Divisible e-cash systems can be truly anonymous

Benito van der Zander

5. Juni 2011

Zusammenfassung

Diese yz Arbeit beschreibt das digitale, Offline-Bezahlsystem von Canard und Gouget, welches als erstes System die Verwendung von vollständig anonymen und teilbaren Münzen ermöglicht. Mit einem solches System kann ein Kunde eine als digitale Münze bezeichnete Signatur mit einem festen Betrag von einer Bank abzuheben, und mit Teilen dieser Münze mehrmals bei unterschiedlichen Händlern bezahlen, wobei garantiert ist, dass der Gesamtbetrag maximal einmal vollständig ausgegeben wird.

Inhaltsverzeichnis

1 Einführung

Digitales Geld überträgt das Konzept des Bargelds in die digitale Welt.

Eine ideale digitale Münze verhält sich wie eine materielle Münze, das heißt, sie ist fälschungssicher, man erhält sie von einer Bank und kann dann anonym mit ihr bei einem Händler bezahlen.

Ein übliches digitales Geldsystem verwendet eine Zufallszahl als Münze, die von der Bank blind, also ohne, dass sie die Zahl kennt, signiert wurde[7]. Durch die Fälschungssicherheit der Signatur kann nur die Bank neue Münzen erstellen und die Blindheit der Signatur ermöglicht das anonyme Ausgeben der Münze.

Allerdings kann natürlich jeder eine solche Zufallszahl beliebig kopieren, weshalb entweder jede Transaktion von der Bank überwacht (Online-System) oder die Identität des Münzenbesitzers in der Münze kodiert werden muss (Offline-System). Bei einem Offline-System ist dementsprechend das größte Problem, die Anonymität jeder ehrlichen Partei zu garantieren und trotzdem die Identität aufzudecken, sobald eine kopierte Münze entdeckt wird.

Eine weitere gewünschte Eigenschaft eines Systems ist die Möglichkeit mit einer einzigen Münze verschiedene Beträge bezahlen zu können, also dass man nur einen Teil der Münze ausgibt und den Rest der Münze behält. Ohne eine teilbare Münze muss man beim Abheben der Münze von der Bank nämlich entweder bereits wissen, welchen Betrag man später benötigt, oder mehrere Münzen abheben, was sehr ineffizient sein kann.

Es gab bereits verschiedene teilbare Offline-Systeme[2][10], aber keines realisiert gleichzeitig absolute Anonymität und Teilbarkeit.

Das nun vorgestellte System von Canard und Gouget[6] ist das erste teilbare Offline-System, das vollständige Anonymität garantiert.

2 Protokollphasen

2.1 Prinzip des Schlüsselbaums

Die meisten teilbaren Münzen basieren auf einem Schlüsselbaum[6][10]. Dabei wird eine Münze mit Wert n durch einen Baum mit n-Blättern dargestellt, so dass das Abspalten einer Münze mit Wert $m \leqslant n$ dem Abtrennen eines Teilbaumes mit m-Blättern entspricht.

Da das Speichern eines kompletten Baumes mit n-Blättern aufwendiger als das Speichern von n Einzelmünzen ist, wird der Baum so konstruiert, dass sich aus einem Knoten des Baumes alle Kindknoten berechnen lassen. Im Falle eines Binärbaumes reicht es dann anstatt einen vollständigen Baum mit Wert $n = 2^L$ und Tiefe L einen einzigen Wurzelknoten zu speichern, und das Abspalten eines Teilbaumes im Wert von $m = 2^{L-l}$ entspricht dem Berechnen und Entfernen eines Knoten auf Ebene l.

Die Berechnung eines untergeordneten Knotens aus dem Elternknoten sollte effizient sein, darf aber nicht umkehrbar sein[1], so dass sich für diese Berechnung modulare Exponentation eignet. Identifiziert man den jeweils i-ten Knoten auf Ebene l mit einem Knotenschlüssel $K_{l,i}$, so kann man mittels zwei Generatoren $g_{l,0}$ und $g_{l,1}$ für jeden Kindknoten markieren, ob er ein linker Kindknoten mit $b = 0$ oder ein rechter Kindknoten mit $b = 1$ ist, indem man als Schlüssel $K_{l,2i+b} = g_{l,b}^{K_{l-1,i}}$ wählt[2]. Dadurch wird der Pfad zum Knoten kodiert und die Position eines Knotens im Baum durch seinen Schlüssel eindeutig festgelegt (siehe Abbildung 1).

Mit dieser Darstellung einer Münze als Schlüsselbaum lassen sich alle weiteren Protokolle auf die (anonymen) Weitergabe von Teilbäumen, den Beweis, dass ein Baum tatsächlich ein Teilbaum ist, und die Entdeckung von Überlappungen zwischen mehreren Teilbäumen reduzieren.

2.2 Initialisierung

In der Initialisierungsphase, Protokoll 1, werden die Parameter des Systems zufällig gewählt. Zu den Parameter gehören die Schlüssel der beteiligten Parteien sowie die verschiedenen Gruppen und Generatoren, welche für die modulare Exponentation bei der Berechnung des Schlüsselbaums benötigt

[1] Sonst wäre jeder Knoten äquivalent zum gesamten Baum.

[2] Man benötigt auf jeder Ebene unterschiedliche Generatoren, da man durch die Exponentiation auf jeder Ebene unterschiedliche Gruppen erhält.

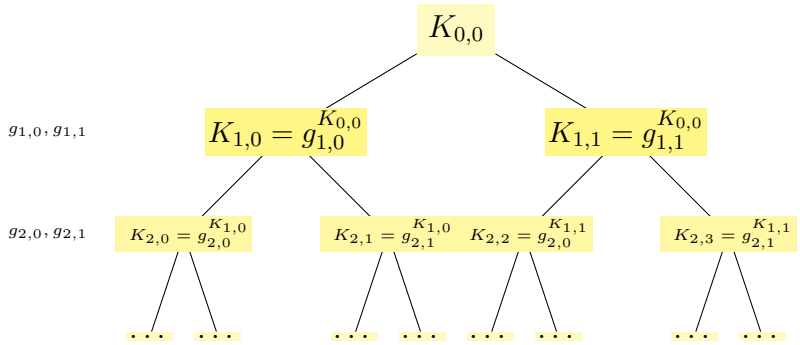


Abbildung 1: Schlüsselbaum

werden. Die Gruppen und Generatoren werden von der Bank gewählt, die persönlichen Schlüssel wählt jede teilnehmende Partei selbst.

Bei der Wahl der Gruppen ist es wichtig, dass sich auf den Gruppen Wurzeln berechnen lassen und jedes Element einer Gruppe $\mathcal{G}_i$ von einer Ebene des Baumes als Exponent für ein Element der Gruppe $\mathcal{G}_{i+1}$ der nächsten Ebene verwendet werden kann, um den Knotenschlüssel im Baum zu berechnen. Ersteres erfordert, dass die Ordnung $\phi(|\mathcal{G}_{i+1}|)$ der jeweils einer Gruppe zugewiesenen Einheitengruppe effizient berechenbar ist, und letzteres dass $\phi(|\mathcal{G}_{i+1}|)$ auch ein Vielfaches der Ordnung der vorherigen Gruppe $\mathcal{G}_i$ ist. Die kleinsten und am einfachsten zu berechnenden Gruppen erhält man daher, wenn man als Ordnungen Primzahlen der Form $|G_{i+1}| = 2|G_i| + 1$ wählt.

Zudem wählt die Bank ein Paar asymmetrischer Schlüssel (sk_B, pk_B) für das dem E-Cash zugrunde liegende Signaturverfahren. Canard und Gouget verwenden hierbei eine CL-Signatur[3], prinzipiell ist aber jedes Signaturverfahren geeignet, dass blinde Signaturen und einen ZKP (siehe Abschnitt 3) der Kenntnis der Signatur unterstützt.

Die Bank veröffentlicht alle berechneten Parameter außer ihrem privaten Signaturschlüssel.

Der private Schlüssel sk_K des Kunden bzw. sk_H des Händlers ist jeweils ein zufälliges Element in $\mathcal{G}$, aus dem der öffentliche Schlüssel $pk_K = g^{sk_K}$ und $pk_H = g^{sk_H}$ durch einfache modulare Potenzierung gewonnen wird.

2.3 Abheben

Beim Abheben, Protokoll 2, interagiert der Kunde mit der Bank, um eine Münze zu erhalten, also einen Schlüsselbaum und eine blinde Signatur σ der

Bank:	*Kunde*:	*Händler*:
Bestimmt die Tiefe L des Schlüsselbaums		
Wählt Gruppe $\mathcal{G}$ mit Ordnung $o_{\mathcal{G}}$		
Wählt Gruppe $\mathcal{G}_1 = \langle g_1 \rangle$ mit $\lvert\mathcal{G}_1\rvert = o_{\mathcal{G}_1}$ und $\mathcal{G} \subset \mathbb{Z}^*_{o_{G_1}}$		
Wählt Gruppe $\mathcal{G}_i = \langle g_i \rangle \subset \mathbb{Z}^*_{o_{\mathcal{G}_{i+1}}}$ mit $\lvert G_i\rvert = o_{\mathcal{G}_i}$		
so dass $o_{G_{i+1}} = 2o_{G_i} + 1 \in \mathbb{P}$rimes ist.		
Wählt Generatoren $g, h_0, h_1, h_2 \in \mathcal{G}$		
Wählt Generatoren $g_{i,0}, g_{i,1}, g_{i,2} \in \mathcal{G}_i$		
Wählt Schlüssel (sk_B, pk_B) für CL-Signatur	Wähle $sk_K \in \mathcal{G}$	
	$pk_K = g^{sk_K}$	Wähle $sk_H \in \mathcal{G}$
		$pk_H = g^{sk_H}$

Protokoll 1: Initialisierung

Bank, welche beweist, dass der Schlüsselbaum von der Bank für genau diesen Kunden ausgestellt wurde.

Der Schlüsselbaum wird dabei eindeutig durch einen Wurzelknotenschlüssel g^s bestimmt[3], der durch die gemeinsame, aber für die Bank geheime Zufallszahl $s = s' + r'$ vom Kunden und der Bank bestimmt wird.

Die Bank signiert den Baum und die Identität, also das Tupel (s, sk_K, r), als von Generatoren verschleiertes Produkt $h_0^{s'} h_1^{sk_K} h_2^r$. Dabei ist r eine weitere Zufallszahl, durch die es für die Bank statistisch unmöglich wird s und sk_K aus $h_0^{s'} h_1^{sk_K} h_2^r$ abzuleiten.

Schließlich beweist der Kunde mit einem ZKP (siehe ZKP 1), dass er tatsächlich die vom Protokoll geforderten Werte gesendet hat.

Bank: pk_B, sk_B, pk_K		*Kunde*: pk_B, pk_K, sk_K
		$r, s' \in \mathbb{Z}^*_G$
		$C' \leftarrow h_0^{s'} h_1^{sk_K} h_2^r$
Überprüfe U	$\xleftarrow{C',\ U,\ pk_K}$	$U \leftarrow PK(\alpha, \beta, \gamma / pk_K = g^\alpha \wedge C' = h_0^\beta h_1^\alpha h_2^\gamma)$
$r' \in \mathbb{Z}^*_G$		
$\sigma \leftarrow \text{CLSign}(C' h_0^{r'})$	$\xrightarrow{r',\ \sigma}$	$s \leftarrow s' + r' \pmod p$
		$\text{CLVerify}(\sigma, (s, sk_K, r))$
		Münze: $C = (s, sk_K, r, \sigma)$

Protokoll 2: Abhebenprotokoll

[3]da sich nach der Konstruktion des Baumes alle weiteren Knoten aus diesem ableiten lassen.

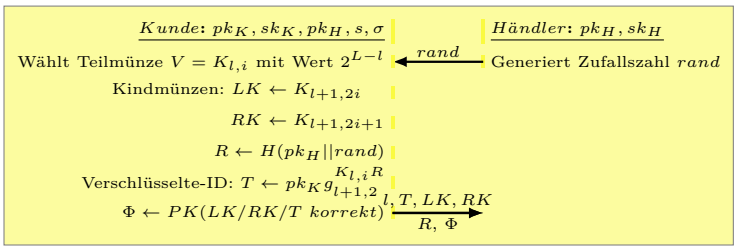
2.4 Bezahlen

Beim Bezahlen eines Betrages 2^{L-l} überträgt der Kunde einen Teilbaum seiner Münze an den Händler, und muss beweisen, dass dieser Teilbaum wirklich von der Bank stammt. Außerdem muss dieser Teilbaum mit den Identitäten des Kunden und Händlers verbunden werden, um zu verhindern, dass einer von ihnen die Münze kopiert.

Da der Händler nicht anonym zu bleiben hat, kann seine Identität pk_H offen im Protokoll verwendet werden, ergänzt um eine vom Händler bestimmte Zufallszahl $rand$, die verhindert, dass ein Kunde ein Transkript ohne direkte Beteiligung des jeweiligen Händler erstellt und den Händler damit beschuldigt, eine Münze kopiert zu haben.

Die Identität pk_K des Kunden muss so verschlüsselt werden, dass sie aus einem Transkript alleine nicht ermittelt werden kann, aus zwei Transkripten mit überlappenden Teilbäumen dagegen schon. Dazu eignet sich der Schlüssel $K_{l,i}$ des Wurzelknotens des Teilbaums, welcher bei zwei überlappenden (vollständigen, nicht identischen) Teilbäumen von zumindest einem Baum bekannt ist.

Der Kunde entfernt daher den Wurzelschlüssel $K_{l,i}$ von dem an den Händler übertragenen Baum, indem er statt der Wurzel K_l, i die ersten beiden Kinder $LK = K_{l+1,2i}$ und $RK = K_{l+1,2i+1}$ sendet.



Protokoll 3: Bezahlenprotokoll

Um zu beweisen, dass ein Kunde dem Protokoll ehrlich gefolgt ist, muss er zeigen, dass er alle Parameter korrekt berechnet hat und der Teilbaum wirklich von der Bank stammt. Letzteres ist genau dann der Fall, wenn es einen von der Bank signierten Wurzelknoten gibt, von dem ein Pfad $K_{0,0} \ldots K_{l,i}$ zu dem Wurzelknoten $K_{l,i}$ des Teilbaums führt (siehe Abbildung 2). Versteckt man diesen Pfad mittels zufälligen Generatoren $\tilde{g}_j$ als öffentliche

Parameter $V_j = \tilde{g}_j^{K_{j-1,\cdot}}$ und den Wurzelknoten des Pfads als $V_0 = \tilde{g}^s \tilde{h}^{\tilde{r}}$ mit zufälligen $\tilde{g}, \tilde{h}, \tilde{r}$ so, ergeben sich daraus die folgenden drei Typen von ZKPs, die gemeinsam im Protokoll 4 übertragen werden[4]:

1. $PK(\sigma, s, sk_K, r, \tilde{r} / \sigma = \text{Sign}(s, sk_K, r) \wedge V_0 = \tilde{g}^s \tilde{h}^{\tilde{r}} \wedge V_1 = \tilde{g}^{g^s})$

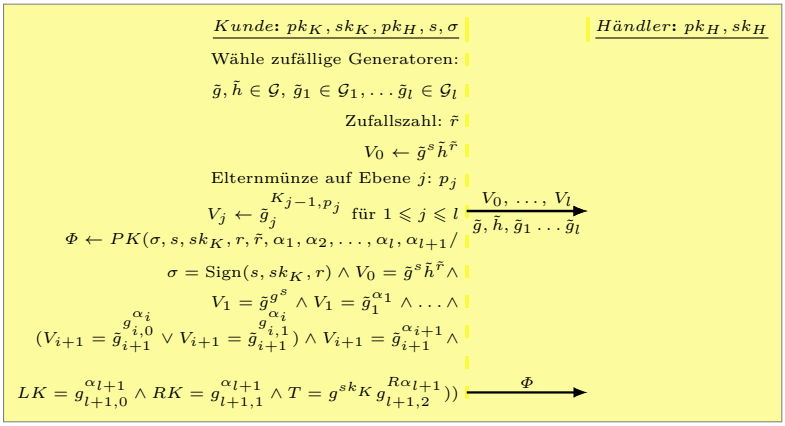
   Es existiert eine Signatur der Bank auf den Wurzelknoten s für den Kunden sk_K und die ersten beiden versteckten Pfadknoten V_0 und V_1 sind korrekt. (siehe ZKP 4)

2. $PK(\alpha_i / V_i = \tilde{g}_i^{\alpha_i} \wedge (V_{i+1} = \tilde{g}_{i+1}^{g_{i,0}^{\alpha_i}} \vee V_{i+1} = \tilde{g}_{i+1}^{g_{i,1}^{\alpha_i}}))$

   Jeder Schritt im Pfad von K_{j-1} nach K_j ist wirklich ein Schritt im Baum, also $K_{j,\cdot} \in \{g_{j,0}^{K_{j-1,\cdot}}, g_{j,1}^{K_{j-1,\cdot}}\}$. (siehe ZKP 3)

3. $PK(sk_K, \alpha_{l+1} / V_{l+1} = \tilde{g}_{l+1}^{\alpha_{l+1}} \wedge LK = g_{l+1,0}^{\alpha_{l+1}} \wedge RK = g_{l+1,1}^{\alpha_{l+1}} \wedge T = g^{sk_K} g_{l+1,2}^{R\alpha_{l+1}})$

   Die finalen Werte LK, RK und T sind gemäß dem Protokoll geformt. (siehe ZKP 1)



Protokoll 4: Übertragung des Bezahlen-ZKP

[4] Bei der Notation $PK(\ldots / \ldots)$ stehen auf der linken Seite des Schrägstrichs immer alle geheimen Variablen, deren Kenntnis bewiesen wird, und auf der rechten die Bedingung, die von den geheimen Variablen erfüllt werden muss.

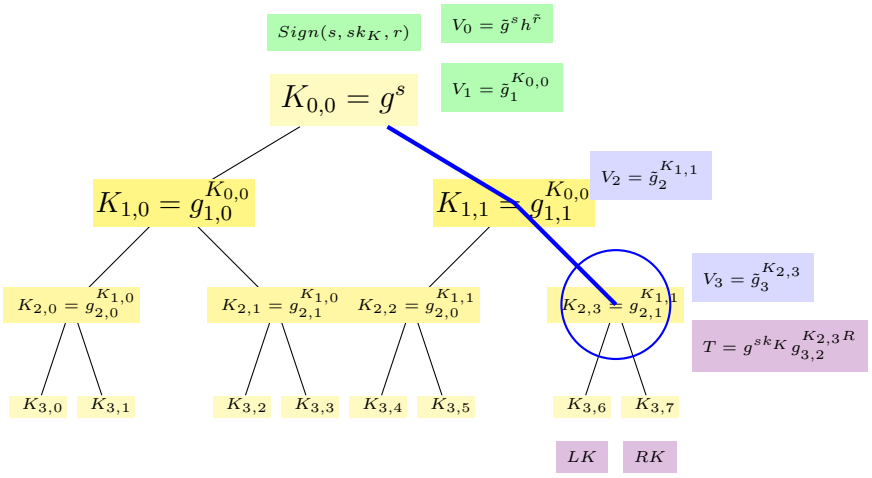


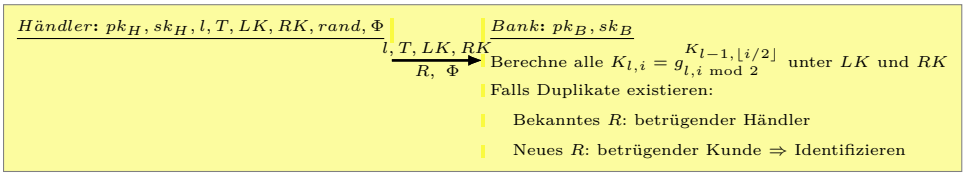
Abbildung 2: Pfad im Schlüsselbaum

2.5 Einzahlen

Beim Einzahlen, Protokoll 5, übergibt der Händler, die vom Kunden erhaltene Münze wieder an die Bank.

Da der Händler außer seiner Zufallszahl und dem Transkript des Bezahlen-Protokolls keine weitere Informationen über die Münze besitzt, sendet er bloß das entsprechende Transkript an die Bank.

Die Bank muss nun überprüfen, ob die Münze gültig und nur einmal ausgegeben wurde. Ersteres wird durch den enthaltenen ZKP garantiert, für letzteres muss sie den eingezahlten Teilbaum auf Überlappungen mit allen jemals eingezahlten Teilbäumen überprüfen. Existiert eine solche Überlappungen, kann sie erkennen, ob die Münze vom Kunden oder vom Händler kopiert wurde, denn jede vom Händler kopierte Münze trägt dieselbe Zufallszahl *rand*. Im anderen Fall kann sie aus einer vom Kunden kopierten Münze die Identität des Kunden ermitteln.



Protokoll 5: Einzahlungsprotokoll

2.5.1 Identifizierung

Falls die Bank eine vom Kunden kopierte Münze entdeckt hat, muss sie die Identität des Kunden ermitteln. Dabei lassen sich zwei Fälle unterscheiden:

Im ersten Fall hat der Kunde zweimal denselben Teilbaum ausgegeben und die Bank besitzt zwei Verschlüsselungen der Identität mit demselben Knotenschlüssel aber unterschiedlichen Zufallszahlen. Dann kann die Bank die Identität pk_K folgendermaßen berechnen:

$$(T^{R'}/T'^{R})^{1/(R'-R)} = (pk_K^{R'} g_{l+1,2}^{K_{l,i}RR'}/pk_K^{R} g_{l+1,2}^{K_{l,i}RR'})^{1/(R'-R)} = (pk_K^{R'-R})^{1/(R'-R)}$$

Im anderen Fall sind die Teilbäume nicht identisch, aber überlappen sich. Da beide Teilbäume vollständige Binärbäume sind, ist der eine Baum im anderen enthalten und die Bank kennt einen Teilbaumwurzelschlüssel $K_{l,i}$. Somit kann sie die Identität direkt als $Tg_{l+1,2}^{-K_{l,i}R} = pk_K g_{l+1,2}^{K_{l,i}R} g_{l+1,2}^{-K_{l,i}R} = pk_K$ berechnen.

3 Zero-Knowledge-Proofs

Die Sicherheit der Protokolle entsteht dadurch, dass in jedem Schritt mittels Zero-Knowledge-Proofs (ZKPs) bewiesen wird, dass die Parteien dem Protokoll ehrlich gefolgt sind.

Dazu benötigt man ZKPs, welche die Kenntnis einer CL-Signatur und verschiedener Kombination von Logarithmen und Doppellogarithmen beweisen:

ZKP 1 beweist allgemein die Kenntnis von u geheimen Logarithmen $\alpha_1, \ldots, \alpha_u$ von w öffentlichen Variablen [4]:

$$PK((\alpha_1, \ldots, \alpha_u)/(y_1 = \prod_{j=1}^{l_1} g_{b_{1j}}^{\alpha_{e_{1j}}}) \wedge \ldots \wedge (y_w = \prod_{j=1}^{l_w} g_{b_{wj}}^{\alpha_{e_{wj}}}))$$

Dabei ersetzt man die α_k durch zufällige r_k, sendet $y_i' = \prod_{j=1}^{l_i} g_{b_{ij}}^{r_{e_{ij}}}$ und zeigt entweder, dass man r_k kennt oder dass man $r_k - \alpha_k$ kennt. [5]

ZKP 2 beweist die Kenntnis eines Logarithmus und eines Doppellogarithmus:

$$PK(\alpha/a = g_1^{\alpha} \wedge b = g_2^{h^{\alpha}})$$

Dieser ZKP entspricht im wesentlichen ZKP 1, wobei man bei der Verifizierung ausnützt, dass $(g^{h^\alpha})^{h^{r-\alpha}} = g^{h^r}$ ist.[11]

Der beim Beweisen der Pfadkorrektheit verwendete ZKP 3 beweist die Kenntnis eines Logarithmus und einem von zwei Doppellogarithmen:

$$PK(\alpha/a = g_1^\alpha \wedge (b = g_2^{h_1^\alpha} \vee b = g_2^{h_2^\alpha}))$$
$$\equiv PK(\alpha/(a = g_1^\alpha \wedge b = g_2^{h_1^\alpha}) \vee (a = g_1^\alpha \wedge b = g_2^{h_2^\alpha}))$$

Dazu beweist man eine Seite der Disjunktion, indem man den ZKP 2 führt, und verwendet auf der anderen Seite den Simulator von ZKP 2.[8]

Der letzte ZKP 4 beweist die Kenntnis einer CL-Signatur auf mehreren geheim gehaltenen Werten:

$$PK(\sigma, s, sk_K, r/\sigma = \text{Sign}(s, sk_K, r))$$

Dazu versteckt man die Signatur mit zufälligen Exponenten und beweist die Kenntnis der entsprechenden Logarithmen.[3]

Jeder dieser ZKPs lässt sich mittels der Fiat-Shamir-Heuristik nicht interaktiv führen, indem man die vom Verifizierer gesendeten Werte durch eine Hashfunktion ersetzt, ohne dass sich dadurch die Sicherheit (im Random-Orakel-Modell) verringert.[9]

4 Sicherheitsbeweise

4.1 Fälschungs-/Kopiersicher

Für diese Sicherheitseigenschaft ist zu zeigen, dass ein Kunde keine Münzen ausgeben kann, die nicht von der Bank stammen; also, dass, wenn er nach m Interaktionen mit der Bank einen Betrag von mehr als $2^L m$ ausgeben kann, eine Sicherheitsannahme widerlegt ist.

Wenn der Kunde unentdeckt mehr als $2^L m$ ausgibt, hat die Bank durch den Kunden mindestens $2^L m + 1$ Blattknoten erhalten.

Für jeden dieser Blattknoten K gibt es einen ZKP, der beweist, dass der Kunde eine Signatur $\sigma = (s, sk_K, r)$ auf einen Wurzelknoten s kennt und dass es einen Pfad von s zu K gibt.

In jedem Schlüsselbaum gibt es aber nur 2^L unterschiedlich Pfade zu Blattknoten, und somit kennt der Kunde durch seine $2^L m + 1$ Pfade mehr als m Bäume und somit mehr als m Signaturen.

Da die Bank in m Interaktionen maximal m Signaturen ausgibt, hat der Kunde entweder eine CL-Signatur oder einen ZKP gefälscht.

Daher folgt diese Sicherheitseigenschaft aus der Sicherheit der CL-Signatur [3] und der Sicherheit der ZKPs.

4.2 Identifizierbar

Falls die CL-Signatur und der ZKP nicht fälschbar sind, wird das Einzahlen einer vom Kunden kopierte Münze, wie im vorherigen Abschnitt bewiesen, erkannt. Es ist aber noch nötig zu zeigen, dass in diesem Fall die Identität des kopierenden Kunden aufgedeckt werden kann.

Falls eine Doppelzahlung eines Kunden erkannt wurde, hat die Bank zwei identische Blattknoten mit verschlüsselten Identitäten T und T' erhalten.

Da der ZKP beweist, dass alle vom Kunden übermittelten Werte korrekt geformt sind, kann die Bank daraus, wie im Abschnitt 2.5.1 beschrieben, die Identität des Kunden berechnen.

4.3 Unlinkbar

Als nächstes ist zu zeigen, dass zwar kopierende Kunden identifizierbar sind, ehrliche Kunden aber vollständig anonym bleiben, also insbesondere, dass sich aus zwei eingezahlte Münzen nicht ermitteln lässt, ob sie vom selben Kunden stammen.

Beim Bezahlen mit einer Teilmünze auf Tiefe l sendet der Kunde das Tupel $(LK, RK, T, V_0, V_1, \ldots, V_l)$ von potentiell kompromittierenden Werten mit:

$$LK = g_{l+1,0}^{V}, RK = g_{l+1,1}^{V}, T = pk_K g_{l+1,2}^{VR}$$

$$V_0 = \tilde{g}^s \tilde{h}^{\tilde{r}}, V_1 = \tilde{g}_1^{K_{0,p_1}}, \ldots, V_l = \tilde{g}_l^{K_{l-1,p_l}}$$

V_0 ist (genauso wie das beim Abheben gesendete $h_0^{s'} h_1^{sk_K} h_2^r$) durch das angehängte $\tilde{h}^{\tilde{r}}$ eine gleichverteilte Zufallszahl und liefert keine Informationen über den Kunden.

Zu zeigen ist nun also, dass sich mittels den beiden der Bank bekannten Tupeln $(LK_0, RK_0, T_0, V_{1,0}, \ldots, V_{l,0}, R_0, \tilde{g}_{1,0}, \ldots, \tilde{g}_{l,0})$ und

$(LK_1, RK_1, T_1, V_{1,1}, \ldots, V_{l,1}, R_1, \tilde{g}_{1,1}, \ldots, \tilde{g}_{l,1})$ nicht erkennen lässt, ob $s_0 = s_1$ gilt[5].

Sei O ein Orakel, dass bei gegebenen Parametern und Tupeln das Identifizierungsproblem, ob die Münzen vom selben Nutzer stammen, entscheidet.

Es muss nun gezeigt werden, dass mit diesem Orakel ein als schwer angenommenes Problem gelöst werden kann. Hierzu wird zunächst ein Diffie-Hellman ähnliches Problem eingeführt, das die Pfadstruktur der Münze abstrahiert:

Dieses Problem besteht aus einem Tupel D_G von Generatoren und einem Fragetupel D_F. Dabei sei $D_G = (h_1, \ldots, h_k, h_{0,0}, h_{0,1}, \ldots, h_{k,0}, h_{k,1})$ und $D_F = (h^x_{0,0}, h^y_{0,1}, h^{h^x_1}_{1,0}, h^{h^y_1}_{1,1}, h^{h^x_1}_{2,0}, h^{h^y_1}_{2,1}, \ldots, h^{h_k^{\cdot^{\cdot^{h^x_1}}}}_{k,0}, h^{h_k^{\cdot^{\cdot^{h^y_1}}}}_{k,1})$. Das heißt D_F enthält zwei Sequenzen $s_0 = x, s_k = h_k^{s_{k-1}}$ und $s'_0 = y, s'_k = h_k^{s'_{k-1}}$, die durch Generatoren $h_{i,0}, h_{i,1}$ versteckt sind.

Die Frage, ob $x = y$ gilt für gegebene D_G, D_F, heiße im folgenden das Extended Multi Decisional Diffie-Hellman(XMDDH)-Problem. Intuitiv scheint dieses Problem schwer zu sein, und im Spezialfall $k = 0$ ist XMDDH äquivalent zum Derived Decisional Diffie-Hellman (DDDH), das $x = y$ für ein Tupel (g_1, g_2, g_1^x, g_2^y) fragt und in vielen Gruppen als schwer gilt[6]. Aber für $k > 0$ enthält es mehr Informationen über x und y, und könnte daher leichter sein.

Es wird nun gezeigt, dass das Identifizierungsproblem mindestens so schwer ist wie XMDDH, indem XMDDH auf das Identifizierungsproblem reduziert wird. Dazu wird ausgenutzt, dass die Struktur des XMDDH-Problems der Pfadstruktur des Baumes entspricht und die Parameter für den Schlüsselbaum und das Bezahlen-Protokoll folgendermaßen gewählt:

Sei $g_{i,0} = h_i$ für $1 \leqslant i \leqslant k$, $g_{k+1,0} = h_{k,0}$ und $g_{k+1,1} = h_{k,1}$ für die Generatoren des Baumes entlang dem Pfad und $\tilde{g}_{i,j} = h_{i,j}$ für $1 \leqslant i < k, 0 \leqslant j \leqslant 1$ für die den Pfad versteckenden Generatoren. Alle weiteren Generatoren, sowie R_0, R_1 und zwei öffentliche Kundenschlüssel $pk_{K,0}$, $pk_{K,1}$ seien zufällig gewählt.

Damit entsprechen die Einträge im XMDDH-Tupel genau dem verstecken Münzenpfad von der Wurzel zum Blatt am linken Rand des Baums und man

[5] O.B.d.A. ist das l in beiden Tupel gleich, da man aus einem Tupel alle längeren Tupel berechnen kann.

kann setzen $V_{i,j} = h_{i,j}^{h_i^{\cdot^{\cdot^{h_1^y}}}}$ für alle $1 \leqslant i < k, 0 \leqslant j \leqslant 1$.

Verwendet man die letzten Einträge im XMDDH-Tupel als $V_0 = h_{k,0}^{h_k^{\cdot^{\cdot^{h_1^x}}}}$ und $V_1 = h_{k,1}^{h_k^{\cdot^{\cdot^{h_1^x}}}}$, so kann man LK, RK, T als $LK_j = g_{k+2,0}^{V_j}$, $RK_j = g_{k+2,1}^{V_j}$ und $T_j = pk_{K,j} g_{k+2,2}^{V_j}$ berechnen. Hierbei entsteht ein kleines, technisches Problem, dass $pk_{K,0} = pk_{K,1}$ genau dann gelten muss, falls $x = y$ gilt. Wählt man aber zufällig einen Fall $pk_{K,0} = pk_{K,1}$ oder $pk_{K,0} \neq pk_{K,1}$, erreicht man immer noch eine Erfolgswahrscheinlichkeit von $\frac{1}{2} \cdot (1 - \varepsilon) + \frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot 0 > \frac{1}{2}$, wobei sich die $\frac{1}{2}$-Wahrscheinlichkeit falsch zu raten in zwei Fälle aufgeteilt hat, im ersten ist die Münze ungültig und das Orakel kann nur raten, im zweiten entstehen zwei unterschiedliche Münzen vom selben Kunden und das Orakel ist immer falsch[6]. Das heißt, man verwendet das Orakel nicht direkt, um eine Identifizierung durchzuführen, sondern rät, ob die Werte von derselben Münze stammen, und fragt das Orakel, ob man korrekt geraten hat.

Damit sind alle Parameter für das Orakel definiert und gemäß der Konstruktion gilt $g^{s_0} = g^{s_1}$ genau dann, wenn $x = y$ gilt. Also kann man mit dem Orakel durch wiederholte Aufrufe mit beliebiger Wahrscheinlichkeit das XMDDH-Problem lösen[7].

Da es im Spezialfall $k = 0$ identisch zum DDDH-Problem ist, folgt daraus, dass zumindest die Ausgabe zweier Teilmünzen im linken und rechten Teilbaum unter der DDH-Annahme unlinkbar ist.

Man kann das Informationsleck durch die $V_{i,j}$ auch beseitigen, indem man die $V_{i,j} = \tilde{g}_{i,j}^K$ durch $V_{i,j} = \tilde{g}_{i,j}^K \tilde{h}^{r_{i,j}}$ mit Zufallszahlen $r_{i,j}$ ersetzt und im ZKP zusätzlich die Kenntnis der $r_{i,j}$ beweist, so dass alle Werte statistisch verteilt sind, und sozusagen immer $k = 0$ gilt.

Allerdings wurde im Beweis davon ausgegangen, dass die Generatoren zufällig gewählt sind, weshalb die Bank die Anonymität trotzdem aufheben

[6] Das ε bezeichnet die vernachlässigbare Wahrscheinlichkeit, dass es zwei unterschiedliche Münzen geben könnte, die trotzdem denselben Pfad enthalten.

[7] Tatsächlich stellt man leicht fest, dass dies auch umgekehrt gilt, die beiden Probleme also äquivalent sind. Zwar muss man in der Gegenrichtung den Pfad von der Wurzel zur ausgegebenen Teilmünze kennen, es ist aber problemlos möglich alle Pfade im Baum auszuprobieren.

kann, indem sie in der Initialisierungsphase betrügt. Dazu wählt sie keine zufälligen Generatoren $g_{i,2}$, sondern Generatoren von denen sie $\log_{g_{i,0}} g_{i,2}$ kennt, daher $g_{i,2} = g_{i,0}^{r_i}$ mit zufälligen r_i.

Damit kann sie aus korrekt geformten Werten LK und T, die Identität als $T \cdot LK^{-r_l R} = pk_K g_{l,2}^{VR} g_{l,0}^{-VRr_l} = pk_K$ berechnen.

Deshalb muss sichergestellt werden, dass die Generatoren wirklich zufällig gewählt werden, zum Beispiel, indem die Bank mit einer Gruppe von Kunden gemeinsame Zufallszahlen wählt oder eine externe, unabhängige Quelle verwendet (z.B.: Sonnenflecken).

4.4 Entlastend

Schließlich muss gezeigt werden, dass das System entlastend ist, das heißt, dass jeder, der ein neues, gültiges Transkript T' erstellen kann, aus dem eine Doppelzahlung durch den Kunden mit öffentlichem Schlüssel $pk_K = g^{sk_K}$ folgt, den privaten Kundenschlüssel sk_K kennt.

Die Kenntnis *eines* Kundenschlüssels sk'_K wird von dem im Transkript kodierten ZKP bewiesen, daher ist nur zu zeigen, dass dies derselbe Schlüssel ist, der von der Bank bei der Identifizierung als $pk''_K = g^{sk''_K}$ ermittelt wird.

Sei $\Upsilon = (LK', RK', L', R', T, \Phi)$ ein gefälschtes Transkript, bei dem die Bank zusammen mit einem anderen Transkript $\Upsilon = (LK, RK, L, R, T, \Phi)$ eine Doppelzahlung erkennt. Nun sind drei Fälle zu unterscheiden:

Im ersten Fall gilt $L' = L$, die beiden Transkripte kodieren also eine Teilmünze desselben Wertes. Wenn die Bank eine Doppelzahlung erkennt, gilt $LKRK = LK'RK'$. Dann kann man davon ausgehen, dass auch die Elternknoten und sogar der gesamte Schlüsselbaum in beiden Transkripten gleich sind, da die ZKPs beweisen, dass es sich um gültige, von der Bank signierte Schlüsselbäume handelt[8]. Daher gilt auch $s = s'$, $V = V'$ und $sk_K = sk'_K$. Da R und R' durch die ZKPs fest sind, verwendet die Bank bei der Berechnung $pk''_K = (T^{R'}/T'^{R})^{1/(R'-R)}$ dieselben R, R' wie der Ersteller der Transkripte und erhält somit $g^{sk''_K} = pk''_K = g^{sk_K} = g^{sk'_K}$. Es gilt also $sk'_K = sk''_K$.

Im zweiten Fall gilt $L' > L$ und es gibt eine Überlappung zwischen den Teilschlüsselbäumen der beiden Transkripte. Dann kennt die Bank ein x mit $g_{l+1,0}^{x} = LK'$. Gemäß dem ZKP in Υ' gibt es ein α_{l+1} mit $LK' = g_{l+1,0}^{\alpha_{l+1}}$

[8]Sind die verwendeten Gruppen nämlich hinreichend groß, ist die Wahrscheinlichkeit der Kollision zweier unabhängiger Zufallszahlen vernachlässigbar.

und $T' = g^{sk'_K} g_{l+1,2}^{\alpha_{l+1}R'}$. Identifiziert nun die Bank den Doppelzahler erhält sie $g^{sk''_K} = pk''_K = T' g_{l+1,2}^{-xR'} = T' g_{l+1,2}^{-\alpha_{l+1}R'} = g^{sk'_K}$, also wieder $sk'_K = sk''_K$.

Im dritten Fall gilt $L' < L$, und Υ ist ein ungefälschtes Transkript eines echten Kunden, da sonst Fall 2 gelten würde. Die Bank erkennt eine Doppelzahlung, indem sie aus LK' und RK' den Wurzelknoten $K_{L,i}$ von Υ und die Identität $pk''_K = Tg^{-K_{L,i}R}$ berechnet. Gilt $pk''_K = pk_K$ kann mit dem gefälschten Transkript Υ' die in Υ kodierte Identität ermittelt werden, was einen Widerspruch zur Anonymitätseigenschaft darstellt. Gilt $pk''_K \neq pk_K$ muss $Tg^{-K_{L,i}R}$ der Identität eines anderen Kunden entsprechen. Der Fälscher hat also ein $K_{L,i}$ gefunden, dass die Gleichung $pk''_K = Tg^{-K_{L,i}R} \Leftrightarrow g^{RK_{L,i}} = T/pk''_K$ für feste g, T, R und ein $pk''_K \in \{\text{Kunden}\}$ löst. Das bedeutet, unter Annahme einer realistischen/polynomiellen Zahl von Kunden, dass er diskrete Logarithmen berechnen kann, was unter anderem der DDH-Annahme widerspricht.

Unter Annahme der DDH-Annahme, der Sicherheit der ZKPs und der gesicherten Anoymität, ist das Verfahren also entlastend.

5 Zusammenfassung

Das E-Cash System von Canard und Gouget scheint die nötige Sicherheit zu erreichen und zeigt damit, dass teilbare, anonyme E-Cash Systeme möglich sind.

Es ist allerdings an manchen Stellen ineffizient, obwohl der Vorteil der Teilbarkeit doch gerade eine Effizienzsteigerung sein soll.

Für die Berechnung des Schlüsselbaums benötigt man für die Gruppenordnungen eine Sequenz von Primzahlen p_i mit $p_i = 2p_{i-1}+1$ und es ist nicht klar, wie sie effizient gefunden werden kann. Wählt man zufällig eine Primzahl und berechnet die entsprechende Sequenz, so liegt die Erfolgswahrscheinlichkeit, dass alle Zahlen prim sind, zwischen 10^{-20} und 10^{-70} je nach Tiefe des Baums und Größe der Gruppen[1]. Andererseits sind diese Gruppen öffentlich und unabhängig von den Schlüsseln; findet man also eine einzige solche Sequenz, können alle Banken mit denselben Gruppen arbeiten.

Der beim Bezahlprotokoll verwendete ZKP 4 ist recht ineffizient, vor allem da er die Kenntnis mehrere Doppellogarithmen zeigen muss.

Und schließlich muss die Bank für die Erkennung einer Doppelzahlung alle Blätter des Schlüsselbaums mit den Blättern aller jemals eingezahlten Bäume vergleichen. Das heißt, das Verfahren hat für die Bank lineare Zeit-

und sogar Speicherkosten[9].

Au, Susilo und Mu [1] schlagen inspiriert vom Canard/Gouget-System ein effizienteres System vor.

Es gibt auch einige theoretische Nachteile. So kann der Kunde die Münze verlieren, und kann nicht beweisen, dass er sie nicht ausgegeben hat. Außerdem kann nur die Bank Doppelzahlungen erkennen – obwohl sie die Transkripte als Beweis einer erkannten Doppelzahlung weitergeben kann – und der Händler muss sich darauf verlassen, dass jeder Kunde ehrlich ist. Hätte das Verfahren keine lineare Speicherkosten, könnte die Bank ihre Datenbank der eingezahlten Blattknoten an alle Händler weitergeben.

Des Weiteren muss der Kunde dem Händler vertrauen, tatsächlich seine Ware zu erhalten. Er kann zwar mit dem Transkript der Bank beweisen, dass er mit der Münze bezahlt hat[10], aber dies garantiert nicht, die Münze oder eine Entschädigung zurückzuerhalten. Ein anderes teilbares und anonymes E-Cash Verfahren von Camenisch und Lysyanskaya[4] ermöglicht dem Kunden, seine Münze in solchen Fällen nachträglich zurückzuziehen.

Literatur

[1] Man Ho Au, Willy Susilo, and Yi Mu. Practical anonymous divisible e-cash from bounded accumulators. In *In Financial Cryptography and Data Security*, 2008.

[2] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *In EUROCRYPT, volume 3494 of LNCS*, pages 302–321. Springer-Verlag, 2005.

[3] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. pages 56–72. Springer-Verlag, 2004.

[4] Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. Endorsed e-cash. *Security and Privacy, IEEE Symposium on*, 0:101–115, 2007.

[5] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of the 17th Annual*

[9] Beziehungsweise sogar exponentielle falls man die Komplexität wie üblich nach der Bitlänge der Beträge misst.

[10] Sofern der Händler die Münze eingezahlt hat.

International Cryptology Conference on Advances in Cryptology, pages 410–424, London, UK, 1997. Springer-Verlag.

[6] Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In *In EUROCRYPT*, pages 482–497, 2007.

[7] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology – Proceedings of CRYPTO 82*, pages 199–203. Plenum Press, 1982.

[8] Ronald Cramer, Ronald Cramer Cwi, and Berry Schoenmakers Cwi. Proofs of partial knowledge and simplified design of witness hiding protocols, 1994.

[9] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. pages 186–194. Springer-Verlag, 1987.

[10] Toru Nakanishi and Yuji Sugiyama. Unlinkable divisible electronic cash. In *Proc. Third International Workshop on Information Security (ISW2000), LNCS 1975, pp.121– 134*, pages 121–134. Springer–Verlag, 2000.

[11] Markus Stadler. Publicly verifiable secret sharing. pages 190–199. Springer-Verlag, 1996.