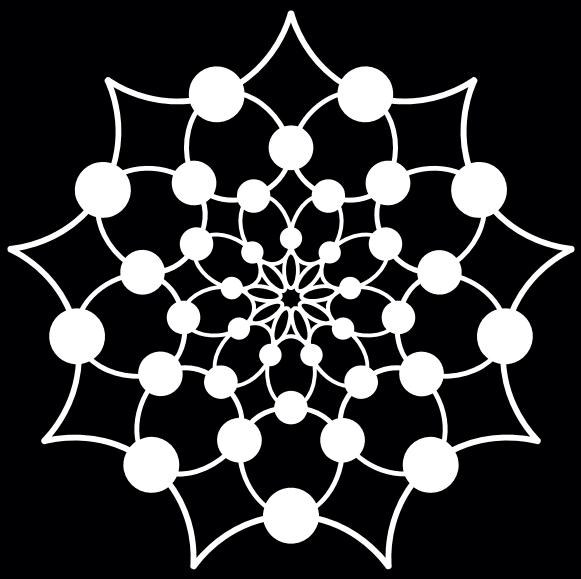
Bitcoin

Ein elektronisches Peer-to-Peer Geldsystem



Satoshi Nakamoto

Deutsche Fassung by

BTC-ECHO

Das Bitcoin Whitepaper

Deutsche Fassung

www.btc-echo.de

BTC-ECHO

BTC-ECHO

Bitcoin: Ein elektronisches Peer-to-Peer Geldsystem

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

Abstract: Eine reine Peer-to-Peer-Version elektronischen Geldes würde es ermöglichen, Online-Zahlungen direkt von einer Partei zur anderen zu senden, ohne dabei eine Finanzinstitution als Mittler zu benötigen. Digitale Signaturen können dies zwar bereits teilweise leisten, die größten Vorteile gehen jedoch trotzdem verloren: Um ein Double-Spending zu verhindern ist immer noch Vertrauen in eine dritte Partei notwendig. Wir schlagen eine Lösung für das Double-Spending-Problem vor, indem wir ein Peer-to-Peer-Netzwerk nutzen. Das Netzwerk versieht Transaktionen mit einem Zeitstempel, indem sie in eine fortlaufende Kette vom so genannten Hash-based Proof-of-Work eingefügt wird. Diese Kette stellt dann ein unveränderliches Protokoll dar. Die längste Kette dient nicht nur als Beweis der Reihenfolge, in der die Vorgänge abgelaufen sind, sondern beweist zudem, dass die größte Menge an Computer-Rechenleistung dafür aufgebracht wurde. So lange die Mehrheit der Rechenleistung von Nodes kontrolliert wird, die nicht miteinander kooperieren, um das Netzwerk anzugreifen, werden diese die längste Kette generieren und Angreifer abhängen. Das Netzwerk selbst benötigt nur eine minimale Struktur. Nachrichten werden nach dem Best-Effort-Prinzip überbracht und Nodes können das Netzwerk beliebig verlassen und wieder beitreten, indem sie die längste Proof-of-Work-Kette als Beweis dessen akzeptieren, was in der Zeit ihrer Abwesenheit geschehen ist.

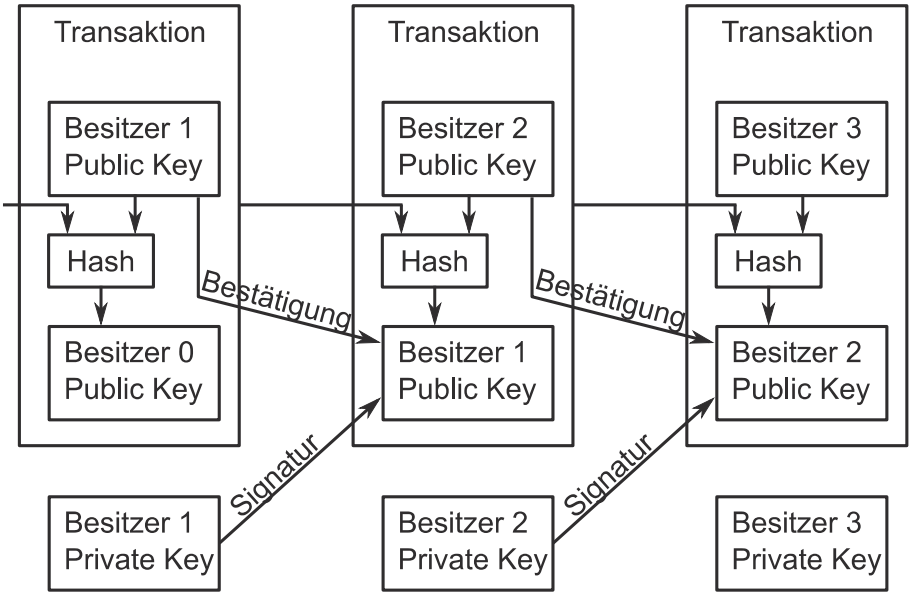
1. Einleitung

Der Internethandel verlässt sich bei der elektronischen Zahlungsabwicklung inzwischen beinahe ausschließlich auf Finanzinstitutionen als dritte Parteien und Mittler, denen die Transaktion anvertraut werden kann. Während dieses System im Falle der meisten Transaktionen ausreichend gut funktioniert, leidet es an den zentralen Schwächen, die mit der Notwendigkeit entsteht, anderen vertrauen zu müssen. Vollständig unumkehrbare Transaktionen sind nicht wirklich möglich, da Finanzinstitutionen es nicht vermeiden können, Meinungsverschiedenheiten schlichten zu müssen. Die anfallenden Schlichtungskosten erhöhen wiederum die Transaktionskosten, limitieren die kleinstmögliche Größe von Transaktionen und machen kleinere Transaktionen unmöglich. Zudem entstehen weitere Kosten im Verlust der Möglichkeit, unumkehrbare Zahlungen für unumkehrbare Leistungen zu erbringen. Wenn es möglich ist, Zahlungen zu widerrufen, steigt die Notwendigkeit von Vertrauen. Händler müssen sich ihrer Kunden bewusst sein und diese nach Informationen fragen, die sie ansonsten nicht benötigen würden. Ein Mindestmaß an Betrug ist zu erwarten und unvermeidbar. Diese Kosten und Zahlungsunsicherheiten können in persönlichem Kontakt dadurch vermieden werden, dass eine physische Währung verwendet wird, doch es gibt keinen Mechanismus, der Zahlungen über einen Kommunikationskanal ermöglicht, ohne dabei eine dritte Partei einzubeziehen, der vertraut werden kann.

Was wir brauchen ist ein elektronisches Zahlungssystem, das auf kryptographischen Beweisen basiert anstatt auf Vertrauen, und das es zwei beliebigen Parteien ermöglicht, direkt untereinander Transaktionen zu tätigen ohne einer dritten Partei vertrauen zu müssen. Transaktionen, die rechnerisch unmöglich zu widerrufen sind, würden Verkäufer vor Betrug schützen und routinierte Treuhandmechanismen zum Schutz von Käufern können leicht eingebaut werden. In diesem Whitepaper präsentieren wir eine Lösung für das Double-Spending-Problem, indem wir einen dezentralen, Peer-to-Peer Server mit Zeitstempeln nutzen um einen rechnerischen Beweis der chronologischen Ordnung von Transaktionen zu generieren. Das System ist so lange sicher, wie ehrliche Nodes im Kollektiv mehr Rechenleistung halten als irgendeine Gruppe miteinander kooperierender Angreifer.

2. Transaktionen

Wir definieren einen elektronischen Coin als eine Kette aus digitalen Signaturen. Ein Nutzer kann einem anderen Nutzer Coins transferieren, indem er einen Hash[1] der vorherigen Transaktion und den Public Key des Empfängers digital signiert. Ein Zahlungsempfänger kann diese Signatur und damit gleichzeitig auch die gesamte Kette des Besitzes verifizieren.



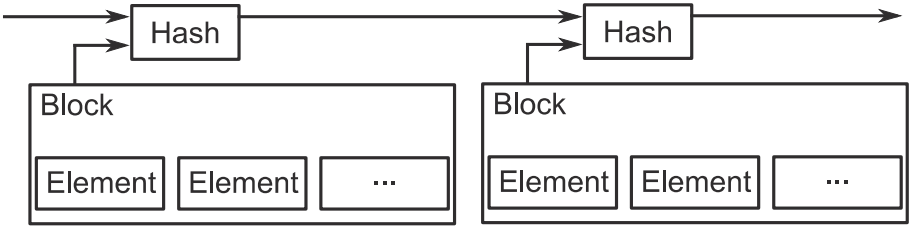
Problematisch an diesem Modell ist natürlich, dass der Zahlungsempfänger nicht verifizieren kann, dass einer der vorherigen Besitzer den Coin nicht doppelt ausgegeben hat. Eine übliche Lösung wäre das Einführen einer zentralen Autorität, etwa einer Prägeanstalt, der alle vertrauen und die alle Transaktionen auf Double-Spending überprüft. Nach jeder Transaktion muss der Coin zur Prägeanstalt zurückgeführt werden, die dann einen neuen Coin herausgibt. Nur Coins, die direkt von der Prägeanstalt kommen, kann vertraut werden. Das Problem dieser Lösung ist, dass das Schicksal des gesamten Geldsystems von der Firma abhängig ist, der die Prägeanstalt gehört, da durch diese, wie durch eine Bank, alle Transaktionen laufen.
Wir brauchen einen Weg, über den der Zahlungsempfänger wissen kann, dass der vorherige Besitzer keine früheren Transaktionen vorgenommen hat. Uns interessiert dabei die erste getätigte Transaktion – wir schauen nicht auf spätere Versuche, einen Coin zwei mal auszugeben. Der einzige Weg, die Abwesenheit einer vorherigen Transaktion auszuschließen, ist die Kenntnis aller getätigten Transaktionen. Im Modell der Prägeanstalt war diese sich stets aller Transaktionen bewusst und konnte entscheiden, welche davon als erstes den Empfänger erreichen sollte. Um dies auch ohne eine dritte Partei erreichen zu können, müssen Transaktionen zum einen öffentlich bekannt gegeben werden und zum anderen braucht es ein System aus Teilnehmern, die sich auf eine einzige Geschichte der Reihenfolge, in der diese angekommen sind, einigen können.

1 Ein Hash ist eine besondere Form der Prüfsumme. Prüfsummen sind Zeichenfolgen, die zur Überprüfung der Integrität von Daten genutzt werden. EAN-Codes oder ISBN-Nummern sind Alltagsbeispiele für Prüfsummen.

Der Zahlungsempfänger braucht den Beweis, dass die Mehrheit der Nodes sich zu der Zeit der jeweiligen Transaktion einig waren, dass er der Erste war, der diese erhalten hat.

3. Zeitstempel-Server

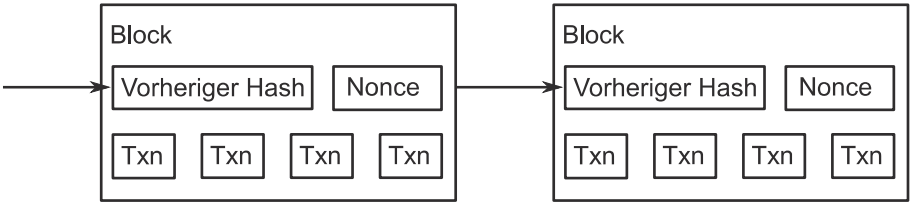
Unser Lösungsansatz beginnt mit einem Zeitstempel-Server. Ein Zeitstempel-Server arbeitet, indem er einen Hash einer gebündelten Menge von Transaktionen[2] mit einem Zeitstempel versieht und den Hash einer breiten Öffentlichkeit zugänglich macht, etwa durch eine Zeitung oder einen Usenet-Post [2-5]. Der Zeitstempel beweist, dass die Daten zu der Zeit existiert haben müssen, um in den Hash zu gelangen. Jeder Zeitstempel beinhaltet den vorherigen Zeitstempel in seinem Hash und bildet so eine Kette, in der jeder neu hinzugefügte Zeitstempel die ihm vorangegangenen bestätigt.



4. Proof-of-Work

Um einen dezentralen Zeitstempel-Server auf Peer-to-Peer-Basis zu implementieren, werden wir statt einer Zeitung oder einem Usenet-Post ein Proof-of-Work-System nutzen müssen, ähnlich Adam Back's Hashcash [6]. Ein Proof-of-Work beinhaltet das Scannen von Werten, deren Hash mit einer Anzahl an Zero Bits (einer bestimmten Menge Nullen) beginnt, wie bei SHA-256[3]. Die durchschnittliche benötigte Arbeit ist exponentiell zu der Anzahl der benötigten Zero Bits und kann durch die Ausführung eines einzigen Hashes verifiziert werden.

Für unser Zeitstempel-Netzwerk implementieren wir den Proof-of-Work indem wir so lange eine Nonce in den Block einsetzen, bis ein Wert gefunden wird, der dem Hash des Blocks die benötigten Zero Bits gibt. Sobald die Rechenleistung des Computers ausgeweitet wurde, um dem Proof-of-Work zu genügen, kann der Block nicht mehr geändert werden, ohne die gesamte Arbeit noch einmal zu leisten. Nachdem weitere Blöcke an die Kette angefügt wurden, würde das Ändern des Blockes bedeuten, die Arbeit aller anderen Blöcke wiederholen zu müssen.



Der Proof-of-Work löst zudem das Problem der Darstellung von Mehrheitsentscheidungen. Wenn die Mehrheit auf dem Prinzip Eine-IP-Eine-Stimme aufgebaut wäre, könnte sie recht einfach von jedem, der in der Lage ist, ausreichend IPs auf sich zu vereinen, eingenommen werden. Proof-of-

2 Diese Menge kann als Block bezeichnet werden.
3 Ein bestimmter Mechanismus zur Generierung von Prüfsummen

Work ist im Grunde ein Eine-Rechenleistung-Eine-Stimme-Prinzip. Die Mehrheitsentscheidung wird von der längsten Kette repräsentiert, die den größten Proof-of-Work-Aufwand in sich vereint. Wird die Mehrheit der Rechenleistung von ehrlichen Nodes kontrolliert, wächst die ehrliche Kette am schnellsten und wird alle konkurrierenden Ansprüche abhängen. Um einen vergangenen Block zu modifizieren, müsste ein Angreifer den Proof-of-Work dieses Blockes sowie aller darauffolgenden Blöcke erbringen und dann zu der Arbeit der ehrlichen Nodes auf- bzw. diese überholen. Wir werden später zeigen, warum die Wahrscheinlichkeit, dass ein langsamerer Angreifer aufholt bei jedem Anfügen eines neuen Blocks exponentiell schwindet.

Um die ansteigende Hardware-Geschwindigkeit und das über die Zeit schwankende Interesse an der Aufrechterhaltung der Nodes auszugleichen, ist die Schwierigkeit des Proof-of-Work darauf ausgelegt, eine festgelegte durchschnittliche Anzahl an Blöcken in einer Stunde zu generieren. Falls die Blöcke zu schnell generiert werden, steigt die Schwierigkeit an.

5. Netzwerk

Die Schritte, um das Netzwerk zu betreiben, sind wie folgend:

1) Neue Transaktionen werden an alle Nodes weitergegeben
2) Jeder Node sammelt neue Transaktionen in einem Block
3) Jeder Node arbeitet daran, einen schwierigen Proof-of-Work für seinen Block zu finden
4) Wenn ein Node einen Prof-of-Work findet, gibt er diesen an alle Nodes weiter
5) Nodes akzeptieren die Blöcke nur, wenn alle enthaltenen Transaktionen gültig sind und nicht bereits ausgegeben wurden
6) Nodes drücken ihre Akzeptanz gegenüber dem Block darin aus, dass sie daran arbeiten, den nächsten Block auf der Kette zu kreieren, indem sie den Hash des akzeptierten Block als vorherigen Hash nutzen.

Nodes sehen die längste Kette immer als die korrekte an und arbeiten daran, diese zu erweitern. Wenn zwei Nodes gleichzeitig verschiedene Versionen des nächsten Blocks verarbeiten, erhalten einige Nodes entweder die eine oder die andere zuerst. In diesem Fall arbeiten sie an dem ersten Block weiter, den sie erhalten, speichern den anderen Zweig jedoch ab, für den Fall, dass dieser der Längere wird. Das Unentschieden wird aufgehoben, wenn der nächste Proof-of-Work erbracht und einer der Zweige länger wird; die Nodes, die an dem anderen Zweig gearbeitet hatten, wechseln dann zu dem längeren.

Die Weitergabe neuer Transaktionen muss nicht notwendigerweise alle Nodes erreichen. So lange genügend Nodes erreicht werden, gehen sie über kurz oder lang in einen Block über. Die Weitergabe von Blöcken ist zudem tolerant gegenüber verloren gegangenen Nachrichten. Falls ein Node einen Block nicht erhält, wird er diesen anfordern, wenn er den nächsten Block erhält, da er bemerkt, dass ihm einer fehlt.

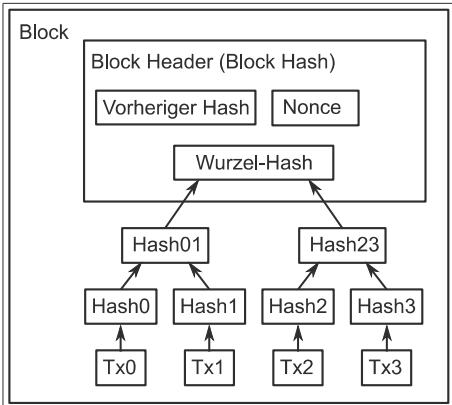
6. Anreiz

Laut Vereinbarung ist die erste Transaktion eines Blocks eine besondere Transaktion, die einen neuen Coin kreiert, der in den Besitz des Blockerzeugers übergeht. Auf diese Weise wird ein Anreiz für Nodes geschaffen, das Netzwerk zu unterstützen. So entsteht ein Weg, erstmalig Coins zu verteilen und in Zirkulation zu führen, da es keine zentrale Autorität gibt, die dies übernehmen könnte. Das fortlaufende Hinzufügen eines konstanten Betrags neuer Coins ist analog zum Schürfen von Gold (engl. Mining), bei dem ein Aufwand erbracht wird, um Gold in Umlauf zu bringen. In unserem Fall ist der Aufwand die Zeit und die Elektrizität, die durch die Rechenleistung in Anspruch genommen wird.

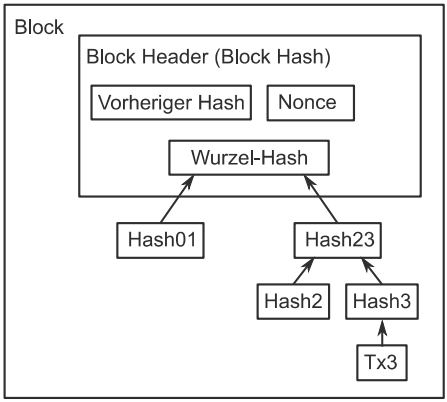
Der Anreiz kann auch durch Transaktionsgebühren finanziert werden. Wenn der Output-Wert einer Transaktion geringer ist als ihr Input-Wert, ist der Unterschied eine Transaktionsgebühr, die auf den Anreizwert des Blockes aufgeschlagen wird, der die Transaktion beinhaltet. Sobald eine vorher bestimmte Anzahl an Coins in die Zirkulation eingetreten sind, kann der Anreiz komplett zu Transaktionsgebühren schwenken und vollständig inflationsbefreit sein.
Der Anreiz kann dabei helfen, Die Nodes dazu zu bewegen, ehrlich zu bleiben. Wenn ein gieriger Angreifer in der Lage ist, mehr Rechenleistung anzusammeln als alle ehrlichen Nodes, müsste er wählen, ob er Menschen hintergehen und seine Zahlung zurückholen möchte oder ob er diese benutzt, um neue Coins zu generieren. Er sollte es profitabler finden, nach den Regeln zu spielen, da ihn diese Regeln stärker damit begünstigen, mehr Coins zu besitzen als alle Anderen zusammen, als damit, das System, und damit die Grundlage für seinen eigenen Reichtum, zu untergraben.

7. Zurückfordern von Festplatten-Kapazitäten

Sobald die jüngste Transaktion in einen Coin unter genügend Blöcken begraben ist, können die vergangenen Transaktionen verworfen werden, um Speicherkapazität auf der Festplatte zu sparen. Um dies umzusetzen ohne den Hash des Blocks zu zerstören, werden Transaktionen in einem Merkle-Tree gehashed [7][2][5], wobei nur die Wurzel im Hash des Blocks beinhaltet ist. Alte Blöcke können dann verdichtet werden, indem Zweige der Baumstruktur entfernt werden. Die einzelne Blätter des Merkle-Trees zusammenfassenden Hashes müssen nicht gespeichert werden.



Transaktionen eines Merkle-Trees

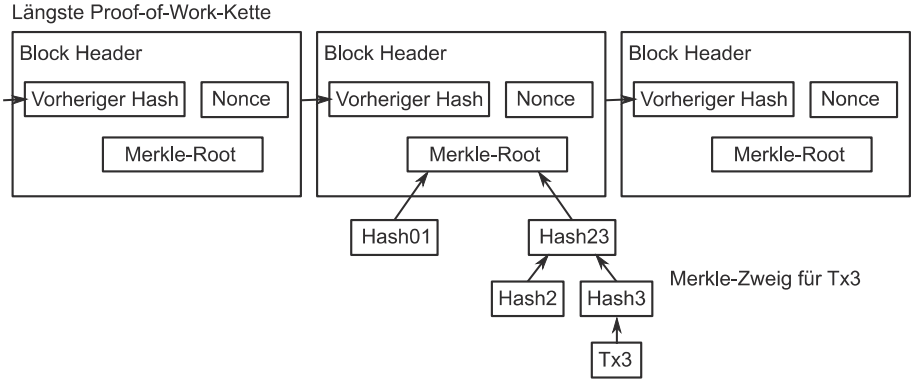


Reduzierter Baum durch Beschneidung von Tx0-2

Ein Dateikopf eines Blocks ohne Transaktionen würde etwa 80 Bytes groß sein. Wenn wir davon ausgehen, dass Blöcke alle 10 Minuten generiert werden, ergeben sich aus 80 Bytes * 6 * 24 * 365 = 4,2MB pro Jahr. Ausgehend von einem gewöhnlichen Computersystemen mit 2GB RAM Stand 2008 und Moore's Gesetz, das ein aktuelles Wachstum von 1,2GB pro Jahr voraussagt, dürfte die Lagerung kein Problem sein, selbst wenn die Dateiköpfe der Blöcke auf der Festplatte bleiben müssten.

8. Vereinfachte Zahlungs-Verifikation

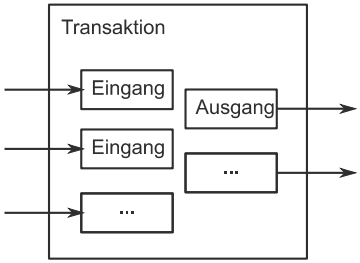
Es ist möglich, Zahlungen zu verifizieren, ohne einen kompletten Netzwerk-Node zu betreiben. Ein Nutzer benötigt lediglich die Kopie eines Dateikopfes der längsten Proof-of-Work-Kette. Diesen kann er erhalten, indem er Netzwerk-Nodes so lange befragt, bis er davon überzeugt ist, die längste Kette zu haben und den Merkle-Zweig erhält, der die Transaktion mit dem Block verbindet, in dem sie zeitgestempelt ist. Er kann die Transaktion nicht eigenständig nachverfolgen, aber dadurch, dass er sie mit einem Ort auf der Kette verbunden hat, kann er sehen, dass ein Netzwerk-Node sie akzeptiert hat. Das Anfügen weitere Blöcke bestätigt darüber hinaus, dass das Netzwerk die Transaktion bestätigt hat.



Die Verifizierung als solche ist verlässlich, solange ehrliche Nodes das Netzwerk kontrollieren – sie wird jedoch anfälliger, wenn das Netzwerk von einem Angreifer gekapert wird. Während Netzwerk-Nodes Transaktionen selbst verifizieren können, kann die vereinfachte Methode von den vorgetäuschten Transaktionen eines Angreifers hinters Licht geführt werden, solange dieser die Macht hat, das Netzwerk zu kontrollieren. Eine Strategie sich gegen ein solches Szenario zu schützen wäre es, Warnungen von Network-Nodes zu akzeptieren, die einen ungültigen Block bemerken und somit die Software des Benutzers dazu veranlassen, den gesamten Block herunterzuladen und die Inkonsistenz der verwarnten Transaktion zu bestätigen. Unternehmen, die regelmäßig Zahlungen erhalten, werden vermutlich nach wie vor ihre eigenen Nodes betreiben wollen, um unabhängige Sicherheit und schnelle Verifikationen zu gewährleisten.

9. Kombinieren und Aufteilen von Werten

Obwohl es möglich wäre, Coins individuell zu handhaben, wäre es unhandlich, für jeden Cent eines Transfers eine separate Transaktion vorzunehmen. Um es Werten zu erlauben, aufgeteilt oder kombiniert zu werden, beinhalten Transaktionen multiple Inputs und Outputs. Normalerweise gäbe es entweder einen einzelnen Input einer großen vorangegangenen Transaktion oder multiple Inputs, die kleinere Beträge zusammenfassen, sowie in der Regel zwei Outputs: einen für die Zahlung und einen zweiten, der die Veränderung, falls es eine gibt, an den Sender zurückberichtet.

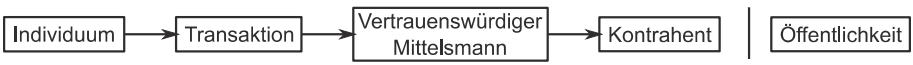


Es sollte festgehalten werden, dass ein Auffächern, indem eine Transaktion von verschiedenen Transaktionen abhängt, die wiederum von weiteren abhängen, hier kein Problem darstellt. Es ist niemals notwendig, eine vollständig alleinstehende Kopie einer Transaktionsgeschichte heranzuziehen.

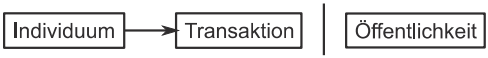
10. Privatsphäre

Das traditionelle Banking-Modell erreicht ein Niveau von Privatsphäre dadurch, dass es den Zugang zu Informationen auf die involvierten Parteien sowie die vertrauenswürdige dritte Partei beschränkt. Die Notwendigkeit, alle Transaktionen öffentlich bekannt zu geben, schließt diese Methode von vorneherein aus. Privatsphäre kann jedoch trotzdem gewährleistet werden, indem der Informationsfluss an einer anderen Stelle unterbrochen wird: Man hält die Public Keys anonym. Die Öffentlichkeit kann zwar sehen, dass jemand einen Betrag an jemand anderen verschickt, jedoch ohne dass die Transaktion dabei mit jemandem verbunden wird. Dies ähnelt dem Grad an Informationen, die an Börsen veröffentlicht werden: dort werden die Zeit und die Größe individueller Trades veröffentlicht, nicht jedoch die Identität der involvierten Parteien.

Privatsphäre im klassischen System



Privatsphäre mit Bitcoin



Als eine zusätzliche Schutzmauer sollte für jede Transaktion ein neues Paar von Private Keys genutzt werden, damit verschiedene Transaktionen nicht auf einen einzelnen Nutzer zurückverfolgt werden können. Ein Mindestmaß an Rückverfolgung ist nicht auszuschließen, da Multi-Input Transaktionen noch immer preisgeben, dass die Zahlungseingänge von dem selben Besitzer zu stammen. Das Risiko besteht, dass wenn der Besitzer eines Keys offenbart wird, dessen Verbindungen andere Transaktionen desselben Besitzers rückverfolgbar machen könnten.

11. Kalkulationen

Wir betrachten das Szenario eines Angreifers, der versucht eine alternative Kette schneller als die ehrliche Kette zu generieren. Selbst falls dies erreicht ist, liefert dies das System nicht willkürlichen Veränderungen aus, etwa dem Erschaffen von Werten aus der Luft heraus oder dem Nehmen von

Geld, das den Angreifern nie gehörte. Nodes werden eine ungültige Transaktion nicht als Bezahlung akzeptieren und ehrliche Nodes werden niemals einen Block akzeptieren, der diese beinhaltet. Ein Angreifer kann lediglich versuchen, eine seiner eigenen Transaktionen rückwirkend zu verändern und Geld zurückgewinnen, das er kurz zuvor ausgegeben hatte.

Der Wettlauf zwischen der ehrlichen Kette und der Version des Angreifers kann als eine binomische Zufallsbewegung betrachtet werden. Als Erfolgsereignis wird die Erweiterung der ehrlichen Kette um einen Block definiert, wodurch sich der Vorsprung um +1 erhöht. Die Niederlage ist konsequent die Verlängerung der Kette des Angreifers um einen Block, wodurch der Abstand zwischen ehrlichem Netzwerk und Angreifer um -1 sinkt.
Die Wahrscheinlichkeit, dass ein Angreifer einen Nachteil aufholt, kann man im Kontext der Spieltheorie als „Ruin des Spielers“ betrachten: Betrachten wir einen Glücksspieler mit unendlich viel Geld, jedoch einem bestimmten Nachteil. Er wird unendlich häufig spielen müssen, um diesen Nachteil aufzuholen. Die Wahrscheinlichkeit, dass er dies schafft – beziehungsweise im betrachteten Beispiel der Angreifer bezüglich der ehrlichen Kette einen Gleichstand erreicht – kann wie folgt definiert werden [8]:

p = Wahrscheinlichkeit, dass ein ehrlicher Node den nächsten Block findet
q = Wahrscheinlichkeit, dass der Angreifer diesen findet
q_z = Wahrscheinlichkeit, dass Angreifer z Blöcke aufholen kann

$$q_z = \begin{cases} 1, wenn\ p \leq q \\ (q/p)^z, wenn\ p > q \end{cases}$$

Unter der Annahme $p > q$ sinkt die Wahrscheinlichkeit exponentiell, da der Angreifer immer mehr Blöcke aufholen muss. Mit einer gegen ihn sprechenden Wahrscheinlichkeit wird die Wahrscheinlichkeit – von frühen Glückstreffern abgesehen – eine erfolgreiche Attacke immer unwahrscheinlicher, da er immer weiter zurückfällt.
Betrachten wir nun, wie lange ein Empfänger warten muss, bevor mit hinreichend großer Sicherheit eine Transaktion nicht mehr geändert werden kann. Sei also der Sender ein Angreifer, der den Empfänger für eine Weile hinsichtlich der Zahlung in Sicherheit wiegen möchte, später jedoch sich selbst das Geld zurückzahlen möchte. Zwar würde der Empfänger auf jeden Fall über eine derartige Änderung vom Netzwerk in Kenntnis gesetzt werden, doch hofft der Angreifer, dass es dann schon zu spät ist.
Der Empfänger generiert also ein neues Schlüsselpaar und gibt dem Sender kurz vor der Signierung der Transaktion den Public Key. Das macht der Empfänger, um dem Sender eine Präparation einer alternativen Blockchain zu erschweren – ansonsten könnte dieser einfach eine alternative Kette von Blöcken privat erzeugen und kurz nach der Transaktion diese neue in das Netzwerk einspeisen. Nach Senden der Transaktion arbeitet jedoch der betrügerische Sender privat an einer parallelen Version der Kette mit einer alternativen Version der Transaktion.
Der Empfänger wartet bis die Transaktion zu einem Block hinzugefügt wurde und z weitere Blöcke der Kette hinzugefügt wurden. Er weiß zwar nicht über den Fortschritt des Angreifers Bescheid, aber unter der Annahme, dass die ehrliche Kette für das Hinzufügen weiterer Blöcke jeweils eine festgelegte Durchschnittszeit benötigt, kann der potentielle Fortschritt des Angreifers in Form einer Poisson-Verteilung dargestellt werden:

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

Die Wahrscheinlichkeit, dass ein Angreifer nun noch aufholen kann, ergibt sich nach Multiplikation der Poisson-Dichte mit jedem Grad an Fortschritt, den er basierend auf oben angegebener Wahrscheinlichkeitsformel hätte erzielen können:

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

Das lässt sich wiederum in eine Reihe ohne unendliche Summe umformen:

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

In C würde das wie folgt implementiert werden können:

```
#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;
}
```

Wir sehen bei Ausführung des Codes, dass die Wahrscheinlichkeit exponentiell mit z abnimmt:

```
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

Für ein P kleiner als 0,1% erhalten wir also:

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. Schlussfolgerung

Wir haben ein System für elektronische Transaktionen vorgeschlagen, das sich nicht auf Vertrauen verlassen muss. Wir haben mit den üblichen Rahmenbedingungen für Coins angefangen, die aus digitalen Signaturen erzeugt werden. Diese bietet ein hohes Maß an Kontrolle über den Besitz, doch fehlt ein Weg, Double-Spending zu verhindern. Um dieses Problem zu lösen, haben wir ein Peer-to-Peer-Netzwerk mit der Nutzung des Proof-of-Work vorgeschlagen, um eine öffentliche Historie von Transaktionen aufzuzeichnen, die für Angreifer sehr bald rechnerisch unmöglich zu ändern ist, solange ehrliche Nodes die Mehrheit der Rechenleistung innehaben. Das Netzwerk ist in seiner unstrukturierten Einfachheit robust. Nodes arbeiten alle gleichzeitig mit minimaler Koordination. Sie müssen nicht identifiziert werden, da Nachrichten nicht an einen bestimmten Ort gerichtet sind und lediglich auf einer Best-Effort-Basis überbracht werden müssen. Nodes können das Netzwerk verlassen und ihm wieder beitreten, wenn sie die Proof-of-Work-Kette als Beweis dessen akzeptieren, was während der Zeit ihrer Abwesenheit passiert ist. Sie stimmen mit ihrer Rechenleistung ab und drücken ihre Akzeptanz gegenüber gültigen Blöcken dadurch aus, dass sie daran arbeiten, sie auszuweiten, während sie ungültige Blöcke dadurch ablehnen, dass sie sich weigern, mit ihnen zu arbeiten. Alle benötigten Regeln und Anreize können mit diesem Konsens-Mechanismus erreicht werden.

Referenzen

[1] W. Dai, "b-money," 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, May 1999.
[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no 2, pages 99-111, 1991.
[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.
[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 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, pages 122-133, April 1980. [8] W. Feller, "An introduction to probability theory and its applications," 1957.