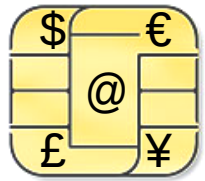
# Untracable Off-line Cash in Wallets with Observers


Seminar Kryptografie

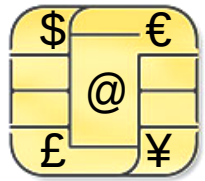
SS 2006




Tim Schweisgut

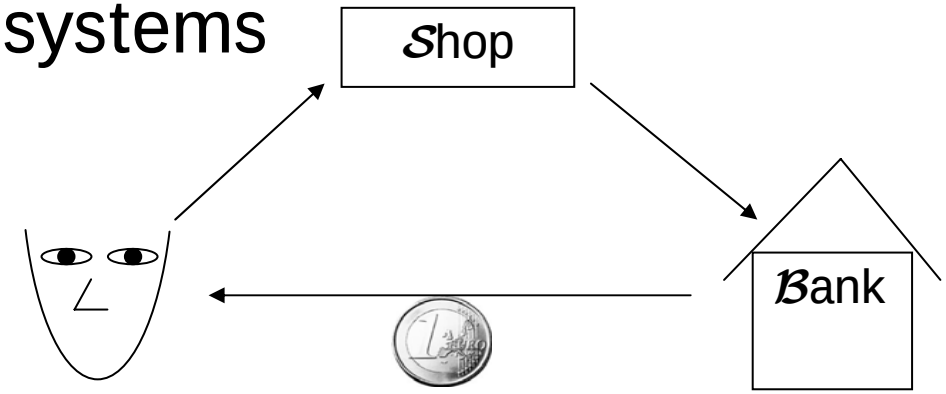
# Inhalt

- Einleitung
  - was sind elektronische offline Geldsysteme
  - was ist zu beachten
  - Double-Spending / Schutz der Privatsphäre
- Brands Geldsystem
  - Darstellungsproblem
  - restriktives Blenden
  - Schnorr Signatur
  - Protokolle
- Wallets mit Observern

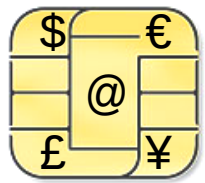


# Elektronische Geldsysteme I

- Aufbau eines Geldsystems
- nur ein Münzwert
- Online / Offline

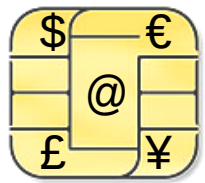



  - online: Die Bank ist permanent online und überprüft die Echtheit der Münzen und sucht nach Double-Spending
    Aus Effizienzgründen nur bei hohen Beträgen
  - offline: Bank überprüft erst später beim Einzahlen



# Elektronische Geldsysteme II

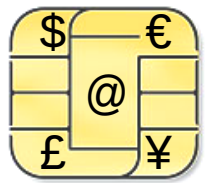
- Bank gibt signierte elektronische Münzen aus
- Privatsphäre muss gewahrt bleiben – wie bei Bargeld
  - Dies wird durch blinde Signaturverfahren erreicht
- elektronische Münzen sind leicht kopierbar =>
  - Double-Spending muss verhindert oder zumindest aufgedeckt werden
  - bei online-Systemen kein Problem



# Elektronische Geldsysteme III

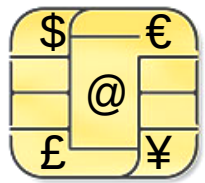
## Double-Spending-Protection

- bei offline-Systemen gibt es verschiedene Möglichkeiten:
  - Verhindern von Double-Spendern im Voraus durch Wallets mit Observern (z.B. Smartcards)
  - Aufdecken von Double-Spendern im Nachhinein (*Double-Spending-Detection*)
    Die Herkunft der Münze wird für die Bank erst sichtbar, wenn sie zum zweiten mal ausgegeben wird



# Brands Geldsystem

- von Stefan Brands 1993 vorgestellt
- das Basismodell verhindert Double-Spending im Nachhinein
- kombinierbar mit Wallets mit Obsververn
- beruht auf:
  - Darstellungsproblem in Gruppen mit Primzahlordnung
  - einem restriktivem blinden Signaturverfahren



# Darstellungsproblem in Gruppen mit Primzahlordnung I

- Betrachten Gruppen $G_q$ wobei q prim
- **Definition (*Erzeugertupel*):**

Sei $k \geq 2$. Ein Erzeugertupel der Länge $k$ ist ein $k$-Tupel $(g_1,...,g_k)$ mit $g_i \in G_q \setminus \{1\}$ und $g_i \neq g_j$ für $i \neq j$.

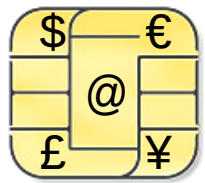
Für jedes $h \in G_q$ ist das Tupel $(a_1,...,a_k)$ eine Darstellung von $h$ bzgl. eines Erzeugertupels $(g_1,...,g_k)$

wobei $a_i \in \mathbb{Z}_q$ für $i = 1,...,k$ so dass $h = \prod_{i=1}^{k} g_i^{a_i}$.

Beispiel 1: $h = 1$

$(0,...,0)$ also $a_1 = a_2 = ... = a_k = 0$ ist eine mögliche Darstellung bzgl. eines beliebigen Erzeugertupels, denn:

$h = 1 = g_1^0 \bullet g_2^0 \bullet ... \bullet g_k^0$




Tim Schweisgut

# Darstellungsproblem in Gruppen mit Primzahlordnung II

Beispiel 2:

$k = 3$

$G_q = \mathbb{Z}_7$

Erzeugertupel: $(2,4,6)$

$h = 6$

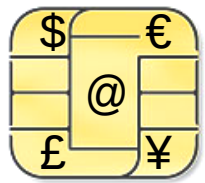
gesucht: Darstellung von $h$ bzgl. $(2,4,6)$

$h = 6 = 2^{a_1} \bullet 4^{a_2} \bullet 6^{a_3}$ mit $a_i \in \mathbb{Z}_7$

mögliche Lösung: $a_1 = a_2 = a_3 = 1$ denn $2^1 \bullet 4^1 \bullet 6^1 = 48 = 6_{\mathbb{Z}_7}$

$\Rightarrow$ Darstellung (1,1,1)

weitere Darstellung: $(0,0,1)$ denn $2^0 \bullet 4^0 \bullet 6^1 = 6$



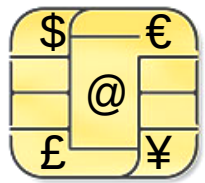

Tim Schweisgut

# Darstellungsproblem in Gruppen mit Primzahlordnung III

- => mehrere Darstellungen einer Zahl bzgl. eines Generatortupels
- **Satz:**
  Für alle $h \in G_q$ und alle Generatortupel der Länge k gibt es genau $q^{k-1}$ Darstellungen von $h$.
- **Beweis:**
  Da $G_q$ Primzahlordnung hat ist jedes $g \in G_q \backslash \{1\}$ ein erzeugendes Element der Gruppe $G_q$. Somit kann man die $a_i$ mit $i=1,\ldots,k-1$ des Darstellungstupels $(a_1,\ldots,a_k)$ frei wählen und $a_k$ ist dann eindeutig bestimmt:

$$h = g_1^{a_1} \bullet \ldots \bullet g_k^{a_k} = \prod_{i=1}^{k} g_i^{a_i} \quad \Leftrightarrow \quad g_k^{a_k} = \frac{h}{\prod_{i=1}^{k-1} g_i^{a_i}}$$

$$\Rightarrow \quad a_k = \log_{g_k} \frac{h}{\prod_{i=1}^{k-1} g_i^{a_i}}$$

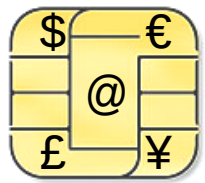



Tim Schweisgut

# Darstellungsproblem in Gruppen mit Primzahlordnung IV

- Somit gibt es $q^{k-1}$ Darstellungen von h
- insgesamt gibt es natürlich $q^k$ $k$-Tupel
- Wahrscheinlichkeit dass ein Algorithmus durch vollständige Suche Darstellung von $h \in G_q$ findet ist geringfügig mit $\frac{q^{k-1}}{q^k} = \frac{1}{q}$

- **Satz:**
  Unter der diskreten Logarithmus-Annahme gibt es keinen effizienten Algorithmus, der bei zufällig gewähltem Erzeugertupel $(g_1,\ldots,g_k)$ und einem $h \in G_q$ eine (nichttriviale, falls $h=1$) Darstellung von $h$ berechnet.

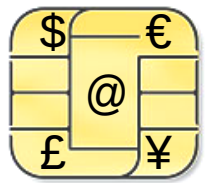


# Restriktive blinde Signatur

- Bank gibt signierte Münze aus => Münze ist authentisch
- Bank sollte aber später nicht mehr wissen wem sie welche Münze gegeben hat => blinde Signaturverfahren
- Eine Münze ist ein Tripel: $A$, $B$, sign($A$, $B$)
- sign($A$, $B$) besteht aus einem Tupel ($z$‘, $a$‘, $b$‘, $r$‘) so dass:

$$(1)\quad g^{r'} = h^{H(A,B,z',a',b')} \bullet a'$$

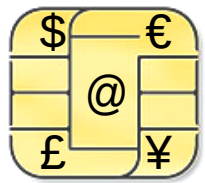
$$(2)\quad \mathrm{A}^{\mathrm{r}'} = z^{H(A,B,z',a',b')} \bullet b'$$




Tim Schweisgut

# Schnorr Signatur

- Das Abhebeprotokoll besteht im Wesentlichen aus zwei Schnorr-Signaturen
  - $g$ sei Generator von $G_q$
  - $x \in \mathbb{Z}_q^*$ der private Schlüssel des Signierers
  - $h=g^{-x}$ der öffentliche Schlüssel des Signierers
  - $\mathcal{H}$: $G_q \times G_q \rightarrow \mathbb{Z}_q^*$ kollisionsresitente Hashfunktion
  - Die Signatur $\mathrm{sign}(m)$ ist ein Tupel $(y,e)$ mit:
    - $g^y h^e = a$
    - $e = \mathcal{H}(a,m)$



# Schnorr Signatur II

- **Ablauf:**

Signierer

$m$ die zu signierende Nachricht

$u \in_R \mathbb{Z}_q^*$

$a \leftarrow g^u$

$e \leftarrow \mathcal{H}(a, m)$

$y \leftarrow u + xe$

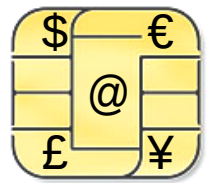
$\text{sign}(m) \leftarrow (y, e)$

$\xrightarrow{m,\ sign(m)}$

Verifizierer

$g^y h^e \overset{?}{=} a$

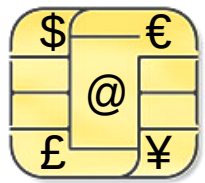
$e \overset{?}{=} \mathcal{H}(a, m)$



# Basismodell

- **Aufbau:**
  - Die Bank wird mit $\mathcal{B}$, der User mit $\mathcal{U}$ und der Shop mit $\mathcal{S}$ bezeichnet
  - Bank erzeugt Zufall-Erzeuger-Tupel: ($g$, $g_1$, $g_2$) von $G_q$
  - privater Schlüssel von $\mathcal{B}$ sei $x \in \mathbb{Z}_q$.
  - öffentlicher Schlüssel von $\mathcal{B}$ sei $h = g^x$
  - kollisionsresistente Hashfunktion für das Abheben

$$H : G_q \times G_q \times G_q \times G_q \times G_q \rightarrow \mathbb{Z}_q^*$$

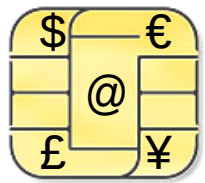


# Basismodell II

- Bei Kontoeröffnung erzeugt der User eine einzigartige Zahl $u_1 \in_R \mathbb{Z}_q$ die auf $\mathcal{U}$ registriert ist
- Wenn $\mathcal{U}$ abheben will rechnet er $I = g_1^{u_1}$ und multipliziert dies mit $g_2$.
- Somit kennt $\mathcal{U}$ eine Darstellung $(u_1,1)$ der Message $m=Ig_2$ bzgl. $(g_1,g_2)$:

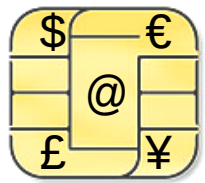
$$m = Ig_2 = g_1^{u_1} g_2^1$$

- $I$ (seine Identität schickt er an die Bank. Diese erzeugt: $z = (Ig_2)^x$ und veröffentlich z.



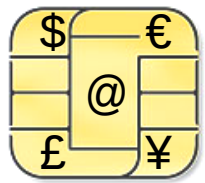
# Das Abhebeprotokoll

| $\mathcal{U}$ | | $\mathcal{B}$ |
|---|---|---|
| | | $w \in_R \mathbb{Z}_q$ |
| | | $a \leftarrow g^w$ |
|  | $\xleftarrow{a,b}$ | $b \leftarrow (Ig_2)^w$ |
| $s \in_R \mathbb{Z}_q^*$ | | |
| $A \leftarrow (Ig_2)^s$ | | |
| $z' \leftarrow z^s$ | | |
| $x_1, x_2, u, v \in_R \mathbb{Z}_q$ | | |
| $B \leftarrow g_1^{x_1} g_2^{x_2}$ | | |
| $a' \leftarrow a^u g^v$ | | |
| $b' \leftarrow b^{su} A^v$ | | |
| $c' \leftarrow \mathcal{H}(A, B, z', a', b')$ | | |
| $c \leftarrow \frac{c'}{u} \bmod q$ | $\xrightarrow{c}$ | |
| | | $r \leftarrow cx + w \bmod q$ |
| $g^r \stackrel{?}{=} h^c a$ | $\xleftarrow{r}$ | |
| $(Ig_2)^r \stackrel{?}{=} z^c b$ | | |
| $r' \leftarrow ru + v \bmod q$ | | |



# Das Abhebeprotokoll II

- Das Protokoll beruht auf zwei Schnorrsignaturen:
  - die erste sorgt dafür dass nur $\mathcal{B}$ Münzen verteilen kann
  - die zweite bezieht sich auf die geblendete Nachricht $A = (Ig_2)^s = m^s$
- Damit Challenge-Response funktioniert kann $\mathcal{U}$ die Nachricht $m$ nur durch potenzieren mit einer Zufallszahl $s \in_R \mathbb{Z}_q$ blenden
- => restriktives Signaturverfahren



# Das Abhebeprotokoll III

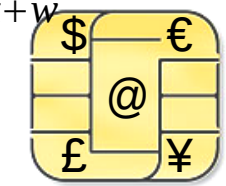
- Signaturverfahren nicht existentiell fälschbar
- Durchführbarkeit des Protokolls:

$$(1)\quad g^r = g^{cx+w} = g^{\frac{c'}{u}x+w} \stackrel{?}{=} h^c a = g^{x^c} a = g^{x\frac{c'}{u}} a$$

$$= g^{\frac{c'}{u}x} g^w = g^{\frac{c'}{u}x+w}$$

$$(2)\quad \left(Ig_2\right)^r = \left(Ig_2\right)^{\frac{c'}{u}x+w} \stackrel{?}{=} z^c b = \left(Ig_2\right)^{xc} b = \left(Ig_2\right)^{\frac{c'}{u}x} b$$

$$= \left(Ig_2\right)^{\frac{c'}{u}x} \left(Ig_2\right)^w = \left(Ig_2\right)^{\frac{c'}{u}x+w}$$

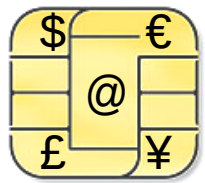


# Das Bezahlprotokoll

- weitere Hashfunktion nötig:

$$\mathcal{H}_0 : G_q \times G_q \times \text{Shop-ID} \times \text{Datum/Zeit} \rightarrow \mathbb{Z}_q$$

- Jeder Shop hat also Shop-ID $I_S$
- Durch Shop-ID und Datum-Zeit Stempel garantiert man richtige Zuordnung der Bezahlung



# Das Bezahlprotokoll II

$\mathcal{U}$ $\mathcal{S}$

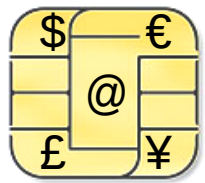
$\xrightarrow{A,B,sign(A,B)}$ $A \overset{?}{\neq} 1$

$d \leftarrow \mathcal{H}_0\left(A, B, I_{\mathcal{S}}, date/time\right)$

$r_1 \leftarrow d(u_1 s) + x_1 \bmod q$ $\xleftarrow{d}$

$r_2 \leftarrow ds + x_2 \bmod q$ $\xrightarrow{(r_1, r_2)}$ überprüfe $sign(A, B)$

$g_1^{r_1} g_2^{r_2} \overset{?}{=} A^d B$

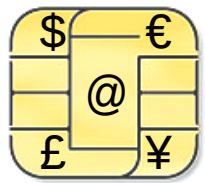


# Das Bezahlprotokoll III

- Durchführbarkeit

$$g_1^{r_1} g_2^{r_2} = g_1^{du_1 s + x_1} g_2^{ds + x_2} \stackrel{?}{=} A^d B = \left(m^s\right)^d g_1^{x_1} g_2^{x_2} = \left(I g_2\right)^{sd} g_1^{x_1} g_2^{x_2}$$

$$= I^{sd} g_2^{sd} g_1^{x_1} g_2^{x_2} = g_1^{u_1 sd} g_2^{sd + x_2} g_1^{x_1} = g_1^{du_1 s + x_1} g_2^{ds + x_2}$$



# Das Bezahlprotokoll IV

- **Korrektheit:**
  Angenommen $\mathcal{U}$ kann auf zwei Challenges d und d‘ antworten. Dann gibt es $(r_1,r_2)$ und $(r_1‘,r_2‘)$ mit:

$$\left.\begin{aligned} g_1^{r_1} g_2^{r_2} &= A^d B \\ g_1^{r_1'} g_2^{r_2'} &= A^{d'} B \end{aligned}\right\} \Rightarrow \begin{aligned} g_1^{r_1} g_2^{r_2} A^{-d} &= B \\ g_1^{r_1'} g_2^{r_2'} A^{-d'} &= B \end{aligned}$$

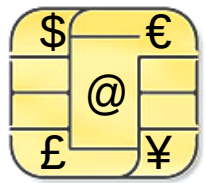
$$\Rightarrow$$

$$g_1^{r_1} g_2^{r_2} A^{-d} = g_1^{r_1'} g_2^{r_2'} A^{-d'}$$

$$g_1^{r_1} g_1^{-r_1'} g_2^{r_2} g_2^{-r_2'} A^{-d} A^{d'} = 1$$

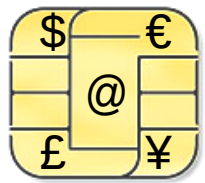
$$g_1^{r_1-r_1'} g_2^{r_2-r_2'} \left(g_1^{u_1 s} g_2^{s}\right)^{d-d'} = 1$$

$$g_1^{r_1-r_1'+u_1 s(d'-d)} g_2^{r_2-r_2'+s(d'-d)} = 1$$



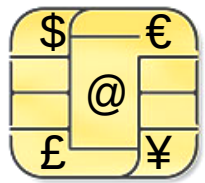
# Einzahlung

- wegen offline mit Zeitverzögerung
- $\mathcal{S}$ sendet die Daten $A$, $B$, $\text{sign}(A,B)$, $(r_1,r_2)$, Zeitstempel der Bezahlung an $\mathcal{B}$
- $\mathcal{B}$ erzeugt selbst nochmals d
- $\mathcal{B}$ überprüft ob $g_1^{r_1} g_2^{r_2} = A^d B$ und ob $\text{sign}(A,B)$ eine gültige Signatur ist
- Wenn Verifikation fehlschlägt wird Einzahlung nicht akzeptiert. Sonst:



# Einzahlung II

- Wenn Verifikation gelingt überprüft $\mathcal{B}$, ob $A$ schon in Datenbank vorhanden, also ob mit Münze irgendwann schon bezahlt wurde
  - $A$ noch nicht in Datenbank => $\mathcal{B}$ speichert (A, Zeitstempel, $r_1$, $r_2$) und schreibt $\mathcal{S}$ das Geld gut
  - $A$ schon in Datenbank:
    - Wenn $I_{\mathcal{S}}$ und Zeitstempel mit Datenbankeintrag identisch: Shop versucht Münze erneut einzulösen
    - Wenn Challenges unterschiedlich: Double-Spending!



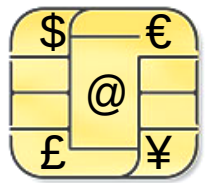
# Einzahlung III

- Die Bank hat nun $(d, r_1, r_2)$ von der zweiten Zahlung und $(d', r_1', r_2')$ von der ersten Zahlung und kann nun

$$g_1^{\frac{r_1 - r_1'}{r_2 - r_2'}} = g_1^{u_1} = I$$

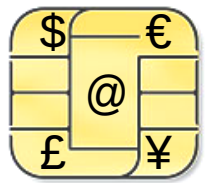
$$\frac{r_1 - r_1'}{r_2 - r_2'} = \frac{du_1 s + x_1 - d' u_1 s - x_1}{ds + x_2 - d' s - x_2} = \frac{du_1 - d' u_1}{d - d'} = u_1$$

berechnen und somit die Identität des Double-Spenders aufdecken



# Double-Spending

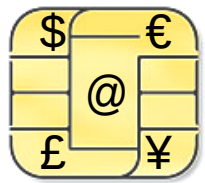
- Die Bank kann somit Double-Spender nach ihrer Tat finden, da man beim Bezahlen einen Punkt einer Gerade preisgibt, die zu seiner Identität führt.
- Sobald der zweite Punkt $(r_1,r_2)$ bekannt wird, kann die Gerade bestimmt werden.
- Wenn $\mathcal{U}$ sich aber korrekt verhält kann Bank seine Zahlungen nicht verfolgen




Tim Schweisgut

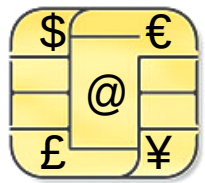
# Wallets with Observer

- Um gar nicht erst die Möglichkeit zum Double-Spending zu bieten kam Idee des Observers auf
- $\mathcal{U}$ bekommt eine Geldbörse (z.B. Smartcard) von der Bank, mit der er bezahlt
- die Bezahlung muss erst von Hardware genehmigt werden
- Observer überprüft, ob Münze bereits verwendet. Wenn ja verweigert er die Zahlung



# Wallets with Observer II

- diese Hardware sollte manipulationssicher sein
- sollte es dennoch gelingen den Observer zu manipulieren, werden Double-Spender von $\mathcal{B}$ nach ihrer Tat entlarvt, wie im Basismodell



# Quellen

- Stefan Brands, Untracable Off-line Cash in Wallets with Observers, Crypto 1993
- Stefan Brands, An Efficient Off-line electronic cash system based on the Representation Problem, 1993