



Sicheres Bezahlen in elektronischen Datennetzen

Patrick Reichert, Christoph Seidler Georg - Cantor - Gymnasium Halle (Saale)

Kurzfassung :

Zum sicheren Bezahlen in elektronischen Datennetzen existieren bis zum heutigen Zeitpunkt mehrere proprietäre Lösungsansätze (digicash, ecash), von denen sich aber keiner als Standard etablieren konnte.

Unser Lösungsansatz zeichnet sich gegenüber anderen Vorschlägen durch eine Zweiteilung aus, zum einen können Zahlungen nach dem Prinzip von Kreditkarten beziehungsweise Schecks geleistet werden, zum anderen aber auch mit sogenannten „elektronischen Münzen“. Die Sicherheit des Kunden beim „kreditkartenbasierten“ Teil soll durch eine 1024- Bit- RSA- Verschlüsselung mit einigen zusätzlichen Sicherheitsmechanismen gewährleistet werden. Der „banknotenbasierte“ Teil gewährleistet eine konsequente Anonymität des Käufers, bei gleichzeitig hoher Sicherheit für Kunden, Bank und Händler. Unser Konzept simuliert also bildlich gesprochen eine elektronische Geldbörse, bei der wahlweise mit namentlich unterzeichneten Schecks oder mit Anonymität gewährleistenden Münzen bezahlt werden kann.

Der Kunde soll sich gegenüber Händler und Bank durch einen geheimen Schlüssel identifizieren, sowie über seine Signatur verifizierbar sein. Auch Händler und Bank lassen sich über Signaturen identifizieren. Der geheime Schlüssel des Kunden soll auf einer Speicherchipkarte gespeichert werden, die von einem Trustcenter für den Kunden personalisiert wird.

Patrick Reichert (17), Fliederweg 35, 06130 Halle (Saale), 0345-4442212

Christoph Seidler (18), Uranusstraße 39, 06118 Halle (Saale), 0345-5229572

Inhalt:

1 EINLEITUNG	3
2 VORBETRACHTUNGEN ZUR PROTOKOLLARCHITEKTUR	3
2.1 Allgemeine Forderungen	3
2.2 Schlüssellänge	4
3 GEPLANTE REALISIERUNG	4
3.1 Chipkarte zur Speicherung der Teilnehmerdaten	5
3.2 Ansätze zur Verhinderung kryptographischer Angriffe	6
3.2.1 Schlüsselwahl	6
3.2.2 Random padding	7
4 KONKRETER ABLAUF EINES HANDELSPROZESSES	8
4.1 Kreditkartenbasiertes Bezahlen	8
4.2 Banknotenbasiertes Bezahlen	9
4.2.1 Erzeugung von neuen Münzen	10
4.2.2 Verifikation der Echtheit von elektronischen Münzen	11
4.2.3 Einlösen von elektronischen Münzen	11
5 TECHNISCHE VORAUSSETZUNGEN	13
6 ZUSAMMENFASSUNG, AUSBLICKE	13
7 ANHANG	14
7.1 Danksagungen	14
7.2 Literaturverzeichnis	15

1 Einleitung

Die fortschreitende Nutzung des Internets für kommerzielle Transaktionen insbesondere die Bezahlung von Waren und Dienstleistungen erfordert dringend die Entwicklung eines Protokollsystems zum digitalen Bezahlen. Derzeit wird weltweit an einer Vielzahl von möglichen Lösungen für dieses Problem gearbeitet. Man muß jedoch feststellen, daß sich bis zum jetzigen Zeitpunkt keine der vorgeschlagenen Lösungen etablieren konnte.

Unser vorliegendes Konzept soll einen weiteren Beitrag in der Diskussion um einen Standard darstellen. In unserer Arbeit werden grundlegende Kenntnisse der Theorie der asymmetrischen Verschlüsselungssysteme, digitaler Signaturen und des RSA- Algorithmus vorausgesetzt. Eine einfache Darstellung hierzu findet sich beispielsweise in [RS97] oder auch in [Beu93].

2 Vorbetrachtungen zur Protokollarchitektur

In diesem Abschnitt sollen einige allgemeine Betrachtungen zur Architektur eines möglichen Protokolls zum elektronischen Bezahlen angestellt werden.

2.1 Allgemeine Forderungen

Die wichtigsten, zunächst allgemeinen Forderungen an ein zu entwickelndes Protokoll zum sicheren elektronischen Bezahlen sind die folgenden:

Gewährleistung von:

- Sicherer Übermittlung der Transaktionsdaten durch asymmetrische oder Hybrid- Verschlüsselung
- Sicherheit des Protokolls über einen Zeitraum von mindestens 5 Jahren (gemessen am derzeitigen Forschungsstand) ohne einschneidende Veränderungen am Protokoll, um den Teilnehmern ein Mindestmaß an Investitionssicherheit zu bieten
- angemessene Geschwindigkeit bei Ver- und Entschlüsselung der anfallenden Daten, geringes Maß an Übertragungsdaten, um die Bandbreite des Netzes nicht unnötig zu belasten. Insbesondere der Verwaltungsaufwand des Protokollsystems sollte gering gehalten werden.
- Authentizität der Teilnehmer des Systems und leichte Verifikation von Teilnehmerzertifikaten
- offener Gestaltung des Protokolls, um eine problemlose Erweiterung und Fortentwicklung des Protokolls zu ermöglichen

Wünschenswert wäre weiterhin eine mögliche Anonymität des Kunden, um bestimmte Produkte und Dienstleistungen aus Datenschutzgründen unpersönlich bezahlen zu können.

2.2 Schlüssellänge

Im Falle der Verwendung des RSA- Algorithmus zur Verschlüsselung der Kreditkartendaten stellt sich die Frage nach einer geeigneten Schlüssellänge. Als Kriterium zur Beurteilung dieser Frage kann hier der Aufwand betrachtet werden, der nötig ist, eine Zahl zu faktorisieren, die Produkt zweier „großer“ Primzahlen ist. Dies ist der Weg, wie beim RSA- Algorithmus aus Kenntnis des öffentlichen Schlüssels auf den privaten Schlüssel geschlossen werden und somit eine Verschlüsselung „geknackt“ werden kann.

Der Aufwand für die Faktorisierung wird in MIPS years (MY) angegeben. Ein MY ist somit die Anzahl von Operationen, die eine Maschine, welche eine Million Integeroperationen pro Sekunde (MIPS) ausführt, in einem Jahr bewältigt. Zur Illustration: ein INTEL Pentium 100 Prozessor hat etwa 50 MIPS, einer der schnellsten Superrechner, der INTEL Paragon rund 50000 MIPS.

Bei der Verwendung des schnellsten bisher bekannten Primfaktorzerlegungsalgorithmus, des allgemeinen Zahlenkörpersiebs (generalized number field sieve, NFS) ergibt sich nach [Sch96] folgender Rechenaufwand:

Schlüssellänge in Bit	Rechenzeit in MY
512	30000
768	$2 \cdot 10^8$
1024	$3 \cdot 10^{11}$
1280	$1 \cdot 10^{14}$
1536	$3 \cdot 10^{16}$
2048	$3 \cdot 10^{20}$

Tabelle: Rechenaufwand bei Faktorisierung mit dem NFS

Ein Superrechner bräuchte also zum Entschlüsseln einer Nachricht mit einem 512 Bit Schlüssel ein knappes Jahr. Trotz des relativ geringen Informationswertes, den ein Jahr alte Transaktionsdaten haben würden, erscheint eine Schlüssellänge von 512 Bit also zu niedrig.

Andererseits muß die geringe Geschwindigkeit des RSA- Algorithmus betrachtet werden, die mit steigender Schlüssellänge weiterhin abnimmt. In Anbetracht dieser Tatsachen erscheint eine Schlüssellänge von 1024 Bit durchaus angebracht, da diese nach heutigem Stand von Wissenschaft (Faktorisierungsalgorithmen) und Technik (Rechenleistung) für einen mittelfristigen Zeitraum als sicher gelten kann.

3 Geplante Realisierung

Unser Konzept simuliert bildlich gesprochen eine elektronische Geldbörse, bei der wahlweise mit namentlich unterzeichneten Schecks oder mit Anonymität gewährleistenden Münzen bezahlt werden kann. Die elektronische Geldbörse zeichnet sich also durch eine konsequente Zweiteilung aus. Es können Zahlungen nach dem Prinzip von Kreditkarten beziehungsweise Schecks geleistet werden, aber auch das Bezahlen mit sogenannten „elektronischen Münzen“ soll möglich sein.

Die Transaktionsdaten beim kreditkartenbasierten Teil sollen mittels einer Verschlüsselung nach dem RSA- Algorithmus mit einer Schlüssellänge von zunächst 1024 Bit verschlüsselt werden. Zur Abwehr kryptographischer Angriffe wurden einige Vorkehrungen getroffen (siehe „Abschnitt Ansätze zur Verhinderung kryptographischer Angriffe“).

Der „banknotenbasierte“ Teil gewährleistet eine konsequente Anonymität des Käufers, bei gleichzeitig hoher Sicherheit für Kunden, Bank und Händler. Der Kunde identifiziert sich gegenüber Händler und Bank durch einen geheimen Schlüssel, Händler und Bank lassen sich über Signaturen identifizieren. Der geheime Schlüssel des Kunden und andere Daten so zum Beispiel Anschrift, Adresse und Bankverbindung, sollen auf einer Speicherchipkarte gespeichert werden.

3.1 Chipkarte zur Speicherung der Teilnehmerdaten

Zur Speicherung der zur Identifikation beim Einkaufsprozeß genutzten Teilnehmerdaten verwenden wir Speicherchipkarten, die mit dem SLE4428 Chip ausgestattet sind. Diese Chipkarten stellen 989 Bytes an frei nutzbarem Speicher zur Verfügung. Jeder Chip ist mit einer fest programmierten, nicht änderbaren Seriennummer versehen und kann byteweise unwiderruflich schreibgeschützt werden (Umwandlung in ROM). Mindestens 10000 Schreib- /Lesezyklen sind vom Hersteller als Lebenszeit angegeben.

Der Datensatz mit den Teilnehmerdaten wird im untenstehenden Format auf der Chipkarte gespeichert, wobei sich die ersten 6 Felder nach der Seriennummer am Aufbau einer Krankenversicherungskarte orientieren. Die weiteren Felder sind speziell auf den Einsatz beim elektronischen Bezahlen ausgerichtet, besonders wichtig ist hier der private Schlüssel des Teilnehmers.

Feldname	Länge in Bytes
ATR- Headre mit fester Seriennummer	32
Vorname	28
Name	28
Straße und Hausnummer	28
PLZ	7
Ort	22
Ländercode	3
Bankleitzahl	8
Kontonummer	16
privater Schlüssel d	128
öffentlicher Schlüssel e	128
öffentliches Modul n	128
Öffentlicher Schlüssel e des Trustcenters	128
Öffentliches Modul n des Trustcenters	128
Hashwert über Datenfelder und Seriennummer	4
Signatur des Trustcenters	128

Tabelle: Datensatzformat der Chipkarte



Abbildung: Speicherchipkarte

Eine nach obigem Schema aufgebaute Teilnehmerchipkarte wird einmalig durch ein sogenanntes Trustcenter beschrieben, welchem gegenüber der Teilnehmer durch ein Personaldokument seine Identität beweisen muß.

Nach dem Beschreiben der Karte mit den Teilnehmerdaten wird diese durch das Trustcenter signiert und schreibgeschützt. Wichtig ist hierbei, daß auch die individuelle, fest programmierte Seriennummer der Karte mit signiert wird, so daß es nicht möglich ist, Änderungen auf der Karte vorzunehmen beziehungsweise ganze Karten zu kopieren.

Da es keine zwei Karten mit identischer Seriennummer gibt, ist es nicht möglich eine identische Kopie zu erstellen. Die auf der kopierten Karte enthaltene Signatur wäre somit falsch.

Es sollte sich auch der öffentliche Schlüssel des Trustcenters mit auf der Karte befinden, um es dem Nutzer während des Einkaufsprozesses zu ermöglichen, Signaturen des Trustcenters und damit dessen Vertrauenswürdigkeit zu überprüfen.

3.2 Ansätze zur Verhinderung kryptographischer Angriffe

3.2.1 Schlüsselwahl

Um das hier teilweise benutzte RSA-Verfahren gegen kryptoanalytische Angriffe zu schützen, sollten folgende Bedingungen erfüllt sein (siehe auch [Bau95]):

1. Das benutzte Modul $n = p \cdot q$ sollte mindestens in der Größenordnung von 1024 Bit liegen, so daß eine eventuelle Faktorisierung von n erschwert wird. Die Erfüllung dieser Bedingung wurde schon in Abschnitt „*Schlüssellänge*“ diskutiert.
2. Die benutzten Primzahlen p und q unterscheiden sich in ihrer Länge um einige Stellen. Somit wird die exhaustive Suche nach einer Darstellung von n als Differenz von 2 Quadraten sofort vereitelt:
$$n = p \cdot q = [(p + q) / 2]^2 - [(p - q) / 2]^2$$
 mit ab \sqrt{n} laufenden Werten für $(p + q) / 2$. Allerdings ist auch zu beachten, daß die Differenz zwischen p und q auch nicht zu groß werden darf: sonst ist ein Faktor von n klein und somit wäre die Faktorisierung von n nicht schwer.
3. $p - 1$ und $q - 1$ enthalten nur große Primfaktoren.
4. $\text{ggT}(p - 1, q - 1)$ ist sehr klein.

Die Eigenschaften 3 und 4 sind in idealer Weise bei **sicheren Primzahlen** erfüllt. Dabei heißt eine Primzahl P genau dann sicher, wenn mit P auch $(P - 1) / 2$ eine Primzahl ist.

5. Sind auch $p' = (p - 1) / 2$ und $q' = (q - 1) / 2$ sichere Primzahlen, so bezeichnet man p und q als **doppelt sichere Primzahlen**. Sind die Bedingungen 3, 4 und 5 nicht gegeben, ist ein *Angriff durch Iteration* möglich: Die mit RSA verschlüsselte Nachricht wird vom Angreifer wiederholt mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Da alle Operationen modulo n ablaufen, wird nach endlich vielen Schritten wieder der Klartext erhalten, ohne daß der private Schlüssel d des Empfängers bekannt ist. Werden die Bedingungen 3 bis 5 nicht erfüllt, so ist die Anzahl der Iterationsschritte entsprechend gering und der Klartext kann schnell ermittelt werden. Wählt man für p und q jedoch doppelt sichere Primzahlen aus, so ist die Anzahl der Iterationsschritte so enorm groß, daß der Aufwand dieser Angriffsmethode etwa vergleichbar mit der Faktorisierung von n ist.

3.2.2 Random padding

Tritt bei der Benutzung des RSA-Verfahrens der Fall auf, daß eine Nachricht m verschlüsselt an verschiedene Personen gesendet wird, so kann ein Angreifer aus Kenntnis dieser verschlüsselten Nachrichten $m_1' = m^{e_1} \bmod (n_1)$, $m_2' = m^{e_2} \bmod (n_2), \dots, m_k' = m^{e_k} \bmod (n_k)$ mit Hilfe des Chinesischen Restsatzes auf die Originalnachricht m zurückschließen.

Um diese Klasse von Angriffen abzuwehren, muß also dafür gesorgt werden, daß keine Nachricht zweimal gleich signiert wird. Zu diesem Zweck bietet es sich an, die betreffende Nachricht immer mit einer Anzahl Zufallsbits aufzufüllen (random padding), um gleiche Signaturen zu verhindern.

Es muß immer mit einer Mindestanzahl von Zufallsbits gepadded werden, sonst könnte ein Angreifer bewußt solche Nachrichten einschleusen, die sehr kurz sind und somit nicht gepadded werden müssen. Einem möglichen Angreifer können aber durchaus Position und Mindestanzahl der Zufallsbits bekannt sein, dies dürfte ihm keinen Vorteil bei der Kryptoanalyse verschaffen.

Ein positiver Nebeneffekt des Random padding ist, daß auch sogenannte *Replay-Attacken* dadurch vermieden werden, daß es niemals zwei gleiche Nachrichten beziehungsweise Signaturen gibt. (Bei einer Replay-Attacke wird eine Nachricht, die bereits einmal gesendet wurde, erneut in das Protokoll geschleust.)

4 Konkreter Ablauf eines Handelsprozesses

4.1 Kreditkartenbasiertes Bezahlen

Folgendes Schema soll den Ablauf einer kreditkartenbasierten Transaktion verdeutlichen:

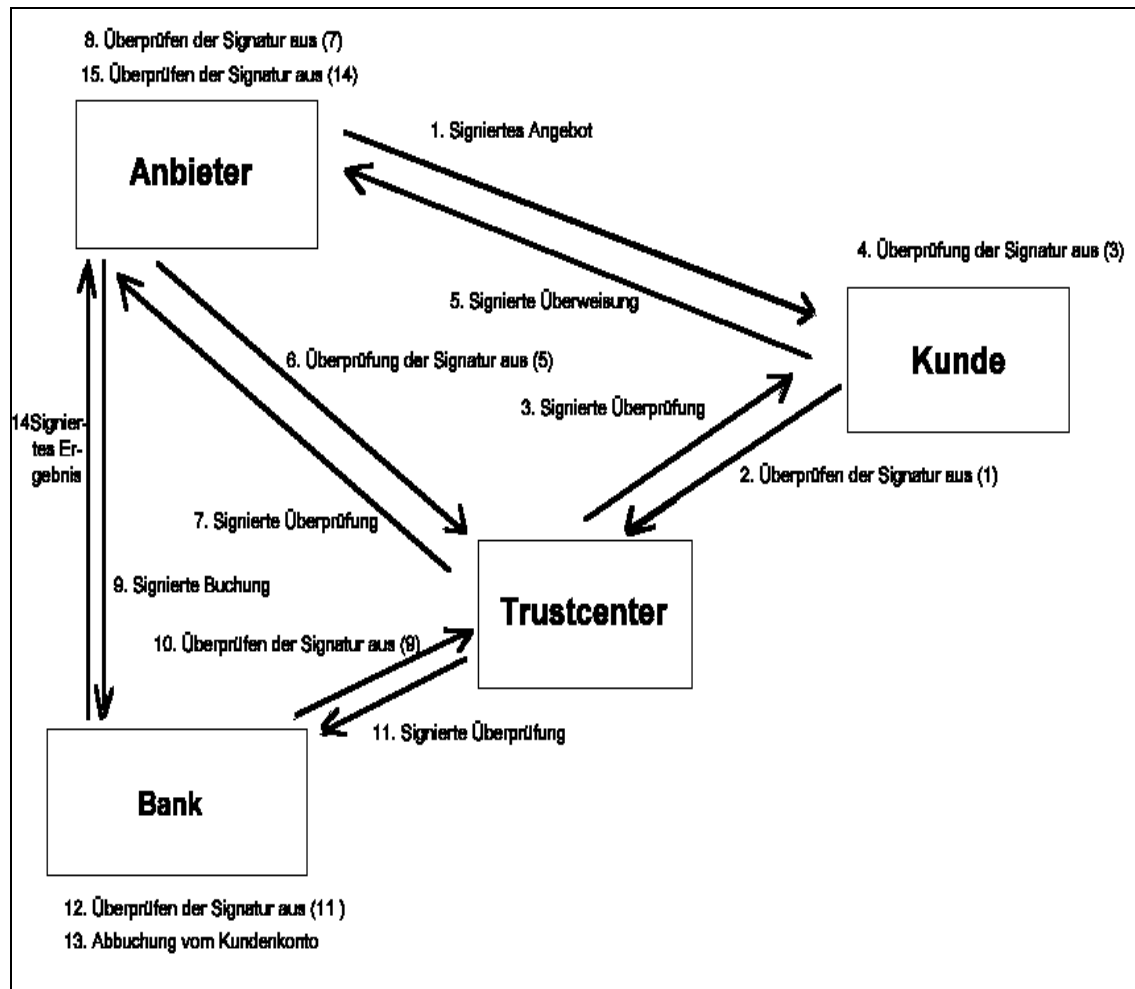


Abbildung: Nachrichtenfluß beim kreditkartenbasierten Zahlen

- Nachdem der Kunde Kaufinteresse bekundet hat, sendet ihm der Händler ein signiertes Preisangebot für diesen Artikel (1). Signiert im Laufe des Einkaufsprozesses eine Instanz eine Nachricht m mit ihrem privaten Schlüssel d , so bestimmt sie $\text{sig}(m) = m^d \bmod n$. Jeder beliebige Teilnehmer des Systems kann die Signatur überprüfen, indem er den öffentlichen Schlüssel der betreffenden Instanz auf $\text{sig}(m)$ anwendet: $\text{sig}(m)^e \bmod n = m^{ed} \bmod n = m$.
- Um sicher zu gehen, daß ihm ein korrektes Angebot vorliegt, überprüft der Kunde die Signatur des Händlers mittels dessen öffentlichen Schlüssels. Diesen erfragt er bei einem Trustcenter (2). Dieses überprüft seinerseits die Echtheit der Kundenkarte mit Hilfe der Angaben von Hashwert sowie Hashwert- Signatur auf der Karte, die der Kunde dem Trustcenter mitteilt. Als Hashfunktion können hier bei der praktischen Umsetzung beispielsweise die Verfahren MD5 oder SHA genutzt werden.
- Um die Authentizität des vom Kunden erfragten Schlüssels zu bestätigen, signiert das Trustcenter auch diese Auskunft (3). Da der öffentliche Schlüssel des Trustcenters fest auf der

Teilnehmerchipkarte gespeichert ist, ist es jedem Teilnehmer zu jedem Zeitpunkt des Prozesses möglich, die Signaturen des Trustcenters zu überprüfen (4).

- Nachdem sich der Kunde nun von der Authentizität des Angebots überzeugt hat, „unterschreibt“ er gewissermaßen einen digitalen Scheck. Mittels des öffentlichen Schlüssels des Verkäufers verschlüsselt er nun mit einer 1024 Bit RSA- Verschlüsselung eine Nachricht, die den gewünschten Betrag, sowie seinen Namen, BLZ und Kontonummer enthält. Anschließend wird die Nachricht noch signiert (5). Der Anbieter überprüft nun mittels der Informationen des Trustcenters die Signatur des Kunden (6-8).
- Nun reicht der Anbieter den erhaltenen Scheck an die Bank weiter, mit der Bitte, seinem Konto den entsprechenden Betrag gutzuschreiben (9). Nachdem die Bank die Signatur des Kunden unter dem Scheck sowie die Signatur des Händlers geprüft hat (10-12), wird der entsprechende Betrag vom Konto des Kunden abgebucht und dem Konto des Händlers gutgeschrieben (13).
- Die Bank teilt dem Anbieter nun das Ergebnis der Transaktion mit (14), so möglicherweise das Mißlingen des Einlösen eines Schecks. Der Händler überprüft schließlich die Integrität der Information der Bank mittels ihrer Signatur (15).

Es wäre nun weiterhin sinnvoll, die Nachrichten jeweils mit einem Zeitstempel zu versehen, um zu verhindern, daß beispielsweise ein Händler einen Scheck zweimal bei der Bank einreicht (replay Attacke). In der Praxis ist ein zuverlässiger Zeitstempel für Nachrichten jedoch schwer zu realisieren, da die Systemzeit beispielsweise auf Personalcomputern frei verstellt werden kann.

Statt dessen kann die Nachricht aber mit Zufallsbits aufgefüllt werden. In diesem Fall spricht man von „random padding“. Man erzielt so, wie bereits im Abschnitt „Ansätze zur Verhinderung kryptographischer Angriffe“ beschrieben, zwei positive Effekte auf einmal: zum einen werden replay Attacken verhindert, zum anderen Angriffe nach dem chinesischen Restsatz.

4.2 Banknotenbasiertes Bezahlen

Im folgenden Abschnitt soll das verwendete Verfahren zum Herstellen, Verifizieren und Einlösen von elektronischen Münzen beschrieben werden.

Dieses Verfahren besitzt folgende grundlegende Eigenschaften:

- **Authentizität:** Nur autorisierte Instanzen (etwa die Zentralbanken) dürfen Geld erzeugen können. Heutzutage besitzen nur die Nationalbanken in Deutschland das Recht, neue Münzen und Geldscheine in Umlauf zu bringen.
- **Verifikation:** Jeder Teilnehmer im System muß in der Lage sein, die Echtheit von elektronischen Münzen überprüfen zu können.
- **Anonymität:** Der Besitzer der Münze bleibt beim Bezahlen vollständig anonym, d.h. wenn eine Münze eingelöst wird, kann die Bank nicht feststellen, an wen sie diese ausgegeben hat.
- **Einmaligkeit** (Schutz gegen mehrfaches Ausgeben): Wenn eine Münze zum zweiten Mal eingelöst werden soll, kann die Bank feststellen, an wen diese Münze ausgegeben wurde.

Bemerkung: Im folgenden Abschnitt wird davon ausgegangen, daß sowohl der Kunde als auch der Verkäufer ihre Bankgeschäfte bei dem gleichen Geldinstitut abwickeln. Diese einschränkende

Bedingung kann aber leicht verallgemeinert werden, da Transaktionen zwischen verschiedenen Bankinstituten schon heute zur Realität gehören.

4.2.1 Erzeugung von neuen Münzen

Die Bank besitzt für jeden möglichen Münzwert (0,10 DM; 0,50 DM; 1 DM; 5 DM; 10 DM; 50 DM; 100 DM) je ein Münzmodul n , einen öffentlichen Münzschlüssel e und einen privaten Münzschlüssel d . Nur die Zahlen n und e werden allen Kunden bereitgestellt, die Zahl d hält die Bank streng geheim.

Das Verfahren zur Münzerzeugung kann in einem Satz zusammengefaßt werden: Ein vom Kunden erzeugter „Münzrohling“ wird durch die Bank mit Hilfe einer „blinden Signatur“ authentifiziert.

Folgende Schritte sind nun zur Münzerzeugung nötig:

1. Der Kunde wählt eine natürliche Zahl m , die folgende Informationen enthält: Eine für den Kunden eindeutige Identifizierungsnummer der Münze (32 Bit groß) und seine Identität, bestehend aus Namen, Bankleitzahl und Kontonummer; diese Informationen befinden sich auf der Chipkarte. (Es muß sichergestellt werden, daß stets $m < n$ gilt.) Würde die Identifizierungsnummer nicht in den Wert von m einfließen, dann könnte der Kunde nur eine Münze für jeden Münzwert erzeugen. Würde der Kunde nämlich 2 identische Münzen besitzen, so würde die zweite Münze nicht angenommen werden, da das hier vorgestellte System sicherstellt, daß jede Münze nur einmal eingelöst werden kann.
2. Weiterhin wählt er 32 zufällige Zahlen k_1, \dots, k_{32} und berechnet 32 Geheimtexte c_1, \dots, c_{32} mit $c_i = f(k_i, m)$. Dabei ist f eine symmetrische Verschlüsslungsfunktion (z.B. DES) mit den Schlüsseln k_i . (Hinweis: Dieses *private key* - Verfahren benötigt im Gegensatz zu den *public key* - Verfahren nur einen Schlüssel, hier k_i , zur Chiffrierung und Dechiffrierung.)
3. Anschließend werden die Werte k_i und c_i (für $i = 1..32$) gehasht, d.h. es erfolgt die Bestimmung von $h(k_i)$ und $h(c_i)$. (Zur Berechnung des Hashwertes können z.B. die Verfahren MD5, SHA oder RIP-MP benutzt werden.)
4. Die Hashwerte werden nicht im Klartext zur Bank geschickt, sondern „geblendet“. Dazu wählt der Kunde eine zufällige natürliche Zahl z , welche ein modulares Inverses z' bezüglich n besitzt. (Es muß $zz' \bmod n = 1$ gelten, solche Zahlenpaare sind mit Hilfe des euklidischen Algorithmus leicht zu finden.) Anschließend werden für jeden Index i folgende Werte bestimmt:

$$\text{blend}(h(k_i)) = h(k_i) \cdot z^e \bmod n \text{ und}$$

$$\text{blend}(h(c_i)) = h(c_i) \cdot z^e \bmod n.$$

Somit hat die Bank nicht die Möglichkeit, die Identität des Kunden und die Hashwerte $h(k_i)$ und $h(c_i)$ zu speichern um somit beim Einlösen der Münze zu bestimmen, an wen sie diese ausgegeben hat.

5. Nun beginnt eine Interaktionsphase mit der Bank, die das Ziel hat, die Identität des Kunden für die Bank zweifelsfrei festzustellen. Der Kunde sendet zunächst Angaben zu seiner Person und den gewünschten Geldbetrag der Münze zur Bank. Die Bank wählt 16 Indizes i zwischen 1 und 32 zufällig aus und fordert den Kunden auf, k_i und c_i offenzulegen. Mit Hilfe der Umkehrfunktion der symmetrischen Funktion f (siehe Schritt 2) kann die Bank das ursprüngliche m bestimmen. (Hinweis: Es gilt $m = f(k_i, c_i)$, da eine symmetrische Funktion gerade die Eigenschaft hat, mit ihrer

Umkehrfunktion übereinzustimmen.) Stimmen die in m enthaltenen Personaldaten des Kunden mit der in diesem Schritt übermittelten Daten überein, so vertraut die Bank dem Kunden und signiert ihm die restlichen 16 Zahlenpaare $\text{blend}(h(k_i))$ und $\text{blend}(h(c_i))$ mit ihrem privatem Münzschlüssel d blind, d.h. der Kunde bekommt die Werte

$$\text{sig}(\text{blend}(h(k_i))) = (h(k_i) \cdot z^e)^d \bmod n = h(k_i)^d \cdot z^{e \cdot d} \bmod n = h(k_i)^d \cdot z \bmod n \text{ und}$$

$$\text{sig}(\text{blend}(h(c_i))) = (h(c_i) \cdot z^e)^d \bmod n = h(c_i)^d \cdot z^{e \cdot d} \bmod n = h(c_i)^d \cdot z \bmod n,$$

da nach der grundlegenden Eigenschaft des RSA-Algorithmus $z^{e \cdot d} \bmod n = z$ ist.

Außerdem erfolgt eine Abbuchung des entsprechenden Geldbetrages vom Kundenkonto.

6. Der Kunde multipliziert die erhaltenen Werte mit z' und erhält die gewünschten signierten Hashwerte:

$$\text{sig}(\text{blend}(h(k_i))) \cdot z' \bmod n = h(k_i)^d \cdot z \cdot z' \bmod n = h(k_i)^d \bmod n = \text{sig}(h(k_i)) \text{ und}$$

$$\text{sig}(\text{blend}(h(c_i))) \cdot z' \bmod n = h(c_i)^d \cdot z \cdot z' \bmod n = h(c_i)^d \bmod n = \text{sig}(h(c_i)).$$

Die elektronische Münze besteht nun also aus den 16 Zahlenpaaren $(\text{sig}(h(k_i)), \text{sig}(h(c_i)))$.

4.2.2 Verifikation der Echtheit von elektronischen Münzen

Der Kunde kann den Verkäufer nun durch folgende Schritte von der Authentizität seiner Münzen überzeugen:

1. Der Kunde sendet dem Verkäufer die 16 Zahlenpaare $(\text{sig}(h(k_i)), \text{sig}(h(c_i)))$. Mit dem öffentlichen Münzschlüssel e der Bank kann dieser zunächst die Signaturen auflösen und erhält die 16 Zahlenpaare $(h(k_i), h(c_i))$.
2. Der Verkäufer erzeugt 16 Zufallszahlen aus der Menge $\{0, 1\}$ und fordert den Kunden bei einer 0 auf, den Wert k_i preiszugeben bzw. analog bei einer 1 den Wert c_i zu übermitteln. Der Kunde sendet also entweder das Urbild von $h(k_i)$ oder das Urbild von $h(c_i)$.
3. Durch Anwendung der Hashfunktion auf die gesendeten Urbilder kann der Verkäufer überprüfen, ob er eine echte Münze vor sich hat oder nicht, das Urbild ist also der Beweis für die Echtheit der Münze.

Bemerkung: Um im praktischen Fall die Anonymität des Kunden gegenüber dem Verkäufer zu gewährleisten, haben wir in unserem Protokollsystem einen „MIX“ zwischen Kunden und Verkäufer geschaltet. Der MIX hat nur die Aufgabe, vom Kunden kommende Informationen anonym an den Verkäufer weiterzuleiten; die Zieladresse wird also behalten, der Absender der Nachricht wird lediglich entfernt. Ferner kann ein MIX dazu benutzt werden, Nachrichten von verschiedenen Kunden in einer anderen Reihenfolge weiterzuleiten. Somit ist es einem eventuellen Beobachter nicht möglich zu entscheiden, welcher Kunde mit welchem Verkäufer kommuniziert hat.

4.2.3 Einlösen von elektronischen Münzen

1. Der Verkäufer reicht bei der Bank sowohl die Münze, als auch die geoffenbarten Urbilder ein, sendet also die folgenden 16 Zahlentripel zu der Bank: $(\text{sig}(h(k_i)), \text{sig}(h(c_i)), k_i \text{ oder } c_i)$.
2. Die Bank überprüft, ob diese Münze schon einmal eingereicht wurde. Wenn nicht, schreibt sie dem Verkäufer den entsprechenden Münzbetrag auf sein Konto gut.

Wenn die Bank feststellt, daß diese Münze schon einmal eingereicht wurde, so ist die Wahrscheinlichkeit nur $1/2^{16} = 1 / 65536$, daß der Verkäufer in dem zweiten Fall der Münzannahme genau die 16 entgegengesetzten Zufallsbits erzeugt hat. Es ist also sehr wahrscheinlich, daß es einen Index i gibt, so daß der Bank nunmehr sowohl der Schlüssel k_i als auch der Geheimtext c_i bekannt ist. Daraus kann sie aber m bestimmen: $m = f(k_i, c_i)$ und besitzt somit die Identität des Betrügers.

Die Anzahl der Schlüssel k_i kann theoretisch von 32 auf jeden beliebigen geraden Wert $2a$ erhöht werden. Gleichzeitig würde die Wahrscheinlichkeit, eine Münze zweimal unbemerkt einzureichen, auf den Wert $1 / 2^a$ fallen. Dieser Fakt einer „Restunsicherheit“ stellt also kein grundsätzliches Sicherheitsproblem des Systems dar.

Allgemein ergibt sich für den Nachrichtenfluß beim „banknotenbasierten“ Bezahlen folgendes Schema:

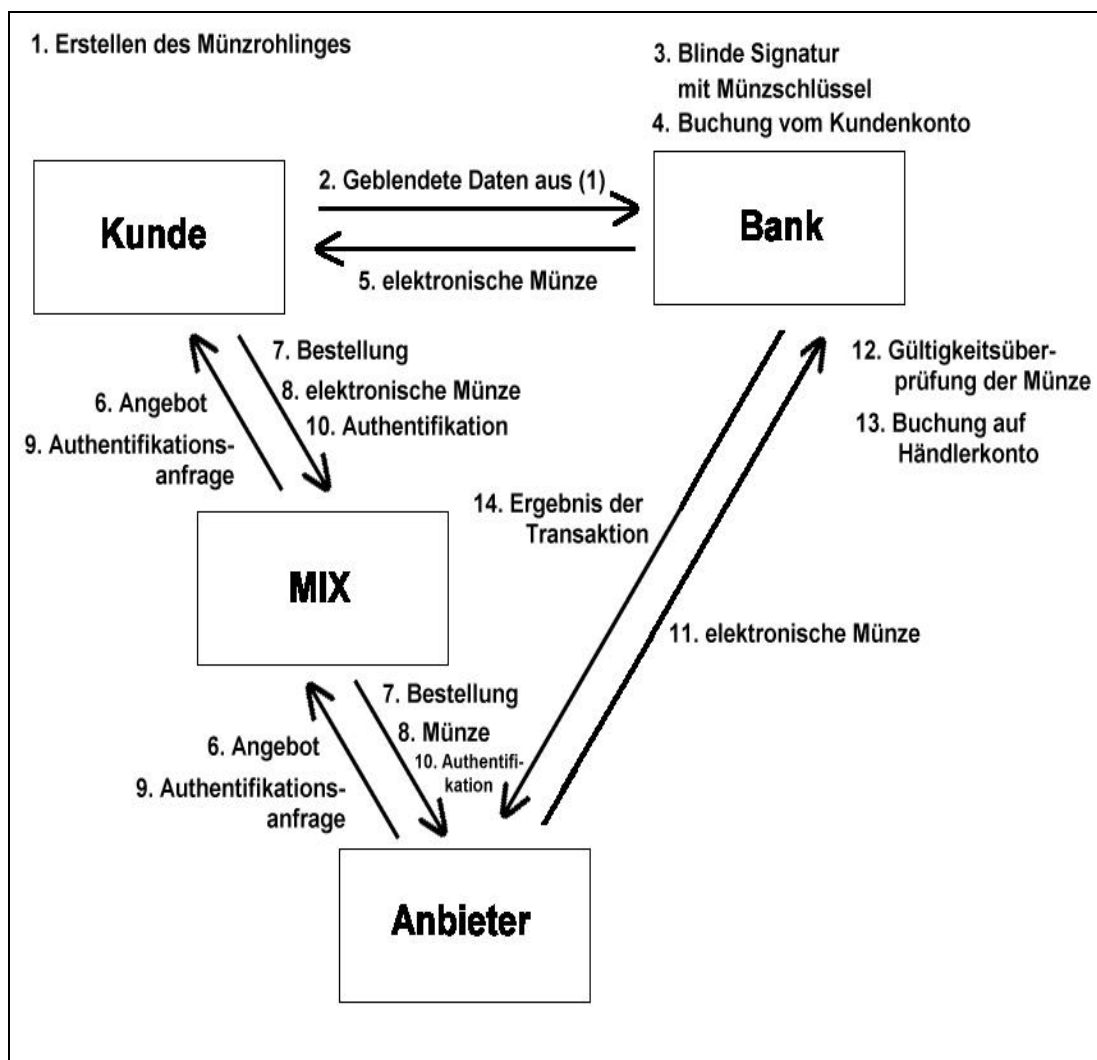


Abbildung: Nachrichtenfluß beim banknotenbasierten Zahlen

5 Technische Voraussetzungen

Zum Lesen und Beschreiben der Chipkarten verwendeten wir in unseren Versuchen ein Schreib/ Lesegerät des Typs „Kartenzwerg“ der Firma towitoko. Es handelt sich hierbei um ein externes Gerät zum Anschluß an die serielle Schnittstelle (RS 232) des Computers. Eine externe Stromversorgung für das Gerät wird nicht benötigt.

Ein Schreib/ Lesegerät dieser Art wäre für jeden Teilnehmer an dem von uns vorgeschlagenen System vonnöten. Der Preis würde bei größeren Abnahmemengen etwa 60 DM betragen. Dieser Betrag könnte beispielsweise in Form eines einmaligen Teilnehmerbeitrages auf die Kunden umgelegt werden.



Abbildung: Verwendetes Schreib/Lesegerät
(Foto: Towitoko electronics GmbH)

Außerdem wird für jeden Teilnehmer eine persönliche Chipkarte mit dem SLE4428 Chip benötigt. Bei größeren Abnahmemengen liegt der Stückpreis hier etwa bei 10 DM.

Die Funktion eines Trustcenters, welches für die Personalisierung der Chipkarten und die Verwaltung der öffentlichen Schlüssel zuständig ist, könnten in der Praxis beispielsweise speziell dafür ausgerüstete Notare übernehmen.

6 Zusammenfassung, Ausblicke

Das von uns vorgestellte System erfüllt alle prinzipiellen Anforderungen an ein System zu sicheren elektronischen Bezahlen. Festzustellen bleibt aber im Vergleich zum „herkömmlichen“ Bezahlen mit Hilfe von Bargeld ein massiv erhöhter Protokollaufwand. Es wird klar, daß die elektronische Umsetzung eines so einfach erscheinenden Prozesses wie der des Bezahlens mit Bargeld durchaus langwierig ist.

Während der Teil des „kreditkartenbasierten“ Zahlens bereits zum jetzigen Entwicklungszeitpunkt ohne weiteres in einem unsicheren Netzwerk umsetzbar ist, ergeben sich beim „banknotenbasierten“ Teil noch einige Punkte, deren theoretische Ausschärfung noch vonnöten ist:

1. In unserem Beispiel nahmen wir an, daß Kunde und Händler ihre Geschäfte mit der selben Bank machen, praktisch wäre hier noch ggf. eine Verständigung der Banken untereinander nötig.
2. Weiterhin haben wir in unserem Beispiel angenommen, daß die Bank eine Vertrauensinstanz darstellt und ihre Kunden nicht betrügt. Es wäre beispielsweise möglich, daß die Bank fälschlicherweise behauptet, eine eingereichte Münze sei bereits einmal eingelöst worden.
3. Es ist zur Zeit nicht möglich, eine elektronische Münze in kleinere Einheiten zu teilen, etwa eine 10 DM Münze in zwei 5 DM Münzen, oder umgekehrt mehrere kleine Münzwerte zu einer

größeren Münze zusammenzufassen. Im Umgang mit „realem“ Geld ist dies jedoch auch nicht ohne weiteres möglich.

4. In dem von uns vorgestellten System muß der Händler eine an ihn gezahlte Münze sofort bei der Bank einlösen und kann diese nicht nutzen, um seinerseits mit ihr zu bezahlen (da er den Empfänger nicht von der Authentizität der Münze überzeugen kann). Weiterhin ist es dem Händler nicht möglich, die Echtheit einer Münze zu überprüfen, dies kann erst die Bank beim Einlösen des Geldstücks vornehmen.
5. Schließlich ergibt sich noch ein großes Problem, daß aus dem Fakt herrührt, daß die Bank alle bisher eingereichten Münzen speichern muß, um zu überprüfen, ob die betreffende Münze bereits eingereicht wurde. Diese Überprüfung braucht bei einem entsprechend großen Datenbestand eine angemessene Zeit und Rechenleistung. Teilweise könnte das Problem dadurch gelöst werden, daß die Bank eventuell nur einen Hashwert der bereits eingereichten Münzen speichert und nicht die kompletten Münzen.

Trotz dieser noch zu lösenden Probleme bleibt festzustellen, daß elektronisches Bezahlen mit einem System wie vorgestellt bereits möglich ist und der Sicherheitsstandard bei finanziellen Transaktionen beispielsweise im Internet bereits mittelfristig entscheidend verbessert werden kann.

7 Anhang

Alle verwendeten Warenzeichen sind Eigentum ihrer Besitzer und wurden in unserer Arbeit ohne Rücksicht auf Schutz- und Patentlage verwendet.

7.1 Danksagungen

Bedanken möchten wir uns bei Prof. Albrecht Beutelspacher, Universität Gießen, der uns bei der Erstellung der Arbeit mit freundlichen Hinweisen und Anregungen unterstützte.

7.2 Literaturverzeichnis

- [Bau95] Friedrich L. Bauer, *Entzifferte Geheimnisse*, Springer Verlag, Berlin Heidelberg 1995
- [Beu93] Albrecht Beutelspacher, *Kryptologie*, Verlag Vieweg, Braunschweig/Wiesbaden 1993
- [Beu97] Albrecht Beutelspacher, *Geheimschriften*, Verlag, C. H. Beck, München 1997
- [BSW95] Albrecht Beutelspacher, Jörg Schwenk, Klaus- Dieter Wolfenstetter, *Moderne Verfahren der Kryptologie*, Verlag Vieweg, Braunschweig/ Wiesbaden 1995
- [Knu97] Donald E. Knuth, *The Art of Computer Programming Volume 2: Seminumerical Algorithms*, Third Edition, Addison Wesley, Reading 1997
- [Rob95] Matt J. B. Robshaw, *Security Estimates for 512- Bit RSA*, matt@rsa.com 1995
- [RSA78] Ronald Rivest, Adi Shamir, Leonard Adleman, *A Method for Obtaining Digital Signatures and Public Key Crypto Systems*, in: *Communications of the ACM* Heft 21 Nummer 2 Februar 1978
- [RS97] Patrick Reichert, Christoph Seidler, *Rechnen mit großen natürlichen Zahlen am Beispiel der Implementation eines RSA- Kryptosystems*, in: *Wurzel* Heft 9+10/ 97
- [Sch96] Bruce Schneier, *Angewandte Kryptographie*, Addison- Wesley, Bonn 1996
- [VV96] Farnesco P. Volpe, Sanfiaz Volpe, *Chipkarten*, Heise, Hannover 1996
- [We96] Ingo Wegener (Hrsg.), *Highlights aus der Informatik*, Springer- Verlag, Berlin/ Heidelberg 1996