{Props})$$

und

$$\texttt{Cerificate} = \text{Hash}\{\texttt{Scrip\_Body} \,||\, \texttt{master\_scrip\_secret}\{\texttt{ID\#}\}\}$$

gilt. Vom Händler erzeugtes Scrip besteht also aus den eigentlichen Daten und der Signatur, die aus dem Hash der Daten und dem master_scrip_secret entsteht.

5.3.7 Scrip-Prüfung durch den Händler

Um eingehendes Scrip zu prüfen, ermittelt der Händler über `ID#` einfach das zugehörige MSS[6] (insofern mehrere existieren) und berechnet die Signatur neu. Wenn die beiden Signaturen identisch sind, wurde das Scrip vom Händler erzeugt. Zusätzlich wird noch das Ablaufdatum `Expires` geprüft. Ist das Scrip noch gültig, muß in der `ID#`-Datenbank gesucht werden, ob das Scrip zum ersten Mal eingereicht wird.

In regelmäßigen Abständen entfernt der Händler alle abgelaufenen `ID#`'s aus der Datenbank. Alle abgelaufenen `ID#`'s werden nicht mehr angenommen. Der Kunde ist dafür verantwortlich, Scrip erneuern zu lassen, das abzulaufen droht. Um zu verhindern, daß alle Kunden Scrip lagern, das eigentlich nicht mehr benötigt wird, kann der Händler für den Refresh-Vorgang eine minimale Gebühr nehmen.

5.3.8 Das `Props`-Feld

Die speziellen Informationen, die im `Props`-Feld gespeichert werden können, kann jeder Händler selbst definieren. Falls Broker mit eingebunden sind, müssen sich Broker und Händler einigen, was eingebunden werden soll und der Broker muß die korrekten Daten liefern.Über das Alter des Kunden kann der Händler beispielsweise entscheiden, an wen„Nur für Erwachsene“-Material geliefert werden soll.

5.3.9 Sicherheitslevel

Millicent bietet 3 Protokoll-Varianten. Scrip kann im Klartext ungeschützt über Netze transportiert werden, Scrip kann verschlüsselt und gesichert transportiert werden und als dritte Alternative kann Scrip gesichert, aber unverschlüsselt übertragen werden.

Scrip im Klartext:

Wenn Scrip 1:1 gesendet wird, kann ein Angreifer das gesendete oder auch das zurückgesendete Scrip selbst nutzen. Wenn der Kunde das Wechselgeld-Scrip dann später nutzen möchte, taucht es in der Double-Spending-Datenbank auf.

Verschlüsselt und abgesichert:

Der Kunde verschlüsselt das Scrip mit einer symmetrischen Chiffre. Beim ersten Scrip-Kauf wird dem Kunden zusammen mit dem Scrip ein geheimer Schlüssel geliefert. Dieser geheime Schlüssel muß auf einem sicheren Kanal übertragen werden. Dieser sichere Kanal kann entweder auf einem sicheren Protokoll außerhalb von Millicent etabliert werden oder er wird in einer schon bestehenden sicheren Millicent-Session übertragen.

[6] `master_scrip_secret`

Damit der Händler nicht alle Kundenschlüssel speichern muß, leitet er den Schlüssel aus der Kundenidentität `Cust_ID#` ab, z.B.

$$\text{Hash}\{\texttt{Cust\_ID\#} \,||\, \texttt{master\_customer\_secret}\{\texttt{Cust\_ID\#}\}\}$$

Wenn der Kunde das verschlüsselte Scrip transferriert, muß er `Cust_ID#` mitliefern, damit derHändler den Schlüssel neu berechnen kann.

Abgesichert, aber unverschlüsselt:

Da auch die symmetrische Verschlüsselung Rechenzeit schluckt und manchmal nicht notwendig ist, die Sicherheit der Authenzität aber trotzdem gewünscht ist, kann der Kunde ebenfalls eine digitale Signatur erzeugen. Dazu wird einfach der Hash über die Bestellung, das Scrip und den geheimen Kundenschlüssel `customer_secret` berechnet:

$$\texttt{ReqSig} = \text{Hash}\{\texttt{Request} \,||\, \texttt{Scrip} \,||\, \texttt{customer\_secret}\}$$

`ReqSig` wird einfach zusammen mit der Kundenanfrage an den Händler gesendet. Der bestimmt aus `Cust_ID#` das Kundengeheimnis neu und rekalkuliert die Signatur. Wenn die beiden übereinstimmen, kam der Auftrag vom Kunden.

5.3.10 Broker im Protokoll

Um Broker ins Protokoll mit einzubauen, gibt es drei Möglichkeiten:

- Im „Scrip-Warenhaus“-Modell kauft der Broker bei allen Händlern regulär Scrip. Wenn ein Kunde Scrip für den Händler benötigt, gibt der Händler dem Kunden das notwendige Scrip. Nach Beendigung aller Käufe liefert der Kunde dem Broker das Rest-Scrip, das der Broker beim Händler zurücktauscht.

  Die Effizienz liegt in der Tatsache, daß der Broker große Blöcke Scrip kaufen kann und so Rabatt bekommt.

- Bei der „lizensierten Scrip-Produktion“ liefert der Händler dem Broker ein `master_scrip_secret`, eine Serie von `ID#`’s (Seriennummern), ein `master_customer_secret` und eine Reihe von Kundennummern (`Cust_ID#`).

  Jetzt kann der Broker Scrip für seine Kunden produzieren. Damit hinterher korrekt abgerechnet werden kann, teilt der Händler mit jedem Broker andere Geheimnisse und Seriennummern sowie Kundennummern. Dadurch kann der Händler den Umsatz für jeden Broker aufsummieren.

  Durch diese Vorgehensweise wird der Händler entlastet, weil er nicht alles Scrip selbst produzieren muß, und der Broker muß nicht für jeden Händler Scrip lagern.

- Bei „mehreren Brokern“ fragt der Kunde seinen Broker nach Scrip eines bestimmten Händlers. Der Broker kontaktiert den Händler, um die Adresse des Händler-Brokers zu erfahren. Der Kunden-Broker kauft beim Händler-Broker Broker-eigenes Scrip, was der Kunden-Broker an seinen Kunden weiterreicht. Der Kunde kauft jetzt beim Händler-Broker Händler-Scrip im Tausch gegen das Broker-Scrip. Nach der ganzen Hin- und Hertauscherei kann der Kunde beim Händler kaufen.

5.3.11 Anonymität

Millicent läßt sich auf verschiedene Arten implementieren. Bei einem Kredit-basierten System erhält der Kunde am Monatsende eine Rechnung über die von ihm erzeugten Kosten. In diesem Modell ist Millicent zwischen Broker und Kunde nicht anonym, ob Anonymität gegenüber dem Händler besteht, hängt von den Daten innerhalb des `Props`-Feldes ab und wie die Bindung zwischen Kunde und `Cust_ID#` realisiert ist.

Wenn ein Debit-Modell realisiert wird, kauft der Kunde im Voraus Scrip beim Broker und ab diesem Zeitpunkt besteht keine Notwendigkeit für Kommunikation zwischen Broker und Kunde bzw. Broker und Händler. Auch hier ist die Frage, wie eng das Scrip auf den Kunden hindeutet.

Generell läßt sich sagen, daß Millicent nicht die Anonymität wie z.B. DigiCash’s Ecash bietet.

5.3.12 Teilbarkeit

Teilbarkeit ist bei Millicent nicht gegeben; das ist aber insofern nicht problematisch, weil man bei Scrip Wechsel-Scrip in sehr kleinen Einheiten bekommt. Die Eigenschaft Teilbarkeit im Sinne binärer Bäume ist hier aber nicht gegeben.

5.3.13 Plattformen und Bedienbarkeit

Da Millicent kein System ITL (In the real life) ist, kann zu unterstützten Plattformen keine Aussage getroffen werden. Prinzipiell läßt sich Millicent aber unter jeder Plattform implementieren.

Aus den gleichen Gründen läßt sich zur Bedienbarkeit keine Aussage treffen. Wenn Millicent beispielsweise in den Web-Browser mit eingebaut wird, entsteht ein einfach zu bedienendes System für Internet-Micrypayments. Es muß nur darauf geachtet werden, daß der Kunde denÜberblick über sein Scrip behalten kann.

5.3.14 Hardware / Software

Millicent ist ein Online-Zahlungssystem, denn die Interaktion mit Broker und Händler geschieht parallel bzw. ohne große Zeitverschiebungen. Wenn das System nur mit Kunden und Händlern betrieben werden würde, wäre eine SmartCard-Variante möglich, um beispielsweise im Copy-Shop Kopien zu verrechnen. Das Problem ist das Verfallsdatum von Scrip. Eine Karte, die nicht

regelmäßig genutzt wird, enthält irgendwann nur noch ungültiges Scrip.

5.3.15 Transaktionskosten

Die Transaktionskosten sind extrem gering, da die Zahlung zusammen mit der Lieferung erfolgt und die notwendigen kryptografischen Berechnungen sehr billig sind. Wenn Millicent für Micropayments genutzt wird und z.B. Ecash für die (echten) Zahlungen an den Broker oder Händler, wäre ein sehr billiges System erreichbar. Durch den Versuch, die Festplattenkapazität der Datenbanken so klein wie nötig zu halten, wird das System immer rentabler, je mehr Transaktionen erfolgen, weil weniger „Kartei-Leichen“ in der Double-Spending-Datenbank liegen.

5.3.16 Skalierbarkeit

Die Skalierbarkeit ist sehr gut, weil jeder Händler mehrere Broker haben kann, jeder Kunde mehrere Brokerdienste nutzen kann und keine zentrale Stelle für Sicherheitsaufgaben von Double-Spending-Prevention notwendig ist. Die Händlerdatenbank muß allerdings leistungsfähig sein, was aber linear mit dem Verkehr auf dem Web-Server ansteigt.

5.3.17 Ausblick

Mit Scrip kann derzeit noch nicht bezahlt werden. DIGITAL hat eine Beispiel-Implementation erstellt, bei der Bibliotheken zum einfachen Einbinden der Funktionen in eigene Applikationen mitgeliefert werden. Um Millicent zu testen, wurde ein Pseudo-Proxy für den Internet-Gebrauch beschrieben, der die Zahlung der bestellten Informationen übernimmt. Die Funktionalität kann relativ leicht in Browser übernommen werden.

Neben dem eigentlichen Bezahlen kann Millicent auch für Accounting-Maßnahmen in firmeneigenen Strukturen verwendet werden. Die oberere Werteschranke für Millicent liegt bei ein paar Mark, da sonst das beschriebene Trust-Modell nicht mehr zugrundeliegt. Die untere Schranke liegt in der Effizienz der Berechnung von Hash-Funktionen. Um beispielsweise für transportierte IP-Pakete im Internet-Backbone zu zahlen, dürfte sogar Millicent zu teuer sein.

Neben Millicent müßte in einem System auch ein zweites Zahlungssystem integriert sein, mit dem das Scrip beim Broker gezahlt werden kann und der sichere Transfer der Kundengeheimnisse möglich ist.

5.3.18 Literatur

[GMA+], [Lan98b], [Kos98], [SFE97, Seiten 70–72], [Way97, Seiten 217–228]

5.4 CyberCoin

Ein anderes, für Micropayments geeignetes Verfahren ist *CyberCoin*, ein von der Firma CyberCash `http://www.cybercash.com/` ebenfalls in die Wallet-Software von em CyberCash integriert wurde. Der Kunde hat hier also zwei Lösungen in einer Anwendung zusammen, wobei CyberCash für die Kreditkartenbasierten Zahlungen (Macropayments) gedacht ist und CyberCoin die Buchungen mit geringem Wert abwickelt. Unter Micropayments versteht CyberCash Buchungen zwischen 40 Pfennig und DM 20.

Wenn von der Wallet-Software die Rede ist, muß auch bei CyberCoin gesagt werden, daß es sich nicht um elektronische Münzen handelt, sondern Autorisierungen, vom Kundenkonto Geld auf das Händlerkonto zu überweisen, d.h., die Bank erzeugt keine Token, die als Geld gelten, sondern führt nur vom Kunden autorisierte Aufträge aus. Daher besteht bei CyberCoin der Bank gegenüber ebensowenig **Anonymität** wie bei CyberCash. Der Händler hingegen muß nicht erfahren, mit wem er ein Geschäft getätigt hat. Für ihn ist nur wichtig, daß die Zahlung in die Wege geleitet wurde.

5.4.1 Eine Transaktion

CyberCoin versucht, die Nutzung von Public-Key-Operationen zu minimieren, um eine vernünftige Performance und preiswerte Transaktionen zu ermöglichen. Um trotzdem Signaturen leisten zu können, werden symmetrische Signaturen mit geheimen Schlüsseln geleistet, ähnlich wie bei Millicent. Die Signaturen werden mit Hash-Funktionen oder symmetrischen Chiffren wie DES erzeugt. Der Kunde mit der Identität C_i und der CyberCoin-Transaktions-Server verfügen über zwei gemeinsame geheime Schlüssel D_i und S_i. Der D_i-Schlüssel hat nur eine eingeschränkte Gültigkeitsdauer (n Tage oder Transaktionen). Symmetrische Schlüssel für den DES werden von D_i abgeleitet. Digitale Signaturen werden mit *keyed*–MD5 erzeugt. Der Schlüssel für die digitalen Signaturen ist S_i.

1. Nachdem sich der Kunde mit dem Händler auf die zu bestellenden Waren geeinigt hat, sendet der Händler dem Kunden den Hash der Bestellung, der mit dem vom Kunden verechneten übereinstimmen muß: $H = \text{Hash}\{\text{data}\}$.

2. Der Kunde stellt eine Nachricht zusammen, in der die Kundenidentität C_i, der Warenwert der Bestellung, der Hash der Bestellung enthalten sind. An die Daten wird noch der geheime Schlüssel S_i angefügt und der Signatur-Hash berechnet:

   $$C_{Sig} = \text{Hash}\{C_i \,||\, \text{Wert} \,||\, \text{Hash}\{\text{data}\} \,||\, S_i\}$$

   Dann wird die Nachricht C_{msg1} mit dem „öffentlichen“ Teil der Bestellung und der Signatur an den Händler gesendet:

   $$C_{msg1} = (C_i \,||\, \text{Wert} \,||\, \text{Hash}\{\text{data}\} \,||\, C_{Sig})$$

3. Jetzt muß der Händler mit der Identität M_j die Nachricht ebenfalls signieren. Dazu hängt er an die Kundennachricht C_{msg1} noch seine Händler-Identität M_j und den geheimen Händler-Schlüssel S_j an, berechnet den Hash und sendet die Händlernachricht an die Bank.

   Die Signatur lautet dann

   $$M_{Sig} = \text{Hash}\left\{C_{msg1} \mid\mid M_j \mid\mid S_j\right\}$$

   und die Nachricht an die Bank lautet:

   $$M_{msg2} = (C_{msg1} \mid\mid M_j \mid\mid M_{Sig})$$

4. Aufgrund der Identitäten und der dem BankServer bekannten Geheimnisse S_i und S_j kann der Server die Signaturen von Kunde und Händler prüfen. Wenn das Geld auf dem Kundenkonto verfügbar ist, erzeugt die Bank eine Nachricht an den Händler, signiert sie mit S_j und sendet sie an den Händler.

5. Der Händler prüft die Signatur der Bank, indem er die den Hash-Wert mit seinem eigenen S_j rekalkuliert und startet die Lieferung an den Kunden.

6. Der Kunde erzeugt nach Erhalt der Lieferung einen Hash über die Ware $\text{Hash}\{\text{data}\}$ und signiert die Prüfsumme durch anfügen von S_i und berechnen des Hashes. Der Hash wird an den Händler geliefert.

7. Nachdem der Händler den Hash an den CyberCoin-Server weitergeleitet hat, transferriert CyberCoin das Geld vom Kundenkonto auf das Händlerkonto.

5.4.2 Kryptografie

Die von CyberCoin eingesetzte Kryptografie sind in der Hauptsache DES und MD5, wobei die Algorithmen auch gegen stärkere Verfahren ausgetauscht werden könnten.

5.4.3 Anonymität

Die Identitäten von Kunde und Händler müssen dem CyberCoin-Server bekannt sein, damit die Buchung korrekt abgewickelt werden kann. Die Identität des Kunden muß dem Händler nicht bekannt sein; der Händler erfährt nur die CyberCoin-Identifikation des Kunden, C_i.

5.4.4 Teilbarkeit

CyberCoin-Münzen sind Transaktionsaufträge, und somit wie Kreditkartenzahlen nicht teilbar. Der Wert kann jedoch individuell eingetragen werden.

5.4.5 Transaktionskosten

Wenn der Kunde in seiner Wallet CyberCoin-Münzen ordert, wird von seinem regulären Konto Geld auf das CyberCoin-Konto überwiesen. Daher werden die Transaktionskosten nicht mehr durch eine Kreditkartentransaktion definiert, sondern durch die Kosten, die auf dem CyberCoin-Server entstehen.

5.4.6 Verlust der Daten und Logging

Da auf der Festplatte des Kunden keine Token gespeichert werden, die den Wert in sich tragen, sondern nur ein Schlüssel, um Transaktionen zu authorisieren, ist der Verlust bei einem Festplatten-Crash geringer, da wie bei einer verlorenen Kreditkarte der Kunde nur bis max. 50 $ haftet. Im Fall eines Crash kann der Kunde die Wallet-Daten in Zusammenarbeit mit CyberCoin rekonstruieren.

Der große Anteil der auf Platte gespeicherten Daten ist das Transaktions-Log, das die Wallet mitführt.

5.4.7 Plattformen und Bedienbarkeit

Die Plattformen, die von CyberCoin unterstützt werden, sind die selben wie bei CyberCash, d.h., MS–Windows und MacOS. Die Bedienbarkeit ist auch hier die selbe wie bei CyberCoin, da beide Systeme im gleichen Software-Programm integriert sind.

5.4.8 Skalierbarkeit

Durch den Einsatz von zentralen Instanzen zur Kontenverwaltung und als Clearing-Stelle entsteht wieder die Problematik, daß die Last auf dem CyberCoin-Server nur schwer verteilt werden kann.

5.4.9 Software / Hardware

CyberCoin ist ein reines Software-Produkt. Da die Kundensoftware unter„unsicheren“ Betriebssystemen läuft, die keinen sicheren Speicherschutz gewährleisten, besteht immer die Gefahr, daß das Kundengeheimnis S_i von einem Virus oder einem unbedacht aus dem Internet heruntergeladenen und ausgeführten Programm ausgespäht wird, sei es, indem die Daten-Datei auf der Platte gelesen und an einen Angreifer weitergemailt wird oder daß eine aktuell laufende Version der Geldbörse ausgespäht wird.

5.4.10 Ausblick

Da der geheime Schlüssel nicht zwangsläufig im Rechner gespeichert werden muß, wäre eine Erweiterung denkbar, mit einem auf SmartCard gespeicherten Schlüssel die eigentliche Signatur außerhalb des Rechners ausführen zu lassen. Doch auch hier ist bei unsicheren Betriebssystemen die Problematik, daß die SmartCard u.U. andere Daten zu Signatur vorgelegt bekommt, als das Wallet auf dem Bildschirm anzeigt.

Zwar ist in der Botschaft C_{Sig} der Händler implizit durch den Hash der Bestellung festgelegt, allerdings wäre meines Erachtens nach die Einbindung der Händleridentität in die signierte Kundenbotschaft eleganter, da abgefangene und von Angreifern eingereichte Botschaften beim Server früher auffallen würden.

5.4.11 Literatur

[Way97, Seiten 155–158], [Sto97, Seiten 71–71], `http://www.cybercash.com/`, [Sie97a, Seiten 112–124]

5.5 PayWord

In [RS96][7] beschreiben RIVEST und SHAMIR, zwei der Namensgeber von RSA, zwei Micropayment-Systeme, *PayWord* und *MicroMint*. In diesem Abschnitt soll PayWord beschrieben werden. Unabhängig von Rivest und Shamir wurden sehr ähnliche Systeme von ROSS ANDERSON, HARRY MANIFAVAS und CHRIS SUTHERLAND in Cambridge [AMS95] und THORBEN P. PEDERSEN in Aarhus [Ped96] entwickelt.

Die Grundlage für PayWord bildet nicht wie bei vielen anderen Systemen die digitale Signatur, sondern die intensive Nutzung von kryptografisch sicheren Hashfunktionen. Ziel der Entwicklung war es, die extrem rechenaufwendigen und damit kostenintensiven RSA-Operationen (Signatur und Verifikation) zu minimieren, aus Hashfunktionen zurückzugreifen, wann immer möglich, und so ein schnelles und kostengünstiges System zu bauen. Public-Key-Signaturen dauern ca. 10.000 mal solange wie ein Hash, Verifikationen noch ca. 100 mal länger als eine Hash-Funktion. Im Schnitt schafft eine normale Workstation 1-2 Signaturen oder 200 Verifikationen oder die Berechnung von 20.000 Hashwerten pro Sekunde.

Speziell im Micropayment-Bereich fallen in sehr kurzen Intervallen viele Zahlungen mit extrem niedrigen Beträgen an. Wenn dann zur Überprüfung der Gültigkeit einer Zahlung teure (weil komplizierte) RSA-Verifikationen durchgeführt werden müssen, sind solche Zahlungen für Banken nicht mehr lohnend, da die Kosten für die Hardware den Umsatz weit übersteigen. Das gleiche gilt auch für Händler, die die Gültigkeit der Münzen prüfen müssen.

5.5.1 Das Grundprinzip

Um Hashfunktionen in diesem Protokoll nutzen zu können, wird ein Schema verwendet, welches stark an LAMPORT (Password authentication with insecure communication) und HALLER (The S/Key one-time password system) angelehnt ist. Um eine Authentisierung durchzuführen wird eine eindeutig festgelegte Folge von Werten berechnet. Wenn der Wert w_{i+1} vorgelegt wird, muß er in eindeutiger und nicht fälschbarer Weise vom vorausgegangenen Wert w_i abhängen. Diese Bedingung kann einfach durch Hashfunktionen realisiert werden. Wenn

$$w_i = h(w_{i+1})$$

gilt, wobei $h(\cdot)$ die Hashfunktion bedeutet, kann der Verifizierer durch anwenden von $h()$ die Echtheit von w_{i+1} prüfen, wenn w_i als echt vorausgesetzt wird. Das Protokoll arbeitet nun wie folgt:

1. Der Kunde muß eine sog. *Hash-Kette*[8] berechnen. Dazu wählt der Kunde einen zufälligen Startwert w_n, welcher die gleiche Bitzahl wir der Output von $h()$ hat. Dann berechnet er

[7] `http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.ps`
[8] engl: hash chain/payword chain

$w_{n-1} = h(w_n)$, $w_{n-2} = h(w_{n-1})$, usw. bis er zu $w_0 = h(w_1)$ gelangt. Er merkt sich alle w_i mit $i \in (0, \ldots, n)$ in einer Datenbank.

2. Der Wert w_0 wird nun auf einem (noch zu erläuternden) sicheren Weg zum Händler gebracht.

3. Um die Authenzität der ersten Zahlung zu beweisen, schickt der Kunde w_1 an den Händler.

4. Der Händler prüft, ob $h(w_1) = w_0$ gilt. Aufgrund der Unumkehrbarkeit der Hashfunktion kann nur der Erzeuger von w_1 auch den Wert w_0 berechnet haben. Da w_0 aber garantiert vom Kunden kommt, muß w_1 auch vom Kunden sein.

Um nun daraus ein funktionsfähiges Zahlungssystem zu bauen, müßen noch Wege zur sicheren Übertragung von w_0 und zur Entgeltverrechnung definiert werden.

Im nachfolgenden halte ich mich eng an [RS96]: Die öffentlichen Schlüssel der Bank[9] B, des Kunden K und des Händlers H seien PK_B, PK_K und PK_H. Deren geheime Schlüssel heißen analog dazu SK_B, SK_K und SK_H. Eine digital signierte Nachricht M sei als $\{M\}_{SK}$ bezeichnet. $h(\cdot)$ sei eine kryptografisch sichere Hashfunktion wie SHA oder RIPEM160 mit 160 bit Hash-Länge. Wichtig ist nur die Einweg-Eigenschaft und Kollisionsfreiheit.

5.5.2 Die Initialisierungsphase

Bank–Kunden-Beziehung

Um PayWord zu nutzen, richtet der Kunde K bei der Bank B einen Account ein. Die Bank stellt dem Kunden ein digital signiertes Zertifikat C_K mit folgenden Daten aus:

- Dem Namen der ausgebenden Bank B,
- dem Namen des Kunden K,
- der IP-Adresse oder sonstige Lieferadresse des Kunden A_K,
- dem öffentlichen Schlüssel PK_K des Kunden,
- dem Gültigeitszeitraum E des Zertifikates und
- weiterer Information, beispielsweise dem Wert einer Zahlung, z.B. 1 Pfennig.

Das Zertifikat hat also die Form $C_K = \{B, K, A_K, PK_K, E, \ldots\}_{PK_B}$. Da der Name des Kunden und seine IP-Adresse im Zertifikat eingebunden sind wird deutlich, daß der Kunde dem Händler gegenüber nicht anonym auftreten kann. Der öffentliche Schlüssel des Kunden wird vom Händler für die Initialisierungsphase benötigt und mit der Gültigkeitsdauer E (Expiration date) schützt

[9] im Original als Broker bezeichnet

die Bank sich vor zu großen Verlusten durch Kunden, die ihre Schulden bei der Bank nicht begleichen, indem kein neues Zertifikat ausgegeben wird. Der Händler kann die Echtheit des Zertifikates mit dem öffentlichen Schlüssel der Bank prüfen.

5.5.3 Eine Transaktion

Händler–Kunden-Beziehung

Wenn der Kunde sich entschließt, bei einem Händler ein Produkt (Webseite, etc.) zu ordern, erzeugt der Kunde auf die oben beschriebene Art und Weise eine Hash-Kette. Diese Hash-Kette gilt nur für den gewählten Händler und auch nur für den aktuellen Tag. Um zu wissen, wie lange die Hash-Kette werden muß, also wie groß n ist, muß der Kunde eine Schätzung treffen, wieviel Ware geordert werden soll[10]. Um die Informationen zum Händler zu schicken, generiert der Kunde eine Zahlungszusage M. Diese Zusage hat die Form $M = \{H, C_K, w_0, Datum, I_{misc}\}_{SK_K}$, wobei folgendes gilt:

- H ist die Identität des Händlers. Damit wird verhindert, daß ein anderer Händler abgehörte Zertifikate nutzen kann.
- C_K bestätigt dem Händler die Echtheit und Glaubwürdigkeit der Signatur SK_K.
- w_0 ist die *Wurzel*, der Startwert der Hash-Kette.
- Das *Datum* schränkt den Gültigkeitszeitraum der Hash-Kette auf den aktuellen Tag ein.
- Die zusätzliche Information I_{misc} kann z.B. die Länge n der Hash-Kette beinhalten.

Diese Zahlungszusage M erlaubt es der Bank B, dem Händler H für eingereichte Werte aus der Hash-Kette $w_1, \ldots, w_n$ Geld gutzuschreiben. Diese Hash-Kette ist somit nur für exakt einen Händler, einen Kunden und einen Tag gültig.

Zahlungen

Wenn der Kunde nun zahlen möchte, sendet er dem Händler den nächsten Wert w_i aus der Hash-Kette zusammen mit dem aktuellen Index i, d.h, eine Zahlung P von K an H hat die Form

$$P_i = (w_i, i)$$

Vom Bandbreitenbedarf her ist so eine Zahlung sehr ökonomisch, zwischen 20 und 30 Bytes Länge. Der Berechnungsaufwand ist (abgesehen von der Decodierung der Zahlungszusage M) einfach die Prüfung, ob $w_{i-1} = h(w_i)$ gilt, also eine Hash-Berechnung. Die Zahlung P_i muß nicht

[10] Wenn die Hash-Kette zu kurz war, kann der Kunde einen neuen Vorgang hinzufügen.

mehr digital signiert werden (wie z.B. bei Digicash), weil w_i die Authentifizierung implizit in sich trägt.

Alle Zahlungen haben eine feste Größe, d.h., wenn der Index i bei einer neuen Zahlung nur um 1 imkrementiert wurde, ist die Zahlung genau 1 Pfennig wert. Um größere Zahlungen (N Pfennig) zu tätigen, müssen nicht N P_i-Werte transferriert werden. Es reicht, einfach den Index i um N zu erhöhen und dann den neuen Hash-Wert zu übertragen. Der Händler muß nun N Hash-Operationen durchführen, um die Gültigkeit von w_{i+N} zu prüfen. Somit wird offenbar, daß der Berechnungsaufwand linear ($O\{n\}$) mit dem Wert der Zahlungen ansteigt.

Händler–Bank-Beziehung

Der Händler muß nicht die komplette Hash-Kette speichern, sondern nur die Zahlungszusage M und die letzte Zahlung P_i (Der Rest der Kette liegt ja zwischen w_i und w_0, was beides bekannt ist.

Um sich das Geld gutschreiben zu lassen, überträgt der Händler am Ende eines Tages die Werte von M und P_i an die Bank. Die Bank prüft die Echtheit ihres eigenen Zertifikates C_K, die Echtheit von M, ob alle in M gegebenen Parameter (Händleridentität, Datum, etc.) gültig sind und berechnet die Hash-Kette von w_i bis zurück nach w_0. Wenn alles stimmt, kann das Geld dem Kunden in Rechnung gestellt und dem Händler gutgeschrieben werden.

5.5.4 Anonymität

Da in den Zertifikation C_K und M die Identität des Kunden und des Händlers festgelegt werden, kann die Bank nachvollziehen, wer mit wem Geschäftsbeziehungen unterhält. Da aber der Händler der Bank nicht die vom Kunden angeforderten Inhalte mitteilt, existiert ein gewisser Schutz des Kunden. Die Bank weiß nicht, was der Kunde bestellt hat; der Händler hat die komplette Information der Vorgänge.

5.5.5 Plattformen

Bei PayWord handelt es sich nicht um ein realisiertes Zahlungssystem, daher existieren keine Implementierungen.

5.5.6 Kryptografie

Im beschriebenen Protokoll wird RSA für die digitalen Zertifikate vorgeschlagen. Dieser Algorithmus ist gegen jeden anderen Public-Key-Algorithmus austauschbar. Die Hashfunktion muß kryptografisch sicher sein. Als geeignete Kandidaten kann man sich hier den SHA oder auch RIPEM160 vorstellen.

5.5.7 Teilbarkeit

Die Paywords sind nicht teilbar. Man kann nur im Vorfeld der Transaktionen den Wert eines einzelnen Paywords P_i festlegen, um auf die benötigte Größe herunterzukommen. Eine neue Größe von Zahlungen kann also in C_K oder in M festgelegt werden.

5.5.8 Skalierbarkeit

Das Protokoll besitzt keine unvermeidbaren Engpässe. Eine Bank kann abhängig vom erwarteten Volumen verschiedene Server bereitstellen, die in einem Cluster zusammengeschaltet werden können. Aufgrund der wenigen Public-Key-Operationen ist das Verfahren aber sowieso relativ ressourcenschonend.

5.5.9 Effizienz

Die Bank muß in regelmäßigen Intervallen neue Kundenzertifikate C_K erstellen. Das kann offline geschehen. Für jedes Zertifikat ist eine Public-Key-Operation notwendig.

Der Kunde muß täglich für jeden zu nutzenden Händler eine neue Zahlungszusage M erzeugen. Hier ist ebenfalls eine Public-Key-Operation notwendig. Der Berechnungsaufwand für die Hash-Ketten ist vernachlässigbar.

Der Händler muß (online) jede Zahlungszusage M und das eingekapselte Kundenzertifikat C_K (zwei Public-Key-Verifikationen) prüfen.

5.5.10 Kleinhändler

Da keine komplizierte Infrastruktur notwendig ist, können auch Kleinhändler das System ohne Probleme nutzen.

5.5.11 Betrugsrisiken

Die erste Schwierigkeit tritt bei der Frage nach der Zahlungsreihenfolge auf („Erst die Ware, dann das Geld?“). Wenn der Kunde zuerst zahlt, kann der Händler die Lieferung verweigern; wenn der Händler zuerst liefert, kann der Kunde die Zahlung verweigern. Da das System hauptsächlich für Micropayments gedacht ist, kann der (betrogene) Partner im Schadensfall einfach die Kommunikation beenden. Er hat keinen Beweis über nicht erfolgte Zahlungen.

5.5.12 Transaktionskosten

Die Transaktionskosten sind minimal. Zu Beginn der Kommunikation muß M übertragen werden, bei jeder Zahlung P_i; da es sich hierbei um wenige Bytes handelt, kann der Aufwand vernachlässigt werden. Da der Händler erst nach Geschäftsschluß zwecks Gutschrift Kontakt mit

der Bank aufnimmt und hier alle Zahlungstupel gesammelt schicken kann, sich auch hier nur minimale Kosten zu erwarten.

5.5.13 Literatur

Hier sei das Originalpaper von RIVEST und SHAMIR [RS96] erwähnt ebenso [AMS95] und [Ped96].

5.6 MicroMint

In [RS96][11] beschreiben RIVEST und SHAMIR, neben *PayWord* auch noch *MicroMint*. MicroMint ist ein MicroPayment-System.

Mint im englischen ist die Münzprägeanstalt. Die Sicherheit bei einer normalen Metall-Münze wird durch den Produktionsvorgang gewährleistet. Die Münze an sich ist extrem billig. Die Ausrüstung, die benötigt wird, um Münzen zu prägen, ist jedoch extrem teuer und schwer zu beschaffen. Die Investitionskosten für eine Münzprägung sind also sehr hoch, aber mit jeder geprägten Münze wird der Prozess rentabler.

MicroMint beschreitet den selben Weg. MicroMint ist ein System, das zur Erzeugung von digitalen Münzen extreme Anschaffungen an Hardware benötigt. Sind die Anschaffungen getätigt, beginnt der Prägevorgang, der ebenfalls der langwierig ist. Die erste Münze ist extrem schwer zu generieren. Mit jeder weiteren Münze verkürzt sich die Zeitspanne bis zur nächsten gültigen Münze. Daraus leitet sich ab, daß die Münzproduktion nur im großen Maßstab ökonomisch sinnvoll ist.

Um die Anschaffung der Hardware für Angreifer nicht doch noch retabel zu machen, veröffentlicht die Bank jeden Monat neue Münzen. Die Gültigkeit der Münzen endet am Monatsende. Kunden können ihre nicht verbrauchten Münzen bei der Bank zurückgeben und gegen neue eintauschen. Händler können die eingenommenen Münzen entweder am Ende eines Buchungstages oder auch am Monatsende zurückgeben.

MicroMint-Münzen müssen einfach auf Gültigkeit prüfbar sein. Ein Design-Ziel war aber die Vermeidung von Public-Key-Operationen. Als einfach zu berechnende, aber schwer umkehrbare Prozesse blieben den Autoren die Hashfunktionen. Die Grundidee hinter MicroMint liegt in der Erzeugung von Kollisionen einer kryptografisch sicheren Hash-Funktion.

MicroMint-Münzen werden durch Kollisionen einer Hash-Funktion dargestellt, d.h., eine Münze besteht aus zwei oder mehreren Werten, die den gleichen Hashwert haben. Eine Hashfunktion h bildet m bit lange Zeicheketten x auf n bit lange Zeichenketten y ab, d.h., $y = h(x)$. Ein Paar von m-bit-Strings (x_1, x_2) wird eine *2-Weg-Kollision* genannt, wenn $y = h(x_1) = h(x_2)$ gilt, wobei y ein n-bit-Vektor ist.

Wenn die h wirklich zufällig und unvorhersagbar arbeitet, müssen im Schnitt $\sqrt{2^n} = 2^{n/2}$ x-Werte

[11] `http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.ps`

gehasht werden und in den Ergebnissen nach gleichen Werten gesucht werden. Der Grund dafür liegt im Geburtstagsparadoxon. Um eine Kollision in einem n-bit-Hashwert zu erzielen, sind also $2^{n/2}$ Versuche nötig. Hat man diese Schwelle jedoch überschritten, werden die nachfolgenden Kollisionen immer schneller gefunden. Wenn die Anzahl der Versuche um den Faktor c gesteigert wird, lassen sich c^2 Kollisionen finden:

> Hashing c times as many x-values as are needed to produce the first collision results in approximately c^2 as many collisions, for $1 \leq c \leq 2^{n/2}$, so producing collisions can be done increasingly efficiently per coin generated, once the threshold for finding collisions has been passed.

Wenn 2-Weg-Kollisionen gesucht werden, muß der Wert für n relativ hoch angesetzt werden. Um den Wert für n kleiner zu wählen und um den Aufwand für Fälscher noch ein bißchen in die Höhe zu treiben, werden in MicroMint keine 2-Weg-Kollisionen gefordert, sondern k-Weg-Kollisionen. Eine k-Weg-Kollisionsmünze besteht aus einem k-Tupel von x-Werten, die alle den gleichen Hashwert haben: $(x_1, x_2, x_3, \ldots, x_k)$.

Bis man die erste k-Kollosion gefunden hat, müssen ca. $2^{n(k-1)/k}$ Werte untersucht werden. Wenn c mal soviele x-Werte durchsucht werden, wobei $1 \leq c \leq 2^{n/k}$ gelten muß, findet man c^k k-Weg-Kollisionen. Die Wahl eines $k > 2$ hat zwei Effekte: zuerst einmal wird die Zeitspanne verlängert, bis die erste Kollision gefunden wird, auf der anderen Seite steigt die Kollisionsrate viel stärker an, wenn die Schwelle erst einmal überschritten ist.

Um nicht immer von Kollisionen, x-Werten und y-Werten sprechen zu müssen, führen RIVEST und SHAMIR den Vergleich mit Bällen und Körben ein. Ein x-Wert entspricht einem Ball und ein y-Wert ist ein nummerierter Korb. Wenn also der zu einem x-Wert gehörige y-Wert berechnet wird, entspricht das dem Wurf des Balles in einen Korb. In welchem Korb der Ball landet, kann vorher nicht bestimmt werden, da die Hash-Funktion ja unvorhersagbar ist.

Um also gültige Münzen zu produzieren, stellt die Bank 2^n Körbe auf und wirft ca. $k \cdot 2^n$ Bälle. Dann werden alle Körbe, in denen k oder mehr Bälle gelandet sind, aussortiert. Diese Bälle bilden jeweils eine Münze. Mit diesen gewählten Parametern gehört jeder Ball mit einer Wahrscheinlichkeit von 0,5 zu einer gültigen Münze.

Wenn mehr als k Bälle in einem Korb gelandet sind, werden die überschüssigen Bälle verworfen. Natürlich könnte man auch aus einem Korb durch Kombinationen verschiedene gültige Münzen machen. Das Problem beginnt, wenn ein Kunde zwei Münzen bekommtm die aus dem selben Korb stammen; dann kann er durch Kombinieren mehr Münzen erzeugen, als ihm eigentlich zustehen. Die Autoren schlagen vor, aus jedem Korb nur eine Münze zu erzeugen, um dieses Problem zu umgehen. Durch diese Vereinfachung wird die Double-Spending-Datenbank der Bank viel einfacher, weil jetzt nur noch 2^n bit benötigt werden, um alle eingezahlten Münzen nachzuhalten.

Das Problem bei dieser Art der Münzerzeugung ist der vefügbare Speicher. Da man während des „Körbewerfens“ die Ergebnisse speichern muß, sprengt das Verfahren alle Festplatten und Speichermedien. Um die Auswahl zu vereinfachen, werden gewisse Bälle aussortiert, wenn sie

nicht ein einer bestimmten Korbsorte gelandet sind, d.h., wenn der Hashwert nicht gewisse Bedingungen erfüllt. Die Bitlänge n des Ergebnisses, also die Ausgangsbitbreite der Hashfunktion, wird in t High-Order-bits und u Low-Order-Bits aufgeteilt, wobei natürlich $n = t + u$ gelten muß. Jeden Monat wird ein anderes Kriterium festgelegt, das für die oberen t bits gelten muß. Wenn ein x-Wert also nicht auf einen y-Wert matcht, der das gewählte Kriterium erfüllt, wird er einfach weggeworfen. Durch geeignete Wahl von t kann die Bank die Münzerzeugung an ihre eigenen Erfordernisse anpassen, d.h., an ihren Speicherplatz und Rechenpower. Durch diese Maßnahme wird die Anzahl der gültigen Körbe um den Faktor 2^t herabgesetzt, während der notwendige Speicherplatz nur noch ein Vielfaches der gewünschten Münzanzahl ist.

Durch die Möglichkeit, die oberen t bit festzulegen, ist auch ein Schema denkbar, bei dem die Bank 31 boolsche Funktionen definiert, denen die oberen t bit genügen müssen. Am ersten Tag des Monats veröffentlicht die Bank die erste Funktion. Damit können Kunden und Händler prüfen, daß ihre Münzen gültig sind. Mit jedem Tag wird eine weitere Funktion veröffentlicht, denen die High-Order-bits genügen müssen. Da die Bank die Kriterien vor Ausgabe der Münzen intern festgelegt hat, erfüllen alle echten Münzen die Kriterien. Falschgeld erfüllt diese Bedingungen nur in seltenen Fällen, da die Fälscher zwar Kollisionen finden konnten, aber nicht wissen, wie die oberen bits definiert sind.

5.6.1 Kosten

In [RS96] wird ausführlich durchgerechnet, welcher Aufwand getrieben werden muß, damit sich eine Münzprägung lohnt. Eine breite Einführung vorausgesetzt, kann sich ein solches System innerhalb eines Monats amortisieren. Die aufgeführten Parameter sind ein Gewinn von 1 Millionen US$ pro Monat, wobei die Bank 10% des Umsatzes einbehält. Für diese Geldmenge müssen 2^{33} Hashoperationen durchgeführt werden, wobei $k = 4$ und $u = 31$ gewählt wurde. Aus der Berechnung ergeben sich 2^{30} gültige Münzen. Der benötigte Speicherplatz beläuft sich auf 128 Gigabytes Festplattenplatz, was zusammen mit FPGA's für schnelle Hash-Berechnung ca. 150.000 US$ Hardwarekosten bedeutet. Das System braucht 500.000 Benutzer, die monatlich einen Umsatz von jeweils 25 US$ machen.

5.6.2 Anonymität

Im Paper werden noch verschiedene Möglichkeiten beschrieben, die Münzen User-spezifisch zu machen, d.h., eine bestimmte Münze kann dann einem Kunden oder einer Kundengruppe zugeordnet werden. Ebenso können händlerspezifische Münzen erzeugt werden, die nur bei einem speziellen Händler eingereicht werden können.

5.6.3 Ausblick

Zum Abschluß ist zu sagen, daß MicroMint ein Verfahren mit sehr interessanten Ansätzen ist. MicroMint ist noch nicht realisiert worden. Durch den Aufbau können Münzen so gut wie gar

nicht gefälscht werden, und auch Betrüger können je nach verwendeter„Geschmacksrichtung“ des Verfahrens auch erkannt werden.

6 Kundenkonten

6.1 Netcheque

Genau wie NetCash wurde *NetCheque* am „Information Science Institute“ der „University of Southern California“ unter der Leitung von CLIFFORD NEUMANN entwickelt. NetCheque überträgt die Funktionalität von Verrechnungsschecks ins Internet. Dabei wird zur Absicherung ein auf KERBEROS [Sch98, NS78, SNS88] basierendes Schlüsselverteilungs- und Linkstrecken-Sicherungssystem verwendet.

Ein NetCheque-Scheck *C* enthält Informationen Geldbetrag, die verwendete Währung, den Namen des Händlers, den Namen der Kunden-Bank, die Kunden-Kontonummer, eine Seriennummer für den Scheck und andere Informationen.

Ein Kunde, der den Scheck an den Händler schicken möchte, fragt beim Kreberos-Server nach einem Ticket für die Bank. Dieses Ticket enthält einen für den Kunden verschlüsselten Session-Key $K_{Kunde,Bank}$, den der Kunde für die Kommunikation mit der Bank verwenden kann sowie ein Paket, das nur die Bank entschlüsseln kann, in dem ebenfalls der Session-Key enthalten ist. Um eine Scheck-Signatur für die Bank zu erzeugen, berechnet er den Hash aus dem Scheck und verschlüsselt den Wert mit dem Session-Key.

Bevor der Händler die vom Kunden erhaltenen Daten (Scheck, Ticket und verschlüsselte Signatur) bei der Bank einreicht, fügt er noch seine Kontodaten hinzu. Wenn der Händler die Daten weiterreicht, kann die Bank sicher sein, daß die Signatur vom Kunde erzeugt wurde. NetCheque bietet zusätzlich die Möglichkeit, auch den Händler mit dem Kerberos-System unterschreiben zu lassen, d.h., auch der Händler besorgt sich ein Ticket für die Bank und signiert den Scheck und seine Kontodaten.

Wenn die Kundenbank und die Händlerbank unterschiedlich sind, wird ein Geldtransfer über bankeninterne Netze eingeleitet.

6.1.1 Skalierbarkeit

Die Leistungfähigkeit des Systems steht und fällt mit der Verfügbarkeit des Kerberos-Servers. Immer wenn ein Ticket erstellt werden soll, muß der„zentralistische Höllenhund“ Kerberos[1] die Arbeit leisten. Die Banken können ihre Last verteilen, aber der Ticket-Service stellt das Nadelör dar.

[1] Zerberus – Torwächter der Hölle in der griechischen Mythologie

6.1.2 Plattformen

Zur Zeit hat das ISI (Institute ...) verschiedene Testversionen der Software für Unix-Workstations (Solaris und HP/UX) freigegeben. NetCheque wird außer in Tests noch nicht eingesetzt.

6.1.3 Ausblick

Durch die Nutzung des Kerberos-Systems wird auf eine wohlbekannte Infrastruktur zurüchgegriffen, die den Vorteil hat, gut durchdacht und untersucht zu sein. Aufgrund der kurzen Verfallszeiten von Kerberos-Tickets besteht aber die Notwendigkeit, die Daten schnell weiterzuleiten, da die Tickets sonst ungültig werden. Das andere Problem mit Kerberos liegt in der Schwierigkeit, daß Ticket-Signaturen nur von der Bank nachvollzogen werden können. Eine Dritte Person kann die Signatur nicht prüfen, da sonst die Geheimschlüssel zwischen Bank und Kerberos-Server bzw. zwischen Kunde und Kerberos-Server aufgedeckt werden müßten. Vorteil des Kerberos-Systems ist die Möglichkeit, auf teuer zu lizensierende Public-Key-Verfahren zu verzichten und schnelle, symmetrische Chiffren einzusetzen.

NetCheque ist kein komplett definiertes und spezifiziertes System, es fehlt eine Standardisierung und Akzeptanz durch die Banken.

6.1.4 Literatur

[SFE97, Seiten 55–57], [Way97, Seiten 131–139], [Sto97, Seiten 60–63]

6.2 NetBill

Netbill (`http://www.netbill.com/`) wurde an der Carnegie-Mellon University in Pittsburgh, Pennsylvania, in Zusammenarbeit mit VISA International entwickelt. Der interessanteste Punkt an diesem auf Kundenkonten basierenden System ist die verschlüsselte Lieferung der Ware, bevor die Zahlung erfolgt. Das System ist für die Bezahlung und den Verkauf von Informationen im Web konzipiert. Der Händler liefert dem Kunden nach einem Kaufauftrag die bestellte Ware; allerdings sind die gelieferten Informationen symmetrisch verschlüsselt und der Schlüssel wird erst nach erfolgter Bezahlung geliefert. Durch diesen Ansatz wird das Risiko zwischen Kunde und Händler gerechter aufgeteilt: Der Kunde hat schon soetwas wie die Ware in den Händen, der Händler kann aber sicher sein, daß der Kunde noch bezahlt.

Um NetBill zu nutzen, richten Kunde und Händler ein Kundenkonto beim NetBill-Server ein. Die Kundensoftware heißt im NetBill-Jargon *checkbook*, die Software des zentralen NetBill-Servers heißt *till*.

6.2.1 Eine Transaktion

1. Der Kunde handelt mit dem Händler die Bedingungen für den Kauf von digitalen Service-Angeboten bzw. digitalisierten Waren aus. Bei Bedarf schickt der Händler dem Kunden ein digital signiertes Angebot bezüglich der Waren zu, auf das sich der Kunde bei der Bestellung beziehen kann. Dadurch wird dem Händler die Möglichkeit gegeben, bei speziellen Kunden individuelle Preise zu machen.

   Um die Preisanfrage zu starten, kontaktiert die Kundensoftware den NetBill-Server, welcher die Anfrage an den Händler weiterleitet.

2. Die Händlersoftware generiert das Preisangebot, signiert es und leitet es über den NetBill-Server an den Kunden weiter. Wenn der Kunde sich entscheidet, zu kaufen, erzeugt die Kundensoftware eine digital signierte Kaufzusage und schickt sie an den Server.

3. Nachdem die Händlersoftware die Anfrage durch den Server zugestellt bekommen hat, verschlüsselt sie die Ware mit einem zufällig gewählten Schlüssel und schickt die verschlüsselte Ware samt Schlüssel an den NetBill-Server.

4. Der NetBill-Server berechnet den Hash der verschlüsselten Ware und leitet die Ware an den Kunde weiter.

5. Nachdem der Kunde die verschlüsselte Ware erhalten hat, berechnet er ebenfalls den Hash-Wert und schickt ihn zusammen mit einem Verfallsdatum für den Scheck, einer Beschreibung der Bestellung und dem letztlich akzeptierten Preis digital signiert zum NetBill-Server.

   Diese Nachricht ist die „*electronic purchase order*“, die die endgültige Zahlungszusage des Kunden darstellt.

6. Der NetBill-Server prüft die Signatur des Kunden und stellt sicher, daß die beiden Hash-Werte identisch sind, d.h., daß die Ware unbeschadet beim Kunden angekommen ist.

   Danach wird die eigentliche Zahlung initiiert, d.h., das Geld wird vom Kundenkonto auf das Händlerkonto transferriert und der Händler bekommt eine Zahlungsbestätigung vom Server.

7. Die letzte Händlernachricht an den Server enthält noch einmal den endgültigen Preis und den symmetrischen Schlüssel, mit dem der Kunde an die Ware gelangt. Der Händler signiert die Nachricht.

8. Der NetBill-Server prüft, ob der Betrag korrekt ist und leitet den symmetrischen Schlüssel an den Kunden weiter, der die Daten entschlüsselt und nutzt.

Durch die Rücklieferung des Hashwertes über die verschlüsselte Ware kann der Käufer den Empfang der Ware nicht abstreiten.

6.2.2 Kryptografie

Das System basiert im wesentlichen auf RSA für Signaturen und Kerberos für den Austausch von symmetrischen Schlüsseln.

6.2.3 Anonymität

Da die ganze Kommunikation über den NetBill-Server läuft, besteht keine Notwendigkeit für den Händler, die Identität des Kunden zu erfahren (es sei denn, um spezielle Angebote zu machen). Für den NetBill-Server ist der gesamte Geschäftsvorgang offenkundig, da alle Transaktionen über den Server laufen.

6.2.4 Skalierbarkeit

Da der Server bei allen Aktionen involviert ist, muß der Server extrem leistungsfähig sein. Speziell der Mittler-Prozess, bei dem der Server die verschlüsselte Ware an den Kunden weiterleitet, führt zu einer starken Belastung des Systems.

6.2.5 Ausblick

Die NetBill-Entwicklungsgruppe hat Händler-Software für UNIX-Workstations (SUN Solaris) entwickelt und versucht, `x-netbill` als neuen MIME-Typ in den HTTP-Spezifikationen unterzubringen.

6.2.6 Literatur

[Sto97, Seiten 75–78], [Way97, Seiten 249–252]

6.3 First Virtual

First Virtual (`http://www.fv.com/`) ist ein Kreditkarten-basiertes System, das Zahlungen über SMTP/S-MIME[2] abwickelt. Vom kryptografischen Standpunkt her gesehen ist First Virtual ausgesprochen uninteressant, da FV keine Kryptografie einsetzt. Die Sicherheit von FV basiert auf der Reihenfolge der Protokollschritte und auf der Sicherheit einer e-mail-Zustellung.

[2] Simple Mail Transfer Protocol / Secure Multipurpose Internet Mail Extensions, siehe RFC822 bzw. RFC1341

6.3.1 Das Zahlungsprotokoll

Der Kunde und der Händler eröffnen bei FV einen Account. Der Kunde teilt FV auf„sicherem“ Weg (wie z.B. Telefon) seine Kreditkartennummer mit. FV teilt dem Kunden eine individuelle PIN, die FV-PIN zu. Diese PIN wird anstelle der Kreditkartennummer zusammen mit der Bestellung an den Händler weitergereicht. Der Händler leitet die Bestellung und die FV-PIN an den FV-Server weiter. Dort wird die Zahlungsfähigkeit des Kunden geprüft. Der FV-Server sendet dem Kunden eine e-mail, in der er den Kunden um Authorisierung der Transaktion bittet. Der Kunde kann mit Zustimmung, Ablehnung oder der Betrugsnachricht antworten. Bei Zustimmung belastet FV das Kreditkartenkonto des Kunden und gibt dem Händler die Erlaubnis, den Warentransfer zu beginnen. Wenn der Kunde die Zahlung ablehnt, teilt FV dem Händler die Ablehnung mit. Wenn der Kunde sagt, er habe die Bestellung nicht initiiert, ist es an FV, die Ermittlungen aufzunehmen.

6.3.2 Kryptografie

First Virtual nutzt keine Kryptografie.

6.3.3 Sicherheit

Die Sicherheit basiert auf der Annahme, daß e-mail’s nicht abgefangen werden können. Es ist unter SMTP vollkommen einfach, eine e-mail mit falschen Daten ins Netz einzuspeisen, da SMTP keine kryptografischen Sicherungen beinhaltet. Wenn ein Angreifer z.B. den Vermittlungsrechner des Internet Service Providers des Kunden übernehmen kann, kann er sämtlichen Verkehr vom und zum Kunden filtern und beliebig Nachrichten erzeugen bzw. deren Zustellung verhindern.

7 Chipkartensysteme

7.1 GeldKarte

Ende März 1996 startete in der Region Ravensburg/Weingarten der erste Feldversuch mit der elektronischen Geldbörse. Mittlerweise (1997) verteilen alle Sparkassen die Geldbörse in jeder Giro- und ec-Karte unter dem Namen *GeldKarte*. Die GeldKarte ist eine SmartCard mit integriertem Microcontroller, der die Transaktionen durchführt. Auf dem Chip sind Prozessor, Speicher und ein Betriebssystem.

Es gibt zwei verschiedene Kartentypen:

- Der Kunde erhält die unter dem Namen GeldKarte laufende Kundenkarte. Diese Karte ist eine Referenz auf ein Schattenkonto, welches bankenintern mitgeführt wird. Mit dieser Karte kann der Kunde in Geschäften, die mit einem Chipkartenleser ausgestattet sind, Einkäufe tätigen.
- Beim Händler muß neben dem offensichtlichen Chipkartenleser noch eine sog. Händlerkarte existieren. Diese Karte weist den Händler aus und dient als Kommunikationspartner für die Kundenkarte, wenn der Kunde einen Kauf tätigt.

7.1.1 Eine Lade-Transaktion

1. Der Kunde führt seine GeldKarte in ein Ladeterminal ein. Das Ladeterminal ist online mit dem Autorisierungssystem der Kundenbank verbunden. Der Kunde gibt seine PIN[1] in die Tastatur des Ladeterminals.
2. Das Ladeterminal übermittelt die PIN an die GeldKarte. Die GeldKarte prüft die PIN und schickt bei Gültigkeit der PIN die Kundendaten an das Ladeterminal.

[1] Persönliche Identifikations-Nummer

3. Das Ladeterminal überprüft die Kundendaten auf Gültigkeit und zeigt dem Kunden im Display den maximal möglichen Ladebetrag an.

4. Nachdem der Kunde den Ladewunsch eingegeben hat, transferiert das Ladeterminal die Daten an die Ladezentrale. Die Ladezentrale leitet die Anfrage an die Kundenbank und an die Kartenevidenzzentrale weiter.

5. Nachdem die Kundenbank die Abhebung autorisiert hat, wird dem Ladeterminal angewiesen, die GeldKarte über den neuen Betrag zu unterrichten. Parallel dazu wird das sog. „*Schattenkonto*“ um den entsprechenden Betrag unterichtet.

   Das Geld an sich wird nicht wie bei Token-basierten Systemen auf die Karte übertragen. Die Kundenbank überweist das Geld zur Kartenevidenzzentrale, in der das Schattenkonto geführt wird. Der Kontostand auf der Karte spiegelt den Betrag wieder, der auf dem Schattenkonto vorhanden sein müßte.

6. Wenn die Buchung durch einen Ladefehler nicht korrekt beendet werden konnte, wird das Geld auf das Kundenkonto zurücktransferiert. Beim nächsten Ladevorgang werden die Daten auf der Karte mit denen auf dem Schattensaldo synchronisiert und der Ladevorgang erneut eingeleitet.

Wenn der Kunde eine Karte verwendet, die nicht an ein Kundenkonto gebunden ist (das kann z.B. bei Kundenkarten von Geschäften, Karten für Touristen oder auch Jugendliche der Fall sein), kann die Karte nur gegen Barzahlung geladen werden. Eine solche„weiße“ Karte wird geladen, indem der Kunde dem Ladeterminalbetreiber den Betrag in Bar gibt und das Geld vom Konto des Ladeterminalbetreibers auf das Schattenkonto überwiesen wird. Bei diesem System ist die PIN zum aufladen der Karte nicht notwendig.

7.1.2 Eine Kauf-Transaktion

1. Der Kunde führt seine GeldKarte in den Chipkartenleser des Händlers ein.

2. Im Display des Lesegerätes erscheint der zu transferrierende Betrag. Der Kunde muß durch Tastendruck seine Bestätigung zur Transaktion geben.

3. Die Händlerkarte nimmt nun Kontakt mit der GeldKarte auf. Nachdem sich die beiden Karten gegenseitig authorisiert haben, erzeugt die die Händlerkarte einen zertifizierten Transaktionsdatensatz der die folgenden Daten enthält:

   - die Höhe des Betrages,
   - die ID der GeldKarte,
   - die ID der Händlerkarte
   - sowie Datum und Uhrzeit

Die GeldKarte und das Händlerterminal speichern den Transaktionsdatensatz. Das Händlerterminal übermittelt bei Geschäftsschluß alle Transaktionsdatensätze an die Händler-Evidenzzentrale. Dort werden alle Transaktionen zusammen mit der Kartenevidenzzentrale abgeglichen und die Beträge von den jeweiligen Schattenkonten auf das Händlerkonto übertragen.

Die GeldKarte speichert die jeweils letzten 15 Transaktionsdatensätze und die letzten 3 Lade-Transaktionsdatensätze. Daher kann der Kunde mit Kartenlesegeräten die letzten 15 Abbuchungen bzw. 3 Ladevorgänge kontrollieren.

7.1.3 Kryptografie

Die Authorisierung der Karten geschieht über Challenge & Response-Prozeduren. Die Händler-Karte kann aus der Seriennummer der GeldKarte des Kunden und dem sog. Reduce-Master-Key den GeldKarten-internen Schlüssel berechnen.

Der Transfer der Transaktionsdatensätze zur Evidenzstelle wird über einen 3DES-MAC gesichert, d.h., die Händlerkarte signiert den Datensatz. Die Evidenz-Zentrale verfügt über den Zertifizierungs-Master-Key, aus dem sie zusammen mit der Händler-ID den Händlerkarten-internen Schlüssel berechnen kann.

7.1.4 Die Kundenkarte

Auf der Kundenkarte (GeldKarte) sind folgende Daten gespeichert:

- Nummer (ID) der GeldKarte (GeldKarte-ID)
- die geheime Nummer für Challenge & Response-Abfragen (GeldKarte-Secret)
- die Karten-PIN des Kunden
- Bankleitzahl BLZ der emittierenden Bank
- Kontonummer des Verrechnungskontos
- Aktivierungsdatum
- Ablaufdatum
- Guthaben
- maximaler Ladebetrag
- maximaler Transaktionsbetrag
- aktuelle Sequenznummer für Lade- bzw. Entladetransaktionen
- aktuelle Sequenznummer für Zahlungstransaktionen

- die letzten 3 Transaktionsdatensätze über Lade- bzw. Entladetransaktionen
- die letzten 15 Transaktionsdatensätze über Zahlungstransaktionen

7.1.5 Die Händlerkarte

Die Händlerkarte wird vom Kreditinstitut des Händlers personalisiert. Es gibt zwei Ausführungen der Händlerkarte: Die Smartcard und die Software-Variante. Die SmartCard ist eine normale Chipkarte oder die Ausführung mit herausbrechbarem Chip. Die Software-Lösung wird virtuelle Händlerkarte genannt. Physische Händlerkarten werden z.B. beim Deutschen Sparkassenverlag personalisiert. Neben den vollständig personalisierten Karten gint es auch teilpersonalisierte Karten, die vom Kreditinstitut direkt vor Ort personalisiert werden können.

7.1.6 Sicherheit

Die Sicherheit des Systems basiert auf der Echtheit der Terminals und verwendeten Karten. Wenn es einem Betrüger gelingt, eine Karte zu erzeugen, die sich bei der Zahlungstransaktion wie eine normale Geldkarte verhält, kann er ohne Risiko Käufe tätigen.

Eine normale Kundenkarte kann nicht einfach aufgeladen werden, da dem Angreifer der geheime Schlüssel zur Berechnung der 16-bit-3DES-MAC fehlt. Darüber hinaus wäre dieser Versuch bei personalisierten Kundenkarten selbst bei Erfolg fatal, da über die ID auf das Kundenkonto und somit auf den Kunden zurückgeschlossen werden kann.

Die Identität der GeldKarte liegt in der ID und der Zufallszahl, die für Challenge & Response verwendet wird. Die Kundenkarte ist durch spezielle Fertigungsverfahren gegen Angriffe von Außen gehärtet. Einen vollkommenen Schutz gegen derartige Hardware-Angriffe gibt es aber nicht. Doch selbst wenn ein Angreifer eine Karte knackt, gelangt er nicht an Informationen, die die Sicherheit des Gesamtsystems kompromittieren. Er hätte das Geld besser einfach so ausgegeben.

Ein lohnenswerteres Ziel scheint da schon die virtuelle Händlerkarte zu sein. Hier müßte sich durch den Einsatz von Disassemblern aus der reinen Software der Reduce-Master-Key gewinnen lassen, mit dem man gültige (GeldKarte-ID, GeldKarte-Secret)-Tupel generieren könnt. Schließlich muß die virtuelle Geldkarte auf der GeldKarte-ID das GeldKarte-Secret berechnen können, um die Response korrekt zu berechnen.

Der nächste Schritt wäre die Fertigung von SmartCards, die mit dem Reduce-Master-Key bei jedem Kaufvorgang eine neue ID und neue Secrets berechnen.

Da der Händler bei der virtuellen Karte Zugriff auf den Reduce-Master-Key hat, wäre auch ein Szenario denkbar, in dem der Händler GeldKarten-Tupel sammelt, um einen soliden Datenbestand zu bilden und dann nach Belieben neue Transaktionen zu generieren.

7.1.7 Anonymität

Bei GeldKarten, die an ein spezielles Konto gebunden sind, kann die Bank bzw. die Evidenz-Zentrale nachvollziehen, wer welche Transaktion bei welchem Händler getätigt hat. Den Handlungsgegenstand, die verkaufte Ware, erfährt die Bank aber nicht, d.h., es können keine detaillierten Kaufprofile erstellt werden. Darüber hinaus beteuern die Banken, nur im Betrugsfall an der Kundenidentität interessiert zu sein.

Bei weißen Karten sieht es mit der Anonymität schon anders aus. Da die zugehörigen Schattensalden gegen Barzahlung geladen werden, hat die Bank keiner Möglichkeit mehr, die Spur zum Kunden zurück zu verfolgen.

7.1.8 Verlust der Karte

Bei Verlust der Karte haftet der Kunde. Da die Karte nicht individuell gesperrt werden kann (schließlich müßte man allen Händlern die ID übermitteln), kann die Bank nur sehen, wie das Schattensaldo sukzessive kleiner wird (wenn der Finder die Karte nutzt).

Wenn die Karte dagegen zerstört wird und der Kunde der Bank die Karte zurückgeben kann, hat die Bank die Gewähr, daß keine Abbuchungen vom Schattensaldo erfolgen können und kann dem Kunden das Guthaben zurückerstatten.

7.1.9 Transaktionskosten

Zur Zeit belaufen sich die Kosten für eine Transaktion auf 0,3 Prozent der Umsatzes, mindestens jedoch 2 Pfennig.

7.1.10 Ausblick

Der nächste Schritt ist der Geschichte der GeldKarte wird die Einbindung von Netzwerken in Form von TCP/IP ins GeldKarten-Protokoll sein, d.h., es wird bald die Möglichkeit geben, die GeldKarte am heimischen PC in den Chipkartenleser zu stecken und dann im Internet via GeldKarte zu bezahlen. Es bleibt die Frage, wie die neuen Protokolle gesichert werden und wie die Akzeptanz durch den Kunden ist.

7.1.11 Literatur

[Kre96], [Sel97]

7.2 Danmønt

Danmønt wird seit Ende 1993 in Dänemark benutzt. Danmønt ist ein SmartCard-basiertes System. Die Karten sind Einweg-Karten, d.h., Debit-Karten ähnlich unseren Telefonkarten, die nach Aufbrauchen des Guthabens weggeworfen werden. Danmønt wurde von der dänischen Telefongesellschaft *Tele Denmark*, dem *Danish Payment Systems* PBS und einigen Banken auf den Markt gebracht.

[SFE97] meinen zu Danmønt:

> [...] Das System besitzt Betriebsoptionen für die Sicherheitsrecherche. Hierbei kann Transaktions–*Clearing* einzeln oder kumuliert erfolgen. Es kann u.a. festgestellt werden, ob Doppeltransaktionen erfolgt sind, d.h., ob ein Terminalfehler vorlag oder ein manipulativer Eingriff in den Zahlungsverkehr durchgeführt wurde. Weiterhin ist Betrug mit den Karten bei solchen Einzel–*Clearings* detektierbar.

[FW96] meinen, daß durch die Auslegung als Speicherkarte Replay-Angriffe möglich sind, weil keine kryptografischen Algorithmen die Sicherheit anheben. Die einzelnen Transaktionen sind keiner Person zuordnenbar.

7.2.1 Literatur

[SFE97, Seite 97], [FW96, Seiten 87–88]

7.3 Proton

Proton wird seit 1995 in Belgien getestet. Es handelt sich bei Proton um ein System mit wiederaufladbaren Wertkarten, die von Händlern mit Terminal-Kassen, Verkaufsautomaten oder auch öffentlichen Telefonen akzeptiert werden. Über die technischen Interna sind keine Einzelheiten bekannt [FW96, Seite 88]:

> [...] Die im System eingesetzten Karten sind Mikroprozessorkarten. Das bedeutet, daß das erreichte Sicherheitsniveau höher liegen wird als in einem System, das auf Speicherkarten beruht.
>
> Mit Sicherheit läßt sich sagen, daß das System auf diversifizierten Geheimschlüsseln basiert. Deswegen sind Transaktionen bedingt rückverfolgbar (kombinierbar). Sämtlich Transaktionen werden üblicherweise im System gespeichert, was die persönliche Zuordnung von Transaktionen mit Hilfe einer Referenztransaktion ermöglicht. Um das Sicherheitsproblem zu lösen, das in Verbindung mit dem Wiederaufladen von Karten in geheimschlüsselbasierten Systemen entsteht, werden Karten in Proton nur über Online-Verbindungen wieder aufgeladen. Das bedeutet, daß nicht jedes Aufladeterminal den Generalschlüssel zum Wiederaufladen

> enthalten muß, sondern die Aufladeterminals online mit dem Emittenten verbunden sind und dieser somit allein im Besitz des Wiederaufladeschlüssels ist. [. . .] Es gibt noch andere Nutzungen des Systems, die zur Offenlegung einer Referenztransaktion führen. Beispielsweise führt Quicklink Verlosungen durch, die auf den Transaktionen und den Kartenkennungen basieren. Die Gewinner müssen sich selbst mittels Kartennummer identifizieren. Es mag zwar nicht beabsichtigt sein, aber der Benutzer liefert dabei die Referenztransaktion, die verwendet werden könnte, um alle mit der Karte durchgeführten Transaktionen zu ent– anonymisieren.

Neben Belgien wird Proton noch von Quicklink in Newcastle/Australien, Interpay in Holland, Telekurs in der Schweiz und Mitel in Brasilien eingesetzt.

7.3.1 Literatur

[FW96, Seiten 88–89]

7.4 Mondex

Mondex ist ein SmartKartenansatz, der seit Jahren in Südengland, im Städtchen Swindon, im Einsatz ist. Das Besondere an Mondex ist die Möglichkeit, daß Nutzer mit Hilfe eines kleinen Zusatzgerätes, der „*electronic wallet*“, das Kartensaldo nicht nur abfragen können, sondern auch Wertetransfers zwischen Kundenkarten durchführen können. Dazu wird das Guthaben von der Sender-Karte in die Wallet gebucht, dann die Karte getauscht und das Guthaben auf die Empfängerkarte übertragen. Auf diese Art und Weise können nicht nur Händler mit speziellem Equipment Zahlungen annehmen, sondern auch Privatkunden. Der Auflade-Prozess seitens der Bank wurde so einfach wie möglich gestaltet: der Kunde kann seine Karte an jeder öffentlichen Telefonzelle aufladen.

Bei Mondex gilt wie bei vielen Smart-Card-Systemen das Motto: „*Security by obscurity!!!*“ Nichts genaues weiß man nicht. Mondex gibt keine Details bezüglich der Transaktionen heraus. Trotzdem läßt sich folgendes sagen:

Mondex-I ist noch ein Geheimschlüssel-basiertes System. Mit Mondex-II möchte Mondex auf Public-Key-Kryptografie wechseln, um dem System verschiedene Schwachstellen zu nehmen. Mondex-I geht davon aus, daß alle privaten Transaktionen auf den jeweiligen Karten vermerkt werden können, d.h., es wird ein Log geführt, welches bei jedem Bankkontakt an die Bank übermittelt wird, um dort Konsistenz-Prüfungen durchzuführen und Transaktionen verschiedener Karten mireinander zu korrelieren, um evtl. auftretendes Falschgeld zu erkennen.

Durch die Mittler-Funktion der Wallet, d.h., durch die Fähigkeit der Wallet, eine Karte aufzuladen, ist das System sehr anfällig gegen Angriffe durch Reverse-Engeneering. Wenn in einem öffentlichen Terminal Geheimschlüssel gespeichert sind, ist das schon eine heiße Sache. Wenn jeder Kunde ein Gerät in die Hände bekommt, an dem nach Belieben geprüft werden kann, ist das ein sehr großes Sicherheitsproblem. Zweifelsohne wird Mondex keine Auskünfte über stattgefundene Betrügereien liefern und es ist bis jetzt nichts derartiges bekannt geworden.

Letztlich muß jedes System irgendwann einmal ein Clearing in Zusammenarbeit mit der Bank durchführen. Bei Mondex wäre es möglich, daß Kundenkarten regen Zahlungsverkehr aufweisen, aber durch Aufladungen durch Privatpersonen keine Notwendigkeit besteht, die Bank zu kontaktieren und somit ein Clearing nicht möglich ist.

Mondex realisiert auf für den Benutzer beeindruckende Weise die Funktionalität von Bargeld, mit einigen tollen Zusatz-Features. Der Kunde kann via Telefon Geld abheben, er kann damit überall bezahlen und auch unter Privatpersonen kann ohne Notwendigkeit für Online-Verbindungen Geld transferriert werden. Es bleibt die Frage, wie sehr dieser Komfort zu Lasten der Sicherheit geht.

7.4.1 Literatur

[SFE97, Seite 96], [FW96, Seiten 90–91]

8 Offene Spezifikationen

8.1 CAFE

CAFE steht für *Conditional Access For Europe*[1].CAFE ist ein im Dezember 1992 von der EU-Kommission ESPRIT gestartetes Projekt, das die Entwicklung einer multifunktionalen Smart-Card für Ausweis– und Geldbörsen-Zwecke zum Ziel hatte. CAFE wurde 1995 erfolgreich beendet und mit *SEMPER*[2] (*Secure Electronic Market Place for EuRope*) wurde sofort das Nachfolge-Projekt ins Leben gerufen. Projektkoordinator am CWI war DAVID CHAUM, der auch schon von DigiCash/Ecash her bekannt ist.

Ziel der Entwicklung war ein System, das neben der Kontrolle von Zugriffsrechten wie Computerzugang und Türschlössern auch ein digitales Zahlungssystem modellieren sollte.

Der Grundbaustein von CAFE ist die „*electronic wallet*“, ein kleines, Taschenrechner-großes Gerät, in das die SmartCard eingeführt werden kann. Die Wallet hat eine eigene Batterie, eine Tastatur, ein Display und eine Infrarot-Schnittstelle. Durch diese Wallet wird ein Baustein bereitgestellt, mit dem der Nutzer vertrauenswürdig Informationen erhalten und eingeben kann (sichere Kommunikationsprotokolle vorausgesetzt). Da jeder Nutzer seine eigene Wallet hat, ist die Integrität von Tastatur und Display, überhaupt des gesamten Gerätes, gesichert. Darüber hinaus ist eine Wallet resistent gegen Viren oder sonstwie modifizierte Software, weil auf der Wallet nur ein festes Programm abläuft. Der Nutzer muß seine private SmartCard auch nicht in nichtvertrauenswürdiges Equipment einführen und behält so physisch die Kontrolle über seine Daten.

8.1.1 Leistungsmerkmale

CAFE bietet im voraus bezahlte offline-Zahlungen. Das System besteht im wesentlichen aus dem anonymen Ecash (siehe auch 5.1) von DigiCash, d.h., es handelt sich um eine in Hardware gegossene Offline-Realisierung von Ecash. Der Kunde kauft im voraus digitale Münzen bei seiner Bank.

Zwei Nutzer können offline elektronische Münzen von der einen Wallet in die andere transferrieren, d.h., im Gegensatz zu Ecash ist kein sofortiger Clearing-Prozess durch die Bank notwendig. Trotzdem muß der Zahlungsempfänger die Münzen bei der Bank gegen neue eintauschen, bevor er sie an einen dritten Nutzer weiterreichen kann, d.h., das Clearing ist doch notwendig,

[1] `http://www.cwi.nl/cwi/projects/cafe.html`
[2] `http://se,per.zurich.ibm.ch/index.html` bzw. `http://www.semper.org`

allerdings kann es zeitversetzt erfolgen.

Abhebungen vom Konto einer Nutzers sind natürlich online-Transaktionen, die an Geldautomaten erfolgen können. Der Kunde kann in seiner Wallet unabhängig voneinander verschiedene Währungen speichern (was auch nach Einführung des Euro noch interessant sein dürfte). Darüber hinaus kann dem Kunden das Geld erstattet werden, wenn er seine SmartCard verliert. Bemerkenswert ist, daß diese Möglichkeit existiert, obwohl es sich um ein anonymes, im voraus bezahltes Zahlungssystem handelt.

CAFE schreibt keine besondere Hardware oder Software fest, sondern beschreibt Protokolle, die ohne weiteres auch in andere Systeme integriert werden können.

8.1.2 Entwicklungsziele Sicherheit

Multi-Party Security: Mit Multi-Party Security ist gemeint, daß jeder nur sich selbst trauen muß. Das Design von CAFE sollte so ausgelegt sein, daß selbst der paranoideste Nutzer sicher sein kann, daß seine Ansprüche gesichert sind. Dieses Feature kommt sowohl den Kunden als auch Banken und Händlern zugute. Im Gegensatz zu vielen Modellen muß die Bank bei CAFE keinen besonderen Vertrauensstatus haben.

Das Design sollte darüber hinaus vollkommen offen spezifiziert sein, d.h., keine proprietären Algorithmen, die unvollständig bekannt sind oder Protokolle, die nicht öffentlich diskutiert werden können. Da der gemeine Nutzer keine Möglichkeit hat, sein Equipment zu untersuchen und zu bewerten, sollte eine Möglichkeit bestehen, daß unabhängige Institutionen die Sicherheit zu untersuchen.

Untraceability: Zahlungen sollen nicht verfolgt werden können. Die Bank sieht, wenn ein Kunde eine digitale Münze abhebt. Wenn ein anderer Nutzer die Münze einreicht, kann die Bank keine Verbindung zwischen der ausgezahlten und der eingezahlten Münze herstellen.

Darüber hinaus bleibt der Käufer auch dem Händler gegenüber unerkannt, d.h., schon der Händler kann aus der Münze keine Käufer-Identität ableiten. Für Händler gilt diese Anonymität nicht. Der Kunde weiß i.A., wem er sein Geld gibt.

Die Auszahlungen und Einzahlungen selbst sind natürlich nicht anonym, da die Bank ja wissen muß, mit welchem Konto der Vorgang zu verrechnen ist.

8.1.3 Kryptografie

Zuerst mal sollte eine SmartCard tamper-resistant sein, d.h., sie sollte nicht geknackt werden können, um an das Geld zu kommen. Darüber hinaus sollte, selbst *wenn* die Wallet geknackt wird, die daraus erhaltenen Informationen einen Angreifer eindeutig identifizieren. Daraus folgt, daß die Nutzer dieser Tamper-resistance überhaupt nicht trauen müssen.

Asymmetrische Unterschriften

Von vorn herein fiel eine Entscheidung zugunsten von asymmetrischen Unterschriften. Bei symmetrischen Unterschriften mit MAC's kann ein Richter im Streitfall nicht entscheiden, welcher der beiden Kommunikationspartner (Prover und Verifier) die Unterschrift erzeugt hat.

Der *Guardian*

Der Guardian ist ein Kryptocontroller (im Speziellen ein IC der Firma Siemens), der in die Wallet mit eingebaut ist und die gesamte Kommunikation überwacht und auch unterbrechen kann. Hauptziel des Guardian ist die Verhinderung von Double-Spending auf Kundenseite.

Dieses Ziel kann der Guardian nicht erreichen. Es gibt immer einen Angreifer, der den aktuellen Sicherheitslevel knacken kann, der den Guardian knackt und durch eine Fälschung ersetzt. Somit stellt der Guardian nur eine hohe Hürde dar.

Off-Line Coins

Um die Sicherheit auch noch ohne Guardian zu gewährleisten, werden elektronische, blinde Signaturen verwendet. Der Unterschied zu den in Ecash verwendeten Signaturen liegt in der Notwendigkeit, die Identität des Nutzers mit einzucodieren. Bei Ecash war das aufgrund des Online-Clearings nicht notwendig. Doppelt ausgegebene Münzen wurden einfach nicht angenommen.

In einem Off-Line-System besteht diese Möglichkeit nicht. Daher wird der Kunde beim Münz-Abheben, d.h., bei der Münzerzeugung, dazu gezwungen, seine Identität fest in die Münze zu codieren. Wenn die Münze einmal ausgegeben wird, kann die Bank bei Einreichung der Münze die Identität nicht rekonstruieren. Wenn die Münze ein zweites Mal ausgegeben wird, erhält die Bank genug Information, um die Identität zu rekonstruieren. Das Verfahren ist in 2.12 auf Seite 17 beschrieben.

8.1.4 Anonymität

Die Auslegung des Systems erlaubt dem Benutzer, den Level Anonymitäts-Level selbst zu bestimmen. Das System ermöglicht nichtzurückverfolgbare Zahlungen.

8.1.5 Transaktionskosten

Da die verwendeten Protokolle die eingesetzte Hardware sehr aufwendig sind, liegen die Transaktionskosten sehr hoch. [FW96]: „*Obwohl der Systementwurf der sicherste aller beschriebenen Systeme ist, stellt er bisher keine kommerziell überlebensfähige Alternative dar.*“

8.1.6 Literatur

[CPS94], [Way97], [FW96, Seite 92]

8.2 SEMPER

SEMPER (`http://www.semper.org/`) bedeutet *Secure Electronic Market Place for Europe* und ist ein EU-Projekt, das im September 1995 ins Leben gerufen wurde. An SEMPER sind neben verschiedenen Banken viele europäische Universitäten und europäischen Krypto-Firmen beteiligt. Die Gesamtleitung des Projektes liegt bei IBM.

Ziel von SEMPER ist die Schaffung einer offenen Umgebung, in der Nutzer verschiedenste Dienste sicher in Anspruch nehmen können. Dazu zählen:

- Mail-Order-Bestellungen aufgeben
- Online Käufe tätigen, Bestellungen aufgeben und bezahlen
- Fair gemeinsam Verträge unterschreiben, trotz räumlicher Abwesenheit verschiedener Vertragspartner
- Die Nutzung von notariellen Diensten, z.B. Timestamping-Services
- Abschluß von Versicherungen
- Auktionen: mitbieten übers Netz
- Online-Kauf von Informationen
- zeitlich Begrenzte Nutzung von Online-Ressourcen wie Zeitschriften, etc.
- Ticket-Nutzung: es können Tickets gekauft werden, gegen deren Einlösung gewisse Services in Anspruch genommen werden können
- Behördengänge via Netz erledigen
- Richterdienste bei Streitigkeiten in Anspruch nehmen können
- Authentische Empfangsbestätigungen

Alle Aktionen sollen sowohl vertraulich, nach Möglichkeit anonym und auch noch nichtabstreitbar sein, d.h., die Belege der einzelnen Teilnehmer gelten vor Gericht als Beweise. Darüber hinaus sollen schon die Protokolle so ausgelegt sein, daß nach der Zusage das Geschäft nicht mehr angebrochen werden kann, d.h., nach Kaufbestätigung wird auch gezahlt. Im Vertragsbereich würden alle Unterschriften erst gemeinsam gültig und Empfänger werden durch die Protokolle gezwungen, für erhaltene Ware nichtabstreitbare Empfangsbestätigungen zu geben.

SEMPER ist ein noch in der Entwicklung befindliches System. Unter SEMPER sollen viele verschiedene Zahlungsmechanismen und mehr unter einem Dach vereint werden. Dazu gehört die Integration von Transportsicherungen ebenso wie Sicherungen auf Applikationsebene.

8.2.1 Literatur

`http://www.semper.org/`, `http://www.semper.org/info/index.html`, [Wai96]

9 Homebanking

9.1 HBCI

HBCI steht für Homebanking Computer Interface und ist eine Spezifikation[1] der deutschen Kreditwirtschaft. HBCI wurde spezifiziert, um das an BTX[2] bzw. T-Online gebundene Homebanking von einem universelleren Verfahren abzulösen.

Da bei HBCI kein Händler mit involviert ist, sondern die Überweisungen abgesichert werden, werden hier keine Unterpunkte wie Teilbarkeit oder Anonymität untersucht.

9.1.1 Entwicklungsziele

- Unabhängigkeit von einem speziellen Übertragungsmedium (auch unsichere Kanäle sollten kein Problem sein). Es soll also Homebanking via Internet möglich sein.
- Unabhängigkeit von Präsentationsdiensten, d.h., jedes Homebanking-Programm soll HBCI-konform sein, unabhängig von der GUI. MS-Money, Quicken, welches Programm auch immer der Kunde wählt, soll bezüglich der Bankschnittstelle den gleichen Standard einhalten.
- Sicherheit: Unabhängig von der Sicherheit des verwendeten Mediums soll die Sicherheit gewährleistet sein.
- Bedienbarkeit: Alle Funktionen, die Kunden benötigen, sollen im Standard festgeschrieben sein, d.h., Girokontoverwaltung, Festgeldverwaltung,Überweisungen, Daueraufträge, Spenden, etc.
- Internationalität: Der Standard soll nicht eine spezielle Landessprache vorschreiben.
- Multibankfähigkeit, d.h., alle Banken sollen HBCI„sprechen“. Jede Kundenbank versteht HBCI; damit kann der Kunde jede Kombination von Software und Bank verwenden.

HBCI soll sowohl auf Kunden-PC’s laufen, aber auch auf öffentlichen Terminals. Um die Sicherheit an öffentlichen Terminals zu gewährleisten, muß der Kunde eine Möglichkeit haben, die ihn

[1] `http://www.siz.de/siz/hbci/hbcispec.htm` und `http://www.hbci-kernel.de`
[2] Bildschirmtext

identifizierenden Daten sicher zu transportieren. Angestrebt ist SmartCard-basierte Lösung, sowohl für öffentliche Terminals als auch für den Kunden-PC zuhause.

Im Gegensatz zu Sicherungsverfahren wie SSL, die in der Transportschicht abgesiedelt sind, erledigt HBCI seinen Dienst in der Anwendungsschicht. Daher ist das darunter liegende Transportmedium egal, es muß nur dialogfähig sein, d.h., e-mail (SMTP) scheidet aus.

9.1.2 Kryptografie

HBCI verwendet in der Anfangsphase zwei Verfahren papallel zueinander:

1. DDV: Das DES-DES-Verfahren DDV verwendet sowohl für die elektronische Signatur als auch für die eigentliche Verschlüsselung Tripel-DES (3DES) mit 2 Schlüsseln, d.h., einer effektiven Schlüsselbreite von 112 bit. DDV verwendet MAC[3] als digitale Signatur.

   Das DDV ist auch die mit Chipkarten umgesetzte Methode.

2. RDH: Das RSA-DES-Hybridverfahren verwendet RSA-EU als Signaturmethode und verschlüsselt auch den 3DES-Session-Key mit RSA.

   RSA ist in der Anfangsphase noch nicht für den Chipkarteneinsatz geplant, da den Banken die Auswahl bei preiswerten und leistungsfähigen RSA-SmartCards noch nicht groß genug ist. PC-basierte Software-lösungen müssen RSA-EU unterstützen, für Endgeräte wie SmartPhones mit MAC-Chipkarte ist RSA noch nicht vorgeschrieben.

RSA:

Die Schlüsselbreite bei RSA ist in HBCI Version 2.0 auf 768 bit festgelegt. Für jeden Teilnehmer sollen zwei Schlüsselpaare existieren: Das eine für die digitale Signatur, das andere als zum Keyexchange.

Hashfunktion:

Die verwendere Hashfunktion ist RIPEMD-160 mit 160 bit (20 Byte) Ergebnis-Breite.

Elektronische symmetrische Signatur:

Die Signatur wird gemäß ANSI X9.19 durch ein Retail CBC-MAC gebildet. Das Gesamtergebnis der MAC ist 64 bit (8 Byte) breit.

[3] *message authentication code*, Einweg-Hashfunktion mit geheimem Schlüssel. Siehe auch [Sch96, Seiten 520–524]

Verschlüsselung:

Als symmetrische Chiffre ist der 3DES nach ANSI X3.92 im CBC[4]-Modus gewählt worden. Die Spezifikation unterbindet die Nutzung von DES-spezifischen schachen und halbschwachen Schlüsseln. Die Sicherung der Nachrichten durch Verschlüsselung ist verpflichtend, d.h., eine einfache Signatur reicht nicht aus.

Schlüssel:

Bei dem in [Luc97a] beschriebenen Projekt der Raiffeisen-Volksbank Mainz erzeugt die Bank das sensitive Schlüsselmaterial und schickt es auf dem Postweg an den Kunden. Sowohl symmetrische Signaturschlüssel als auch RSA-Schlüsselpaare werden von den Banken erzeugt.

Replay-Angriffe:

Um das Wiedereinspielen von Nachrichten zu verhindern, werden alle HCBI-Messages mit Sequenznummern versehen. Diese Sequenznummern werden beid er Bank nachgehalten. Da die Verarbeitung der Nachrichten beim Kunden u.U. asynchron läuft, toleriert die Bank eine Nichteinhaltung der Reihenfolge innerhalb der eintreffenden Nachrichten. Trotzdem werden die Sequenznummern nachgehalten, um Replay-Angriffe zu verhindern.

9.1.3 Zertifikate

Durch die bankeninterne Schlüsselerzeugung besteht kein Bedarf an Trustcentern, da alle Schlüssel sozusagen im Trustcenter generiert werden und daher implizit echt sind.

9.1.4 Ausblick

Wenn HBCI von allen Banken unterstützt wird, verspricht es ein komfortables System für den Homebankingsektor zu werden. Solange noch keine RSA-Karten verfügbar sind, bleibt die Problematik, daß Viren oder trojanische Pferde die geheimen RSA-Schlüssel und Kundendaten vom magnetischen Datenträger lesen können. Vom Gesamtkonzept aber präsentiert sich HBCI als gut durchdachte Spezifikation. Es bleibt zu hoffen, daß die eigentlichen Implementationen keine ernsten Sicherheitslöcher durch schlechte Zufallszahlengenerierung etc. erzeugen.

9.1.5 Literatur

[Int97a], [Int97f], [Int97b], [Int97c], [Int97d], [Int97e], [Luc97b], [Luc97a]

[4] *cipher block chaining*, Verkettung von Chriffrentext- und Klartext-Blöcken

9.2 MeCHIP

Der MeCHIP (http://www.mechip.com) ist ein Produkt der Firma ESD GmbH (http://www.esd.de/) aus Deutschland. Grund für die Entwicklung dieses Hardware-Produktes war die Annahme, daß alle Homebanking-Lösungen, die nur aus Software basieren, zu anfällig gegen Viren und trojanische Pferde sind.

Der MeCHIP ist eine Steckkarte, die in den PC gebaut wird. Die Tastatur des Rechners wird nicht mehr an den normalen Tastaturport angeschlossen, sondern an die Steckkarte. Daher müssen alle Benutzereingaben zwangsläufig durch die Steckkarte, bevor sie im PC weiterverarbeitet werden können. Auf der Steckkarte läuft ein festes Programm, das nicht durch Viren oder andere Programme umgangen werden kann. Wenn der PC nun eine Verbindung zum Bankserver aufbaut, übernimmt der MeCHIP die gesamte Verschlüsselung der Daten. Nicht-verschlüsselte Daten sollen nicht mehr in den Speicher des PC's gelangen. Im MeCHIP werden die geheimen Schlüssel des Benutzers gespeichert. Dadurch kann der Chip alle Daten digital signieren und so gegen Veränderung durch böswillige Programme schützen.

Die erste wirkliche Anwendung erfolgte im März 1996 durch die *Sparda Bank Hamburg* (http://www.sparda-hh.de/), die den MeCHIP als Homebanking-Lösung unterstützte. Auf dem PC läuft parallel zum MeCHIP das Programm MeWallet, welches die Übertragung der Daten in Richtung Bank und die Präsentation übernimmt.

ESD bietet für die Server-Seite eine ganze Fülle von Produkten an, damit Banken das System in ihre schon bestehende Rechnerlandschaft integrieren können. Direkt nach dem Netzzugang folgt eine Firewall, die die Datenpakete vom MeCHIP auf ihre Gültigkeit überprüft. Darüber hinaus wird ein Signatur-Server mit angebunden, um RSA-Signaturen zu prüfen. Ein Kommunikationsserver, ein dynamischer Kartenserver und ein Protokollserver sind die letzten Glieder der Kette, bevor die eigentlichen Buchungsdaten ins Bankennetz übergeben werden.

9.2.1 Kompatibilität

ESD plant die Unterstützung von SET, HBCI, OFX oder auch der GeldKarte, d.h., das Laden und Entladen von Zuhause aus.

9.2.2 Kryptografie

ESD setzt auf RSA mit bis zu 2048 bit Schlüssellänge als asymmetrische Chiffre, DES, 3DES und IDEA als symmetrische Chiffren und MD5, SHA und RIPEMD160 als Hashfunktionen.

Wenn man sich die verwendeten Algorithmen ansieht, findet man die zur Zeit sichersten Vertreter der jeweiligen Gattungen.

9.2.3 Sicherheit

Der Ansatz, gegen Viren resistente Hardware einzusetzen, ist vernünftig. Das Problem, das Hybridverfahren von Hardware und Software mit sich bringen ist die Diskrepanz zwischen den auf dem Bildschirm dargestellten Daten und den im Rechnerinneren tatsächlich signierten Daten. Auch wenn Viren den eigentlichen Signatur- bzw. Verschlüsselungsvorgang nicht modifizieren können, können dem Chip trotzdem Daten untergeschoben werden, die nicht denen auf dem Bildschirm entsprechen.

9.2.4 Ausblick

Durch den Einsatz von sicherer Hardware wird die Angriffsschwelle durch Hacker deutlich angehoben. Trotzdem lassen sich auch hier Angriffe noch nicht komplett ausschalten.

Die andere Schwierigkeit liegt in der Herstellerbindung. Im Gegensatz zu offenen Standards wie HBCI oder SET ist der MeCHIP ein Gerät, das nur von einem einzigen Hersteller erhältlich ist. Solche Lösungen führen häufig zu einem Markt, in dem viele zueinander inkompatible Produkte um die Vorherrschaft ringen. Trotzdem ist das ESD-System auf die Unterstützung offener Standards ausgelegt, was diesem Dorn ein wenig die Härte nimmt.

9.3 X*PRESSO

Das *X*PRESSO Security Package* ist ein Produkt der Böblinger Firma Brokat (`http://www.brokat.de`). X*PRESSO bietet 128-bit-Verschlüsselung auf Applikationsebene. Da bis vor kurzem die aus den USA exportierten Softwareprodukte eine maximale Schlüssellänge von 40 bit aufweisen durften, bestand Bedarf an Lösung, die einfach in bestehende Browser-Systeme integriert werden konnte.

Um diese Integration zu ermöglichen, wurde eine auf Java basierende Lösung erarbeitet. Ein Java-Applet führt die Verschlüsselungsarbeit durch. Die *X*PRESSO Java Security Classes* können in die Kundenapplikation eingebaut werden, Serverseitig lautet das Pendant *X*PRESSO Security Server*. Alternativ zu den Java Security Classes existiert noch ein Security-Plug-In für Browser und die *C++ Security Classes*, um C++—Programme mit den Brokat-Sicherheitsmaßnahmen auszustatten. Auf X*PRESSO basierend setzt das SRT-Protokoll[5] auf. SRT [Luc97d] stellt praktisch eine 1:1 Unsetzung von SSL in Java dar.

9.3.1 Kryptografie

Der asymmetrische Teil wird von RSA mit 1024 bit Schlüsselbreite übernommen, die symmetrische Verschlüsselung wird von IDEA mit 128 bit übernommen. Die verwendeten Hash-

[5] Secure Request Technology

Funktionen sind MD5 und SHA, optional auch RIPEMD. Die einzelnen Nachrichten werden über MACs gesichert.

9.3.2 Eine Kommunikationsverbindung

Bevor das Java-Applet vom Server geladen wird, baut der Browser eine SSL-Verbindung zum Server auf. Dann wird das digital signierte Applet über die 40-bit-SSL-Verbindung an den Client geschickt. Der Client startet das Applet, welches die Applet-spezifische Verschlüsselung startet (zusätzlich zur schon bestehenden SSL-Verbindung).

9.3.3 Ausblick

SSL/SRT/X*PRESSO ist kein Zahlungssystem, sondern in der Transportschicht angesiedelt. Zur Zeit sichern die *Deutsche Bank* und die *Bank 24* ihre Homebanking-Verbindungen mit X*PRESSO ab.

[Sto97, Seiten 92–97], [Luc97d]

10 Perspektiven

Ich hoffe, daß diese Arbeit aufzeigen konnte, wieviel unterschiedliche Ansätze es im Bereich des Computer-gestützten Geldgeschäftes gibt. Es gibt die tokenbasierten Ansätze, die kleine Informationsbrocken weiterreichen, als wären es Münzen. Die andere große Sparte sind die scheckbasierten Verfahren. Irgendwo sind der Absender und der Empfänger einer Zahlung immer festgehalten. Bei den scheckbasierten Verfahren wird versucht, die Schecks immer sicherer zu machen und die Verarbeitung möglichst zu automatisieren.

Trotz aller Mühen findet man fast kein System, das alle Vorteile in sich vereint. Jedes System hat seine ganz spezifische „Geschmacksrichtung“, und es bleibt dem Anwendungsfall überlassen, was für einen ganz speziellen Fall am besten paßt.

Ein wichtiger Schritt für die Zukunft scheint eine Migration der wichtigsten Systeme unter einem Dach zu sein; beispielsweise wäre ein Zahlungssystem, bestehend aus SET, Ecash und Millicent, oder alternativ dazu Payword, das System, das große Anwendungsfelder abdecken würde. Der Kunde hätte große, mittlere und kleine Zahlungsmechanismen sicher und komfortabel unter einem Dach vereint.

Der andere Schritt muß die sichere Spezifikation von angriffssicherer Hardware sein. Hier sind noch nicht einmal Module gemeint, die der Ionenfräse im Mikroelektroniklabor standhalten, sondern System-Entwürfe, bei denen der Kunde sicher sein kann, daß keine Viren oder amoklaufenden Java-Applets seinen Zahlungsdaten gefährlich werden können. Beispielsweise Chipkartenleser mit Display, die Signaturen durchführen können, wobei im Display die zu signierenden Daten angezeigt werden. Das könnte teils die Schwierigkeiten beheben, die entstehen, wenn die bei einer SmartCard zur Signatur eingereichten Daten andere sind, als auf dem Computerbildschirm zu sehen sind.

Darüber hinaus sind wirkungsvolle Schutzkonzepte innerhalb der Betriebssysteme vonnöten, die effektiven Speicherschutz garantieren. Eine Sensibilisierung der Kunden, ihren Rechner nicht mit Software unbekannter Herkunft vollzustopfen, wäre auch ein Schritt in die richtige Richtung.

Da der normale Internet-Surfer aber immer die neueste Software auf seiner Rechnerplattform installieren möchte, wären auch autonome Zahlungssysteme denkbar, die als Peripherie an den Computer angeschlossen werden. Diese Geräte könnten ebenfalls mit Displays ausgerüstet sein, die den Zahlungsempfänger anzeigen und mit Speicher, um die elektronischen Münzen oder die geheimen Schlüssel zu speichern.

Meines Erachtens nach ist die Tamper-Resistance nicht so wichtig, wie sie oft gesehen wird. Wenn der Protokollentwurf von vorn herein davon ausgeht, daß sowohl der Kunde als auch der Händler Betrügereien starten möchten und vollen Zugriff auf die gesamte Hardware haben, dann

wäre ein durchweg sicherer Standpunkt gefunden. CAFE ist hier ein gutes Beispiel. Jeder Teilnehmer kann den gewünschten Sicherheitslevel selbst wählen, und die Protokolle gehen davon aus, daß auch Spezialhardware wie ein Guardian überlistet werden kann. Im Endeffekt sind diese Sperren nur ein Wettlauf zwischen Entwicklern und Angreifern.

Bei zusätzlicher Hardware kann der Kunde ein gutes Feeling bezüglich der Datensicherheit haben. Der Kunde bekommt mit einem abnehmbaren Gerät die Möglichkeit, die sensitiven Daten einfach zu nehmen und an einem sicheren Ort wegzuschließen. Darüber hinaus kann er auf seinem Rechner arbeiten, installieren und löschen, was das Zeug hält.

Ich hoffe, Sie hatten beim Lesen der Arbeit wenigstens ein klein wenig Spaß . . . ;-)

Düsseldorf, den 25. Februar 1998

Literaturverzeichnis

[AJSW97] Asokan, N., Janson, Phillipe, Steiner, Michael und Waidner, Michael. State of the Art in Electronic Payment Systems. *IEEE Computer*, September 1997.

[Alb97] Thomas Albinus. Schlüssel zum Internet-Kommerz? *DOS*, April 1997.

[AMS95] Anderson, Ross, Manifavas, Harry und Sutherland, Chris. A practical electronic cash system. *Technical Report, Cambridge University*, 1995.

[AMS96] Anderson, Ross, Manifavas, Charalampos und Sutherland, Chris. NetCard – A Practical Electronic-Cash System. In Lomas [Lom96], Seiten 49–58.

[And93] Ross Anderson. Why cryptosystems fail. *Proceedings of the First ACM Conference on Computer and Communications Security*, 1993.

[And94] Ross Anderson. Liability and Computer Security: Nine Priciples. In Gollmann [Gol94], Seiten 231–245.

[ASW] Asokan, N., Schunter, Matthias und Waidner, Michael. Optimistic Protocols for Fair Exchange. *Internal Report*.

[Bar97] George Barwood. FAQ: Kryptografie mit elliptische Kurven, v1.8. *Internet:* `http://www.fitug.de/ulf/faq/ec_faq.html`, 1997.

[BBC+94] Boly, J.-P., Bosselaers, A., Cramer, R., Michelsen, R., Mjolsnes, S., Muller, F., Pedersen, T., Pfitzmann, B., de Rooij, P., Schoenmakers, B., Schunter, M., Vallee, L. und Waidner, M. The ESPRIT Project CAFE – High Security Digital Payment Systems. In Gollmann [Gol94], Seiten 217–230.

[BC97] Baldwin, Robert und Chang, Victor. Locking the e-safe. *IEEE SPECTRUM*, Februar 1997.

[Bey98] Hans-Bernhard Beykirch. OTP: Open Trade Protocol: Weltweit handeln. *iX*, Seiten 122–126, Marz 1998.

[BGH+95] Bellare, Mihir, Garay, Juan, Hauser, Ralf, Herzberg, Amir, Krawczyk, Hugo, Steiner, Michael, Tsudik, Gene und Waidner, Michael. iKP — A Family of Secure Electronic Payment protocols - Externded Abstract. `http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/`, Juli 1995.

[BKMM96] Bertino, Elisa, Kurth, Helmut, Martella, Giancarlo und Montolivo, Emilio, Hrsg. *Computer Security – ESORICS 96*. Springer Verlag, 1996.

[Bra93] Stefan Brands. Untraceable Off-Line Cash in Wallets with Observers. In unbekannt [unb93], Seiten 302–318.

[Bra95] Stefan Brands. Restrictive blindings of secret-key certificates. In Guillou, Louis C. und Quisquater, Jean-Jaques [GQ95], Seiten 231–247.

[BS97] Buth, Karl Heinz und Sommer, Carsten. Sicherer Geschäftsverkehr über das Internet. *Mit Sicherheit in die Informationsgesellschaft*, Seiten 413–425, 1997.

[Bun97a] Bundesamt für Sicherheit in der Informationstechnik. Elektronisches Bezahlen. *Faltblatt Ö 18*, Januar 1997.

[Bun97b] Bundesamt für Sicherheit in der Informationstechnik. *Mit Sicherheit in die Informationsgesellschaft*. SecuMedia Verlag Ingelheim, 1997.

[CFN95] Chaum, D., Fiat, A. und Naor, M. Untraceable Electronic Cash. In Goldwasser [Gol95], Seiten 319–327.

[Cha83] David Chaum. Blind Signatures for Untraceable Payments. In unbekannt [unb83], Seiten 199–203.

[Cha92] David Chaum. Achieving electronic privacy. *Scientific American*, August 1992.

[Chr96a] Jürgen Christ. Bitzahler – Die Banken entdecken den Kommerz im Netz. *iX*, Seiten 92–103, August 1996.

[Chr96b] Jürgen Christ. Elektronisches Geld. *Funkschau*, 8, 1996.

[CL96] Crispo, Bruno und Loman, Mark. A Certification Scheme for Electronic Commerce. In Lomas [Lom96], Seiten 19–32.

[CMS96] Camenisch, J.L., Maurer, U. und Stadler, M.A. Digital Payment Systems with Passive Anonymity-Revoking. In Bertino, Elisa et al. [BKMM96], Seiten 33–43.

[Cop95] Don Coppersmith, Hrsg. *Advances in Cryptology — CRYPTO '95*. Springer Verlag, 1995.

[CPS94] Camenisch, J.L., Piveteau, J.M. und Stadler, M.A. An Efficient Electronic Payment System Protecting Privacy. In Gollmann [Gol94], Seiten 207–215.

[Dar95] I.B. Darmgård. Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals. In Goldwasser [Gol95], Seiten 328–335.

[DBR98] Dowling, Michael, Beuker, Ralf und Rexrodt, Günther. Stratigien für Electronic Commerce. *Spektrum der Wissenschaft: Dossier – Die Welt im Internet*, Seiten 80–84, 1998.

[DD97] Dresen, Stephan und Dunne, Thomas. Penny Lane – Wie das ecash-Projekt der Deutschen Bank praktisch funktioniert. *iX*, Seiten 102–107, Dezember 1997.

[ECS94] Eastlake, Donald, Crocker, Stephen und Schiller, Jeffrey. Randomness Recommendations for Security. *Request for Comments: 1750*, Dezember 1994.

[Fab97] Carsten Fabich. Plastikdschungel: Kreditkarten im Überblick. *c't Report 3: Geld online*, 1997.

[Fan97] Carol Fancher. In your pocket smartcards. *IEEE SPECTRUM*, Februar 1997.

[Fei92] J. Feigenbaum, Hrsg. *Advances in Cryptology — CRYPTO '91*. Springer Verlag, 1992.

[Flo96] Udo Flohr. Electric money. *Byte*, Juni 1996.

[FO96] Fujisaki, Eiichiro und Okamoto, Tatsuaki. Practical Escrow Cash System. In Lomas [Lom96], Seiten 33–48.

[FW96] Furche, Andreas und Wrightson, Graham. *Computer Money: Zahlungssysteme im Internet*. dpunkt Verlag, 1996.

[Gem97] Peter Gemell. Traceable e-cash. *IEEE SPECTRUM*, Februar 1997.

[GMA+] Glassmann, Steve, Manasse, Mark, Abadi, Martin, Gauthier, Paul und Sobalvarro, Patrick. The Millicent Protocol for Inexpensive Electronic Commerce. `http://www.research.digital.com/SRC/personal/Mark_Manasse/common/millic%ent/millicent.html`.

[Gol94] Dieter Gollmann, Hrsg. *Computer Security – ESORICS 94*. Springer Verlag, 1994.

[Gol95] Shafi Goldwasser, Hrsg. *Advances in Cryptology — CRYPTO '88*. Springer Verlag, 1995.

[GP98] Grimm, Rüdiger und Pordesch, Ulrich. Wie sicher ist die digitale Signatur. *Spektrum der Wissenschaft: Dossier – Die Welt im Internet*, Seiten 60–62, 1998.

[GQ95] Guillou, Louis C. und Quisquater, Jean-Jaques, Hrsg. *Advances in Cryptology — EUROCRYPT '95*. Springer Verlag, 1995.

[Gri97] Rüdiger Grimm. Elektronisches Geld. *DuD*, Juli 1997.

[Gro97] Andreas Grote. Mit offenen Karten: Was der Verbraucher beim bargeldlosen Zahlungsverkehr zu beachten hat. *c't Report 3: Geld online*, 1997.

[GS97] Garfinkel, Simson und Spafford, Gene. *Web Security & Commerce*. O'Reilly & Associates, Inc., 1997.

[GZ96] Grimm, Rüdiger und Zangeneh, Kambiz. Cybermoney im Internet. *à la CARD*, Februar 1996.

[GZ97] Grimm, Rüdiger und Zangeneh, Kambiz. Cybermoney in the Internet: An Overview over new Payment Systems in the Internet. *???*, 1997.

[Ham97a] Scott Hamilton. E-Commerce for the 21st Century. *IEEE Computer*, Mai 1997.

[Ham97b] Christoph Hammerschmidt. SERIE Electronic Cash, Folge 3: Sicherheit im elektronischen Handel ist heute schon problemlos realisierbar. *Computer Zeitung Nr. 47*, 1997.

[Int97a] Homebanking Computer Interface. Das HBCI-Kompendium. `http://www.siz.de/siz/hbci/hbkomp20.pdf.pdf`, 1997.

[Int97b] Homebanking Computer Interface. Teil A: Allgemeines. `http://www.siz.de/siz/hbci/hbci201a.pdf`, 1997.

[Int97c] Homebanking Computer Interface. Teil B: Sicherheit. `http://www.siz.de/siz/hbci/hbci201a.pdf`, 1997.

[Int97d] Homebanking Computer Interface. Teil C: Geschäftsvorfälle. `http://www.siz.de/siz/hbci/hbci201a.pdf`, 1997.

[Int97e] Homebanking Computer Interface. Teil D: Anlagen. `http://www.siz.de/siz/hbci/hbci201a.pdf`, 1997.

[Int97f] Homebanking Computer Interface. Vortrag „Sicherheit im Homebanking". `http://www.siz.de/siz/hbci/cthm_kr4.pdf`, 1997.

[Jak95] Markus Jakobsson. Ripping coins for a fair exchange. In Guillou, Louis C. und Quisquater, Jean-Jaques [GQ95], Seiten 220–230.

[JMLW93] Johnson, D., Matyas, S., Le, A. und Wilkins, J. Design of the Commercial Data MAsking Facility Data Privacy Algorithm. *Proceedings of the First ACM Conference on Communications and Computer Security*, 1993.

[jo97] jo. Mit Peanuts im Web bezahlen. *c't 13/97*, 1997.

[JW95] Janson, Phillipe und Waidner, Michael. Electronic Payment over Open Networks. `http://www.zurich.ibm.com/~sti/g-kk/publications/1995/JaWa95.dir/JaWa95%e.html`, April 1995.

[JW96] Janson, Phillipe und Waidner, Michael. Electronic Payment Systems. *DuD*, Juni 1996.

[Kam97] Ajit Kambil. Doing business in the wired world. *IEEE Computer*, Mai 1997.

[Kle97] Max Kleiner. Unter aller Augen: Sicherheit im Bankenbusineß. *c't Report 3: Geld online*, 1997.

[Kos98] Axel Kossel. Micropayment von Digital: Bezahlen über das Internet. *c't – 3/98*, Seite 22, 1998.

[Kre96] Thomas Krebs. Elektronische Geldbörse: Grundlagen, Erfahrungen, Perspektiven. *SIZ-SPEZIAL*, 1996.

[Kun97] Michael Kunze. Kartentausch: Web-Konsortium will den Schutz der Privatsphäre im Netz vereinheitlichen. *c't Report 3: Geld online*, 1997.

[KW97] Kossel, Axel und Wronski, Hans–Jürgen. Bare Bytes – Online bezahlen im Internet. *c't – 16/97*, Seiten 66–69, 1997.

[Lan97] Barbara Lange. Secure Electronic Transactions: Kreditkarten im Internet. *iX*, Seiten 120–124, Oktober 1997.

[Lan98a] Barbara Lange. Bücher über Cybergeld – Online gebucht. *iX*, Seiten 78–79, Februar 1998.

[Lan98b] Barbara Lange. Mausklick-Preise: Abrechnung von Kleinstbeträgen im Internet. *iX*, Seiten 119–121, Januar 1998.

[LL96] Lynch, Daniel und Lundquist, Leslie. *digital money: The new era of internet commerce*. John Wiley n Sons, 1996.

[LL97] Lynch, Daniel und Lundquist, Leslie. *Zahlungsverkehr im Internet*. Carl Hanser Verlag, 1997.

[Loh97] Marc Lohmann. Seminarausarbeitung: Digitales Bargeld. *Universität Paderborn: Seminar SS 1997*, 1997.

[Lom96] Mark Lomas, Hrsg. *Security Protocols*. Springer Verlag, 1996.

[LR97] Le, Dinh Khoi und Rauer, Matthias. Seminarausarbeitung: Elliptische Kurven. *Universität Paderborn: Seminar SS 1997*, 1997.

[Luc97a] Norbert Luckhardt. Auf dem Weg zum Standard? *c't Report 2: Geld online*, 1997.

[Luc97b] Norbert Luckhardt. Die Mutter der Porzellankiste: Wie Kryptografie vor den Gefahren im Internet schützt. *c't Report 2: Geld online*, 1997.

[Luc97c] Norbert Luckhardt. ME-Chip: Härtere Ware? *c't Report 2: Geld online*, 1997.

[Luc97d] Norbert Luckhardt. SRT: Browser genügt. *c't Report 2: Geld online*, 1997.

[Luc97e] Norbert Luckhardt. Trend-SETter: Sichere Kartenzahlungen über unsichere Netze. *c't Report 3: Geld online*, 1997.

[Mao96a] Wenbo Mao. Light-Weight Micro-cash Payment for the Internet. In Bertino, Elisa et al. [BKMM96], Seiten 15–32.

[Mao96b] Wenbo Mao. On Cryptographic Techniques for On-line Bankcard Payment Transactions Using Open Networks. In Lomas [Lom96], Seiten 1–18.

[Mey97a] Carsten Meyer. Gold und Plastik: Was lauert unter den Kontakten einer Chipkarte? *c't Report 3: Geld online*, 1997.

[Mey97b] Carsten Meyke. Tele-Banking mit dem Home Banking Computer Interface. *Mit Sicherheit in die Informationsgesellschaft*, Seiten 445–459, 1997.

[MN93] Medvinsky, Gennady und Neumann, Clifford. NetCash: A design for practical electronic currency on the Internet. *Proceedings of the First ACM Conference on Computer and Communications Security*, 1993.

[NS78] Needham, R.M. und Schroeder, M.D. Using Encryption for Authentication in Large Networks od Computers. *Communications of the ACM, Band 21, Heft 12*, Seiten 993–999, 1978.

[Oka95] Tatsuaki Okamoto. An Efficient Divisible Electronic Cash Scheme. In Coppersmith [Cop95], Seiten 438–451.

[OO92] Okamoto, Tatsuaki und Ohta, K. Universal Electronic Cash. In Feigenbaum [Fei92], Seiten 324–337.

[Ped95] T.P. Pedersen. Electronic Payments of small amounts. *Technical Report PB-495, Aarhus University, Denmark*, August 1995.

[Ped96] Torben P. Pedersem. Electronic Payments of Small Amounts. In Lomas [Lom96], Seiten 59–68.

[RK97] Reif, Holger und Kossel, Axel. Bits statt Bares: Elektronisches Geld im Internet. *c't Report 2: Geld online*, 1997.

[RS96] Rivest, Ronald L. und Shamir, Adi. PayWord and MicroMint: Two simple micropayment schemes. *Technical Report, Massachusetts Institute of Technology, Cambridge, Massachusetts*, Mai 1996.

[RSA78] Rivest, R.L., Shamir, A. und Adleman, L. M. A method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, Februar 1978.

[Sch95] Wolfgang Schneider. Schutzregeln für den globalen Marktplatz gesucht. *NET-Spezial*, Januar 1995.

[Sch96] Bruce Schneier. *Angewandte Kryptographie*. Addison Wesley, 1996.

[Sch97] Christiane Schulzki-Haddouti. Nummer Sicher: Neue ec-Karten mit besserer Verschlüsselung. *c't Report 3: Geld online*, 1997.

[Sch98] Jeffrey Schiller. Sicherheit im Daten-Nahverkehr. *Spektrum der Wissenschaft: Dossier – Die Welt im Internet*, Seiten 52–57, 1998.

[Sel97] Frank Seliger. Informationen zur ec-GeldKarte. `http://www.darmstadt.gmd.de/~seliger/GeldKarte/exindex.html`, 1997.

[SFE97] Schuster, Rolf, Färber, Johannes und Eberl, Markus. *Digital Cash: Zahlungssysteme im Internet*. Springer Verlag, 1997.

[Sie97a] Richard Sietmann. *Electronic cash: Der Zahlungsverkehr im Internet*. Schäffer-Poeschel Verlag Stuttgart, 1997.

[Sie97b] Richard Sietmann. Zweifelsfrei authentisch: Ablösung des PIN-Codes als Zugangsschutz überfällig. *c't Report 3: Geld online*, 1997.

[SK97] Strembeck, Mark und Kloster, Rolf. Seminarausarbeitung: Abwicklung von elektronischen Transaktionen. *Universität GH Essen: Seminar SS 1997*, 1997.

[SNS88] Steiner, J.G., Neumann, B.C. und Schiller, J.I. Kerberos: An Authentication Service for Open Network Systems. *Usenix Conference Proceedings, Dallas (Texas)*, Seiten 191–202, 1988.

[SPC95] Stadler, M.A., Piveteau, J.M. und Camenisch, J.L. Fair blind signatures. In Guillou, Louis C. und Quisquater, Jean-Jaques [GQ95], Seiten 209–219.

[Ste96] Michael Steiner. Keine Angst um Ihr Geld! Entwicklung des sicheren Zahlungsverkehrs im Internet. *Neue Zürcher Zeitung (NZZ)*, Oktober 1996.

[Sto97] Markus Stolpmann. *Elektronisches Geld im Internet: Grundlagen, Kontepte, Perspektiven*. O Reilly Verlag, 1997.

[Sza97] Arnulf Szameitat. *GeldKarte - und mehr: System, Nutzen, Markt*. Deutscher Sparkassenverlag Stuttgart, 1997.

[TCH97] Tenenbaum, Jay, Chowdhry, Tripatinder und Hughes, Kevin. Eco system; An internet commerce architecture. *IEEE Computer*, Mai 1997.

[unb83] unbekannt, Hrsg. *Cryptography, Proceedings — CRYPTO '82*. Plenum Press, 1983.

[unb93] unbekannt, Hrsg. *Advances in Cryptology — CRYPTO '93*. Springer Verlag, 1993.

[unb96] unbekannt. Harte Zeiten für Schmiergeld - ecash von DigiCash. *unix/mail*, 14, 1996.

[unb98] unbekannt. Die elektronische Geldbörse ist keine Zukunftsmusik. *Computerwoche 4/98*, 1998.

[VIS97a] VISA. Book 1: Business Description. *SET Secure Electronic Transaction Specification*, Mai 1997.

[VIS97b] VISA. Book 2: Programmer's Guide. *SET Secure Electronic Transaction Specification*, Mai 1997.

[VIS97c] VISA. Book 3: Formal Protocol Definition. *SET Secure Electronic Transaction Specification*, Mai 1997.

[Wai96] M. Waidner. Development of a Secure Electronic Marketplace for Europe. In Bertino, Elisa et al. [BKMM96], Seiten 1–14.

[Way97] Peter Wayner. *Digital Cash: Commerce on the net*. Academic Press London, 1997.

[Wei97] Rüdiger Weis. Sichere Datenübertragung mit SSL. *DOS*, April 1997.

[Wei98] Rüdiger Weis. Banken prägen Software-Geld: Mit Bits bezahlen. *PC Magazin*, Seiten 234–236, Marz 1998.

[Wob97] Reinhard Wobst. *Abenteuer Kryptologie*. Addison Wesley, 1997.