Seminar

Kryptologie

Untracable Off-line Cash in Wallets with Observers

Mathematik

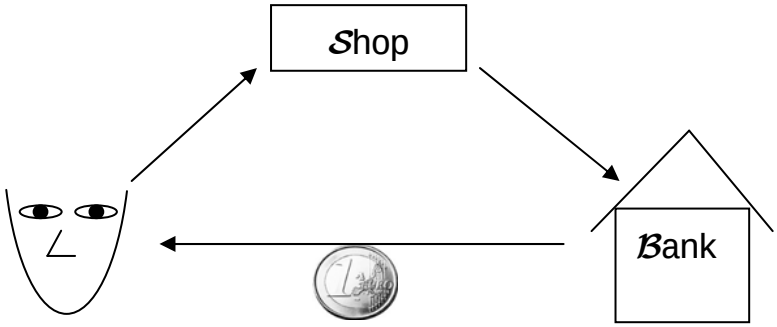
Sommersemester 2006

Inhalt

Einleitung

Was sind Off-line Geldsysteme

Das Internet hat sich in den letzten Jahren sehr stark verbreitet und im Zusammenhang damit werden auch viel mehr Geschäfte über das Internet getätigt. Hier gibt es meistens die Bezahlmöglichkeiten: Nachnahme, Vorkasse, und sehr häufig Kreditkarte. Nur sehr selten bekommt man die Möglichkeit als privater Nutzer per Rechnung zu bezahlen. Kreditkartenzahlungen sind zwar sehr bequem, allerdings auch sehr riskant. Das größte Risiko besteht bei der nicht vorhandenen Authentifikationsmöglichkeit beim Bezahlen.
Aus diesem Grund versucht man weiterhin elektronische Geldsysteme zu entwickeln, mit denen gewährleistet ist, dass die Kunden anonym bleiben, keine sensiblen Daten übertragen werden müssen und es keine Möglichkeit gibt, Geld zu bekommen, was einem nicht zusteht.



Bei einem elektronischen Geldsystem gibt die Bank elektronische Münzen an die Kunden aus. Diese Münzen werden von der Bank signiert. Durch diese Signatur wird sichergestellt, dass die Münzen echt sind und nicht gefälscht werden können.
Beim Betrachten von elektronischen Geldsystemen geht man davon aus, dass es nur einen Münzwert gibt. Es lassen sich aber durch Verwendung verschiedener Schlüsselpaare auch unterschiedliche Münzwerte realisieren.
Man unterteilt elektronische Geldsysteme in *offline* und *online Systeme*. Bei online Systemen ist die Bank (zu der das Geld früher oder später wieder zurück muss) jederzeit online und überprüft, ob die Münzen mit denen gehandelt wird echt sind und bisher noch nicht zuvor zum Bezahlen verwendet wurden. Online-Systeme sind hauptsächlich für große Beträge geeignet.
Bei offline Systemen überprüft die Bank irgendwann, zeitlich versetzt zum Bezahlen, nämlich wenn der Shopbesitzer die Münze einlösen will, ob die Münzen echt sind.

Was ist bei elektronischen Geldsystemen zu beachten

Es gibt zwei wesentliche Punkte, auf die man bei der Entwicklung eines elektronischen Geldsystems achten sollte. *Double-Spending* und *Privacy-Protecting*.

Double-Spending

Elektronisches Geld besteht natürlich nur aus Bits und Bytes. Münzen sind somit nur digitale Informationen welche – wenn nichts dagegen getan wird – leicht kopiert werden können. Somit könnte ein User eine Münze mit der er schon in einemShop seine Turnschuhe bezahlt hat auch noch in einem anderen Shop ein T-Shirt bezahlen. Dies nennt man auch *Double-Spending*.

Bei der Entwicklung von elektronischen Geldsystemen muss darauf geachtet werden, dass die Nutzer keine Möglichkeit haben elektronische Münzen die selbst kopiert wurden auszugeben und zu verwenden.
Bei online Systemen ist Double-Spending kein Problem, da die Bank jederzeit überprüft, ob die Münze bereits einmal verwendet wurde.
Bei offline Systemen sind zwei Varianten entwickelt worden: *Double-Spending-Protection* und *Double-Spending-Detection*. Bei der Double-Spending-Protection wird Double-Spending durch so genannte *Wallets mit Observer* im Voraus verhindert. Diese Wallets mit Observer sind zum Beispiel Smartcards, mit dessen Hilfe die User bezahlen müssen (siehe auch letztes Kapitel).
Die Double-Spending-Detection sorgt dafür, dass die Identität von Usern die eine Münze mehrfach ausgegeben haben aufgedeckt werden kann. Dies geschieht natürlich erst, wenn der Schaden schon entstanden ist.
Diese Detection sorgt aber dafür dass der User der sich legal verhält anonym bezahlen kann.

Privacy Protecting

Es ist nicht schwierig mit digitalen Signaturen ein elektronisches Geldsystem zu entwickeln. Jede Münze ist ein einzigartiges Stück digitaler Information zusammen mit der digitalen Signatur der Bank. Die Bank verfolgt den Weg aller Münzen und kann im Nachhinein feststellen, wenn eine Münze doppelt ausgegeben wurde. Auch dieses System könnte man noch mit Observern ausstatten um Double-Spending im Voraus zu verhindern. Jedoch sollte eine elektronischen Geldsystem die Anonymität des legalen Users gewährleisten, genauso wie es beim Bargeld der Fall ist. Wenn man im Geschäft mit einer Euromünze bezahlt und diese irgendwann vom Geschäftsinhaber bei der Bank eingezahlt wird, weiß die Bank nicht von wem die Münze ursprünglich kommt und was damit bezahlt wurde.
Es ist allerdings klar, dass die Sicherheit und Effizienz (aus Sicht der Bank) in einem offline-System, bei dem alle Zahlungen überwacht und rückverfolgbar sind, größer ist als in einem anonymen System.
Dies ist der zweite wichtige Punkt, der für ein elektronisches Geldsystem gelten sollte.

Brands Geldsystem

Das elektronische Geldsystem [1], welches Stefan Brands entwickelt hat, verhindert Double-Spending und wahrt die Anonymität des sich legal verhaltenen Users. Die Identität des Users wird erst dann aufgedeckt, wenn er eine Münze ein zweites Mal ausgibt. Man kann das System aber auch ohne Schwierigkeiten mit Observer-Nutzung kombinieren.
Das System beruht auf zwei Konzepten: Dem „Darstellungsproblem in Gruppen mit Primzahlordnung“ und den „restriktiven blinden Signaturverfahren“.

Darstellungsproblem in Gruppen mit Primzahlordnung

Im Folgenden betrachten wir Gruppen G_q, wobei q eine Primzahl ist.

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:

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

Diese Darstellung heißt auch *triviale Darstellung*.

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 $2^0 \bullet 4^0 \bullet 6^1 = 6$

Somit gibt es zu einem Erzeugertupel mehrere Darstellungen. Wie viele es gibt klärt folgender Satz:

Satz: Für alle $h \in G_q$ und alle Erzeugertupel der Länge k gibt es genau q^{k-1} Darstellungen von h.

Beweis: Da G_q Primzahlordnung hat ist jedes $g \in G_q \setminus \{1\}$ ein erzeugendes Element der Gruppe G_q. Somit kann man die a_i mit i=1,…,k-1 des Darstellungstupels (a_1,…,a_k) frei wählen und a_k ist dann eindeutig bestimmt:

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

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

Es gibt also q^{k-1} Darstellungen von h. Die Mächtigkeit der Menge aller k-Tupel ist q^k. Somit ist die Wahrscheinlichkeit, dass es ein Algorithmus in polynomieller Laufzeit durch eine

vollständige Suche schafft eine nichttriviale Darstellung von $h \in G_q$ zu finden $\frac{q^{k-1}}{q^k} = \frac{1}{q}$ also geringfügig.

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.

Ein Beweis zu dem Satz steht in [3].

Restriktive blinde Signaturverfahren

Wenn die Bank eine Münze ausgibt signiert sie diese. Dadurch ist gewährleistet, dass die Münze authentisch ist. Jedoch besteht noch das Problem, dass die Bank weiß, wem sie die Münze gegeben hat, und die Bezahlungen so zurückverfolgen kann. Da dies in einem elektronischen Geldsystem, welches die Privatsphäre schützt nicht der Fall sein sollte, wurden blinde Signaturverfahren entwickelt. Das Prinzip besteht darin, dass der User die Nachricht (die Münze) blendet indem er etwas dazumultipliziert, bzw. die Nachricht mit einer Zufallszahl potenziert. Somit kann die Bank die Münze nicht mehr zurückverfolgen.
Das im Folgenden verwendete System nennt man auch restriktiv, da der User beim Blenden der Nachricht eingeschränkt ist. Er kann die Nachricht nur durch potenzieren mit einer Zufallszahl $s \in_R \mathbb{Z}_q^*$ blenden. Würde er sie noch anders blenden würde das Challenge-Response-Verfahren nicht mehr funktionieren.

Protokolle

Ein elektronisches Geldsystem besteht aus drei Protokollen:

1. Dem Abheben der Münzen bei der Bank
2. Dem Bezahlen mit den Münzen bei einem Shop
3. Der Shop möchte die Münzen die er eingenommen hat wieder bei der Bank einlösen

Die blinde Schnorr Signatur

Das Abhebe-Protokoll besteht im Wesentlichen aus zwei Schnorr Signaturen. Eine blinde Schnorr Signatur funktioniert nach folgendem Schema [4]:

- g sei Generator von G_q
- $x \in \mathbb{Z}_q^*$ der private Schlüssel der Bank
- $h=g^{-x} \bmod q$ der öffentliche Schlüssel der Bank
- $\mathcal{H}: \mathbb{Z}_q \times G_q \rightarrow \mathbb{Z}_q^*$ kollisionsresitente Hashfunktion
- A ist die zu signierende Nachricht
- Die Signatur sign(A) ist ein Tupel (y,e) mit:
  - $g^y h^e = a'$
  - $e = \mathcal{H}(a', A)$

Der Ablauf des Schnorr-Signatur-Schemas:

User		Bank
		$w \in_R \mathbb{Z}_q$
	$\xleftarrow{a}$	$a \leftarrow g^w$
$u, v \in_R \mathbb{Z}_q$		
$a' \leftarrow a^u g^v := g^{uk+v}$		
$c' \leftarrow \mathcal{H}(a, A)$		
$c \leftarrow \frac{c'}{u} \bmod q$	$\xrightarrow{c}$	$r \leftarrow cx + w \bmod q$
	$\xleftarrow{r}$	
$r' \leftarrow ru + v \bmod q$		
$\text{sign}(A) \leftarrow (c', r')$	$\xrightarrow{A,\ sign(A)}$	$g^{r'} h^{c'} \overset{?}{=} a^u g^v = a'$

Aufbau des Systems

Eine Münze im Nachfolgenden Geldsystem ist ein Tripel *A*, *B*, sign(*A*,*B*).
sign(*A*,*B*) besteht aus einem Tupel (*z'*, *a'*, *b'*, *r'*) so dass:

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

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

Wie dies funktioniert wird deutlich, wenn man sich die Protokolle des Geldsystems anschaut.

Die Bank wird im Folgenden mit $\mathcal{B}$ und der User mit $\mathcal{U}$ bezeichnet.
Die Bank erzeugt ein Zufallgeneratortupel (g,g_1,g_2) von G_q und sich selbst einen privaten Schlüssel $x \in \mathbb{Z}_q$.
Der öffentliche Schlüssel der Bank sei $h=g^x$
Für das Abheben benötigt man noch eine kollisionsresistente Hashfunktion:
$H : G_q \times G_q \times G_q \times G_q \times G_q \to \mathbb{Z}_q^*$
Sowohl G_q als auch das Erzeugertupel und die Hashfunktion sind öffentlich.

Wenn die Bank nun ein Konto eröffnet wählt sich der User per Zufall eine einzigartige Zahl $u_1 \in \mathbb{Z}_q$. Der User berechnet $g_1^{u_1} = I$ und multipliziert dies mit g_2. Dies geschieht einmal bei der Kontoeröffnung. Somit kennt der User eine Darstellung (u_1,1) der Zahl $m=Ig_2$ bzgl (g_1, g_2): $m = Ig_2 = g_1^{u_1} g_2^1$.
I schickt der User dann an die Bank und hält u_1 geheim. Die Bank errechnet $z = \left(Ig_2\right)^x$ und schickt *z* an den User. Die Bank kennt also nicht $\log_{g_1} I = u_1$.

Abhebe-Protokoll

Nach diesen Vereinbarungen kann nun der Ablauf des Abhebeprotokolls erklärt werden:

Schritt 1: $\mathcal{B}$ wählt $w \in_R \mathbb{Z}_q$ und sendet $a=g^w$ und $b=m^w=(Ig_2)^w$ and $\mathcal{U}$.

Schritt 2: $\mathcal{U}$ wählt sich Zufallszahlen: $s \in_R \mathbb{Z}_q^*$,

$$x_1, x_2, u, v \in_R \mathbb{Z}_q$$

Anschließend blendet $\mathcal{U}$ $m=Ig_2$ zu A mit $(Ig_2)^s=A$ und z zu $z'=z^s$

$\mathcal{U}$ erzeugt $B = g_1^{x_1} g_2^{x_2}$

Des weiteren werden a zu $a'=a^u g^v$ und b zu $b'=b^{su}A^v$ geblendet

Mit diesen Werten kann $\mathcal{U}$ die Challenge $c'=H(A,B,z',a',b')$

Zum Schluss wird c' geblendet zu $c = \frac{c'}{u} \bmod q$ und c wird an $\mathcal{B}$ geschickt

Schritt 3: Die Bank berechnet die Response $r=cx+w \bmod q$ und schickt diese an $\mathcal{U}$.

Schritt 4: $\mathcal{U}$ akzeptiert die Antwort nur, wenn $g^r=h^c a$ und $m^r=(Ig_2)^r=z^c b$ gilt.

Stimmen diese Gleichungen hat der User eine gültige Signatur erhalten und berechnet sich noch $r'=ru+v \bmod q$ und besitzt dann die Münze $(A, B, (z',a',b',r'))$

Das Protokoll beruht auf zwei Schnorr-Signaturen. Die erste Signatur sorgt dafür, dass nur die Bank Münzen ausstellen kann, da der private Schlüssel der Bank dazu benötigt wird. Die zweite Signatur bezieht sich auf die geblendete Nachricht $A=(Ig_2)^s=m^s$. Hierdurch steckt die Identität des Users in jeder Münze und es wird – wie später erklärt – möglich sein Double-Spender zu entlarven.
Dieses blinde Signaturverfahren ist nicht existentiell fälschbar, selbst wenn bereits polynomiell viele Signaturen erzeugt wurden.

Das Protokoll im Überblick:

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

Dieses Protokoll ist durchführbar denn die beiden Gleichungen die zum Schluss überprüft werden stimmen bei korrekter Durchführung:

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

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

Bezahl-Protokoll

Das nächste Protokoll in einem elektronischen Geldsystem ist natürlich das Bezahlprotokoll, denn die man hebt das Geld in der Regel nur von der Bank ab, wenn man vor hat dieses über kurz oder lang auszugeben.
Für das Bezahlprotokoll wird eine weitere kollisionsresistente Hashfunktion benötigt, welche die Bank zur Verfügung stellt:

$$H_0 : G_q \times G_q \times \text{Shop-ID} \times \text{Datum/Zeit} \rightarrow \mathbb{Z}_q$$

Jeder Shop hat also eine einmalige ID. Hierdurch wird gewährleistet, dass jeder Shop mit großer Wahrscheinlichkeit unterschiedliche Challenges erzeugt. Zusätzlich werden die Zahlungen mit einem Datum/Zeit-Stempel versehen, was dafür sorgt, dass ein Shop für jede Zahlung eine andere Challenge erzeugt.

Der Ablauf des Protokolls:

Schritt 1: Der User schickt seine Münze *A*, *B* und die Signatur sign(*A*,*B*) an den Shop $\mathcal{S}$.

Schritt 2: Der Shop erzeugt die Challenge $d=H_0(A, B, I_S, \text{Date/Time})$ und sendet *d* an $\mathcal{U}$.

Schritt 3: $\mathcal{U}$ erzeugt die Antworten r_1 und r_2 und schickt diese an $\mathcal{S}$ zurück.

$r_1=d(u_1 s)+x_1 \bmod q$
$r_2=ds+x_2 \bmod q$

Der Shop akzeptiert die Bezahlung, wenn sign(*A*,*B*) wirklich eine Signatur ist und $g_1^{r_1} g_2^{r_2} = A^d B$ gilt.

Das Protokoll im Überblick:

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A,B,sign(A,B)}$	$A \overset{?}{\neq} 1$
		$d \leftarrow \mathcal{H}_0(A, B, I_{\mathcal{S}}, date/time)$
$r_1 \leftarrow d(u_1 s) + x_1 \bmod q$	$\xleftarrow{d}$	
$r_2 \leftarrow ds + x_2 \bmod q$	$\xrightarrow{(r_1,r_2)}$	überprüfe $sign(A, B)$
		$g_1^{r_1} g_2^{r_2} \overset{?}{=} A^d B$

Die Durchführbarkeit lässt sich leicht zeigen:

$$g_1^{r_1} g_2^{r_2} = g_1^{du_1 s+x_1} g_2^{ds+x_2} \overset{?}{=} A^d B = \left(m^s\right)^d g_1^{x_1} g_2^{x_2} = \left(Ig_2\right)^{sd} g_1^{x_1} g_2^{x_2}$$
$$= I^{sd} g_2^{sd} g_1^{x_1} g_2^{x_2} = g_1^{u_1 sd} g_2^{sd+x_2} g_1^{x_1} = g_1^{du_1 s+x_1} g_2^{ds+x_2}$$

Wie sieht es aber mit der Korrektheit aus? Angenommen, der User kann auf zwei Challenges *d* und *d*' antworten. Dann gibt es die Antworten (r_1, r_2) und (r_1', r_2') für die die Verifikationsgleichung gilt. Also:

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

$$\Rightarrow$$

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

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

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

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

Da aber d und d' unterschiedliche Challenges sind, sind die Exponenten ungleich Null und der User hätte somit eine nichttriviale Darstellung der Eins berechnet, was aber wie im Kapitel über das Darstellungsproblem festgestellt wurde nicht sein kann. Somit ist das Protokoll korrekt.

Geld einzahlen

Der Shopbesitzer möchte natürlich seine Einnahmen bei der Bank auf sein einzahlen. Da es sich bei dem hier betrachteten Geldsystem um ein Offline-System handelt geschieht diese Einzahlung mit einer Zeitverzögerung nach der Bezahlung im Shop.
Bei der Einzahlung sendet der Shop die Daten des Bezahlprotokolls, also (*A*, *B*, sign(*A*,*B*)), (r_1, r_2) und den Zeitstempel der Bezahlung an die Bank.
$\mathcal{B}$ erzeugt sich selbst erneut die Challenge *d* mit der Hashfunktion H_0 und dem übermittelten Zeitstempel.
Mit Hilfe von *d* überprüft die Bank ob sign(*A*,*B*) eine gültige Signatur von (*A*, *B*) ist und ob $g_1^{r_1} g_2^{r_2} = A^d B$ gilt.
Wenn eine Verifikation fehlschlägt akzeptiert die Bank die Einzahlung nicht. Gelingt die Verifikation, überprüft die Bank, ob A bereits in der geführten Datenbank vorhanden ist, also ob schon einmal mit der Münze bezahlt wurde. Wenn nicht, wird die Einzahlung akzeptiert, dem Shop das Geld gutgeschrieben und (A, Zeitstempel, r_1,r_2) in die Datenbank eingetragen.
Sollte A aber schon in der Datenbank sein werden folgende Fälle unterschieden:

1. Wenn $I_{\mathcal{S}}$ und Zeitstempel identisch sind, versucht der Shop eine Münze ein zweites Mal einzulösen.
2. Wenn die Challenges aber unterschiedlich sind, handelt es sich um Double-Spending eines Users.
   Die Bank besitzt nun (*d*, r_1, r_2) der zweiten Zahlung und (*d*', r_1', r_2') der ersten Zahlung und kann damit die Identität des Double-Spenders aufdecken:

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

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

Die Bank hat also die Möglichkeit Double-Spender zu entlarven, da der User beim Bezahlen mit einer Münze immer einen Punkt einer Gerade preisgibt. Sobald er das zweite mal damit bezahlt und (r_1, r_2) preisgibt, kann die Bank die Gerade bestimmen, die zu seiner Identität führt.
Verhält sich der User aber korrekt, hat die Bank keine Möglichkeit die Münzen zurückzuverfolgen.

Observer

Man kann wie schon erwähnt Double-Spending auch im Voraus verhindern, und nicht nur, wie oben gezeigt erst dann aufdecken, wenn es schon zu spät ist. Hierzu wurden die „Wallets with Observers“ entwickelt. Die Bank verteilt an ihre Kunden elektronische Geldbörsen (zum Beispiel Smartcards), die manipulationssicher sind.
Sollte es dennoch einem User gelingen, den Observer so zu manipulieren, dass er Münzen mehrfach ausgeben kann, greift immer noch die zweite Sicherung bei der Einzahlung der Münzen in der Bank, und die Identität des Double-Spenders wird im Nachhinein aufgedeckt.

Der Aufbau des Bezahlsystems ist im Grunde wie das oben erklärte. Es gibt nur ein paar Erweiterungen:

Kontoeröffnung:

Die Bank übergibt nach der normalen Registrierung des Kunden mit Speicherung von $g_1^{u_1}$ dem Kunden den Observer $\mathcal{O}$ in dem eine Zufallszahl $o_1 \in \mathbb{Z}_q^*$ gespeichert ist, die dem User nicht bekannt ist.
Im Folgenden ist $A_{\mathcal{O}} = g_1^{o_1}$.
Die Bank erzeugt $I = A_{\mathcal{O}}\left(g_1^{u_1}\right)$ und $z = \left(Ig_2\right)^x$ und schickt dann $A_{\mathcal{O}}$ und z an $\mathcal{U}$, welcher u_1, z und $A_{\mathcal{O}}$ speichert.
I ist die Kontonummer und hat die gleiche Aufgabe wie in dem oben beschriebenen Basissystem. Allerdings kennt der User diesmal nicht $\log_{g_1} I$.

Abhebe-Protokoll

Nachdem der User der Bank bewiesen hat, dass er der Besitzer des Kontos ist, wird folgendes Protokoll in Gang gesetzt:

Schritt 1: $\mathcal{O}$ erzeugt eine Zufallszahl $o_2 \in_R \mathbb{Z}_q$ und berechnet $B_{\mathcal{O}} = g_1^{o_2}$. Nach der Berechnung wird $B_{\mathcal{O}}$ an $\mathcal{U}$ geschickt.

Schritt 2: $\mathcal{B}$ wählt $w \in_R \mathbb{Z}_q$ und sendet $a=g^w$ und $b=m^w=(Ig_2)^w$ and $\mathcal{U}$.

Schritt 3: $\mathcal{U}$ wählt sich Zufallszahlen: $s \in_R \mathbb{Z}_q^*$,

$$x_1, x_2, e, u, v \in_R \mathbb{Z}_q$$

Anschließend blendet $\mathcal{U}$ $m=Ig_2$ zu A mit $(Ig_2)^s=A$ und z zu $z'=z^s$

$\mathcal{U}$ erzeugt $B = g_1^{x_1} g_2^{x_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}}$

Des weiteren werden a zu $a'=a^u g^v$ und b zu $b'=b^{su}A^v$ geblendet

Mit diesen Werten kann $\mathcal{U}$ die Challenge $c'=H(A,B,z',a',b')$

Zum Schluss wird c' geblendet zu $c = \frac{c'}{u} \bmod q$ und c wird an $\mathcal{B}$ geschickt

Schritt 4: Die Bank berechnet die Response $r=cx+w \bmod q$ und schickt diese an $\mathcal{U}$.

Schritt 5: $\mathcal{U}$ akzeptiert die Antwort nur, wenn $g^r=h^c a$ und $m^r=(Ig_2)^r=z^c b$ gilt.

Das Protokoll im Überblick:

$\mathcal{O}$	$\mathcal{U}$		$\mathcal{B}$	
			$w \in_R \mathbb{Z}_q$	
$o_2 \in_R \mathbb{Z}_q$			$a \leftarrow g^w$	
$B_{\mathcal{O}} \leftarrow g_1^{o_2}$	$\xrightarrow{B_{\mathcal{O}}}$	$s \in_R \mathbb{Z}_q^*$	$\xleftarrow{a,b}$	$b \leftarrow (Ig_2)^w$
	$A \leftarrow (Ig_2)^s$			
	$z' \leftarrow z^s$			
	$x_1, x_2, u, v \in_R \mathbb{Z}_q$			
	$B \leftarrow g_1^{x_1} g_2^{x_2}$			
	$a' \leftarrow a^u g^v$			
	$b' \leftarrow b^{su} A^v$			
	$c' \leftarrow \mathcal{H}(A, B, z', a', b')$			
	$c \leftarrow \frac{c'}{u} \bmod q$	$\xrightarrow{c}$		
			$r \leftarrow cx + w \bmod q$	
	$g^r \stackrel{?}{=} h^c a$	$\xleftarrow{r}$		
	$(Ig_2)^r \stackrel{?}{=} z^c b$			
	$r' \leftarrow ru + v \bmod q$			

Bezahl-Protokoll

Der Ablauf des Protokolls:

Schritt 1: Der User schickt seine Münze *A*, *B* und die Signatur sign(*A*,*B*) an den Shop $\mathcal{S}$.

Schritt 2: Der Shop erzeugt die Challenge $d=H_0(A, B, I_S, \text{Date/Time})$ und sendet *d* an $\mathcal{U}$.

Schritt 3: Wenn o_2 noch im Speicher vorhanden ist, berechnet $\mathcal{O}$ die Antwort $r_1' = d'o_1 + o_2 \bmod q$ und schickt diese an $\mathcal{U}$. Wenn o_2 allerdings schon gelöscht worden ist, sperrt der Observer die Zahlung. Anschließend löscht der Observer o_2 aus dem Speicher.

Schritt 4: $\mathcal{U}$ überprüft, ob $g_1^{r_1'} = A_{\mathcal{O}}^{d'} B_{\mathcal{O}}$. Wenn die Gleichung gilt, erzeugt $\mathcal{U}$ die Antworten r_1 und r_2 und schickt diese an $\mathcal{S}$ zurück.

$r_1=r_1'+d(u_1s)+x_1 \bmod q$
$r_2=ds+x_2 \bmod q$

Der Shop akzeptiert die Bezahlung, wenn sign(*A*,*B*) wirklich eine Signatur ist und $g_1^{r_1} g_2^{r_2} = A^d B$ gilt.

Das Protokoll im Überblick:

$\mathcal{O}$	$\mathcal{U}$	$\mathcal{S}$
	$\xrightarrow{A,B,sign(A,B)}$	$A \overset{?}{\neq} 1$
		$d \leftarrow \mathcal{H}_0(A, B, I_{\mathcal{S}}, date/time)$
	$\xleftarrow{d}$	
	$d' \leftarrow s(d+e) \bmod q$	
$\xleftarrow{d'}$		
o_2 noch im Speicher?		
$r_1' \leftarrow d'o_1 + o_2 \bmod q$		
$\xrightarrow{r_1'}$		
	$g_1^{r_1'} \overset{?}{=} A_{\mathcal{O}}^{d'} B_{\mathcal{O}}$	
	$r_1 \leftarrow r_1' + d(u_1s) + x_1 \bmod q$	
	$r_2 \leftarrow ds + x_2 \bmod q$	
	$\xrightarrow{(r_1,r_2)}$	überprüfe $sign(A,B)$
		$g_1^{r_1} g_2^{r_2} \overset{?}{=} A^d B$

Der Observer prüft also beim Bezahlen, ob die Münze bereits einmal ausgegeben wurde. Wenn ja verweigert er die Bezahlung. Wenn nein, gibt er die Bezahlung frei.

Da $\mathcal{U}$ keine Darstellung von I kennt, kann der User auch keine Darstellung einer Münze kennen, wenn es sich beim Abhebeprotokoll um ein restriktives blindes Signaturverfahren handelt.
Der User kann keine Münze ausgeben, wenn der Observer nicht kooperiert. Das Einzahlprotokoll ist das gleiche wie im Basissystem. Die Bank hat hier auch die Möglichkeit Double-Spender zu entdecken, wenn Sie es geschafft haben sollten, den Observer zu manipulieren.

Quellen

[1] Stefan Brands, Untracable Off-line Cash in Wallets with Observers, Crypto 1993

[2] Stefan Brands, An Efficient Off-line Electronic Cash System Based On The Representation Problem, 1993

[3] Thomas Schwarzpaul, Eine faire elektronische Geldbörse basierend auf de Problem des diskreten Logarithmus, März 2003

[4] Claus-Peter Schnorr, Efficient Signature Generation by Smart Cards, 1991