RaiBlocks: Ein gebührenfreies verteiltes Kryptowährungsnetzwerk

Colin LeMahieu
clemahieu@gmail.com

***Übersicht*—In letzter Zeit haben ein hoher Bedarf und beschränkte Skalierbarkeit zu einer Erhöhung der durchschnittlichen Transaktionszeiten und Gebühren beliebter Krptowährungen geführt, was in einem unbefriedigenden Nutzungserlebnis resultierte. In diesem Whitepaper stellen wir RaiBlocks vor, eine Kryptowährung mit einer neuartigen Blockgitter-Architektur, in der jedes Konto über seine eigene Blockchain verfügt. Dadurch werden nahezu sofortige Transaktionen und unbeschränkte Skalierbarkeit ermöglicht. Jeder Nutzer verfügt über seine eigene Blockchain, die er asynchron zum restlichen Netzwerk aktualisieren kann, was zu schnellen Transaktionen mit minimalem Mehraufwand führt. Transaktionen überwachen anstelle von Transaktionsbeträgen das verbleibende Konto-Guthaben, was eine aggressive Datenbankreinigung ohne Abstriche bei der Sicherheit ermöglicht. Zum gegenwärtigen Zeitpunkt hat das RaiBlocks-Netzwerk bereits 4,2 Millionen Transaktionen verarbeitet, bei einer unbereinigten Ledgergröße von nur 1,7 GB. RaiBlocks gebührenfreie, sekundenschnelle Transaktionen machen es zur primären Kryptowährung für Verbrauchertransaktionen.**

***Index-Begriffe*—Kryptowährung, Blockchain, Raiblocks, Distributed-Ledger, Digital, Transaktionen**

I. Einführung

SEIT der Implementierung von Bitcoin im Jahre 2009, gab es einen zunehmenden Übergang von traditionellen, staatlich unterstützten Währungen und Finanzsystemen zu modernen, kryptografischen Zahlungsmethoden, die eine vertrauensfreie und sichere Aufbewahrung und Überweisung von Geldmitteln ermöglichen [1]. Um effektiv funktionieren zu können, muss eine Währung einfach zu überweisen und unumkehrbar sein, und darf über keine oder nur beschränkte Gebühren verfügen. Die erhöhten Transaktionszeiten, hohen Gebühren und die ungewisse Skalierbarkeit seines Netzwerks warfen Fragen zur Tauglichkeit von Bitcoin als eine Währung für den alltäglichen Gebrauch auf.

In diesem Papier stellen wir RaiBlocks vor, eine Niedriglatenz-Kryptowährung, die auf einer innovativen Blockgitter-Datenstruktur entwickelt wurde und sich durch uneingeschränkte Skalierbarkeit und Gebührenfreiheit auszeichnet. Per Design handelt es sich bei RaiBlocks um ein einfaches Protokoll mit dem alleinigen Zweck, als eine Hochleistungskryptowährung zu funktionieren. Das RaiBlocks-Protokoll kann auf Niedrigstrom-Hardware ausgeführt werden, was es zu einer praktischen, dezentralisierten Kryptowährung für den Alltag macht.

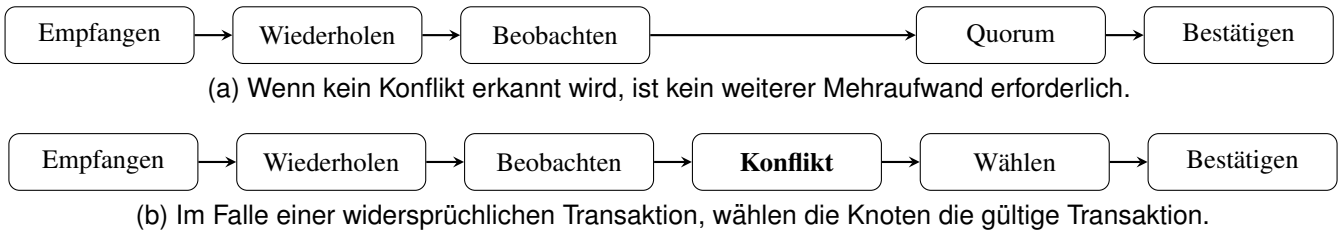
Die in diesem Papier zitierten Kryptowährungsstatistiken waren zum Veröffentlichungszeitpunkt wahrheitsgetreu.

II. Hintergrund

2008 veröffentlichte eine anonyme Person unter dem Pseudonym Satoshi Nakamoto ein Whitepaper, in dem die weltweit erste dezentralisierte Kryptowährung, Bitcoin, beschrieben wurde [1]. Eine der wichtigsten Innovationen, die mit Bitcoin eingeführt wurde, war die Blockchain – eine öffentliche, unveränderliche und dezentralisierte Datenstruktur, die als ein Ledger (Kontobuch) für die Transaktionen der Währung benutzt wird. Leider traten im Laufe der weiteren Entwicklung von Bitcoin diverse Probleme innerhalb des Protokolls auf, die Bitcoin für viele Anwendungszwecke unbrauchbar gemacht haben:

1) Schlechte Skalierbarkeit: Jeder Block in der Blockchain kann nur eine beschränkte Menge an Daten speichern, was bedeutet, dass das System nur eine bestimmte Anzahl an Transaktionen pro Sekunde verarbeiten kann, wodurch der Platz innerhalb eines Blocks zu einer Ware wird. Momentan beträgt die mediane Transaktionsgebühr $10,38 [2].
2) Hohe Latenz: Die durchschnittliche Bestätigungszeit beträgt 164 Minuten [3].
3) Ineffizienter Stromverbrauch: Das Bitcoin-Netzwerk verbraucht geschätzt 27,28 TWh pro Jahr, wobei im Durchschnitt 260 KWh pro Transaktion verbraucht werden [4].

Bitcoin, ebenso wie andere Kryptowährungen, funktioniert durch das Erreichen eines Konsens auf seinem globalen Ledger, um so legitime Transaktionen zu verifizieren und böswillige Akteure abzuwehren. Bitcoin erreicht den Konsens mit Hilfe einer ökonomischen Maßnahme namens Proof of Work (PoW). In einem PoW-System konkurrieren die Teilnehmer um die Berechnung einer Zahl namens *Nonce*, so dass der Hash eines gesamten Blocks sich im Zielspektrum befindet. Dieses gültige Spektrum ist umgekehrt proportional zur kumulativen Rechenkraft des gesamten Bitcoin-Netzwerks, um eine konsistente Durchschnittszeit zur Entdeckung einer gültigen Nonce zu gewährleisten. Der Finder einer gültigen Nonce darf anschließend den Block zur Blockchain hinzufügen; daher spielen Teilnehmer, die mehr Rechenkraft zur Berechnung einer Nonce aufwenden, eine größere Rolle für den Status der Blockchain. PoW bietet Widerstand gegen eine Sybil-Attacke, bei der eine Entität sich als mehrere Entitäten ausgibt, um in einem dezentralisierten System mehr Macht zu gewinnen, und reduziert auch signifikant die Race Conditions (Wettlaufsituationen), die beim Zugriff auf eine globale Datenstruktur inhärent auftreten.



(a) Wenn kein Konflikt erkannt wird, ist kein weiterer Mehraufwand erforderlich.

(b) Im Falle einer widersprüchlichen Transaktion, wählen die Knoten die gültige Transaktion.

Abb. 1. RaiBlocks erfordert für typische Transaktionen keinen zusätzlichen Mehraufwand. Bei widersprüchlichen Transaktionen, müssen die Knoten die Transaktion wählen, die beibehalten werden soll.

Ein alternatives Konsensprotokoll, Proof of Stake (PoS), wurde 2012 durch Peercoin eingeführt [5]. In einem PoS-System stimmen die Teilnehmer mit dem Gewichtsäquivalent der Geldmenge ab, die sie in einer bestimmten Kryptowährung besitzen. Bei diesem Arrangement erhalten Teilnehmer mit einer größeren finanziellen Investition mehr Macht und verfügen über einen inhärenten Anreiz, die Ehrlichkeit des Systems aufrecht zu erhalten, um nicht dem Risiko ausgesetzt zu sein, ihre Investition zu verlieren. PoS beseitigt den verschwenderischen Konkurrenzkampf um Rechenleistung und erfordert nur leichtgewichtige Software, die auf Niedrigstrom-Hardware betrieben werden kann.

Das ursprüngliche RaiBlocks-Papier und die erste Beta-Implementierung wurden im Dezember 2014 veröffentlicht, was RaiBlocks zu einer der ersten Kryptowährungen macht, die auf einem Directed Acyclic Graph (DAG) basieren [6]. Kurz darauf begann die Entwicklung anderer DAG-Kryptowährungen, allen voran DagCoin/Byteball und IOTA [7], [8]. Diese DAG-basierten Kryptowährungen brachen mit der Blockchain-Form und verbesserten die Systemleistung und Sicherheit. Byteball erreicht einen Konsens, indem es sich auf eine "Hauptkette" verlässt, die aus ehrlichen, angesehenen und von Nutzern vertrauten "Zeugen" besteht, während IOTA seinen Konsens über ein kumulatives PoW gestapelter Transaktionen etabliert. RaiBlocks erreicht einen Konsens durch Guthaben-gewichtete Abstimmungen zu widersprüchlichen Transaktionen. Dieses Konsenssystem bietet schnellere, deterministischere Transaktionen, während es nach wie vor ein starkes, dezentralisiertes System gewährleistet. RaiBlocks setzt diese Entwicklung weiter fort und hat sich als eine der leistungsstärksten Kryptowährungen positioniert.

III. RaiBlocks-Komponenten

Bevor wir die gesamte RaiBlocks-Architektur beschreiben, definieren wir zunächst die individuellen Komponenten, aus denen das System sich zusammensetzt.

A. Konto

Ein Konto ist die Public-Key-Hälfte eines digitalen Signaturschlüsselpaars. Der Public-Key, auch bekannt als Adresse, wird mit anderen Netzwerk-Teilnehmern geteilt, während der Private-Key geheim gehalten wird. Ein digital signiertes Datenpaket garantiert, dass die Inhalte vom Inhaber des Private-Keys genehmigt wurden. Ein Nutzer kann viele Konten besitzen, aber es kann nur eine öffentliche Adresse pro Konto existieren.

B. Block/Transaktion

Die Begriffe "Block" und "Transaktion" werden oft austauschbar verwendet, wobei ein Block eine einzelne Transaktion beinhaltet. "Transaktion" bezieht sich speziell auf die Handlung, während "Block" die digitale Verschlüsselung der Transaktion bezeichnet. Transaktionen werden mit dem Private-Key signiert, der zum Konto gehört, auf dem die Transaktion durchgeführt wird.

C. Ledger

Der Ledger ist das globale Kontenbuch, worin jedes Konto über eine eigene Transaktionskette verfügt (Abbildung 2). Dies ist eine zentrale Designkomponente, die den Ersatz einer Laufzeit-Vereinbarung mit einer Designzeit-Vereinbarung darstellt; alle Teilnehmer stimmen über die Signaturprüfung zu, dass nur der Kontoinhaber seine eigene Kette modifizieren kann. Dadurch wird eine scheinbar geteilte Datenstruktur, ein Distributed-Ledger, in eine Reihe von ungeteilten Datenstrukturen konvertiert.

D. Knoten

Ein *Knoten* ist eine Software, die auf einem Computer ausgeführt wird, dem RaiBlocks-Protokoll entspricht und sich am RaiBlocks-Netzwerk beteiligt. Die Software verwaltet den Ledger und alle Konten, die der Knoten kontrolliert (falls vorhanden). Ein Knoten kann entweder den gesamten Ledger

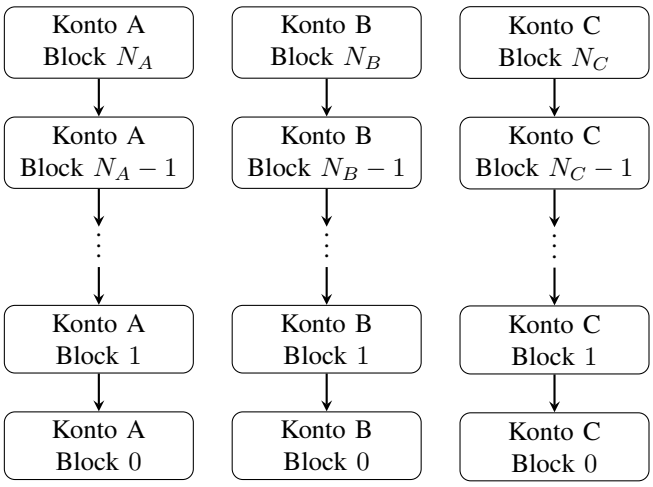


Abb. 2. Jedes Konto verfügt über seine eigene Blockchain, die den Guthaben-Verlauf des Kontos enthält. Block 0 muss eine offene Transaktion sein (Section IV-B)

oder einen gereinigten Verlauf speichern, der nur die letzten Blocks der Blockchain jedes Kontos enthält. Bei der Einrichtung eines neuen Knotens wird empfohlen, den gesamten Verlauf zu verifizieren und lokal zu reinigen.

IV. Systemübersicht

Im Gegensatz zu den Blockchains vieler anderer Kryptowährungen, verwendet RaiBlocks eine *Blockgitter*-Struktur. Jedes Konto verfügt über seine eigene Blockchain (Kontokette), die dem Transaktions-/Guthabenverlauf des Kontos entspricht (Abbildung 2). Jede Kontokette kann nur vom Konto-Inhaber aktualisiert werden; dadurch lässt sich jede Kontokette sofort und asynchron zum Rest des Blockgitters aktualisieren, was schnelle Transaktionen ermöglicht. Das RaiBlocks-Protokoll ist extrem leichtgewichtig; jede Transaktion passt in die minimale, für die Internet-Übertragung erforderliche UDP-Paketgröße. Die Hardware-Anforderungen für Knoten sind ebenfalls minimal, da Knoten für die meisten Transaktionen lediglich Blocks aufzeichnen und erneut senden müssen (Abbildung 1).

Das System wird mit einem *Genesis-Konto* initiiert, in dem das *Genesis-Guthaben* enthalten ist. Das Genesis-Guthaben ist ein Fixbetrag und kann niemals erhöht werden. Das Genesis-Guthaben wird aufgeteilt und über die Sende-Transaktionen, die auf der Genesis-Kontokette registriert sind, an andere Konten gesendet. Die Guthaben-Summe auf allen Konten wird niemals das anfängliche Genesis-Guthaben überschreiten, wodurch das System eine Quantitätsschranke erhält, die nicht erhöht werden kann.

Dieser Abschnitt behandelt, wie unterschiedliche Transaktionsarten konstruiert und innerhalb des Netzwerks propagiert werden.

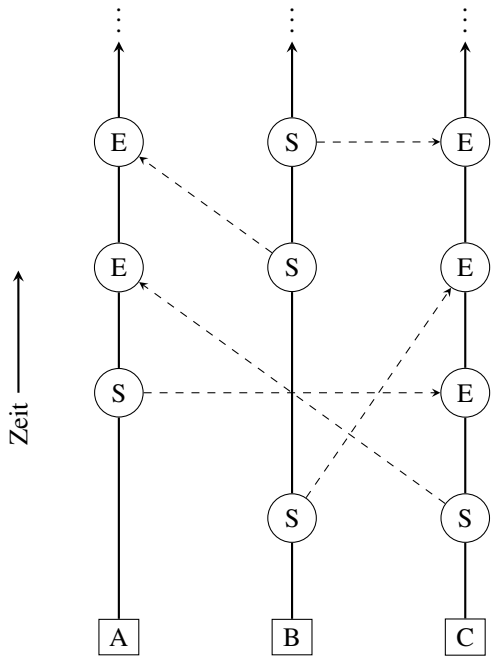


Abb. 3. Visualisierung des Blockgitters. Jede Überweisung erfordert einen Sendeblock (S) und einen Empfangsblock (E), von denen jeder vom jeweiligen Eigentümer der Kontokette signiert werden muss (A,B,C).

A. Transaktionen

Für Überweisungen von einem Konto an ein anderes Konto werden zwei Transaktionen benötigt: eine *Sendetransaktion*, die den entsprechenden Betrag vom Guthaben des Senders abzieht, und eine *Empfangstransaktion*, die den Betrag zum Guthaben des Empfängerkontos addiert (Figure 3).

Die Überweisung von Beträgen als separate Transaktionen in den Konten des Senders und Empfängers erfüllt einige wichtige Funktionen:

1) Sequenzierung eingehender Überweisungen, die inhärent asynchron sind.
2) Kleinhaltung von Transaktionen, damit diese in UDP-Pakete passen.
3) Erleichterung der Ledgerreinigung durch Minimierung des Datenfußabdrucks.
4) Isolierung abgeschlossener und ausstehender Transaktionen.

Wenn von mehr als einem Konto an die selbe Zieladresse überwiesen wird, handelt es sich dabei um eine asynchrone Operation; Netzwerklatenz und die Tatsache, dass die Sendekonten nicht zwangsläufig mit einander kommunizieren, haben zur Folge, dass es keine universal akzeptable Methode gibt, zu wissen, welche Transaktion zuerst stattgefunden hat. Da die Addition assoziativ ist, spielt die Reihenfolge der Eingaben keine Rolle, weshalb lediglich eine globale Übereinstimmung benötigt wird. Dies ist eine zentrale Designkomponente, die eine Laufzeit-Vereinbarung in eine Designzeit-Vereinbarung konvertiert. Das Empfängerkonto hat die Kontrolle darüber, zu entscheiden, welche Überweisung zuerst ankam. Dies wird durch die signierte Reihenfolge der eingehenden Blöcke ausgedrückt.

Falls ein Konto eine große Überweisung senden möchte, die als eine Reihe kleinerer Überweisungen ankommt, wollen wir dies auf eine Weise ausdrücken, die in ein UDP-Paket hineinpasst. Wenn ein Empfängerkonto Eingabeüberweisungen in einer Reihenfolge anordnet, behält es eine laufende Summe seines Kontostandes bei, so dass es jederzeit einen Betrag mit einer fixen Transaktionsgröße überweisen kann. Dies unterscheidet sich vom Eingabe-/Ausgabe-Transaktionsmodell, das von Bitcoin und anderen Kryptowärungen verwendet wird.

Einige Knoten sind nicht daran interessiert, Ressourcen zu verbrauchen, um den vollen Transaktionsverlauf eines Kontos zu speichern; sie interessieren sich nur für das gegenwärtige Guthaben jedes Kontos. Wenn ein Konto eine Transaktion durchführt, verschlüsselt es sein kumuliertes Guthaben, so dass diese Knoten lediglich den letzten Block überwachen müssen. Dadurch können sie historische Daten verwerfen und trotzdem korrekt bleiben.

Selbst mit dem Schwerpunkt auf Designzeit-Vereinbarungen, gibt es bei der Validierung von Transaktionen ein Verzögerungsfenster, um böswillige Akteure im Netzwerk zu identifizieren und zu bekämpfen. Da Vereinbarungen in RaiBlocks schnell zustande kommen (innerhalb von Millisekunden bis Sekunden), können wir dem Nutzer zwei vertraute Kategorien eingehender Transaktionen präsentieren: Abgeschlossen und ausstehend. Transaktionen sind abgeschlossen, wenn ein Konto bereits

Empfangsblöcke generiert hat. Transaktionen sind ausstehend, wenn sie noch nicht zum kumulativen Guthaben des Empfängers addiert wurden. Dies ist ein Ersatz für die komplexeren und unvertrauteren Bestätigungsmetriken anderer Kryptowährungen.

B. Ein Konto erstellen

Um ein Konto zu erstellen, muss eine *offene* Transaktion erstellt werden (Abbildung 4). Eine offene Transaktion ist immer die erste Transaktion jeder Kontokette und kann beim ersten Empfang einer Einzahlung erstellt werden. Das Feld *account* speichert den Public-Key (die Adresse), der aus dem Private-Key generiert wird, welcher zur Signierung dient. Das Feld *source* enthält den Hash der Transaktion, über welche die Zahlung gesendet wurde. Bei der Erstellung eines Kontos muss ein Vertreter gewählt werden, der im Namen dieses Kontos abstimmen soll; dies kann nachträglich geändert werden (Abschnitt IV-F). Das Konto kann sich selbst als seinen eigenen Vertreter erklären.

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

Abb. 4. Anatomie einer offenen Transaktion

C. Konto-Guthaben

Das Konto-Guthaben wird im Ledger selbst aufgezeichnet. Anstatt den Betrag einer Transaktion aufzuzeichnen, erfordert die Verifizierung (Abschnitt IV-I) eine Überprüfung der Differenz zwischen dem Guthaben des Sendeblocks und dem Guthaben des vorangegangenen Blocks. Das Empfängerkonto kann anschließend das vorherige Guthaben in Übereinstimmung mit dem errechneten Endguthaben erhöhen, das im neuen Empfangsblock enthalten ist. Dies dient zur Erhöhung der Geschwindigkeit beim Herunterladen von hohen Blockvolumen. Bei der Abfrage des Kontoverlaufs sind die Beträge bereits enthalten.

D. Von einem Konto senden

Um von einer Adresse senden zu können, muss die Adresse bereits über einen bestehenden offenen Block und ergo über ein Guthaben verfügen (Abbildung 5). Das Feld *previous* enthält den Hash des vorherigen Blocks in der Kontokette. Das Feld *destination* enthält das Konto, an das Geld überwiesen werden soll. Ein Sendeblock kann nachdem er bestätigt wurde nicht verändert werden. Sobald es an das Netzwerk übertragen wird, wird das Geld sofort vom Guthaben des Sendekontos abgezogen und als *pending* (ausstehend) behandelt werden, bis der Empfänger einen Block signiert, um dieses Geld anzunehmen. Ausstehende Transaktionen warten nicht auf Bestätigungen; sie sind auf dem Konto des Senders praktisch verbraucht und der Sender kann die Transaktion nicht widerrufen.

```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

Abb. 5. Anatomie einer Sendetransaktion

E. Eine Transaktion erhalten

Um eine Transaktion abzuschließen, muss der Empfänger des überwiesenen Geldes einen Empfangsblock auf seiner eigenen Kontokette erstellen (Abbildung 6). Das Feld “Quelle” bezieht sich auf den Hash der entsprechenden Sendetransaktion. Sobald dieser Block erstellt und übertragen wird, wird das Guthaben des Kontos aktualisiert und das überwiesene Geld offiziell diesem Konto gutgeschrieben.

```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

Abb. 6. Anatomie einer Empfangstransaktion

F. Einen Vertreter einsetzen

Die Fähigkeit von Kontoinhabern, einen Vertreter zu wählen, der in ihrem Namen wählen kann, ist ein leistungsstarkes Dezentralisierungswerkzeug, das über kein ebenbürtiges Analog in den Proof of Work und Proof of Stake Protokollen verfügt. In konventionellen PoS-Systemen, muss der Knoten des Kontoinhabers aktiv sein, um wählen zu können. Das kontinuierliche Betreiben eines Knotens ist für viele Nutzer unpraktisch; die Übertragung des Stimmrechts an einen Vertreter lockert diese Voraussetzung. Kontoinhaber verfügen über die Fähigkeit, den Konsens jederzeit einem anderen Konto zuzuweisen. Eine *Wechseln*-Transaktion wechselt den Vertreter eines Kontos, indem das Stimmgewicht vom alten Vertreter abgezogen und dem neuen Vertreter zugewiesen wird (Abbildung 7). Bei dieser Transaktion wird kein Geld übertragen und der Vertreter verfügt nicht über das Recht, das Guthaben des Kontos auszugeben.

```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

Abb. 7. Anatomie einer Wechseln-Transaktion

G. Forks und Abstimmungen

Ein Fork ereignet sich, wenn j signierte Blöcke $b_1, b_2, \ldots, b_j$ den selben Block als ihren Vorgänger ansehen (Abbildung 8). Diese Blöcke erzeugen eine widersprüchliche Ansicht zum Status eines Kontos, die gelöst werden muss. Nur der Inhaber eines Kontos ist in der Lage, Blöcke in ihre Kontokette zu signieren, daher muss ein Fork zwangsläufig aus schlechter Programmierung oder der böswilligen Absicht (doppelte Ausgabe) des Kontoinhabers entstehen.

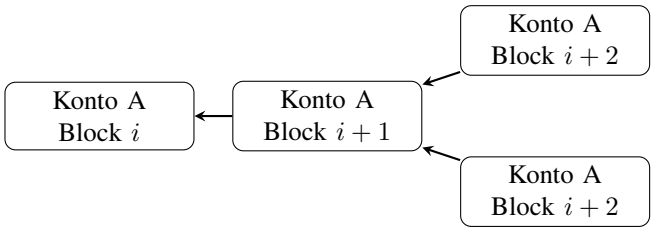


Abb. 8. Ein Fork entsteht, wenn zwei (oder mehr) signierte Blöcke den selben vorangegangenen Block referenzieren. Ältere Blöcke sind links; neuere Blöcke sind rechts.

Nach der Erkennung eines Forks, erstellt ein Vertreter in seinem Ledger eine Abstimmung zum Block $\hat{b}_i$ und sendet diese an das Netzwerk. Das Stimmgewicht eines Knotens, w_i, ist die Summe der Guthaben aller Konten, die ihn als Vertreter ernannt haben. Der Knoten beobachtet die eingehenden Stimmen der anderen M Online-Vertreter und führt eine kumulative Aufzählung für 4 Abstimmungszeiträume durch (Gesamtdauer von 1 Minute), anschließend bestätigt er den Block, der gewonnen hat (Gleichung 1).

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

Der beliebteste Block b^* wird die meisten Stimmen erhalten und im Ledger des Knotens gespeichert werden (Gleichung 2). Die Blöcke, die die Abstimmung verlieren, werden entsorgt. Falls ein Vertreter einen Block in seinem Ledger ersetzt, erstellt er eine neue Abstimmung mit einer höheren Sequenznummer und sendet die neue Abstimmung an das Netzwerk. Dies ist das **einzige** Szenario, bei dem Vertreter abstimmen.

Unter einigen Umständen können kurze Netzwerk-Verbindungsstörungen dazu führen, dass ein gesendeter Block nicht von allen Peers akzeptiert wird. Alle folgenden Blöcke auf diesem Konto werden von Peers, die die ursprüngliche Übertragung nicht gesehen haben, als ungültig angesehen und ignoriert. Eine erneute Übertragung dieses Blocks wird von den verbleibenden Peers akzeptiert werden und alle nachfolgenden Blöcke werden automatisch abgerufen. Selbst wenn ein Fork oder fehlender Block sich ereignet, sind davon nur die Konten betroffen, die in der entsprechenden Transaktion referenziert werden; der Rest des Netzwerks verarbeitet weiterhin die Transaktionen aller anderen Konten.

H. Proof of Work

Alle vier Transaktionsarten haben einen Arbeitsbereich, der korrekt ausgefüllt werden muss. Der Arbeitsbereich ermöglicht es dem Transaktionsersteller, eine Nonce zu berechnen, so dass der Hash der Nonce, die mit dem vorherigen Bereich in der Empfangen-/Senden-/Wechseln-Transaktion oder dem Kontobereich in einer offenen Transaktion verkettet ist, einen bestimmten Schwellwert nicht übersteigt. Im Gegensatz zu Bitcoin wird das RaiBlocks-PoW lediglich als ein Anti-Spam-Werkzeug benutzt (ähnlich wie bei Hashcash) und kann innerhalb von Sekunden berechnet werden [9]. Sobald eine Transaktion gesendet wird, kann das PoW für den folgenden Block vorberechnet werden, da der vorangegangene Blockbereich bekannt ist; auf diese Weise erscheinen Transaktionen für den Endnutzer sofort, so lange die Zeit zwischen Transaktionen größer ist, als die Zeit, die zur Berechnung des PoW benötigt wird.

I. Transaktionsverifizierung

Damit ein Block als gültig betrachtet wird, muss er über die folgenden Eigenschaften verfügen:

1) Der Block darf noch nicht im Ledger sein (doppelte Transaktion).
2) Der Block muss vom Kontoinhaber signiert sein.
3) Der vorherige Block ist der Kopfblock der Kontokette. Falls der Block existiert, aber nicht der Kopfblock ist, handelt es sich um einen Fork.
4) Das Konto muss über einen offenen Block verfügen.
5) Der berechnete Hash erfüllt die PoW-Schwellwertbedingung.

Falls es sich um einen Empfangsblock handelt, muss geprüft werden, ob der Quellblock-Hash ausstehend ist, was bedeutet, dass er noch nicht eingelöst wurde. Falls es sich um einen Sendeblock handelt, muss das Guthaben größer sein als das vorherige Guthaben.

V. Angriffsflächen

So wie andere dezentralisierte Kryptowährungen, ist auch RaiBlocks dem Risiko eines Angriffs durch böswillige Parteien ausgesetzt, die sich finanziellen Gewinn erhoffen oder das System zerstören möchten. In diesem Bereich schildern wir einige mögliche Angriffsszenarien, die Konsequenzen solcher Angriffe, und die Präventivmaßnahmen des RaiBlocks-Prokolls.

A. Blocklücken-Synchronisierung

Im Abschnitt IV-G, haben wir ein Szenario diskutiert, in dem ein Block nicht ordnungsgemäß übertragen wird, was das Netzwerk dazu veranlasst, nachfolgende Blöcke zu ignorieren. Falls ein Knoten einen Block erkennt, der nicht über den referenzierten vorherigen Block verfügt, hat der Knoten zwei Optionen:

1) Den Block ignorieren, da er ein schädlicher Abfallblock sein könnte.
2) Resynchronisierung mit einem anderen Knoten beantragen.

Im Falle einer Resynchronisierung, muss eine TCP-Verbindung mit einem Bootstrapping-Knoten hergestellt werden, um den erhöhten Datenverkehr abzudecken, der mit einer Resynchronisierung einhergeht. Falls es sich bei dem Block allerdings um einen böswilligen Block gehandelt hat, war die Resynchronisierung nicht erforderlich und hat unnötig den Datenverkehr im Netzwerk erhöht. Hierbei handelt es sich um einen Netzwerk-Verstärkungsangriff, welcher zu einem Denial of Service führt.

Um eine unnötige Resynchronisierung zu vermeiden, warten Knoten ab, bis ein bestimmter Schwellwert an Stimmen für einen möglicherweise böswilligen Block erreicht wird, bevor sie eine Verbindung mit einem Bootstrap-Knoten herstellen und die Synchronisierung einleiten. Falls ein Block nicht genug Stimmen erhält, ist anzunehmen, dass es sich dabei um Junk-Daten handelt.

B. Transaktionsüberflutung

Ein böswilliger Akteur könnte viele unnötige aber dennoch gültige Transaktionen zwischen von ihm kontrollierten Konten veranlassen, um zu versuchen, so das Netzwerk zu überlasten. Ohne Transaktionsgebühren kann dieser Angriff endlos fortgesetzt werden. Allerdings beschränkt das für jede Transaktion erforderliche PoW die Transaktionsrate, die von der böswilligen Partei generiert werden kann, ohne signifikant in Rechenleistung zu investieren. Selbst unter einem solchen Angriff, bei dem versucht wird, den Ledger zu überlasten, können Knoten, welche nicht den vollständigen Verlauf speichern, alte Transaktionen einfach aus ihrer Kette reinigen; dadurch wird die Speicherauslastung durch einen solchen Angriff für nahezu alle Nutzer eingeschränkt.

C. Sybil-Attacke

Eine Partei könnte Hunderte von RaiBlocks-Knoten auf einem einzigen Gerät erstellen; da das Abstimmungssystem allerdings anhand des Konto-Guthabens gewichtet wird, erhält der Angreifer durch zusätzliche Knoten in einem Netzwerk keine zusätzlichen Stimmen. Aus diesem Grund ist es nicht möglich, sich durch eine Sybil-Attacke einen Vorteil zu verschaffen.

D. Penny-Spend-Attacke

Bei einer Penny-Spend-Attacke lädt ein Angreifer die Guthaben einer großen Anzahl von Konten mit Kleinstbeträgen auf, um die Speicherressourcen der Knoten zu verschwenden. Die Blockveröffentlichungsrate ist durch das PoW beschränkt, daher ist die Erstellung von Konten und Transaktionen zu einem gewissen Maß eingeschränkt. Knoten, die nicht den vollen Verlauf speichern, können Konten unterhalb einer statistischen Metrik reinigen, unter der ein Konto aller Wahrscheinlichkeit nach unauthentisch ist. Darüber hinaus ist RaiBlocks darauf ausgerichtet, minimalen Speicherplatz zu verbrauchen, daher ist der Speicher, der für ein zusätzliches Konto benötigt wird, proportional zur Größe eines offenen Blocks + Indexierung $= 96\text{B} + 32\text{B} = 128\text{B}$. Das bedeutet, dass in 1GB Speicher 8 Millionen Penny-Spend-Kontos gespeichert werden können. Falls Knoten eine aggressivere Reinigung vornehmen wollen, können sie eine auf Zugriffshäufigkeit basierte Verteilung berechnen und selten genutzte Konten in den langsameren Speicher delegieren.

E. Vorberechnete PoW-Attacke

Da der Inhaber eines Kontos die einzige Entität ist, die Blöcke zur Kontokette hinzufügen kann, ist es möglich, sequentielle Blöcke und ihr PoW zu berechnen, bevor sie an das Netzwerk übertragen werden. Hierbei erstellt der Angreifer über einen längeren Zeitraum eine Unmenge an sequentiellen Blöcken, von denen jeder von minimalem Wert ist. Ab einem bestimmten Punkt führt der Angreifer einen Denial of Service (DoS) durch, indem er das Netzwerk mit vielen gültigen Transaktionen überflutet, die von anderen Knoten so schnell wie möglich verarbeitet und wiederholt werden. Dies ist eine fortgeschrittene Version der in Abschnitt V-B beschriebenen Transaktionsüberflutung. Ein solcher Angriff würde nur kurzzeitig funktionieren, könnte aber in Kombination mit anderen Angriffen wie der >50% Attacke (Abschnitt V-F) verwendet werden, um die Effektivität zu erhöhen. Momentan werden Transaktionsratenbeschränkung und andere Techniken erwogen, um diese Angriffe abzuschwächen.

F. >50% Attacke

Die Konsensmetrik für RaiBlocks ist ein Guthaben-basiertes Abstimmungssystem. Falls ein Angreifer mehr als 50% der Stimmstärke erreichen kann, ist er in der Lage, einen Konsensumschwung innerhalb des Netzwerks zu bewirken und so das System zu brechen. Ein Angreifer kann das dafür aufzuwendende Guthaben verringern, indem die Beteiligung guter Knoten an der Abstimmung durch ein Netzwerk-DoS verhindert wird. RaiBlocks unternimmt die folgenden Maßnahmen, um einen solchen Angriff zu verhindern:

1) Die primäre Verteidigung gegen diese Angriffsart, ist die Tatsache, dass Stimmgewicht mit Investitionen in das System verbunden ist. Ein Konto-Inhaber verfügt über einen inhärenten Anreiz, die Ehrlichkeit des Systems zu wahren, um seine Investition zu schützen. Ein Versuch, den Ledger zu manipulieren, wäre zerstörerisch für das gesamte System und würde die Investition des Angreifers zerstören.
2) Die Kosten dieses Angriffs sind proportional zur Marktkapitalisierung von RaiBlocks. In PoW-Systemen können Technologien entwickelt werden, die disproportionale Kontrolle im Vergleich zum investierten Geld

zulassen, und im Falle eines erfolgreichen Angriffs, kann diese Technologie zweckentfremdet werden. Bei RaiBlocks skalieren die Kosten eines Angriffs auf das System mit dem System selbst, und falls ein erfolgreicher Angriff gelingen sollte, kann die Investition in den Angriff nicht zurückgewonnen werden.

3) Um ein maximales Quorum von Wählern zu gewährleisten, ist die nächste Verteidigungslinie ein repräsentatives Wahlsystem. Konto-Inhaber, die sich aufgrund von Verbindungseinschränkungen nicht regelmäßig an Abstimmungen beteiligen können, können einen Vertreter ernennen, der mit dem Gewicht ihrer Stimmen wählen kann. Die Maximierung der Anzahl und Vielfalt an Vertretern erhöht die Widerstandsfähigkeit des Netzwerks.
4) RaiBlocks-Forks passieren niemals zufällig, daher können Knoten sich für Richtlinien für den Umgang mit geforkten Blöcken entscheiden. Die einzige Situation, in der Nicht-Angreifer-Konten anfällig für Blockforks sind, ist, wenn sie ein Guthaben von einem Angreifer-Konto empfangen. Konten, die vor Blockforks geschützt sein wollen, können eine kurze oder lange Zeit abwarten, bevor sie eine Überweisung von einem Konto empfangen, das Forks generiert hat, oder diese prinzipiell ablehnen. Empfänger haben auch die Möglichkeit, separate Konten für den Empfang von Überweisungen zu erstellen, die von dubiosen Konten ausgehen, um so ihre anderen Konten zu isolieren.
5) Eine letzte Verteidigungslinie, die noch nicht implementiert wurde, ist *Blockzementierung*. RaiBlocks unternimmt die größtmöglichen Bemühungen, um geforkte Blocks schnell über Abstimmungen zu lösen. Knoten könnten konfiguriert werden, um Blöcke zu zementieren, was verhindern könnte, dass diese nach einer bestimmten Zeit zurückgezogen werden. Das Netzwerk ist durch die Fokussierung auf eine schnelle Abschlusszeit ausreichend geschützt, um mehrdeutige Forks zu vermeiden.

Eine intelligentere Version des $> 50\%$ Angriffs ist in Abbildung 9 geschildert. “Offline” bezeichnet den Prozentwert der Vertreter, die ernannt wurden, aber nicht online sind, um an Abstimmungen teilzunehmen. “Anteil” bezeichnet den Investitionsbetrag, mit dem der Angreifer abstimmt. “Aktiv” bezeichnet die Vertreter, die online sind und in Übereinstimmung mit dem Protokoll wählen. Ein Angreifer kann den benötigten Anteilsbetrag mindern, indem er andere Wähler über eine Netzwerk-DoS-Attacke offline nimmt. Falls diese Attacke gelingt, verlieren die angegriffenen Vertreter ihre Synchronisierung, was durch “Unsync” dargestellt wird. Abschließend kann der Angreifer einen kurzen Schub in relativer Stimmstärke erhalten, indem er seine Denial of Service Attacke gegen eine andere Reihe an Vertretern richtet, während die vorherige Reihe dabei ist, ihren Ledger zu resynchronisieren. Dies wird durch “Attacke” dargestellt.

Falls ein Angreifer mit Hilfe einer Kombination dieser Umstände Anteil >Aktiv bewirken kann, könnte er erfolgreich auf Kosten seines Anteils die Stimmen auf dem Ledger

Offline	Unsync	**Attacke**	Aktiv	Anteil

Abb. 9. Ein mögliches Stimmarrangement, dass die Anforderungen für eine 51% Attacke senken könnte.

umschwingen. Wir können einschätzen, wie viel ein Angriff dieser Art kosten würde, indem wir die Marktkapitalisierung anderer Systeme betrachten. Wenn wir annehmen, dass 33% aller Vertreter offline sind oder per DoS angegriffen werden, müsste ein Angreifer 33% der Marktkapitalisierung aufkaufen, um das System per Abstimmungsverfahren anzugreifen.

G. Bootstrap Poisoning

Je länger ein Angreifer einen alten Private-Key mit einem Guthaben behalten kann, desto höher ist die Wahrscheinlichkeit, dass Konten, die zur selben Zeit existiert haben, über keine teilnehmenden Vertreter mehr verfügen werden, da ihre Guthaben oder Vertreter an neue Konten übertragen wurden. Das bedeutet, dass falls ein Knoten an eine alte Repräsentation des Netzwerks gebootstrapped ist, in welcher der Angreifer gegenüber den Vertretern, die zu diesem Zeitpunkt existiert haben, über ein Quorum des Stimmanteils verfügt, die Entscheidungsgewalt an diesen Knoten übertragen werden könnte. Falls dieser neue Nutzer mit jemand anderem als dem Angreiferknoten kommunizieren wollte, würden all seine Transaktionen abgelehnt werden, da sie über unterschiedliche Kopfblöcke verfügen. Das Nettoergebnis ist, dass Knoten die Zeit neuer Knoten im Netzwerk verschwenden können, indem sie diese mit schlechten Informationen versorgen. Um dies zu verhindern, können Knoten mit der Datenbank der ursprünglichen Knoten und bekannten, guten Blockköpfen gepaart werden; dies dient als Alternative zum Downloaden der gesamten Datenbank bis zum Genesis-Block. Je aktueller der Download ist, umso größer ist die Wahrscheinlichkeit, vor dieser Attacke geschützt zu sein. Diese Attacke ist letztendlich nicht schlimmer, als beim Bootstrapping Junk-Daten in einen Knoten einzuspeisen, da diese nicht in der Lage wären, mit anderen Instanzen zu interagieren, die über eine aktuelle Datenbank verfügen.

VI. Implementierung

Die aktuelle Referenzimplementierung verwendet C++ und veröffentlicht seit 2014 Releases auf Github [10].

A. Design-Funktionen

Die RaiBlocks-Implementierung entspricht dem Architektur-Standard, der in diesem Papier dargelegt wurde. Weitere Spezifizierungen werden hier beschrieben.

1) Signatur-Algorithmus: RaiBlocks verwendet einen modifizierten ED25519 elliptischen Kurvenalgorithmus mit Blake2b-Hashing für alle digitalen Signaturen [11]. ED25519 wurde für schnelles Signieren, schnelle Verifizierung und ein hohes Maß an Sicherheit ausgewählt.

2) Hashing-Algorithmus: Da der Hashing-Algorithmus nur verwendet wird, um Netzwerk-Spam zu vermeiden, ist die Auswahl des Algorithmus im Vergleich zu Mining-basierten Kryptowährungen weniger wichtig. Unsere Implementierung verwendet Blake2b als Digest-Algorithmus gegen Blockinhalte [12].

3) Schlüssel-Ableitungsfunktion: Im Referenz-Wallet werden die Schlüssel mit einem Passwort verschlüsselt, welches aus einer Schlüssel-Ableitungsfunktion bezogen wird, um gegen ASIC-Cracking-Versuche geschützt zu sein. Momentan ist Argon2 [13] der Gewinner des einzigen öffentlichen Wettbewerbs zur Entwicklung einer widerstandsfähigen Schlüssel-Ableitungsfunktion.

4) Block-Intervall: Da jedes Konto über seine eigene Blockchain verfügt, können Aktualisierungen asynchron zum Stand des Netzwerks durchgeführt werden. Daher gibt es keine Block-Intervalle und Transaktionen können umgehend veröffentlicht werden.

5) UDP-Nachrichtenprotokoll: Unser System ist dazu designt worden, über einen unendlich langen Zeitraum unter Verwedung einer minimalen Menge an Rechenressourcen betrieben zu werden. Alle Nachrichten innerhalb des Systems wurden designt, um zustandslos zu sein und in ein einziges UDP-Paket hineinzupassen. Auf diese Weise ist es auch einfacher für Lite-Peers mit schwankender Verbindung sich am Netzwerk zu beteiligen, ohne kurzzeitige TCP-Verbindungen wiederherstellen zu müssen. TCP ist nur für neue Peers gedacht, die die Blockchains en masse bootstrappen wollen.

Ein Knoten kann sich sicher sein, dass das Netzwerk seine Transaktion empfangen hat, indem er den Transaktionsdatenverkehr anderer Knoten beobachten, da mehrere Kopien von diesen widergespiegelt werden sollten.

B. IPv6 und Multicast

Durch auf verbindungslosem UDP basierende Fortentwicklung können zukünftige Implementierungen IPv6 Multicast als einen Ersatz für traditionelle Transaktionsflutung und Stimmübertragung verwenden. Dadurch wird der Bandbreitenverbrauch des Netzwerks reduziert und Knoten erhalten mehr Flexibilität bei der Festlegung von Richtlinien.

C. Leistung

Zum gegenwärtigen Zeitpunkt hat das RaiBlocks-Netzwerk 4,2 Millionen Transaktionen verarbeitet, woraus sich eine Blockchain-Größe von 1,7 GB ergibt. Die Transaktionszeiten werden in Sekunden gemessen. Eine aktuelle Referenzimplementierung, die auf SSDs betrieben wird, kann über 10 000 Transaktionen pro Sekunde verarbeiten, was vor allem mit IO-Beschränkungen zusammenhängt.

VII. Ressourcenverbrauch

Dies ist eine Übersicht der Ressourcen, die von einem RaiBlocks-Knoten verbraucht werden. Zusätzlich betrachten wir einige Ideen zur Reduzierung der Ressourcen-Nutzung für spezifische Anwendungszwecke. Reduzierte Knoten werden normalerweise als leichtgewichtige, gereinigte oder vereinfachte Zahlungsverifizierungsknoten bezeichnet.

A. Netzwerk

Die Netzwerkaktivität hängt davon ab, wie viel das Netzwerk zur Gesundheit des Netzwerks beiträgt.

1) Vertreter: Ein Vertreterknoten erfordert maximale Netzwerkressourcen, da er den Abstimmungsdatenverkehr anderer Vertreter überwacht und seine eigenen Stimmen veröffentlicht.

2) Vertrauensfrei: Ein vertrauensfreier Knoten ähnelt einem Vertreterknoten, aber fungiert ausschließlich als Beobachter, er umfasst keinen Private-Key eines Vertreterkontos und veröffentlicht keine eigenen Stimmen.

3) Vertrauensbasiert: Ein vertrauensbasierter Knoten überwacht den Abstimmungsdatenverkehr eines Vertreters, dem er vertraut, einen korrekten Konsens herzustellen. Dadurch wird der eingehende Abstimmungsdatenverkehr verringert, den Vertreter an diesen Knoten senden.

4) Leichtgewichtig: Ein leichtgewichtiger Knoten ist auch ein vertrauensbasierter Knoten, der ausschließlich Datenverkehr für Konten überwacht, an denen er interessiert ist – auf diese Weise wird das Netzwerk nur minimal genutzt.

5) Bootstrap: Ein Bootstrap-Knoten überträgt den gesamten Ledger oder Teile des Ledgers an Knoten, die dabei sind, online zu gehen. Dies findet über eine TCP-Verbindung anstelle einer UDP-Verbindung statt, da bei diesem Prozess große Datenmengen involviert sind, die eine erweiterte Datenflusskontrolle benötigen.

B. Disk-Kapazität

In Abhängigkeit von den Nutzeranforderungen, verfügen unterschiedliche Knotenkonfigurationen über unterschiedliche Speicheranforderungen.

1) Historisch: Ein Knoten, der eine komplette Verlaufsaufzeichnung aller Transaktionen speichern möchte, benötigt den meisten Speicherplatz.

2) Aktuell: Aufgrund des Designs, in dessen Rahmen akkumulierte Guthaben in Blocks gespeichert werden, müssen Knoten nur den letzten Block bzw. den Kopfblock jedes Kontos speichern, um am Konsens teilzunehmen. Falls ein Knoten nicht daran interessiert ist, einen vollen Verlauf zu speichern, kann er stattdessen nur die Kopfblöcke speichern.

3) Leichtgewichtig: Ein leichtgewichtiger Knoten speichert keine lokalen Ledger-Daten und beteiligt sich nur am Netzwerk, um Aktivitäten auf Konten zu überwachen, an denen er interessiert ist – oder er erstellt optionale Transaktionen mit seinen Private-Keys.

C. CPU

1) Transaktionsgenerierung: Ein Knoten, der neue Transaktionen erstellen möchte, muss eine Proof-of-Work-Nonce erstellen, um den Drosselungsmechanismus von RaiBlocks zu passieren. Die Rechenleistung unterschiedlicher Hardware wird im Anhang A verglichen.

2) Vertreter: Ein Vertreter muss Signaturen für Blocks und Stimmen verifizieren und auch seine eigenen Signaturen produzieren, um sich am Konsens zu beteiligen. Die Menge an CPU-Ressourcen, die für einen Vertreterknoten benötigt werden, sind wesentlich niedriger, als der Ressourcenaufwand zur Transaktiosgenerierung. Eine einzige CPU in einem modernen Computer sollte ausreichend sein.

3) Beobachter: Ein Beobachter-Knoten erzeugt keine eigenen Stimmen. Da der Mehraufwand zur Signaturgenerierung minimal ist, entsprechen die CPU-Anforderungen nahezu genau den Voraussetzungen zum Betreiben eines Vertreterknotens.

VIII. Schlussfolgerung

Wir haben in diesem Papier das Rahmenwerk für eine vertrauensfreie, gebührenfreie Niedriglatenz-Kryptowährung präsentiert, die eine neuartige Blockgitter-Struktur und delegierte Proof-of-Stake-Abstimmungen verwendet. Das Netzwerk erfordert nur minimale Ressourcen, benötigt keine stromaufwändige Mining-Hardware, und kann einen hohen Transaktionsdurchsatz verarbeiten. All das wird mit Hilfe von individuellen Blockchains für jedes Konto erreicht, welche die Zugriffsprobleme und Ineffektivitäten einer globalen Datenstruktur eliminieren. Wir haben mögliche Angriffsflächen gegenüber dem System identifiziert und Argumente dazu vorgestellt, weshalb RaiBlocks gegen diese Arten von Angriffen immunisiert ist.

Anhang A
PoW-Hardware-Benchmarks

Wie zuvor erwähnt, dient das PoW von RaiBlocks dazu, Netzwerk-Spam zu reduzieren. Unsere Knoten-Implementierung bietet eine Beschleunigung, die von GPUs Gebrauch machen kann, die mit OpenCL kompatibel sind. Tabelle I bietet einen praktischen Benchmark-Vergleich unterschiedlicher Hardware. Momentan ist die PoW-Schwelle fixiert, aber eine anpassbare Schwelle kann in Zukunft als Reaktion auf eine wachsende durchschnittliche Rechenleistung implementiert werden.

TABELLE I
Hardware-PoW-Leistung

Gerät	Transaktionen pro Sekunde
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

Wie zuvor erwähnt, dient das PoW von RaiBlocks dazu, Netzwerk-Spam zu reduzieren. Unsere Knoten-Implementierung bietet eine Beschleunigung, die von GPUs Gebrauch machen kann, die mit OpenCL kompatibel sind. Tabelle I bietet einen praktischen Benchmark-Vergleich unterschiedlicher Hardware. Momentan ist die PoW-Schwelle fixiert, aber eine anpassbare Schwelle kann in Zukunft als Reaktion auf eine wachsende durchschnittliche Rechenleistung implementiert werden.

Anerkennung

Wir möchten Brian Pugh für die Kompilierung und Formattierung dieses Papiers danken.

Verweise

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] “Bitcoin median transaction fee historical chart.” [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html
[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.
[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.
[8] S. Popov, “The tangle,” 2016.
[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf