RaiBlocks: Een Gedistribueerd Cryptocurrency Netwerk Zonder Kosten

Colin LeMahieu
clemahieu@gmail.com

***Abstract*—Tegenwoordig hebben de grote vraag en de beperkte schaalmogelijkheden van populaire cryptocurrencies de gemiddelde transactietijden verhoogd, wat resulteert in een slechte gebruikerservaring. RaiBlocks, een cryptocurrency met een nieuwe block-lattice-architectuur, waarbij elk account zijn eigen blockchain heeft en met een bijna onmiddellijke transactiesnelheid en onbeperkte schaalbaarheid, biedt hierop een antwoord. Elke gebruiker heeft zijn eigen blockchain, waardoor deze asynchroon kan worden bijgewerkt naar de rest van het netwerk, wat resulteert in snelle transacties met minimale overheadkosten. Transacties houden rekeningevenwichten bij in plaats van transactiebedragen, waardoor in de database kan worden gesnoeid zonder de beveiliging in gevaar te brengen. Tot op heden heeft het RaiBlocks-netwerk 4,2 miljoen transacties verwerkt met een niet-ondersteunde grootboekgrootte van slechts 1,7 GB. RaiBlock's kosteloze, split-second transacties maken het de eerste cryptocurrency voor consumententransacties.**

***Index Terms*—cryptocurrency, blockchain, raiblocks, gedistribueerd grootboek, digitaal, transacties**

I. Introductie

SINDS de implementatie van Bitcoin in 2009 is er een toenemende verschuiving gaande van traditionele, door de overheid gesteunde valuta's en financiële systemen naar moderne betalingssystemen op basis van cryptografie, die de mogelijkheid bieden om fondsen op een betrouwbare en veilige manier op te slaan en over te dragen [1]. Om effectief te kunnen functioneren, moet een valuta gemakkelijk overdraagbaar en onomkeerbaar zijn, en beperkte of geen kosten bevatten. De verhoogde transactietijden, hoge kosten en twijfelachtige schaalbaarheid van het netwerk hebben vragen doen rijzen over de bruikbaarheid van Bitcoin als dagelijkse valuta.

In dit artikel introduceren we RaiBlocks, een cryptocurrency met lage latentie, gebouwd op een innovatieve block-lattice-gegevensstructuur, die onbeperkte schaalbaarheid en geen transactiekosten biedt. RaiBlocks is volgens ontwerp een eenvoudig protocol met als enige doel een high-performance cryptocurrency te zijn. Het RaiBlock-protocol kan op low-power-hardware draaien, waardoor het een praktische, gedecentraliseerde cryptocurrency voor dagelijks gebruik is.

Cryptocurrency-statistieken die in dit document worden vermeld, zijn accuraat vanaf de publicatiedatum.

II. Achtergrond

In 2008 publiceerde een anonieme bron onder de schuilnaam Satoshi Nakamoto, een whitepaper, die 's werelds eerste gedecentraliseerde cryptocurrency, Bitcoin, uitlijnde [1]. Een sleutelinnovatie die Bitcoin tot stand bracht was de blockchain. Een publieke, onveranderlijke en gedecentraliseerde datastructuur die wordt gebruikt als een grootboek voor de transacties van de valuta. Naarmate Bitcoin volwassen werd, maakten verschillende problemen in het protocol Bitcoin onbruikbaar:

1) Slechte schaalbaarheid: elk blok in de blockchain kan een beperkte hoeveelheid gegevens opslaan. Dat betekent dat het systeem slechts zoveel transacties per seconde kan verwerken, wat van deze blokken een grondstof maakt. Momenteel zijn de gemiddelde transactiekosten $10.38 [2].
2) Hoge latentie: de gemiddelde bevestigingstijd is 164 minuten [3].
3) Energie inefficiënt: Het Bitcoin netwerk verbruikt een geschatte 27.28TWh per jaar, met een gemiddelde van 260KWh per transactie [4].

Bitcoin en andere cryptocurrencies functioneren door consensus te bereiken over hun wereldwijde grootboeken om legitieme transacties te verifiëren, terwijl ze kwaadwillende actoren weerstaan. Bitcoin bereikt een consensus via een economische maatregel genaamd Proof of Work (PoW). In een PoW-systeem concurreren deelnemers om een getal, genaamd *nonce*, te berekenen, zodat de hash van de hele blockchain binnen doelbereik ligt. Dit geldige bereik is omgekeerd evenredig met het cumulatieve rekenvermogen van het gehele Bitcoin-netwerk om een consistente gemiddelde tijd te behouden die is bereikt voor een geldige nonce. De vinder van een geldige nonce wordt dan toegestaan om het blok aan de blockchain toe te voegen; daarom zullen degenen die meer computerkracht hebben, een grotere rol spelen in het berekeningsproces van de blockchain. PoW biedt weerstand tegen een Sybil-aanval, waarbij een entiteit zich als meerdere entiteiten voordoet om extra kracht te krijgen in een gedecentraliseerd systeem, en ook de racetoestanden die inherent aanwezig zijn tijdens toegang tot een wereldwijde gegevensstructuur aanzienlijk vermindert.

Een alternatief consensusprotocol, Proof of Stake (PoS), werd voor het eerst geïntroduceerd door Peercoin in 2012 [5]. In een PoS-systeem stemmen de deelnemers met een gewicht dat overeenkomt met het aandeel dat ze bezitten in een bepaalde cryptocurrency. Met deze regeling hebben degenen met een grotere financiële investering meer macht gekregen en worden ze inherent gestimuleerd om de eerlijkheid van het systeem te behouden, of ze lopen hey risico om hun investering te verliezen. PoS maakt komaf met de verkwistende rekenkrachtwedstrijd, waardoor alleen lichtgewicht software nodig is die draait op hardware met een laag vermogen.

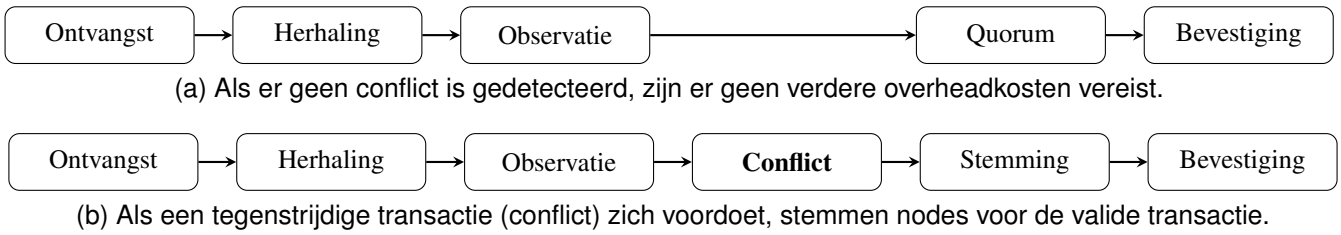

Fig. 1. RaiBlocks vereist geen additionele overheadkosten voor typische transacties. Als er tegenstrijdige transacties zijn, moeten nodes stemmen voor de transactie die behouden wordt

De oorspronkelijke RaiBlocks-paper en de eerste beta-implementatie werden gepubliceerd in december 2014, waardoor het een van de eerste Direct Acyclic Graph (DAG) cryptocurrencies werd [6]. [6]. Kort daarna begonnen zich andere DAG-cryptocurrencies te ontwikkelen, met name DagCoin / Byteball en IOTA [7], [8]. Deze op DAG gebaseerde cryptocurrencies braken de blockchain-vorm, waardoor de systeemprestaties en -beveiliging werden verbeterd. Byteball bereikt een consensus door te vertrouwen op een "hoofdketen" bestaande uit eerlijke, gerenommeerde en door de gebruiker vertrouwde "getuigen". In tegenstelling presteert IOTA via de cumulatieve PoW van gestapelde transacties. RaiBlocks bereikt consensus via een balansgewogen stem over tegenstrijdige transacties. Dit consensussysteem biedt snellere en meer deterministische transacties, terwijl er nog steeds een sterk en gedecentraliseerd systeem wordt onderhouden. RaiBlocks zet deze ontwikkeling voort en heeft zich gepositioneerd als één van de best presterende cryptocurrencies.

III. RaiBlocks Componenten

Voordat we de algemene RaiBlocks-architectuur beschrijven, moeten we eerst de individuele componenten waaruit het systeem bestaat definiëren.

A. Account

Een account is het openbare sleutelgedeelte van een digitaal handtekenings-sleutelpaar. De openbare sleutel, ook wel het adres genoemd, wordt gedeeld met andere netwerkdeelnemers. De privésleutel dient volstrekt geheim te blijven. Een digitaal ondertekend pakket van gegevens zorgt ervoor dat de inhoud wordt goedgekeurd door de houder van de privésleutel. Eén gebruiker kan veel accounts beheren, maar er kan per account slechts één openbaar adres bestaan.

B. Blok/Transactie

De term "blok" en "transactie" worden vaak onderling verwisselbaar gebruikt, waarbij een blok een enkele transactie bevat. Transactie verwijst specifiek naar de actie terwijl blok naar de digitale codering van de transactie verwijst. Transacties worden ontworpen door de privésleutel die hoort bij het account waarop de transactie wordt uitgevoerd.

C. Grootboek

Het grootboek is de wereldwijde set van rekeningen, of *accounts*, waarbij elke rekening een eigen transactieketen heeft (Figuur 2). Dit is een belangrijke ontwerp-component die valt onder de categorie van het vervangen van een looptijd overeenkomst met een ontwerptijd overeenkomst; iedereen gaat akkoord via handtekeningscontrole dat alleen een rekening-eigenaar zijn eigen keten kan wijzigen. Dit converteert een schijnbaar gedeelde gegevensstructuur, een gedistribueerd grootboek, naar een reeks niet-gedeelde grootboeken.

D. Node

Een *node* is een stuk software dat wordt uitgevoerd op een computer die voldoet aan het RaiBlocks-protocol en deelneemt aan het RaiBlocks-netwerk. De software beheert het grootboek en eventuele accounts die de node al dan niet beheert. Een node kan ervoor kiezen om het hele grootboek op te slaan, of alleen een beknopte geschiedenis die de laatste blokken van de blockchain van elk account bevat. Bij het opzetten van een nieuwe node wordt het aanbevolen om de gehele geschiedenis en stam lokaal te verifiëren.

IV. Systeemoverzicht

In tegenstelling tot blockchains die in veel andere cryptocurrencies worden gebruikt, gebruikt RaiBlocks een *blok-rooster* structuur. Elk account heeft zijn eigen blockchain (account-chain) equivalent aan de transactie/balans-geschiedenis van het account (Figuur 2). Elke accountketen kan alleen worden bijgewerkt door de eigenaar van het account; hierdoor

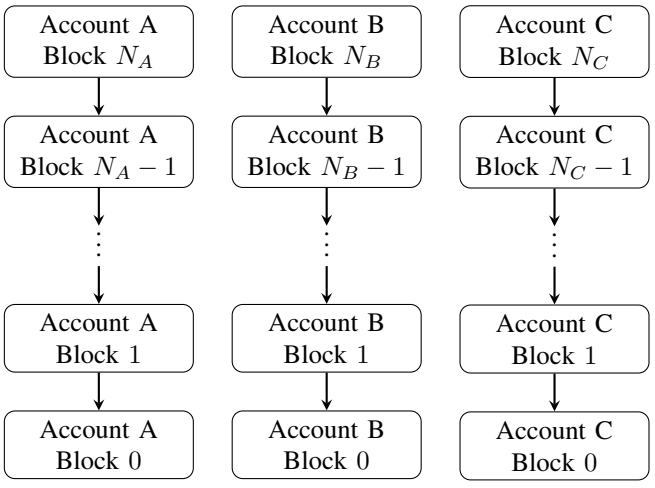

Fig. 2. Elk account heeft zijn eigen blockchain die de geschiedenis van de rekeningbalans bevat. Blok 0 moet een open transactie zijn (Section IV-B)

kan elke accountketen onmiddellijk en asynchroon worden bijgewerkt naar de rest van het blok-rooster, wat resulteert in snelle transacties. Het protocol van RaiBlocks is extreem licht; elke transactie past binnen de vereiste minimale UDP-pakketgrootte voor verzending via internet. Voorwaarden voor hardwarevereisten zijn ook minimaal, omdat nodes alleen de blokken voor de meeste transacties hoeven te registreren en opnieuw moeten uitwerken (Figuur 1).

Het systeem wordt geïnitieerd met een *genesis-account* dat de *genesis-balans* bevat. De genesis-balans is een vaste hoeveelheid en kan nooit worden verhoogd. De genesis-balans wordt gedeeld en verzonden naar andere accounts via verzend-transacties die zijn geregistreerd op de genesis-accountketen. De som van de saldi van alle accounts overschrijdt nooit de oorspronkelijke genesis-balans die het systeem dus een bovengrens geeft voor de maximale hoeveelheid. Er is dus ook geen mogelijkheid om de totale balans te verhogen.

In dit gedeelte wordt beschreven hoe verschillende soorten transacties worden geconstrueerd en verspreid door het hele netwerk.

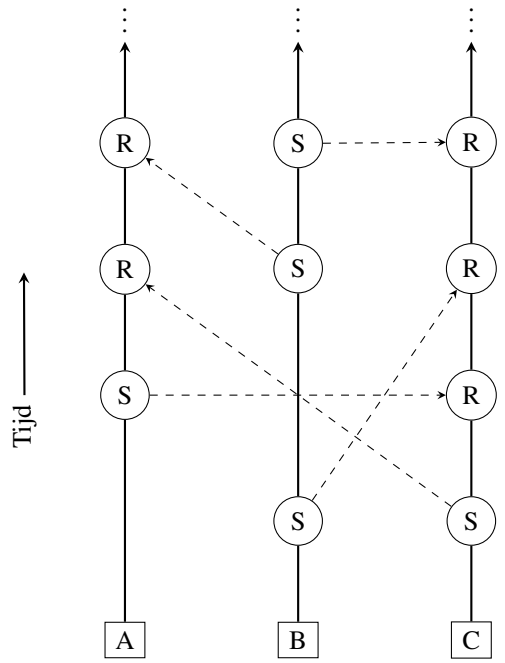

Fig. 3. Visualisatie van het blok-rooster. Elke transactie van fondsen vereist een zend blok (S) en een ontvangst blok (R), beiden ondertekend door hun rekeninghouder (A,B,C)

A. Transacties

Voor het overboeken van bedragen van het ene naar het andere account zijn twee transacties nodig: een *zending* die het bedrag van het account van de zender verminderd en een *ontvangst* die de bedragen optelt bij de balans van de ontvanger (Figuur 3).

Het overboeken van bedragen als afzonderlijke transacties in de rekeningen van de afzender en de ontvanger dient een aantal belangrijke doelen:

1) Het ordenen van inkomende transfers die inherent asynchroon zijn.
2) Transacties klein houden om in UDP-pakketten te passen.
3) Mogelijkheid tot beperken van het grootboek door het minimaliseren van de data-voetafdruk.
4) Het isoleren van afgehandelde transacties tegenover niet afgehandelde transacties.

Meerdere accounts die worden overgedragen naar hetzelfde bestemmingsaccount, vormen een asynchrone bewerking. Doordat netwerklatentie en verzendaccounts niet noodzakelijk in communicatie met elkaar zijn, betekent dit dat er geen algemeen aanvaardbare manier is om te weten welke transactie het eerst is gebeurd. Omdat optellen associatief is, is de volgorde van de inputs niet van belang, en daarom hebben we eenvoudigweg een wereldwijde overeenkomst nodig. Dit is een sleutelontwerp-component die een looptijd-overeenkomst converteert naar een overeenkomst voor ontwerptijd. Het ontvangende account heeft controle over de beslissing welke overdracht het eerst is binnengekomen en wordt uitgedrukt door de ontworpen volgorde van de inkomende blokken.

Als een account een grote overschrijving wil doen die werd ontvangen als een set van veel kleine overschrijvingen, willen we dit weergeven op een manier die past in een UDP-pakket. Wanneer aan een ontvangstrekening inkomende reeksen worden overgedragen, houdt deze een lopend totaal van haar rekeningsaldo bij, zodat het op elk moment de mogelijkheid heeft om elke hoeveelheid met een transactie van een vaste afmeting over te dragen. Dit verschilt van het input / output-transactiemodel dat door Bitcoin en andere cryptocurrencies wordt gebruikt.

Sommige nodes hebben geen interesse in het besteden van bronnen om de volledige transactiegeschiedenis van een account op te slaan; ze zijn alleen geïnteresseerd in het huidige saldo van elk account. Wanneer een account een transactie uitvoert, codeert het zijn geaccumuleerde saldo en hoeven deze nodes alleen het laatste blok bij te houden, waardoor ze historische gegevens kunnen verwijderen met behoud van de juistheid.

Zelfs met een focus op ontwerptijd-overeenkomsten is er een uitstel-venster voor het valideren van transacties, vanwege het identificeren en verwerken van slechte actoren in het netwerk. Omdat overeenkomsten in RaiBlocks snel worden bereikt, in de orde van milliseconden tot seconden, kunnen we de gebruiker twee vertrouwde categorieën van inkomende transacties presenteren: afgehandeld en niet-afgehandeld. Afgehandelde transacties zijn transacties waarbij een account ontvangstblokken heeft gegenereerd. Niet-afgehandelde transacties zijn nog niet opgenomen in het cumulatieve saldo van de ontvanger. Dit is een vervanging voor de meer complexe en onbekende bevestigingsmetriek in andere cryptocurrencies.

B. Een Account maken

Als u een account wilt maken, moet u een *openstaande* transactie uitvoeren (Figuur 4). Een open transactie is altijd de eerste transactie van elke accountketen en kan worden aangemaakt bij de eerste ontvangst van een bedrag. Het *account* veld bewaart de publieke sleutel (adres) afgeleid van

de privésleutel die gebruikt wordt voor de handtekening. Het *bron* veld bevat de hash van de transactie die de bedragen heeft verzonden. Bij het maken van een account moet een vertegenwoordiger worden gekozen om namens u te stemmen; dit kan later worden gewijzigd (Sectie IV-F). Het account kan zichzelf tot eigen vertegenwoordiger verklaren.

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

Fig. 4. Anatomie van een open transactie

C. Rekening-balans

Het rekeningsaldo wordt binnen het grootboek zelf geregistreerd. In plaats van het bedrag van een transactie op te nemen, zal verificatie (Sectie IV-I) het controleren van het verschil tussen de balans op het verzend-blok en de balans van het vorige blok vereisen. Het ontvangende account kan dan het vorige saldo verhogen zoals gemeten in het eindsaldo dat is gegeven in het nieuwe ontvang-blok. Dit wordt gedaan om de verwerkingssnelheid te verbeteren bij het downloaden van grote hoeveelheden blokken. Bij het aanvragen van de accountgeschiedenis zijn bedragen al gegeven.

D. Zenden van een Account

Om van een adres te verzenden, moet het adres reeds een openstaand blok hebben, en dus ook een balans (Figuur 5). Het *vorige* veld bevat de hash van het vorige blok in de rekening-keten. Het *bestemming* veld bevat de rekening voor de te verzenden bedragen. Een verzend-blok is onveranderbaar zodra het is bevestigd. Zodra het op het netwerk is uitgezonden, worden de bedragen onmiddellijk van het saldo van de afzenders-rekening afgetrokken en worden beschouwd als *wachtend* totdat de ontvanger een blok signeert om de bedragen te aanvaarden. Geld in afwachting moet niet worden beschouwd als een wachtende bevestiging, omdat ze zo goed als uitgegeven zijn door het afzender-account en de afzender de transactie niet kan intrekken.

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

Fig. 5. Anatomie van een verzend-transactie

E. Een Transactie Ontvangen

Om een transactie te voltooien, moet de ontvanger van verzonden bedragen een ontvangstblok maken in zijn eigen accountketen (Figuur 6). Het bronveld verwijst naar de hash van de bijbehorende verzend-transactie. Zodra dit blok is gemaakt en uitgezonden, wordt het accountsaldo bijgewerkt en zijn de bedragen officieel op hun rekening geplaatst.

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

Fig. 6. Anatomie van een ontvangst-transactie

F. Een Vertegenwoordiger Toewijzen

Rekeninghouders die de mogelijkheid hebben om een vertegenwoordiger te kiezen die namens hen stemt, is een krachtige decentralisatie-tool die geen sterke analogie heeft met Proof of Work, noch Proof of Stake protocollen. In conventionele PoS-systemen moet de node van de rekeninghouder draaien om deel te nemen aan de stemming. Het continu uitvoeren van een node is voor veel gebruikers onpraktisch; een vertegenwoordiger de mogelijkheid geven namens een account te stemmen, versoepelt deze vereiste. Rekeninghouders kunnen op elk moment consensus toevoegen aan een rekening. Een *veranderings-* transactie wijzigt de vertegenwoordiger van een rekening door het stemgewicht af te trekken van de oude vertegenwoordiger en het gewicht toe te voegen aan de nieuwe vertegenwoordiger (Figuur 7). Er worden geen bedragen verplaatst in deze transactie en de vertegenwoordiger heeft geen koopkracht over de tegoeden van het account.

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

Fig. 7. Anatomie van een veranderings-transactie

G. Vorken en Stemmen

Een vork treedt op wanneer j ondertekende blokken $b_1, b_2, \ldots, b_j$ dezelfde blok als hun voorgaande opeisen (Figuur 8). Deze blokken veroorzaken een conflicterende weergave van de status van een account en moeten worden opgelost. Alleen de eigenaar van de rekening heeft de mogelijkheid om blokken in de account-keten te ondertekenen, dus een vork moet het resultaat zijn van slechte programmering

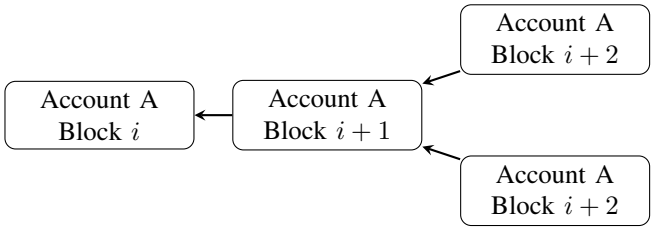

Fig. 8. Een vork treedt op wanneer twee (of meer) ondertekende blokken verwijzen naar hetzelfde vorige blok. Oudere blokken staan aan de linkerkant; nieuwere blokken staan aan de rechterkant

of kwaadwillige intentie (dubbele uitgaven) door de eigenaar van het account.

Na detectie creëert een vertegenwoordiger een stem die verwijst naar het blok $\hat{b}_i$ in het grootboek en zendt het naar het netwerk. Het stemgewicht van een node, w_i, is de som van de saldi van alle accounts die het als zijn vertegenwoordiger hebben genoemd. De node neemt de inkomende stemmen van de andere online vertegenwoordigers van M waar en houdt een cumulatief aantal stemperioden van 1 minuut bij en bevestigt het winnende blok 1).

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

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

Het meest populaire blok b^* zal de meerderheid van de stemmen halen en zal worden bewaard in het grootboek (Vergelijking 2). De blokken die de stemming verliezen worden genegeerd. Als een vertegenwoordiger een blok in het grootboek vervangt, maakt het een nieuwe stem met een hoger volgnummer en zendt het de nieuwe stem uit naar het netwerk. Dit is het **enige** scenario waarin de vertegenwoordigers stemmen.

In sommige omstandigheden kunnen korte netwerkconnectiviteitsproblemen ertoe leiden dat een uitgezonden blok niet door alle peers wordt geaccepteerd. Elk volgend blok in dit account zal als ongeldig worden genegeerd door peers die de eerste uitzending niet hebben gezien. Een heruitzending van dit blok wordt door de resterende peers geaccepteerd en de daaropvolgende blokken worden automatisch opgehaald. Zelfs wanneer een vork of een ontbrekend blok voorkomt, worden alleen de rekeningen beïnvloed die in de transactie worden genoemd; de rest van het netwerk gaat verder met verwerkingstransacties voor alle andere accounts.

H. Proof of Work

Alle vier de transactietypen hebben een werkveld dat correct moet worden ingevuld. Het werkveld maakt het de transactiemaker mogelijk om een nonce te berekenen, zodanig dat de hash van de nonce samengevoegd met het vorige veld in ontvang / verzend / wijzigings transacties of het accountveld in een open transactie onder een bepaalde drempelwaarde ligt. In tegenstelling tot Bitcoin, wordt de PoW in RaiBlocks eenvoudig weg gebruikt als antispamhulpmiddel, vergelijkbaar met Hashcash, en kan deze worden berekend in de orde van seconden [9]. Zodra een transactie is verzonden, kan de PoW voor het volgende blok vooraf worden berekend sinds het vorige blokveld bekend is; hierdoor zullen transacties ogenblikkelijk aan een eindgebruiker verschijnen, zolang de tijd tussen transacties groter is dan de tijd die nodig is om de PoW te berekenen.

I. Transactie Verificatie

Om een blok als geldig te kunnen beschouwen, moet het blok de volgende attributen hebben:

1) Het blok mag zich niet al in het grootboek bevinden (dubbele transactie).
2) Moet door de eigenaar van het account worden ondertekend.
3) Het vorige blok is het hoofdblok van de accountketen. Als het bestaat, maar niet het hoofd is, is het een vork.
4) Het account moet een open blok hebben.
5) De berekende hash voldoet aan de PoW-drempelvereiste.

Als het een ontvangst blok is, controleer dan of de bronblok hash in behandeling is, wat betekent dat het nog niet is ingewisseld. Als het een verzendblok is, moet het saldo kleiner zijn dan het vorige saldo.

V. Aanvalsvectoren

RaiBlocks kan, net als alle gedecentraliseerde cryptocurrencies, worden aangevallen door kwaadwillende partijen voor poging tot financieel gewin of het vernietigen van het systeem. In deze sectie schetsen we enkele mogelijke aanvalscenario's, de gevolgen van een dergelijke aanval en hoe het protocol van RaiBlocks preventieve maatregelen neemt.

A. Block Gap Synchronisatie

In Sectie IV-G, hebben we het scenario besproken waarbij een blok mogelijk niet goed wordt uitgezonden, waardoor het netwerk de volgende blokken negeert. Als een node een blok waarneemt dat niet het vorige blok bevat, heeft het twee opties:

1) Negeer het blok want het kan kwaadaardig zijn.
2) Vraag een hersynchronisatie met een andere node.

In het geval van een hersynchronisatie moet een TCP-verbinding worden gevormd met een bootstrap-node om de toegenomen hoeveelheid verkeer die een hersynchronisatie nodig heeft te vergemakkelijken. Als het blok echter een slecht blok zou zijn geweest, zou de hersynchronisatie onnodig zijn en het verkeer op het netwerk zou onnodig zijn toegenomen. Dit is een Network Amplification Attack en resulteert in een denial-of-service.

Om onnodige hersynchronisatie te voorkomen, wachten nodes tot een bepaalde drempel van stemmen is vastgesteld voor een mogelijk kwaadaardig blok, voordat een verbinding met een bootstrap-node wordt gestart om te synchroniseren. Als een blok onvoldoende stemmen krijgt, kan worden aangenomen dat dit ongewenste gegevens zijn.

B. Het netwerk overspoelen met transacties

Een kwaadwillende entiteit kan veel onnodige maar geldige transacties verzenden tussen accounts onder zijn beheer in een poging het netwerk te verzadigen. Zonder transactiekosten kunnen ze deze aanval voor onbepaalde tijd voortzetten. De vereiste PoW voor elke transactie beperkt echter de transactiewaarde die de kwaadwillende entiteit zou kunnen genereren zonder significant te moeten investeren in computationele bronnen. Zelfs onder een dergelijke aanval in een poging om het grootboek op te blazen, zijn nodes die geen nodes met een volledige historie zijn in staat om oude transacties uit hun keten te snoeien; dit beperkt het opslaggebruik van dit type aanval voor bijna alle gebruikers.

C. Sybil-Aanval

Een entiteit kan honderden RaiBlocks-nodes maken op één enkele machine; Aangezien het stemsysteem echter wordt gewogen op basis van het rekeningsaldo, resulteert het toevoegen van extra nodes aan het netwerk niet tot extra stemmen voor aanvallers. Daarom is er geen voordeel te behalen via een Sybil-aanval.

D. Penny-Spend-Aanval

Bij een Penny-Spend-aanval zal een aanvaller oneindige spam-hoeveelheden aan een groot aantal accounts besteden om de opslagbronnen van nodes te verspillen. Blok publicatie wordt beperkt door het PoW-bestand, dus dit beperkt het maken van accounts en transacties tot op zekere hoogte. Nodes die geen volledige historische nodes zijn, kunnen accounts onder een statistische metriek snoeien waarbij het account hoogstwaarschijnlijk geen geldig account is. Ten slotte is RaiBlocks afgestemd om minimale permanente opslagruimte te gebruiken, dus ruimte die nodig is om één extra account op te slaan is evenredig met de grootte van een open block + indexing $= 96\text{B} + 32\text{B} = 128\text{B}$. Dit staat gelijk aan het op kunnen slaan van 8 miljoen penny-accounts in 1GB. Als nodes agressiever accounts willen inkorten, kunnen ze een distributie berekenen op basis van de toegangsfrequentie en niet vaak gebruikte accounts delegeren naar langzamere opslag.

E. Precomputed PoW Aanval

Aangezien de eigenaar van een account de enige entiteit is die blokken aan de accountketen toevoegt, kunnen opeenvolgende blokken worden berekend, samen met hun PoW, voordat ze naar het netwerk worden uitgezonden. Hier genereert de aanvaller een groot aantal opeenvolgende blokken, elk met een minimale waarde, over een langere periode. Op een bepaald moment voert de aanvaller een Denial of Service (DoS) uit door het netwerk te overspoelen met veel geldige transacties, die andere nodes zo snel mogelijk verwerken en echoën. Dit is een geavanceerde versie van de overslag van transacties beschreven in Sectie V-B. Een dergelijke aanval zou slechts kort werken, maar zou samen met andere aanvallen kunnen worden gebruikt, zoals een >50% Aanval (Section V-F) om de effectiviteit te vergroten. Momenteel worden transactiesnelheid-beperkende en andere technieken onderzocht om aanvallen te verminderen.

F. >50% Aanval

De mate van consensus voor RaiBlocks is een evenwichtig gewogen stemsysteem. Als een aanvaller in staat is om meer dan 50% van de stemkracht te krijgen, kan hij ervoor zorgen dat het netwerk een consensus forceert waardoor het systeem kapot gaat. Een aanvaller kan de hoeveelheid balans die hij moet inleveren verlagen door te voorkomen dat goede nodes via een DoS netwerk stemmen. RaiBlocks neemt de volgende maatregelen om een dergelijke aanval te voorkomen:

1) De belangrijkste verdediging tegen dit soort aanvallen is dat het stemgewicht wordt gekoppeld aan de investering in het systeem. Een rekeninghouder is inherent gestimuleerd om de eerlijkheid van het systeem te handhaven om zijn investering te beschermen. Een poging om het grootboek om te draaien zou destructief zijn voor het systeem als geheel, wat hun investering zou vernietigen.
2) De kosten van deze aanval zijn evenredig aan de marktkapitalisatie van RaiBlocks. In PoW-systemen kan technologie worden uitgevonden die een onevenredige controle geeft in vergelijking met monetaire investeringen en als de aanval succesvol is, kan deze technologie worden hergebruikt nadat de aanval is voltooid. Met RaiBlocks worden de kosten van een aanval op het systeem geschaald met het systeem zelf en zelfs als een aanval succesvol zou zijn, kan de investering van de aanval niet worden teruggevorderd.
3) Om het maximale aantal van kiezers te handhaven, is de volgende verdedigingslinie representatief stemmen. Rekeninghouders die wegens connectiviteitsredenen niet op betrouwbare wijze kunnen deelnemen aan het stemmen, kunnen een vertegenwoordiger aanduiden die kan stemmen met het gewicht van hun saldo. Het maximaliseren van het aantal en de diversiteit van vertegenwoordigers verhoogt de netwerkbestendigheid.
4) Vorken in RaiBlocks gebeuren nooit per ongeluk, dus nodes kunnen beleidsbeslissingen nemen over hoe om te gaan met gevorkte blokken. De enige tijd dat niet-aanvallersaccounts kwetsbaar zijn voor blokvorken is als ze een saldo van een aanvallend account ontvangen. Accounts die beveiligd willen zijn tegen blokvorken kunnen een beetje of veel langer wachten voordat ze een account ontvangen dat vorken genereert of ervoor kiest om nooit te ontvangen. Ontvangers kunnen ook aparte accounts genereren om te gebruiken bij het ontvangen van geld van dubieuze accounts om andere accounts te isoleren.
5) Een laatste verdedigingslinie die nog niet is geïmplementeerd, is blokcementering. RaiBlocks doet er alles aan om blokvorken snel te vereffenen via stemmen. Nodes kunnen worden geconfigureerd om blokken te cementeren, wat zou voorkomen dat ze na een bepaalde periode worden teruggerold. Het netwerk is voldoende beveiligd door zich te concentreren op een snelle insteltijd om dubbelzinnige vorken te voorkomen.

Een meer geavanceerde versie van een $> 50\%$ aanval is gedetailleerd in Figuur 9. “Offline” is het percentage vertegenwoordigers dat is genoemd maar niet online is om te stemmen.

"Stake" is de hoeveelheid investering waarmee de aanvaller stemt. "Active" zijn vertegenwoordigers die online zijn en stemmen volgens het protocol. Een aanvaller kan het bedrag van de te verbeuren inzet verrekenen door andere kiezers offline te slaan via een netwerk-DoS-aanval. Als deze aanval kan worden volgehouden, worden de aangevallen vertegenwoordigers niet gesynchroniseerd en dit wordt aangetoond door "Unsync." Ten slotte kan een aanvaller een korte opstoot in relatieve stemkracht krijgen door zijn Denial of Service-aanval om te zetten in een nieuwe set vertegenwoordigers, terwijl de oude set hun grootboek opnieuw synchroniseert, dit wordt gedemonstreerd door "Attack."

Offline	Unsync	**Attack**	Active	Stake

Fig. 9. Een mogelijk stemmingsarrangement dat de aanvalsbehoeften van 51% kan verlagen.

Als een aanvaller een Stake >Active kan veroorzaken door een combinatie van deze omstandigheden zouden ze de stemmen op het grootboek kunnen omdraaien ten koste van hun inzet. We kunnen inschatten hoeveel dit type aanval kan kosten door de marktkapitalisatie van andere systemen te onderzoeken. Als we schatten dat 33% van de vertegenwoordigers offline is of wordt aangevallen via DoS, dan moet een aanvaller 33% van de marktkapitalisatie kopen om het systeem via stemmen aan te vallen.

G. Bootstrap Poisoning

Hoe langer een aanvaller in staat is om een oude privésleutel met een saldo te behouden, des te groter de kans dat saldi die op dat moment bestaan, geen deelnemende vertegenwoordigers zullen hebben, omdat ze hun saldo of vertegenwoordigers aan nieuwere rekeningen hebben overgedragen. Dit betekent dat als een node wordt gebootstrapped naar een andere representatie van het netwerk waar de aanvaller een quorum van stembus heeft ten opzichte van vertegenwoordigers op dat moment in tijd, ze in staat zouden zijn om de stem-beslissing op de node te laten oscilleren. Als deze nieuwe gebruiker wilde communiceren met alle andere aanvallers, zouden al hun transacties genegeerd worden omdat ze verschillende hoofdblokken hebben. Het netto resultaat is dat nodes de tijd van nieuwe nodes in het netwerk kunnen verspillen door hen slechte informatie te geven. Om dit te voorkomen, kunnen nodes worden gekoppeld aan een eerste database met accounts en bekende goede blok koppen; dit is een vervanging voor het downloaden van de database helemaal terug naar het genesisblok. Hoe dichter de download is, hoe hoger de kans op een nauwkeurige verdediging tegen deze aanval. Uiteindelijk is deze aanval waarschijnlijk niet slechter dan het voeden van ongewenste gegevens aan nodes die bootstrappen, omdat ze niet in staat zouden zijn om transacties met iemand met een hedendaagse database uit te voeren.

VI. Implementatie

Momenteel is de referentie-implementatie geïmplementeerd in C++ en zijn er releases uitgebracht op GitHub sinds 2014 [10].

A. Ontwerp Kenmerken

De RaiBlocks implementatie houdt zich aan de architectuurstandaard die geschetst is in deze paper. Aanvullende specificaties worden hier beschreven.

1) Ondertekenings Algoritme: RaiBlocks gebruikt een aangepast ED25519 elliptic curve algoritme met Blake2b hashing voor alle digitale ondertekeningen [11]. Er is gekozen voor ED25519 vanwege snelle ondertekening, snelle verificatie, en een hoge mate van beveiliging.

2) Hashing Algoritme: Aangezien het hashing algoritme alleen wordt gebruikt om netwerk spam tegen te gaan, is de keuze voor het algoritme minder belangrijk vergeleken met op mining gebaseerde cryptocurrencies. Onze implementatie gebruikt Blake2b als een digest algoritme tegen blokinhoud [12].

3) Sleutel Afleidings Functie: In de referentieportefeuille worden sleutels gecodeerd met een wachtwoord en wordt het wachtwoord door een sleutelafleidingsfunctie geleid om het te beschermen tegen pogingen tot ASIC-cracking. Momenteel is Argon2 [13] de winnaar van de enige publieke competitie gericht op het creëren van een veerkrachtige sleutelafleidingsfunctie.

4) Blok Interval: Omdat elk account zijn eigen blockchain heeft, kunnen updates asynchroon worden uitgevoerd naar het netwerk. Daarom zijn er geen blok-intervallen en kunnen transacties direct worden gepubliceerd.

5) UDP Message Protocol: Ons systeem is ontworpen om voor onbepaalde tijd te werken met behulp van de kleinst mogelijke hoeveelheid computer-hulpbronnen. Alle berichten in het systeem zijn ontworpen om stateloos te zijn en passen in een enkel UDP-pakket. Dit maakt het ook gemakkelijker voor lite-peers met intermitterende connectiviteit om deel te nemen aan het netwerk zonder opnieuw korte TCP-verbindingen tot stand te brengen. TCP wordt alleen gebruikt voor nieuwe peers wanneer zij de blockchains in bulk willen opstarten.

Nodes kunnen er zeker van zijn dat hun transactie wordt ontvangen door het netwerk door transactie-uitzendverkeer van andere nodes te observeren, omdat verschillende exemplaren zouden moeten worden teruggekaatst naar zichzelf.

B. IPv6 en Multicast

Door te bouwen op verbindingsloze UDP kunnen toekomstige implementaties IPv6-multicast gebruiken als vervanging voor traditionele transactiestromingen en stemuitzendingen. Dit zal het verbruik van netwerkbandbreedte verminderen en zal meer beleidsflexibiliteit bieden aan nodes die voorwaarts gaan.

C. Prestaties

Op het moment van schrijven zijn er 4,2 miljoen transacties verwerkt door het RaiBlocks-netwerk, wat een blockchain-omvang van 1,7 GB heeft opgeleverd. Transactietijden worden gemeten in de orde van seconden. Een huidige referentie-implementatie die werkt op standaard SSD's kan meer dan 10.000 transacties per seconde verwerken, voornamelijk IO-gebonden.

VII. Bronverbruik

Dit is een overzicht van bronnen die worden gebruikt door een RaiBlocks-node. Daarnaast bespreken we ideeën voor het verminderen van het gebruik van hulpbronnen voor specifieke use-cases. Gereduceerde nodes worden meestal lichte, gesnoeide of vereenvoudigde betalingsverificatie (SPV) nodes genoemd.

A. Netwerk

De hoeveelheid netwerkactiviteit hangt af van hoeveel het netwerk bijdraagt aan de gezondheid van een netwerk.

1) Representatief: Voor een representatieve node zijn maximale netwerkbronnen nodig, aangezien het het stemverkeer van andere vertegenwoordigers waarneemt en zijn eigen stemmen publiceert.

2) Vertrouwd: Een vertrouwde node lijkt op een representatieve node maar is alleen een waarnemer, het bevat geen persoonlijke sleutel van een representatief account en publiceert geen eigen stemmen.

3) Vertrouwend: Een vertrouwende node observeert het stemmingsverkeer van de ene vertegenwoordiger die hij vertrouwt om een juiste consensus te bereiken. Dit vermindert de hoeveelheid inkomend stemverkeer van vertegenwoordigers die naar deze nodes gaan.

4) Lichte node: Een lichte node is ook een vertrouwensnode die alleen verkeer waarneemt voor accounts waarin het geïnteresseerd is, wat een minimaal netwerkgebruik mogelijk maakt.

5) Bootstrap: Een bootstrap-node biedt delen van het grootboek aan aan nodes die zichzelf online brengen. Dit wordt gedaan via een TCP-verbinding in plaats van UDP, omdat het om een grote hoeveelheid gegevens gaat waarvoor geavanceerde stroomregeling vereist is.

B. Opslagcapaciteit

Afhankelijk van de gebruikerseisen, vereisen verschillende nodeconfiguraties verschillende opslagvereisten.

1) Historisch: Een node die geïnteresseerd is in het bijhouden van een volledig historisch overzicht van alle transacties, heeft de maximale hoeveelheid opslagruimte nodig.

2) Actueel: Vanwege het ontwerp van het bijhouden van verzamelde saldi met blokken, hoeven nodes alleen de nieuwste of hoofdblokken voor elk account bij te houden om deel te nemen aan consensus. Als een node niet geïnteresseerd is in het bijhouden van een volledige geschiedenis, kan deze ervoor kiezen om alleen de hoofdblokken te behouden.

3) Lichte node: Een lichte node bewaart geen gegevens in het grootboek en neemt alleen deel aan het netwerk om activiteiten te observeren op accounts waarin het geïnteresseerd is of om optioneel nieuwe transacties te maken met privésleutels die het bevat.

C. CPU

1) Transactie Genererend: Een node dat geïnteresseerd is in het maken van nieuwe transacties moet een bewijs van nonce produceren om het throttling-mechanisme van RaiBlocks te doorstaan. Een referentiekader van verschillende hardware is te vinden in Bijlage A.

2) Vertegenwoordiger: Een vertegenwoordiger moet handtekeningen verifiëren voor blokken, stemmen en ook zijn eigen handtekeningen produceren om deel te nemen aan consensus. De hoeveelheid CPU-bronnen voor de representatieve node is aanzienlijk minder dan die voor het genereren van transacties en zou moeten werken met elke afzonderlijke CPU in een moderne computer.

3) Waarnemers-node: Een waarnemers-node genereert geen eigen stemmen. Aangezien de overhead van de handtekeninggeneratie minimaal is, zijn de CPU-vereisten vrijwel identiek aan het uitvoeren van een representatieve node.

VIII. Conclusie

In deze paper hebben we een framework besproken voor een gedistribueerd cryptocurrency netwerk met hoge snelheid en zonder kosten, dat gebruik maakt van een nieuwe *block-lattice* structuur en *delegated Proof of Stake* stemming. Het netwerk vereist minimale middelen, geen mining hardware met hoog energieverbruik, en kan een hoge doorvoer van transacties verwerken. Dit wordt bereikt door individuele blockchains voor ieder account, waardoor toegangsproblemen en inefficiënties van een globale datastructuur worden geëlimineerd. We hebben mogelijke aanvalsvectoren op het systeem geïdentificeerd en bespraken argumenten over hoe RaiBlocks resistent is tegen deze vormen van aanvallen.

Appendix A
PoW Hardware referentiekader

Zoals eerder vermeld is de PoW in RaiBlocks om netwerk spam te verminderen. Onze node implementatie biedt versnelling die kan profiteren van OpenCL compatibele GPUs. Tabel I verstrekt een actueel referentiekader van verschillende hardware. Momenteel is de PoW-drempel vaststaand, maar een adaptieve drempel zou geïmplementeerd kunnen worden naarmate de gemiddelde rekenkracht van computers toeneemt.

TABEL I
Hardware PoW Prestaties

Apparaat	Transacties Per Seconde
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

Dankwoord

We willen Brian Pugh bedanken voor het compileren en formatteren van deze paper.

REFERENTIES

[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