Bitcoin: Ein elektronisches Peer-to-Peer- Cash-System

Translated in German from bitcoin.org/bitcoin.pdf by 99Bitcoins

Satoshi Nakamoto satoshin@gmx.com www.bitcoin.org

Abstrakt. Eine reine Peer-to-Peer-Version von elektronischem Bargeld würde es ermöglichen, Online-Zahlungen direkt von einer Partei an eine andere zu senden, ohne ein Finanzinstitut bemühen zu müssen. Digitale Signaturen stellen einen Teil der Lösung dar, aber die Hauptvorteile gehen verloren, wenn eine vertrauenswürdige dritte Partei weiterhin erforderlich ist, um Doppelausgaben zu vermeiden. Wir schlagen eine Lösung für das Problem der doppelten Ausgaben unter Verwendung eines Peer-to-Peer-Netzwerks vor. Die Netzwerk-Timestamp tasten Transaktionen ab, indem sie sie in eine fortlaufende Kette von Hash-basierten Arbeitsnachweisen einteilen, die einen Datensatz bilden, der nicht geändert werden kann, ohne den Arbeitsnachweis zu wiederholen. Die längste Kette dient nicht nur als Beweis für die beobachtete Ereignisfolge, sondern auch als Beweis dafür, dass sie aus dem größten Pool an CPU-Leistung stammt. Solange ein Großteil der CPU-Leistung von Knoten gesteuert wird, die nicht zusammenarbeiten, um das Netzwerk anzugreifen, generieren sie die längsten Ketten- und Outpace-Angreifer. Das Netzwerk selbst benötigt eine minimale Struktur. Die Nachrichten werden nach bestmöglichen Bemühens gesendet, und die Knoten können nach Belieben das Netzwerk verlassen und sich wieder an das Netzwerk anschliessen, wobei sie die längste Beweisstückkette akzeptieren, als Beweis dafür, was passiert ist, während sie weg waren.

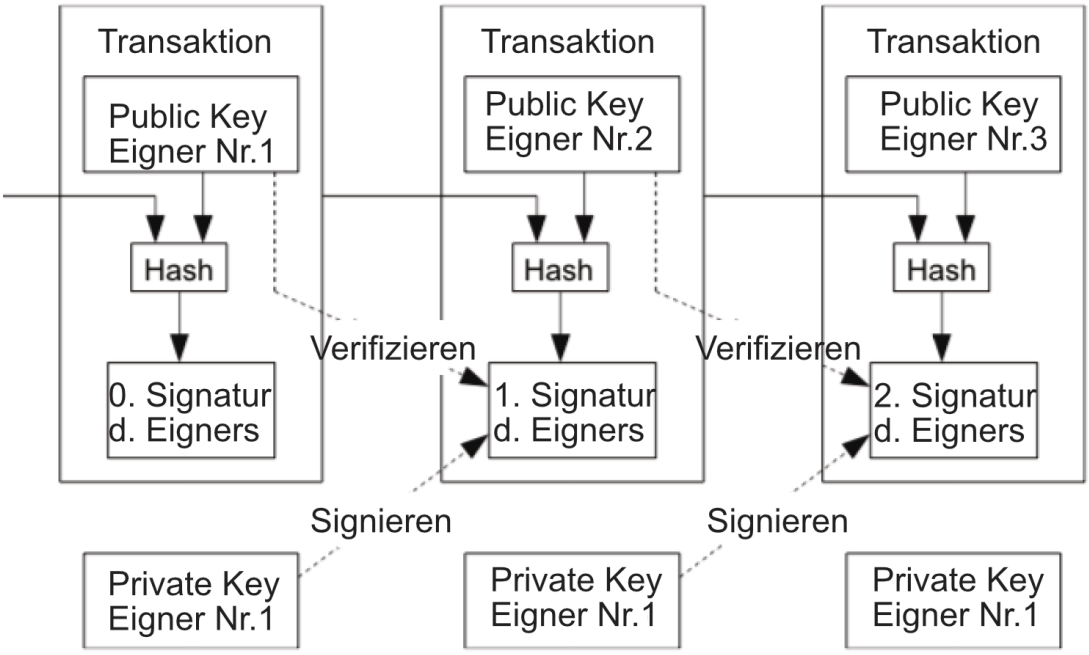
1. Einführung

Der Handel im Internet stützt sich fast ausschließlich auf Finanzinstitute, die als vertrauenswürdige Dritte für die Abwicklung elektronischer Zahlungen agieren. Während das System für die meisten Transaktionen gut genug funktioniert, leidet es immer noch unter den inhärenten Schwächen des auf Vertrauen basierenden Modells. Vollständig nicht umkehrbare Transaktionen sind nicht wirklich möglich, da Finanzinstitute die Vermittlung von Streitigkeiten nicht vermeiden können. Die Kosten einer Mediation erhöhen die Transaktionskosten, schränken die praktische Mindesttransaktionsgröße ein und verhindern die Möglichkeit für kleine Gelegenheitstransaktionen, und der Verlust der Fähigkeit, nicht reversible Zahlungen für nicht umkehrbare Dienste zu leisten, wird breiter angelegt. Mit der Möglichkeit der Umkehrung breitet sich das Bedürfnis nach Vertrauen aus. Händler müssen sich vor ihren Kunden in Acht nehmen und sie um mehr Informationen bitten, als sie sonst benötigen würden. Ein gewisser Prozentsatz an Betrugs wird als unvermeidbar akzeptiert. Diese Kosten und Zahlungsunsicherheiten können durch die Verwendung physischer Währung persönlich vermieden werden, aber es gibt keinen Mechanismus, um Zahlungen über einen Kommunikationskanal ohne eine vertrauenswürdige Partei zu tätigen.

Was benötigt wird, ist ein elektronisches Zahlungssystem, das auf einem kryptografischen Beweis statt auf Vertrauen basiert, wodurch zwei bereitwillige Parteien direkt miteinander arbeiten können, ohne dass eine vertrauenswürdige dritte Partei benötigt wird. Transaktionen, die rechnerisch nicht umkehrbar sind, würden Verkäufer vor Betrug schützen, und routinemäßige Anderskonto-Mechanismen könnten leicht implementiert werden, um Käufer zu schützen. In diesem Artikel schlagen wir eine Lösung für das Problem der doppelten Ausgabe unter Verwendung eines verteilten Peer-to-Peer-Zeitstempels vor, um einen rechnerischen Nachweis der chronologischen Reihenfolge der Transaktionen zu erstellen. Das System ist sicher, solange vertrauenswürdige Knoten gemeinsam mehr CPU-Leistung steuern als kooperierende Gruppen von Angreiferknoten.

2. Transaktionen

Wir definieren eine elektronische Münze als eine Kette von digitalen Signaturen. Jeder Besitzer überträgt die Münze auf die nächste, indem er ein Hash der vorherigen Transaktion und den öffentlichen Schlüssel des nächsten Besitzers digital signiert und diese zum Ende der Münze hinzufügt. Ein Zahlungsempfänger kann die Signaturen überprüfen, um die Eigentumskette zu überprüfen.

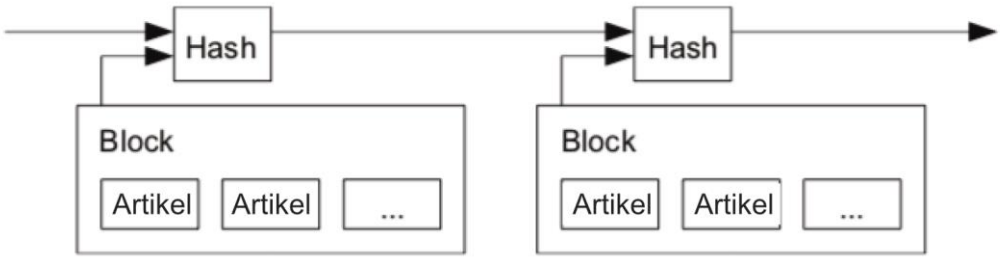


Das Problem ist natürlich, dass der Zahlungsempfänger nicht verifizieren kann, dass einer der Besitzer die Münze nicht doppelt ausgegeben hat. Eine gängige Lösung ist die Einführung einer vertrauenswürdigen zentralen Behörde (mint), die jede Transaktion auf doppelte Ausgaben überprüft. Nach jeder Transaktion muss die Münze an die Münzstätte zurückgegeben werden, um eine neue Münze auszugeben, und nur Münzen, die direkt von der Münzstätte ausgegeben werden, werden nicht doppelt ausgegeben. Das Problem bei dieser Lösung ist, dass das Schicksal des gesamten Geldsystems davon abhängt, dass die Münzanstalt die Münzstätte betreibt, und dass jede Transaktion genau wie eine Bank durch sie gehen muss.

Wir brauchen einen Weg für den Zahlungsempfänger, um zu wissen, dass die früheren Eigentümer keine früheren Transaktionen unterzeichnet haben. Für unsere Zwecke ist die früheste Transaktion die, die zählt, so dass uns spätere Versuche, doppelt auszugeben, egal sind. Die einzige Möglichkeit, das Fehlen einer Transaktion zu bestätigen, besteht darin, alle Transaktionen zu kennen. Im mintbasierten Modell (behördenbasierten Modell) war die Münzstätte über alle Transaktionen informiert und entschied, welche zuerst eintraf. Um dies ohne eine vertrauenswürdige Partei zu erreichen, müssen Transaktionen öffentlich angekündigt werden [1], und wir brauchen ein System für die Teilnehmer, um sich auf einen einzelnen Verlauf der Reihenfolge zu einigen, in der sie empfangen wurden. Der Zahlungsempfänger benötigt den Nachweis, dass zum Zeitpunkt jeder Transaktion die Mehrheit der Knoten zustimmte, dass es der erste Empfänger war.

3. Zeitstempelserver

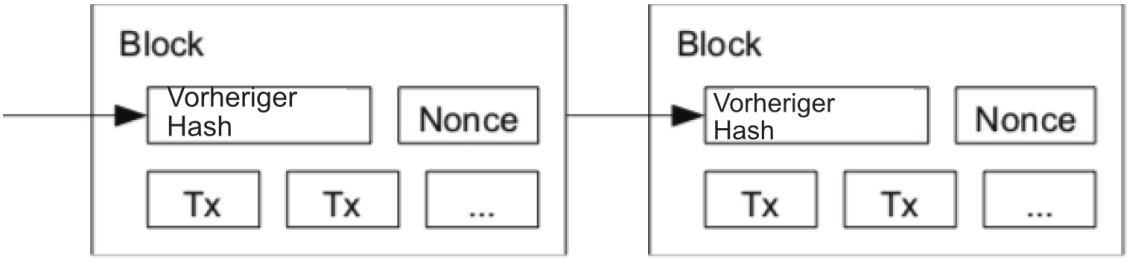
Die von uns vorgeschlagene Lösung beginnt mit einem Zeitstempelserver. Ein Zeitstempel-Server arbeitet mit einem Hash eines Blocks von Elementen, die mit einem Zeitstempel versehen werden sollen, und publiziert den Hash weit, beispielsweise in einer Zeitung oder einem Usenet-Post [2-5]. Der Zeitstempel beweist, dass die Daten offensichtlich zu der Zeit existieren mussten, um in den Hash zu gelangen. Jeder Zeitstempel enthält den vorherigen Zeitstempel in seinem Hash, der eine Kette bildet, wobei jeder zusätzliche Zeitstempel die vorherigen verstärkt.



4. Arbeitsnachweis

Um einen Verteilte Zeitstempel Server auf Peer-to-Peer-Basis zu implementieren, müssen wir ein Ausführungsbeweis-System verwenden, das Adam Hacks HashCash [6] ähnelt, statt Zeitungen oder Usenet-Posts. Der Arbeitsbeweis umfasst das Scannen nach einem Wert, der bei der Hash-Verarbeitung, wie bei SHA-256, mit einer Anzahl von Null-Bits beginnt. Die erforderliche durchschnittliche Arbeit ist exponentiell in der Anzahl der erforderlichen Null-Bits und kann verifiziert werden, indem ein einzelner Hash ausgeführt wird.

Für unser Zeitstempel-Netzwerk implementieren wir den Arbeitsbeweis, indem wir eine Nonce im Block inkrementieren, bis ein Wert gefunden wird, der dem Block-Hash die erforderlichen Null-Bits gibt. Sobald der CPU-Aufwand aufgebraucht ist, um den Arbeitsbeweis zu erfüllen, kann der Block nicht geändert werden, ohne die Arbeit erneut auszuführen. Da spätere Blöcke nach ihm verkettet werden, würde die Arbeit zum Ändern des Blocks das Wiederholen aller nachfolgenden Blöcke umfassen.



Der Arbeitsnachweis löst auch das Problem der Repräsentation in der Mehrheitsentscheidung. Wenn die Mehrheit auf einer IP-Adresse mit einer Stimme basiert, könnte sie von jedem unterlaufen werden, der viele IPs vergeben kann. Arbeitsnachweis ist im Wesentlichen eine CPU - eine Stimme als Beweis. Die Mehrheitsentscheidung wird durch die längste Kette dargestellt, in die der größte Arbeitsaufwand investiert wird. Wenn ein Großteil der CPU-Leistung durch vertrauenswürdige Knoten gesteuert wird, wächst die ehrliche Kette am schnellsten und übertrifft alle konkurrierenden Ketten. Um einen vergangenen Block zu modifizieren, müsste ein Angreifer den Funktionsnachweis des Blocks und aller Blöcke danach wiederholen und dann die Arbeit der vertrauenswürdigen Knoten einholen und überholen. Wir werden später zeigen, dass die Wahrscheinlichkeit eines Aufholens eines langsameren Angreifers exponentiell abnimmt, wenn nachfolgende Blöcke hinzugefügt werden.

Um die zunehmende Hardwaregeschwindigkeit und das variierende Interesse an Laufknoten im Laufe der Zeit zu kompensieren, wird die Schwierigkeit des Nachweises der Arbeit durch einen gleitenden Durchschnitt bestimmt, der auf eine durchschnittliche Anzahl von Blöcken pro Stunde abzielt. Wenn sie zu schnell generiert werden, erhöht sich die Schwierigkeit.

5. Netzwerk

Die Schritte zum Ausführen des Netzwerks lauten wie folgt:

1. 1) Neue Transaktionen werden an alle Knoten gesendet.
2. 2) Jeder Knoten sammelt neue Transaktionen in einem Block.
3. 3) Jeder Knoten arbeitet daran, einen schwierigen Arbeitsbeweis für seinen Block zu finden.
4. 4) Wenn ein Knoten einen Arbeitsnachweis findet, sendet er den Block an alle Knoten.
5. 5) Knoten akzeptieren den Block nur dann, wenn alle darin enthaltenen Transaktionen gültig und nicht bereits verbraucht sind.
6. 6) Knoten drücken ihre Akzeptanz des Blocks aus, indem sie an der Erstellung des nächsten Blocks in der

   Kette, wobei der Hash des akzeptierten Blocks als der vorherige Hash verwendet wird.

Knoten halten immer die längste Kette für die richtige und arbeiten weiter daran, sie zu verlängern. Wenn zwei Knoten gleichzeitig verschiedene Versionen des nächsten Blocks senden, können einige Knoten das eine oder das andere zuerst empfangen. In diesem Fall arbeiten sie an der ersten, die sie erhalten haben, aber speichern Sie die andere Verzweigung für den Fall, dass sie länger wird. Die Verbindung wird unterbrochen, wenn der nächste Arbeitsnachweis gefunden wird und ein Zweig länger wird; Die Knoten, die an dem anderen Zweig arbeiteten, werden dann zu dem längeren Zweig wechseln.

Neue Transaktionsübertragungen müssen nicht notwendigerweise alle Knoten erreichen. Solange sie viele Knoten erreichen, geraten sie in Kürze in einen Block. Block-Sendungen sind auch tolerant gegenüber gelöschten Nachrichten. Wenn ein Knoten keinen Block empfängt, wird er es anfordern, wenn er den nächsten Block empfängt und erkennt, dass er einen verpasst hat.

6. Anreiz

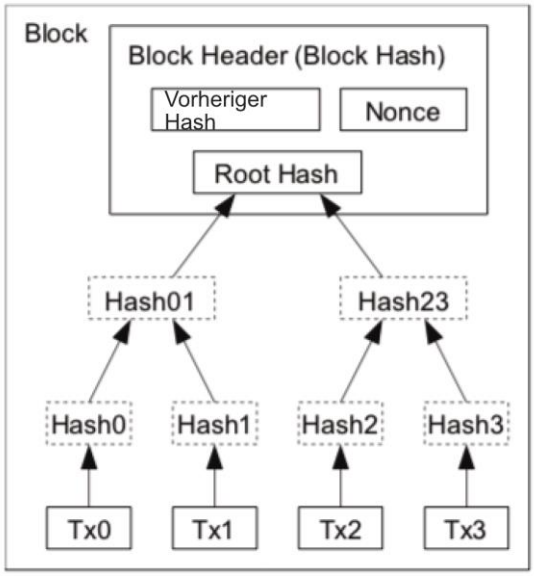
Gemäß der Konvention ist die erste Transaktion in einem Block eine spezielle Transaktion, die eine neue Münze startet, die dem Ersteller des Blocks gehört. Dies stellt einen Anreiz für die Knoten dar, das Netzwerk zu unterstützen, und bietet eine Möglichkeit, Münzen zunächst in Umlauf zu bringen, da es keine zentrale Behörde gibt, um diese auszugeben. Die stetige Addition einer konstanten Menge neuer Münzen ist analog zu Goldminen, die Ressourcen ausgeben, um Gold in den Umlauf zu bringen. In unserem Fall werden CPU-Zeit und Strom verbraucht.

Der Anreiz kann auch mit Transaktionsgebühren finanziert werden. Wenn der Ausgabewert einer Transaktion kleiner ist als der eingegebene Wert, ist die Differenz eine Transaktionsgebühr, die zum Anreizwert des Blocks hinzugefügt wird, der die Transaktion enthält. Sobald eine vorbestimmte Anzahl von Münzen in Umlauf gelangt ist, kann der Anreiz vollständig auf Transaktionsgebühren übergehen und vollständig inflationsfrei sein.

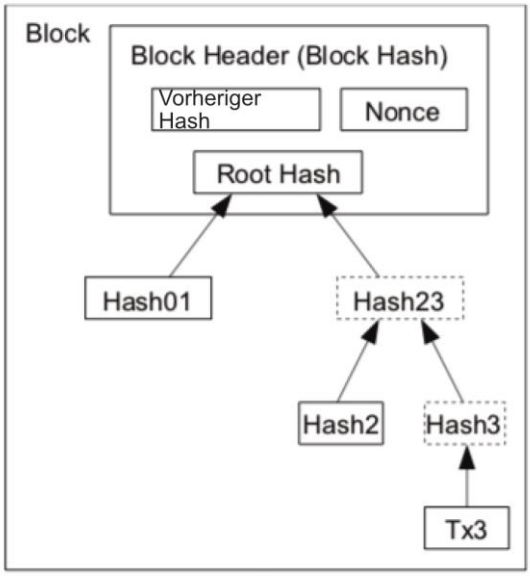
Der Anreiz kann dazu beitragen, Knoten zu ermutigen, vertrauenswürdig zu bleiben. Wenn ein gieriger Angreifer in der Lage ist, mehr CPU-Leistung zu sammeln als alle ehrlichen Knoten, müsste er sich entscheiden, ob er damit Menschen betrügen könnte, indem er seine Zahlungen stehlen oder neue Münzen generieren würde. Er sollte es profitabler finden, nach den Regeln zu spielen, die ihn mit mehr neuen Münzen begünstigen als alle anderen zusammen, als das System und die Gültigkeit seines eigenen Reichtums zu untergraben.

7. Speicherplatz zurückfordern

Sobald die letzte Transaktion in einer Münze unter genügend Blöcken begraben ist, können die zuvor getätigten Transaktionen verworfen werden, um Speicherplatz zu sparen. Um dies zu erleichtern, ohne den Hash des Blocks zu knacken, werden Transaktionen in einem Merkle-Baum [7] [2] [5] gehashed, wobei nur der root im Hash des Blocks enthalten ist. Alte Blöcke können dann verdichtet werden, indem man Äste des Baumes abstößt. Die inneren Hashes müssen nicht gespeichert werden.



In Merkle Baum gehashte Transaktionen



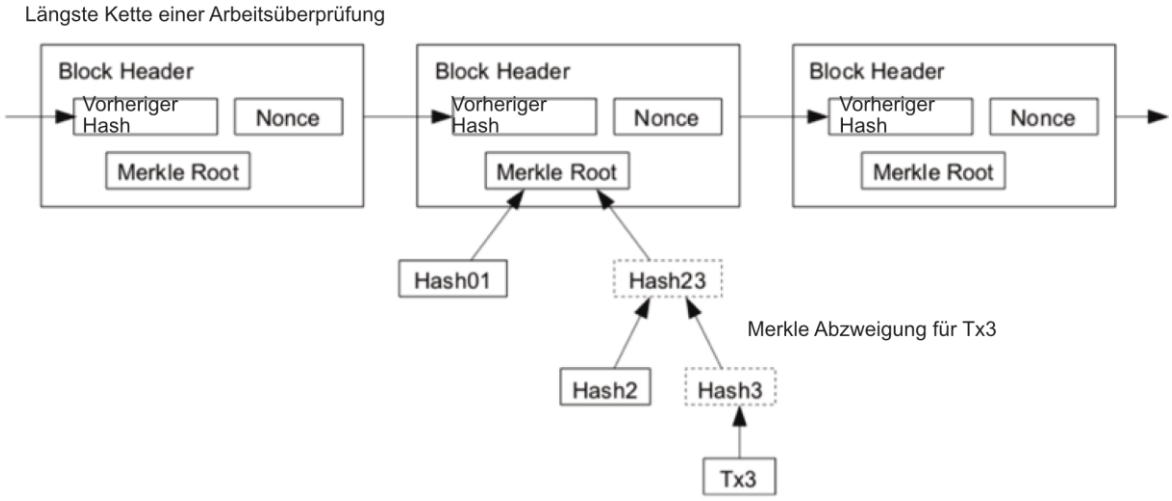
Nachdem Tx 0-2 von dem Block entfernt wurde

Ein Blockheader ohne Transaktionen würde ungefähr 80 Bytes umfassen. Wenn wir annehmen, dass Blöcke alle 10 Minuten erzeugt werden, 80 Bytes * 6 * 24 * 365 = 4,2 MB pro Jahr. Mit Computersystemen, die typischerweise ab 2008 mit 2 GB RAM verkauft werden, und dem Mooreschen Gesetz, das ein aktuelles Wachstum von 1,2 GB pro Jahr vorhersagt, sollte der Speicher kein Problem sein, selbst wenn die Blockheader im Speicher gehalten werden müssen.

4

8. Vereinfachte Zahlungsüberprüfung

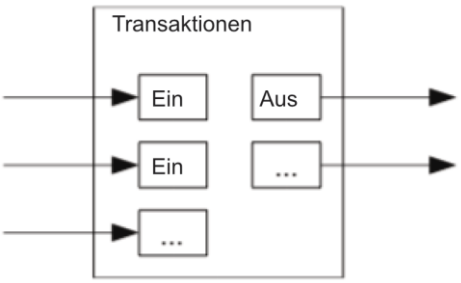
Es ist möglich, Zahlungen zu überprüfen, ohne einen vollständigen Netzwerkknoten auszuführen. Ein Benutzer muss nur eine Kopie der Block-Header der längsten Arbeitsbeweis aufbewahren, die er durch Abfragen von Netzwerkknoten erhalten kann, bis er überzeugt ist, dass er die längste Kette hat, und den Merkle-Zweig erhält, der die Transaktion mit dem Block verbindet Es ist Zeitstempel in. Er kann die Transaktion nicht für sich selbst überprüfen, sondern indem er sie mit einem Ort in der Kette verknüpft, kann er sehen, dass ein Netzwerkknoten ihn akzeptiert hat und Blöcke hinzufügen, nachdem er weiter bestätigt hat, dass das Netzwerk ihn akzeptiert hat.



Daher ist die Verifizierung zuverlässig, solange ehrliche Knoten das Netzwerk steuern, aber anfälliger sind, wenn das Netzwerk von einem Angreifer überlastet wird. Während Netzwerkknoten Transaktionen für sich selbst verifizieren können, kann die vereinfachte Methode durch die fabrizierten Transaktionen eines Angreifers getäuscht werden, solange der Angreifer weiterhin das Netzwerk überwältigen kann. Eine Strategie zum Schutz davor wäre, Warnungen von Netzwerkknoten zu akzeptieren, wenn sie einen ungültigen Block erkennen, und die Software des Benutzers dazu auffordert, den vollständigen Block und die alarmierten Transaktionen herunterzuladen, um die Inkonsistenz zu bestätigen. Unternehmen, die häufig Zahlungen erhalten, werden wahrscheinlich immer noch ihre eigenen Knoten für mehr unabhängige Sicherheit und schnellere Verifizierung betreiben wollen.

9. Wert kombinieren und teilen

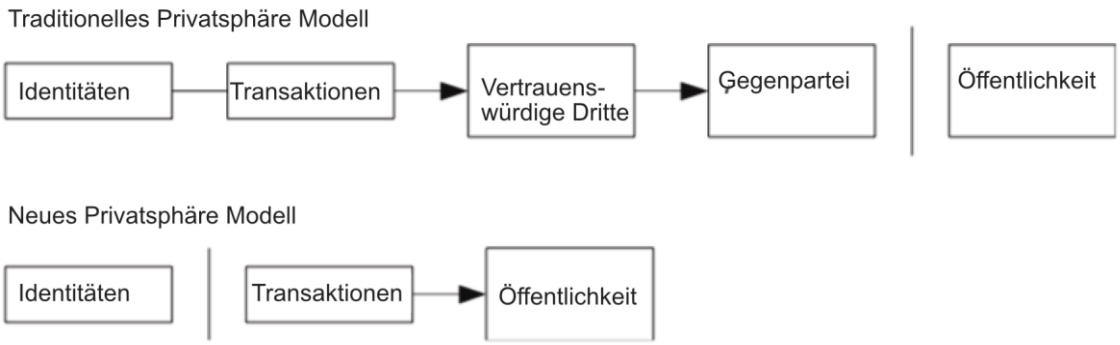
Obwohl es möglich wäre, Münzen einzeln zu handhaben, wäre es unhandlich, für jeden Cent einer Überweisung eine separate Transaktion durchzuführen. Damit der Wert aufgeteilt und kombiniert werden kann, enthalten Transaktionen mehrere Eingaben und Ausgaben. Normalerweise gibt es entweder eine einzelne Eingabe von einer größeren vorherigen Transaktion oder mehrere Eingaben, die kleinere Mengen kombinieren, und höchstens zwei Ausgaben: eine für die Zahlung und eine, die die Änderung, falls vorhanden, zurück an den Absender zurückgibt.



Es sollte angemerkt werden, dass Fan-Out, bei dem eine Transaktion von mehreren Transaktionen abhängt und diese Transaktionen von viel mehr abhängen, kein Problem ist. Es ist niemals erforderlich, eine vollständige eigenständige Kopie des Transaktionsverlaufs zu extrahieren.

10. Privatsphäre

Das traditionelle Bankenmodell erreicht ein gewisses Maß an Privatsphäre, indem es den Zugang zu Informationen für die beteiligten Parteien und die vertrauenswürdige dritte Partei einschränkt. Die Notwendigkeit, alle Transaktionen öffentlich anzukündigen, schließt diese Methode aus, aber die Privatsphäre kann weiterhin aufrechterhalten werden, indem der Informationsfluss an anderer Stelle unterbrochen wird: indem öffentliche Schlüssel anonym bleiben. Die Öffentlichkeit kann sehen, dass jemand einen Betrag an jemand anderen sendet, aber ohne Information, die die Transaktion mit irgendjemandem verbindet. Dies ist ähnlich dem Informationsniveau, das von den Börsen herausgegeben wird, wo die Zeit und die Größe der einzelnen Geschäfte, das "Band", veröffentlicht werden, ohne jedoch zu sagen, wer die Parteien waren.



Als zusätzliche Firewall sollte für jede Transaktion ein neues Schlüsselpaar verwendet werden, damit sie nicht mit einem gemeinsamen Eigentümer verknüpft werden. Einige Verknüpfungen sind bei Transaktionen mit mehreren Eingängen immer noch unvermeidlich, die notwendigerweise zeigen, dass ihre Eingaben demselben Besitzer gehören. Das Risiko besteht darin, dass bei der Offenlegung des Besitzers eines Schlüssels durch die Verknüpfung andere Transaktionen angezeigt werden können, die demselben Besitzer gehören.

11. Berechnungen

Wir betrachten das Szenario eines Angreifers, der versucht, eine alternative Kette schneller als die ehrliche Kette zu erzeugen. Selbst wenn dies erreicht wird, wird das System nicht für beliebige Änderungen geöffnet, wie zum Beispiel um Werte aus der Luft zu schaffen oder Geld zu nehmen, das niemals dem Angreifer gehörte. Knoten akzeptieren keine ungültige Transaktion als Zahlung, und ehrliche Knoten akzeptieren niemals einen Block, der sie enthält. Ein Angreifer kann nur versuchen, eine seiner eigenen Transaktionen zu ändern, um das Geld zurückzuerhalten, das er kürzlich ausgegeben hat.

Das Rennen zwischen der ehrlichen Kette und einer Angreiferkette kann als ein Binomial-Zufallslauf charakterisiert werden. Das Erfolgsereignis ist, dass die ehrliche Kette um einen Block erweitert wird, wobei der Vorsprung um +1 erhöht wird, und das Fehlerereignis ist, dass die Kette des Angreifers um einen Block verlängert wird, wodurch die Lücke um -1 verringert wird.

Die Wahrscheinlichkeit, dass ein Angreifer von einem bestimmten Defizit aufholt, ist analog zu einem Gambler's Ruin Problem. Angenommen, ein Spieler mit unbegrenztem Kredit beginnt mit einem Defizit und spielt potenziell eine unendliche Anzahl von Versuchen, um zu versuchen, den Breakeven zu erreichen. Wir können die Wahrscheinlichkeit berechnen, mit der er jemals die Gewinnschwelle erreicht, oder dass ein Angreifer die ehrliche Kette jemals einholt, wie folgt [8]:

p = Wahrscheinlichkeit, dass ein ehrlicher Knoten den nächsten Block findet

q = Wahrscheinlichkeit, dass der Angreifer den nächsten Block findet

qz = Wahrscheinlichkeit, dass der Angreifer jemals von Z-Blöcken dahinter aufholt

$$q_z = \begin{cases} 1 & if\ p \le q \\ (q/p)^z & if\ p > q \end{cases}$$

Unter der Annahme, dass p > q sinkt die Wahrscheinlichkeit exponentiell, da die Anzahl der Blöcke, die der Angreifer einholen muss, steigt. Mit den Chancen gegen ihn, wenn er früh nicht einen glücklichen Ausfall macht, werden seine Chancen verschwindend klein, wie er weiter hinten fällt.

Wir betrachten nun, wie lange der Empfänger einer neuen Transaktion warten muss, bevor er ausreichend sicher ist, dass der Absender die Transaktion nicht ändern kann. Wir gehen davon aus, dass der Absender ein Angreifer ist, der den Empfänger glauben machen will, dass er ihn für eine Weile bezahlt hat, und ihn dann nach einiger Zeit wieder an sich selbst zurückzahlen lässt. Der Empfänger wird benachrichtigt, wenn dies geschieht, aber der Absender hofft, dass es zu spät sein wird.

Der Empfänger generiert ein neues Schlüsselpaar und gibt den öffentlichen Schlüssel kurz vor dem Signieren an den Absender aus. Dies verhindert, dass der Absender eine Kette von Blöcken im Voraus vorbereitet, indem er kontinuierlich daran arbeitet, bis er das Glück hat, weit genug voraus zu sein, und dann die Transaktion in diesem Moment ausführt. Sobald die Transaktion gesendet wurde, beginnt der unredliche Absender geheim an einer parallelen Kette zu arbeiten, die eine alternative Version seiner Transaktion enthält.

Der Empfänger wartet, bis die Transaktion zu einem Block hinzugefügt wurde und z Blöcke wurden danach verknüpft. Er weiß nicht genau, wie viel Fortschritt der Angreifer gemacht hat, aber unter der Annahme, dass die ehrlichen Blöcke die durchschnittlich erwartete Zeit pro Block genommen haben, ist der potentielle Fortschritt des Angreifers eine Poisson-Verteilung mit dem erwarteten Wert:

$$\lambda = z \frac{q}{p}$$

Um die Wahrscheinlichkeit zu ermitteln, mit der der Angreifer noch aufholen könnte, multiplizieren wir die Poisson-Dichte für jede Fortschrittsmenge mit der Wahrscheinlichkeit, dass er von diesem Punkt aus aufholen könnte:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

Neuanordnung, um zu vermeiden, den unendlichen Schwanz der Verteilung zu summieren ...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Konvertieren in C-Code ...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```


Einige Ergebnisse können wir sehen, dass die Wahrscheinlichkeit exponentiell mit z abfällt.

```
q = 0,1
z = 0 P = 1,0000000
z = 1 P = 0,2045873
z = 2 P = 0,0509779
z = 3 P = 0,0131722
z = 4 P = 0,0034552
z = 5 P = 0,0009137
z = 6 P = 0,0002428
z = 7 P = 0,0000647
z = 8 P = 0,0000173
z = 9 P = 0,0000046
z = 10 P = 0,0000012
q = 0,3
z = 0 P = 1,0000000
z = 5 P = 0,1773523
z = 10 P = 0,0416605
z = 15 P = 0,0101008
z = 20 P = 0,0024804
z = 25 P = 0,0006132
z = 30 P = 0,0001522
z = 35 P = 0,0000379
z = 40 P = 0,0000095
z = 45 P = 0,0000024
z = 50 P = 0,0000006
```

Lösung für P weniger als 0,1% ...

```
P < 0,001
q = 0,10 z = 5
q = 0,15 z = 8
q = 0,20 z = 11
q = 0,25 z = 15
q = 0,30 z = 24
q = 0,35 z = 41
q = 0,40 z = 89
q = 0,45 z = 340
```

12. Fazit

Wir haben ein System für elektronische Transaktionen vorgeschlagen, ohne auf Vertrauen angewiesen zu sein. Wir begannen mit dem üblichen Rahmen von Münzen, die aus digitalen Unterschriften hergestellt wurden, was eine starke Eigentumskontrolle ermöglicht, aber unvollständig ist, ohne einen doppelten Aufwand zu verhindern. Um dieses Problem zu lösen, haben wir ein Peer-to-Peer-Netzwerk vorgeschlagen, das eine öffentliche Historie von Transaktionen aufzeichnet, die für einen Angreifer schnell unpraktisch werden, wenn ehrliche Knoten einen Großteil der CPU-Leistung steuern. Das Netzwerk ist robust in seiner unstrukturierten Einfachheit. Knoten arbeiten alle auf einmal mit nur wenig Koordination. Sie müssen nicht identifiziert werden, da Nachrichten nicht

an einen bestimmten Ort geroutet werden und nur auf der Grundlage des bestmöglichen Aufwands geliefert werden müssen. Knoten können das Netzwerk nach Belieben verlassen und wieder betreten, indem sie die Beweisstückkette als Beweis dafür akzeptieren, was passiert ist, während sie weg waren. Sie stimmen mit ihrer CPU-Leistung ab und drücken ihre Akzeptanz gültiger Blöcke aus, indem sie daran arbeiten, sie zu erweitern und ungültige Blöcke abzulehnen, indem sie sich weigern, an ihnen zu arbeiten. Alle erforderlichen Regeln und Anreize können mit diesem Konsensmechanismus durchgesetzt werden.

8

Verweise

1. W. Dai, "B-Geld", http://www.weidai.com/bmoney.txt, 1998.
2. H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, Mai 1999.
3. S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, , Bd. 3, Nr 2, Seiten 99-111, 1991.
4. [4] D. Bayer, S. Haber, WS Stornetta, "Verbesserung der Effizienz und Zuverlässigkeit der digitalen Zeitstempel" Im Sequenzen II: Methoden in Kommunikation, Sicherheit und InformatikSeiten 329-334, 1993.
5. [5] S. Haber, WS Stornetta, "Sichere Namen für Bitstrings", In Proceedings der 4. ACM Konferenz auf Computer- und Kommunikationssicherheit, Seiten 28-35, April 1997.
6. A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.
7. R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, Seiten 122-133, April 1980.
8. W. Feller, "An introduction to probability theory and its applications," 1957.