RaiBlocks: Besplatna Distribuirana Kriptovalutna Mreža

Colin LeMahieu
clemahieu@gmail.com

***Sažetak*—Nedavno, velika potražnja i ograničena skalabilnost, povećale su prosječnu brzinu i cijenu transakcije postojećih kriptovaluta, ostavljajući za sobom loše korisničko iskustvo. Predstavljamo Vam RaiBlocks, kriptovalutu sa *novel block-lattice* arhitekturom u kojoj svaki korisnik ima svoj *blockchain*, omogućujući gotovo instantne brzine transakcija i neograničenu skalabilnost. Svaki korisnik ima svoj vlastiti *blockchain*, što mu omogućuje da se asinkrono sinkronizira sa ostatkom mreže, što rezultira brzim transakcijama sa minimalnim opterećenjem. Transakcije vode evidenciju o saldima računa, a ne o količini transakcije, što omogućuje agresivno optimiziranje baze podataka bez komprimitiranja sigurnosti. Do danas, RaiBlocks mreža je procesuirala 4.2 milijuna transakcija sa ne optimiziranom bazom podataka veličine 1.7GB. RaiBlock-ove besplatne i instantne transakcije čine ga savršenim izborom za korištenje.**

***Indeks Pojmovi*—kriptovalute, blockchain, raiblocks, distribuirana glavna knjiga, digitalne, transakcije**

I. Uvod

OD pojave Bitcoina 2009. godine, dešava se velik pomak sa tradicionalnog sustava plaćanja, prema modernom sustavu plaćanja temeljnog na kriptografiji, koji je stvorio mogućnost da novac spremamo i razmjenjujemo na pouzdan i siguran način. [1]. Da bi bila efikasna, valuta mora biti lako prenosiva, i imati male ili nepostojeće transakcijske naknade. Povećno vrijeme transakcija, velike naknade i upitna mrežna skalabilnost, podigle su pitanje o praktičnosti Bitciona kao svakodnevne valute.

U ovom dokumentu predstavljamo vam RaiBlocks, niskolatentnu kriptovalutu izgrađenu na inovativnoj *block-lattice* podatkovnoj strukturi, koja omogućuje neograničenu skalabilnost i besplatne transakcije. RaiBlocks je po svom dizajnu jednostavan protokol sa jedinom svrhom da bude kritpovaluta visokih performansi. RaiBlocks protokol može raditi na hardveru male snage, što mu omogućuje da bude praktična, decentralizirana kriptovaluta za svakodnevnu upotrebu.

Statistike o kriptovalutama iznesene u ovom dokumentu su točne na dan objave dokumenta.

II. Pozadina

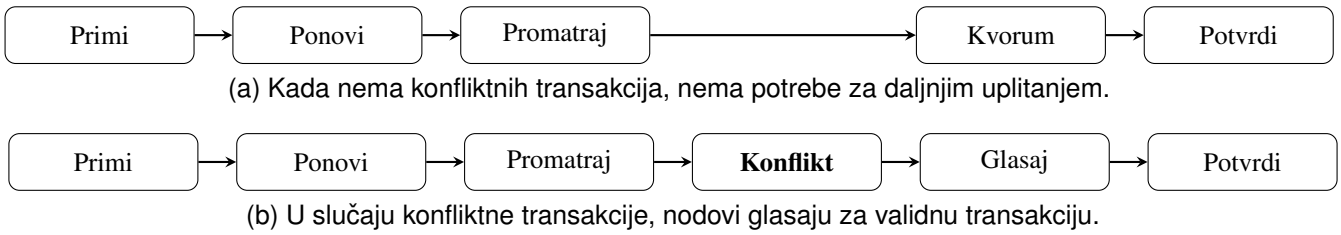
2008. godine, anonimac pod pseudonimom Satoshi Nakamoto, objavio je dokument u kojemu je opisao prvu svjetsku decentraliziranu kriptovalutu, Bitcoin [1]. Ključna inovacija u njegovom dokumentu, bio je *blockchain*, javna, nepromjenjiva i decentralizirana podatkovna struktura koja se koristi kao knjiga salda za transakcije te valute. Nažalost, kako je Bitcoin sazrijevao, nekoliko nedostataka u njegovom protokolu izašlo je na vidjelo:

1) Slaba skalabilnost: Svaki blok u *blockchainu* može spremiti ograničenu količinu podataka, što znači da sistem može procesuirati ograničenu količinu transakcija, što je stvorilo mjesto u bloku, a time i samu transakciju skupom. Trenutno, prosječna cijena transakcije je $10.38 [2].
2) Visoka latentnost: Prosječno vrijeme za konfirmaciju tranaskcije je 164 minute. [3].
3) Potrošnja struje: Bitcoin mreža u prosjeku konzumira 27.28TWh struje godišnje, trošeći prosječnih 260KWh po transakciji [4].

Bitcoin, i ostale kriptovalute, funkcioniraju donoseći suglasnost o svojim globalnim knjigama salda da bi verificirale legitimne transakcije, dok istovremeno onemogućuju lažne trankcije. Bitcoin postiže suglasnost ekonomskom mjerom zvanom Proof of Work (PoW). U PoW sustavu, sudionici se natječu u računanju broja, koji zovemo *dokaz*, tako da je haš cijeloga bloka u ciljanom rasponu. Taj ciljani raspon je obrnuto proporcionalan kumulativnoj računalnoj snazi cijele Bitcoin mreže, sa namjerom da se dobije konzistentno prosječno vrijeme potrebno da se pronađe validni dokaz. Pronalazniku validnog dokaza, je tada dozvoljeno da dodaje novi blok u blockchain; prema tome, oni koji potroše više računalne snage na izračunavanje dokaza igraju veću ulogu u stanju *blockchain-a*. PoW omogućava otpornost na Sybil napade, gdje entitet oponaša mnoštvo entiteta da bi dobio veću moć u decentraliziranom sustavu, i ujedno uvelike smanjuje uvjete utrke koja postoji dok pristupa globalnoj strukturi podataka.

Alternativni protokol suglasnosti, Proof of Stake (PoS), je prvi predstavio Peercoin 2012 godine [5]. U PoS sustavu, sudionici glasaju sa težinom koja je ekvivalentna količini bogatstva koje posjeduju u datoj kriptovaluti. Tim uređenjem, oni koji su više financijski investirani, imaju veću moć koja ih potiče da održavaju sigurnost sustava ili riskiraju gubitak svoje investicije. PoS na taj način zaobilazi potrebu za rasipnim natjecanjem u računalnoj snazi, i omogućuje korištenje laganog softvera koji radi na energetski efikasnom hardveru.

Originalni RaiBlocks dokument i prva beta implementacija je objavljena u prosincu, 2014. godine, što ju čini prvom Directed Acyclic Graph (DAG) kriptovalutom [6]. Ubrzo nakon toga, ostale DAG kriptovalute započele su svoj razvoj, među značajnijima su DagCoin/Byteball i IOTA [7], [8]. Navedene kriptovalute temeljene na DAG-u izašle su iz *blockchain* kalupa, unaprijedile performanse sustava i njegovu sigurnost. Byteball postiže suglasnost oslanjajući se na lanac koji se sastoji od poštenih, reputabilnih i provjerenih ”svjedoka”, dok IOTA postiže suglasnost kumulativno dodajući POW transak-



Slika 1. RaiBlocks ne zahtjeva nikakav dodatan rad u slučaju validne transakcije. U slučaju konfliktne transakcije, nodovi glasaju koju transakciju će zadržati.

cijama. RaiBlocks postiže dogovor uravnoteženom težinom suglasnosti kod konfliktnih transakcija. Takav sustav suglasnosti pruža brze i deterministične transakcije dok istovremeno zadržava snažan decentralizirani sustav. RaiBlocks nastavlja svoj razvoj i pozicionirao se kao jedna od najperformantnijih kriptovaluta.

III. RaiBlocks Komponente

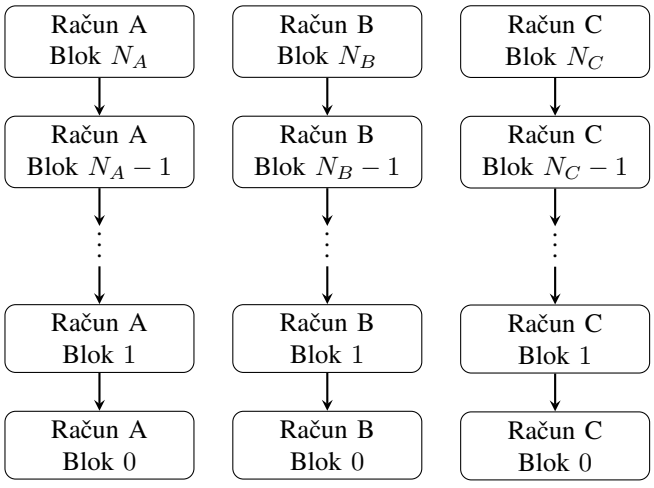
Prije nego opišemo kompletnu RaiBlocks arhitekturu, definirat ćemo individualne komponente koje čine njen sustav.

A. Račun

Račun je *public-key* dio digitalnog potpisa. *Public-key*, također korišten kao adresa dijeli sa ostalim učesnicima u mreži, dok je *private-key* tajan. Digitalno potpisani paket podataka brine se da je sadržaj bio odobren od vlasnika koji posjeduje *private-key*. Jedan korisnik može kontrolirati nekoliko računa, ali samo jedna adresa može postojati po jednom računu.

B. Blok/Transakcija

Nazive *blok* i *transakcija* često koristimo izmjenično, gdje blok sadrži jednu transakciju. Transakcija se specifično odnosi na akciju dok se blok odnosi na digitalno kodiranje transakcije. Transakcije potpisuje *private-key* koji pripada računu na kojem se transakcija izvršava.



Slika 2. Svaki račun ima svoj blockchain, koji sadrži saldo računa. Blok 0 mora biti prva transakcija. (Section IV-B)

C. Knjiga salda

Knjiga salda je globalni set računa gdje svaki račun ima svoju tablicu transakcija (Figure 2). Ovo je ključna komponenta dizajna kojom mijenjamo *run-time* dogovor sa *design-time* dogovorom; svi sudionici, provjerom potpisa, slažu se da samo vlasnik može modificirati svoju tablicu transakcija. To pretvara naizgled dijeljenu strukturu podataka, distribuiranu knjigu salda, u set ne dijeljenih knjiga salda.

D. Nod

Nod je računalni program koji se izvršava na računalu, usklađen je sa RaiBlocks protokolom i sudjeluje u RaiBlocks mreži. Softver kontrolira glavnu knjigu i sve račune koje taj nod kontrolira, ako postoje. Nod može spremiti kompletnu knjigu salda, ili samo optimiziranu povijest koja sadrži samo posljednjih nekoliko blokova svakog računa. Kod podešavanja novog noda preporučeno je verificirati kompletnu povijest i optimizirati ju lokalno.

IV. Pregled Sustava

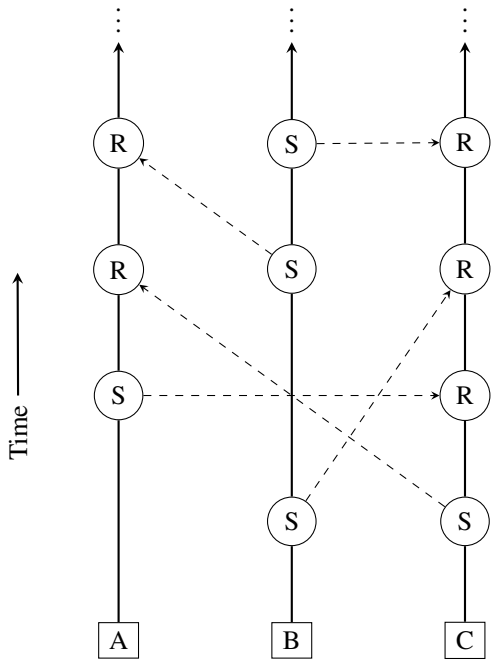
Za razliku od mnogih ostalih kriptovaluta koje koriste *blockchain*, RaiBlocks koristi *block-lattice* podatkovnu strukturu. Svaki račun ima svoj vlastiti *blockchain* koji je ekvivalent povijesti transakcija/salda tog računa (Figure 2). Svaki račun može mijenjati samo njegov vlasnik; to omogućuje da račun bude mijenjan instantno i asinkrono u odnosu na *block-lattice*, što rezultira brzim transakcijama. RaiBlocks protokol je izuzetno lagan; svaka transakcija stane u minimalni UDP paket koji je se šalje internetom. Hardverski zahtjevi za nodove su također minimalni, s obzirom da nodovi za većinu transakcija moraju samo voditi evidenciju i slati blokove transakcija (Figure 1).

Sistem je iniciran sa *prvotnim računom* koji sadrži *prvotni saldo*. Prvotni saldo je fiksan i nikada se nemože povećati. Prvotni saldo je podijeljen i poslan ostalim računima *send transakcijom* koja je registrirana u prvotnom računu. Zbroj svih salda svi računa nikada neće biti veći od inicijalnog prvotnog računa, što sustavu daje granicu i onemogućuje mu da ju povisi.

Ovaj odlomak objasniti će kako su različiti tipovi transakcija konstruirani i odaslani prema mreži.

A. Transakcija

Slanje sredstava sa jednoga računa na drugi zahtjeva dvije transakcije: *send* koja oduzima sredstva sa pošiljateljevog



Slika 3. Vizualizacija *block-lattice*. Svaki transfer novca zahtjeva *send* blok (S) i *receive* blok (R), svaki potpisan privatnim ključem njegova vlasnika (A,B,C)

računa i *receive* koja dodaje sredstva na primateljev račun. (Figure 3).

Slanje sredstava koristeći odvojene transakcije na računima pošiljatelja i primatelja ima nekoliko prednosti:

1) Nizanje dolaznih transakcija koje su nasljedno asinkrone.
2) Održavanje transakcija malima kako bi stale u UDP paket.
3) Olakšavanje optimiziranja glavne knjige, minimizirajući njenu veličinu
4) Odvajanje potvrđenih transakcija od nepotvrđenih

Kada više računa sredstva šalje prema istom računu to je asinkrona operacija: latencija mreže i računi koji šalju sredstva ne moraju nužno biti u komunikaciji jedno sa drugim, što znači da ne postoji univerzalni dogovoreni način da znamo koja transakcija se dogodila prva. S obzirom da je dodavanje transakcija asocijativno, redoslijed kojim su dolazne transakcije poredane nije bitan, i potreban je globalni dogovor. Ovo je ključna komponenta dizajna koja pretvara *run-time* dogovor u *design-time dogovor*. Račun koji prima sredstva ima kontrolu da odluči koja je transakcija stigla prva i to čini potpisujući dolazne blokove.

Ako račun želi napraviti veliku transakciju koja je primljena kao niz nekoliko manjih transakcija, to želimo reprezentirati na način koji stane u UDP paket. Kada račun primatelja prima transakcije, konstantno pamti saldo svoga računa, tako da u svakom trenutku ima mogućnost poslati bilo koji saldo u transakciji fiksne veličine. To je različito od ulazno/izlaznih tranakcijskog modela korištenog u Bitcoin i ostalim kriptovalutama.

Neki nodovi nisu zainteresirani za trošenje resursa i spremanje kompletne povijesti transakcija; zainteresirani su samo za trenutno stanje računa. Kada račun radi transakciju, on enkodira akumulirani saldo i prema tome nodovi trebaju pratiti samo posljednji blok, što im dozvoljava da zanemare povijest transakcija, dok istovremeo zadržavaju ispravnost.

Čak i sa fokusom na *design-time* suglasnost, postoji mogućnost kašnjena pri validiranju transakcija zbog identificiranja i obrade malicioznih sudoionika u mreži. S obzirom da se dogovori u RaiBlocks mreži riješavaju brzo, na razini milisekunda do sekunde, korisniku možemo prezentirati dvije slične kategorije dolaznih transakcija: riješene i neriješene transakcije. Riješene transakcije su transakcije za koje je račun primatelja generirao *receive* blok. Neriješene transacije još nisu dodane u ukupan primateljev saldo. To je zamjena za puno kompleksniji i nepoznatiji način konfirmacije transakcija u ostalim kriptovalutama.

B. Stvaranje Računa

Da bismo stvorili račun, potrebno je poslati *open* transakciju (Figure 4). Ta transakcija je uvijek prva transakcija na svakom računu i može biti stvorena prilikom prvog primanja sredstava. Polje *account* sprema *public-key* (adresu) izvedenu iz *private-key* ključa, koji se koristi za potpisivanje. Polje *source* sadrži haš transakcije koja je poslala sredstva. Pri stvaranju računa potrebno je izabrati predstavnika, koji će glasati u ime stvorenog računa; predstavnik naknadno može biti promjenjen (Section IV-F). Račun može sam sebe proglasiti svojim predstavnikom.

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

Slika 4. Anatomija stvaranja transakcije

C. Saldo računa

Saldo računa je spremljen unutar same knjige salda. Umjesto zapisivanja količine sredstava neke transakcije, za verifikaciju (Section IV-I) je potrebno provjeriti razliku između salda *send* bloka i salda prethodnog bloka. Račun koji prima transakciju tada može povisiti prethodni saldo u finalni saldo koji je dat u novom dolazećem bloku. To je učinjeno da bi se povećala brzina procesuiranja pri skidanju velike količine blokova. Kada zatražimo povijest računa, salda su već poznata.

D. Slanje sa Računa

Da bismo poslali transakciju, adresa mora imati postojeći otvoreni blok, i prema tome saldo (Figure 5). Polje *previous* sadrži haš prethodnog bloka tog računa. Polje *destination* sadrži račun na koji šaljemo sredstva. Odaslani blok je nepromjenjiv jednom kada je potvrđen. Jednom kada je poslan

u mrežu, sredstva su oduzeta sa pošiljateljevog računa i čekaju u polju *pending* dok primatelj ne potpiše blok da bi prihvatio poslana sredstva. Sredstva koja su u tijeku, ne treba gledati kao na sredstva koja čekaju odobrenje, jer su potrošena sa pošiljateljevog računa i on ne može poništiti transakciju.

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

Slika 5. Anatomija *send* transakcije

E. Primanje Transakcije

Da bismo završili transakciju, primatelj mora stvoriti blok na svom vlastitom računu (Figure 6). *Source* polje pokazuje na haš pripadajuće poslane transakcije. Kad je taj blok kreiran i odaslan u mrežu, saldo računa je osvježen i sredstva su službeno primljena na njegov račun.

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

Slika 6. Anatomija primanja transakcije

F. Dodijeljivanje Predstavnika

Mogućnost da vlasnici računa mogu izabrati predstavnika, koji će glasati za njih je moćan decentralizacijski alat koji ne postoji u ostalim Proof of Work i Proof of Stake protokolima. U konvencionalnom PoS sustavu, nod vlasnika računa mora biti aktivan da bi sudjelovao u glasanju. To je nepraktično za mnoge korisnike; mogućnost da korisnik ima predstavnika koji može glasati za njega rješava taj problem. Vlasnici računa imaju mogućnost promjene predstavnika u bilo kojem trenutku. *Change* transakcija mjenja predstavnika računa, tako da oduzima moć glasanja sa starog predstavnika i daje ju novom predstavniku (Figure 7). Za tu transakciju se ne troše sredstva, a predstavnik nema mogućnost trošenja sredstava računa koji ga je izabrao kao predstavnika.

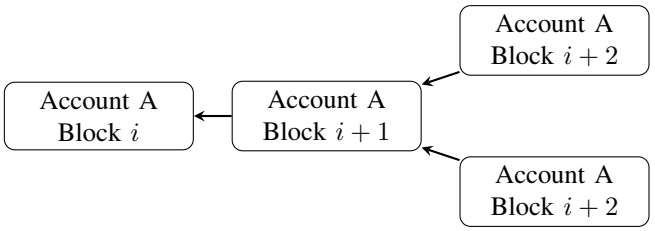
G. Grananje i Glasanje

Do grananja dolazi kada j potpiše blokove $b_1, b_2, \ldots, b_j$ i potražuje pravo na isti blok kao njegov predhodnik (Figure 8). Ti blokovi stvaraju konflikt u statusu računa i moraju biti razriješeni. Samo vlasnik računa ima mogućnost potpisivanja

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

Slika 7. Anatomija mjenjanja transakcije

blokova u svom računu, pa grananje može biti rezultat lošeg programiranja ili malicioznog pokušaja (double-spend) od strane vlasnika računa.



Slika 8. A fork occurs when two (or more) signed blocks reference the same previous block. Older blocks are on the left; newer blocks are on the right

U slučaju detekcije, predstavnik će pokrenuti glasanje koje pokazuje na blok $\hat{b}_i$ u svojoj knjizi salda i odaslati ga u mrežu. Težina njegovog glasa, w_i, je zbroj svih salda svih računa koji su ga izabrali kao predstavnika. Nod ce tada promatrati dolazeće glasove ostalih M aktivnih predstavnika i čuvati kumulativni zbroj 4 glasačka perioda, u ukupnom trajanju 1 minute, i potvrditi pobjednički blok (Formula 1).

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

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

Najpopularniji blok b^* će imati većinu glasova i biti će zadržan u nodovoj knjizi salda (Formula 2). Blok koji izgubi glasanje je zanemaren. Ako predstavnik zamjeni blok u svojoj glavnoj knjizi, to će stvoriti novo glasanje sa većim rednim brojem koje ce biti odaslano u mrežu. To je **jedini** scenarij u kojem predstavnici glasuju.

U nekim okolnostima, kratke smetnje u mreži mogu prouzročiti da odaslani blok nije prihvaćen od strane svih sudionika u mreži. Svaki sljedeći blok na tom računu, kojega sudionici nisu vidjeli, će biti zanemaren kao nevažeći. Ponovno odašiljanje bloka će prihvatiti preostali sudionici a naknadni blokovi će biti prihvaćeni automatski. Čak i kada se pojavi grananje ili nepostojeći blok, samo računi upleteni u transakciju su pogođeni; ostatak mreže nastavlja sa procesuiranjem transakcija za ostale račune.

H. Proof of Work

Sve četiri tipa transakcije imaju *work* polje koje mora biti ispravno popunjeno. *Work* polje omogućuje kreatoru transakcije

da izračuna *dokaz* na način da je haš *dokaza* u vezi sa *previous* poljem u *receive/send/change* transakciji ili da je polje računa u otvorenoj transakciji ispod određene granične vrijednosti. Za razliku od Bitcoin-a, PoW u RaiBlocks je jednostavno *anti-spam* alat, sličan Hashcash-u, i može biti izračunat za nekoliko sekundi [9]. Kada je transakcija poslana, PoW za sljedeći blok može biti unaprijed izračunat, s obzirom da je prethodni blok poznat; na taj način transakcije djeluju instantne krajnjem korisniku, sve dok je vrijeme između transakcija veće od vremena potrebnog za računanje PoW.

I. Verifikacija Transakcije

Da bi se blok smatrao validnim, mora imati sljedeća svojstva:

1) Blok ne smije postojati u knjizi salda (dupla transakcije).
2) Mora biti potpisan od strane vlasnika računa.
3) Prethodni blok je čelni blok računa. Ako postoji, ali nije glavni, riječ je o grananju.
4) Račun mora imati otvoreni blok.
5) Izračunati haš je u unutar razine PoW granica.

U slučaju *receive* bloka, provjerava se dali haš izvornog bloka čeka, što znači da još nije preuzet. Ako se radi o *send* bloku, saldo mora biti manji nego prethodni saldo.

V. Mogućnosti Napada

RaiBlocks, kao i sve decentralizirane kriptovalute, može biti napadnut od strane malicioznih korisnika, sa motivacijom financijske dobiti ili uništenja sustava. U ovom poglavlju istaknut ćemo nekoliko mogućih scenarija napada, posljedice takvih napada, i koje preventivne mjere za njihovo spriječavanje poduzima RaiBlock's protokol.

A. Block Gap Synchronization

U poglavlju IV-G, diskutirali smo o scenariju u kojem blok nije primjereno poslan, što će uzrokovati da mreža zanemari sljedeće blokove. Ako nod promatra blok koji nema referencu prema prošlom bloku, ima dvije opcije:

1) Ignorirati blok jer se može raditi o malicioznom bloku.
2) Zatražiti resinkroniziranje sa drugim nodom.

U slučaju resinkroniziranja, TPC veza mora biti formirana samopokrenutim nodom koji će omogućiti povećanu količinu prometa koja je potrebna za resinkroniziranje. No, ako je blok zaista bio loš blok, tada je resinkroniziranje bilo nepotrebno i stvaramo nepotreban promet na mreži. To je *Network Amplification Attack* koji rezultira napadom uskraćivanjem resursa (DoS napadom).

Da bismo izbjegli nepotrebno resinkroniziranje, nodovi će čekati dok se promotri određen broj glasova za potencijalno maliciozni blok prije iniciranja resinkroniziranja noda. Ako blok ne primi dovoljno glasova, možemo predpostaviti da je nod smeće.

B. Transaction Flooding

Maliciozni entitet bi mogao poslati puno nepotrebnih, ali validnih transakcija između računa koje kontrolira, sa namjerom da zaguše mrežu. Sa obzirom na besplatne transakcije, napad može trajati neograničeno vrijeme. Svejedno, PoW potreban za svaku transakciju ograničava broj transakcija koje maliciozni entiteti mogu generirati bez značajnog invesitranja u računalne resurse. Čak i u slučaju takvog napada sa namjerom da zaguši knjigu salda, nodovi koji ne prate povijest svih transakcija imaju mogućnost optimiziranja i uklanjanja starih transakcija sa svoga računa; to u slučaju takvog napada onemogućuje korištenje diskovnog prostora korisnika.

C. Sybil Attack

Entitet bi mogao stvoriti tisuće Raiblock nodova na jednom računalu: iako, pošto je sustav glasovanja temeljen na težini saldo računa, dodavanje dodatnih nodova u mrežu napadaču neće osigurati dodatne glasove. Prema tome ne postoji prednost koju bi mogao dobiti koristeći *Sybil attack*.

D. Penny-Spend Attack

Penny-spend napad je napad u kojemu napadač šalje beskonačno male količine salda velikom broju računa sa namjerom da troši diskovni prostor nodova. Objavu blokova ograničava PoW, pa to do određene granice ograničava mogućnost kreiranja računa i stvaranja transakcija. Nodovi koji ne prate kompletnu povijest transakcija mogu optimizirati račune do razine gdje statističkim mjerilima možemo utvrditi račune koji nisu validni. Naposljetku, RaiBlocks je podešen da koristi minimalnu količinu prostora, pa je prostor potreban da se spremi dodatni račun, proporcionalan veličini open block + indexing $= 96\text{B} + 32\text{B} = 128\text{B}$. Iz toga proizlazi da 1GB može spremiti 8 milijuna *penny-spend* računa. Ako nodovi žele optimizirati još agresivnije, mogu kalkulirati distribuciju temeljenu na pristupnoj frekvenciji i premjestiti rijeđe korištene račune, na sporiji diskovni prostor.

E. Precomputed PoW Attack

S obzirom da je vlasnik računa jedini entitet koji dodaje blokove u svoj račun, budući blokovi mogu biti unaprijed stvoreni, zajedno sa njihovim PoW, prije nego se pošalju u mrežu. Ovdje napadač generira mnoštvo budućih blokova, svaki minimalne vrijednosti, kroz duži period vremena. U određenom trenutku, napadač izvršava *Denial of Service* (DoS) napad zagušujući mrežu sa mnoštvom validnih transakcija, koje će ostali nodovi procesirati i proslijediti u najbržem roku. Ovo je napredna verzija *Transaction Flooding* napada opisana u Poglavlju V-B. Takav napad bi djelovao samo kratko, ali bi mogao biti korišten zajedno sa ostalim napadima, kao $>50\%$ Attack (Section V-F) da bi bio efikasniji. Ograničenje broja transakcija i ostale tehnike se trenutno istražuju, da bi se se onemogućili takvi napadi.

F. >50% Attack

Mjera suglasnosti za RaiBlocks je sustav glasanja temeljen na težini glasa. Ako napadač uspije ostvariti više od 50% glasačke snage, može poljuljati sustav glasanja i prouzročiti pad sustava. Napadač može smanjiti količinu salda koji mora posjedovati, onemogućujući validnim nodovima glasanje DoS napadom. RaiBlocks preduzima sljedeće mjere da bi onemogućio takve napade:

1) Primarna obrana od ovoga napada je ta što je težina glasa vezana za investiciju u sustav. Vlasnik računa je prinuđen održavati mrežu zdravom da bi zaštitio svoju investiciju. Pokušaj da našteti mreži bio bi destruktivan za cijelu mrežu, pa tako i za njegovu investiciju.
2) Cijena ovog napada je proporcionalna tržišnoj vrijednosti RaiBlocks-a. U PoW sustavu, moguće je osmisliti tehnologiju koja daje neproporcionalnu kontrolu u odnosu na financijsku investiciju, i ako je napad uspješan, ta tehnologija može biti ponovo korištena. Kod RailBlocks-a cijena napada na sustav povećava se sa samim sustavom i ako je napad uspješan investicija u napad se nemože vratiti.
3) Da bi se održao maksimalan broj prisutnih glasača, sljedeća linija obrane je glasanje predstavnika. Vlasnici računa koji nisu u mogućnosti pouzdano sudjelovati u glasanju, mogu imenovati predstavnika koji može glasati za njih. Povećanjem broja i raznolikosti predstavnika povećava se otpornost mreže.
4) Grananja u RaiBlocks-u nikad nisu slučajna, nodovi imaju mogućnost donošenja odluke kako pristupati razgranatim blokovima. Jedini slučaj u kojemu su računi korisnika koji nije napadač ranjivi, je kada primaju sredstva sa računa napadača. Korisnik koji se želi osigurati, može čekati manje ili više vremena prije nego primi sredstva sa računa koji je generirao grananje, ili odlučiti nikad ne primiti sredstva. Primatelji tagođer mogu generirati odvojene račune za primanje novca sa sumnjivih računa i na taj način izolirati ostale račune.
5) Posljednja linija obrane koja još nije implementirana je *block cementing*. Raiblocks je otišao jako daleko u tome da razriješi grananje blokova brzo kroz glasanje. Nodovi mogu biti konfigurirani tako da *cementiraju* blokove, što bi im onemogućilo izmjene nakon određenog vremena. Mreža je dovoljno sigurna kroz fokus na brzoj uspostavi transakcija koja onemogućuje grananja.

Nešto sofisticiranija verzija $> 50\%$ napada je objašnjena na skici9. *Offline* je postotak predstavnika koji su izabrani, ali nisu aktivni u glasanju. *Stake* je količina investicije sa kojom napadač glasa. *Active* je predstavnik koji je aktivan i glasa prema protokolu. Napadač može pomaknuti količinu uloga koju mogu izgubiti tako da ostale predstavnike izbaci sa mreže koristeći DoS napad. Ako taj napad bude ustaljen, predstavnici neće biti sinkronizirani a to je demonstrirano sa *Unsync*. Naposljetku, napadač može ostvariti kratkoročne nalete u relativnoj snazi glasanja tako da prebacuje DoS napade na novi set predstavnika, dok se prethodno napadnuti predstavnici sinkroniziraju, što je demonstrirano sa *Attack*.

Ako je napadač sposoban prouzročiti situaciju u kojoj je

Offline	Unsync	**Attack**	Active	Stake

Slika 9. Potencijalni dogovor glasanja koji smanjuje 51% napad

Ulog >Aktivnosti kombinacijom navedenih napada, mogao bi uspješno preokrenuti glasove u knjizi salda po cijenu svojeg uloga. Možemo procjeniti koliko bi takav napad koštao razmatrajući tržišnu vrijednost ostalih sustava. Ako procjenimo da je 33% predstavnika offline ili napadnuto DoS napadom, napadač bi trebao kupiti 33% tržišne vrijednosti da bi mogao izvršiti takav napad.

G. Bootstrap Poisoning

Što duže je napadač sposoban držati stari privatni ključ sa saldom, veća je vjerovatnost da saldo koji je posjedovao u to vrijeme neće imati aktivnog predstavnika zato što su se njegova salda i predstavnici preselili na druge račune. To znači da ako je nod samopokrenut na staru reprezentaciju mreže gdje napadač drži kvorum glasačkog uloga u odnosu na predstavnika u tom trenutku, mogli bi zaljuljati odluke glasanja za taj nod. Ako bi novi korisnik želio interakciju sa bilo kim drugim osim sa nodom napadačem, sve njegove transakcije bi bile odbijene iz razloga što sadržavaju različitu glavu bloka. Zaključak je da nodovi mogu trošiti vrijeme novih nodova u mreži, davajući im netočne informacije. Da bismo to spriječili, nodovi mogu biti upareni sa inicijalnom bazom računa i provjereno točnih blokova; to je zamjena za skidanje kompletne baze, sve do genesis bloka. Što je baza bliža tome da bude aktualna, veća je vjerovatnost uspješne obrane od ovog napada. Na kraju, taj napad nije ništa gori od slanja netočnih podataka nodovima koji se samopokreću, s obzirom da oni nemogu komunicirati sa nikim tko ima suvremenu bazu podataka.

VI. Implementacija

Trenutno, opisana implementacija je izvedena u C++ programskom jeziku i izdaje nova izdanja od 2014. godine na Github-u [10].

A. Mogućnosti Dizajna

RaiBlocks arhitektura prati standarde opisane u ovom dokumentu. Sve dodatne implementacije koje će biti opisane su ovdje.

1) Algoritam Potpisa: RaiBlocks koristi modificirani ED25519 algoritam eliptične krivulje sa Blake2b haširanjem za sve digitalne potpise [11]. ED25519 je izabran zbog brzine potpisivanja, brzine verifikacije i visoke razine sigurnosti.

2) Algoritam Haširanja: S obzirom da se algoritam haširanja koristi samo za kontrolu gušenja mreže, izbor algoritma je manje važan u odnosu na kriptovalute koje su bazirane na rudarenju. Naša implementacija koristi Blake2b za prožimanje sadržaja blokova [12].

3) Funkcija za derivaciju ključa: U određenom novčaniku, ključevi su kriptirani koristeći lozinku koji ulazi u funkciju za derivaciju ključa radi zaštite od ASIC napada. Trenutni pobjednik je Argon2 [13] jedinog javnog nadmetanja sa ciljem kreiranja otporne funkcije za deriviranje ključa.

4) Interval Bloka: S obzirom da svaki račun ima svoj blockchain, osvježavanja se mogu izvršavati asinkrono u odnosu na ostatak mreže. Prema tome ne postoje intervali bloka i transakcije mogu biti objavljene instantno.

5) UDP Podatkovni Protokol: Naš sustav je dizajniran da radi beskonačno koristeći minimalne količine računalnih resursa. Sve poruke u sustavu su dizajnirane da stanu u jedan UDP paket. To olakšava komunikaciju i sudjelovanje u mreži bez potrebe za uspostavljanjem kratkoročne TCP veze. TCP se koristi samo za nove korisnike kada žele masovno samopokrenuti blockchain.

Nodovi mogu biti sigurni da su njihove poruke primljene u mrežu promatrajući komunikaciju transakcija drugih nodova jer bi trebale primiti nekoliko kopija natrag.

B. IPv6 and Multicast

Gradeći povrh UDP u budućnosti omogućuje implementaciju IPv6 odašiljanje kao zamjenu za tradicionalo transakcijsko gušenje i odašiljanje glasova. To će smanjiti konzumaciju prometa na mreži i dati veću fleksibilnost nodovima.

C. Performanse

U trenutku pisanja, RaiBlocks mreža je procesuirala 4.2 milijuna transakcija, što je dovelo do baze veličine 1.7GB. Brzina transakcije se mjeri u sekundama. Trenutna implementacija koja se izvršava na običnom SSD-u, omogućava 10,000 transakcija u sekundi i jedino usko grlo je I/O tvrdoga diska.

VII. Korištenje Resursa

Ovo je pregled korištenja resursa Raiblocks noda. Dodatno, govorimo o idejama za smanjenje korištenja resursa u određenim slučajevima. Ograničeni nodovi se tipično zovu lagani nodovi, optimizirani ili jednostavno, jednostavni platni verifikator (JPV) nodovi.

A. Mreža

Količina mrežne aktivnosti, ovisi o tome koliko mreža doprinosi zdravlju mreže.

1) Predstavnik: Predstavnik nod zahtjeva maksimalnu količinu mrežnih resursa jer promatra promet glasanja ostalih predstavnika i objavljuje vlastite glasove.

2) Nepovjerljiv: Nepovjerljivi nod je sličan predstavniku, ali je samo promatrač, ne posjeduje privatni ključ predstavnika i samostalno ne objavljuje glasove.

3) Povjerljiv: Povjerljivi nod promatra promet glasova jednog predstavnika kojemu vjeruje da će ispravno donjeti odluku. To smanjuje količinu prometa jer predstavnik nema potrebe komunicirati sa nodom.

4) Lagan: Lagani nod je istovremeno i povjerljivi nod koji promatra promet na računima za koje je zainteresiran minimalno opterećujući mrežu.

5) Samopokrenuti: Samopokrenuti nod poslužuje dio, ili cijelu knjigu salda nodovima koji se spajaju na mrežu. To se radi koristeći TCP vezu umjesto UDP jer su potrebne velike količine podataka što zahtjeva naprednije kontrole prometa.

B. Kapacitet Diska

U ovisnosti od zahtjeva korisnika, različite konfiguracije nodova imaju različite zahtjeve za diskovnim prostorom.

1) Povijesni: Nod zainteresiran za čuvanje kompletne povijesti svih transakcija zahtjeva najveću potrebu za diskovnim prostorom.

2) Trenutni: Zbog dizajna koji čuva akumulirana salda sa blokovima, nodovi moraju čuvati samo informaciju o posljednjoj glavi bloka za svaki račun ako žele sudjelovati u glasanju. Ako nod nije zainteresiran za čuvanje potpune povijesti, može čuvati samo glavu bloka.

3) Lagani: Lagani nod ne čuva nikakve informacije o knjizi salda i samo sudjeluje u mreži na način da promatra aktivnosti računa za koje je zainteresiran ili opcionalno stvara nove transakcije sa privatnim ključem koji posjeduje.

C. Procesor

1) Generiranje Transakcije: Nod zainteresiran za kreiranje transakcije mora proizvesti *Proof of Woork* da bi prošao kroz Raiblocks-ov prigušni mehanizam. Performanse različitog hardware-a su prikazane u Appendix A.

2) Predstavnik: Predstavnik mora verificirati potpise za blokove, glasove i također proizvesti svoje vlastite potpise da bi sudjelovao u odlukama. Količina procesorske snage potrebne predstavniku je osjetno manja od one potrebne za generiranje transakcije i može raditi i na jednojezgrenom procesoru suvremenog računala.

3) Promatrač: Promatrač nod ne generira svoje glasove. S obzirom da je generiranje potpisa minimalno, procesorski zahtjevi su gotovo identični kao i za nod predstavnika.

VIII. Zaključak

U ovom dokumentu, prezentiramo vam kriptovalutu koja je sigurna, besplatnih transakcija, koja koristi neuobičajenu *block-lattice* strukturu i delegirano "Proof of Stake" glasanje. Mreža zahtjeva minimalne resurse, ne treba skupi hardver za rudarenje, i može procesuirati velik broj transakcija. Sve ovo je postignuto korištenjem individualnih blockchain-ova za svaki korisnički račun, eliminirajući probleme pristupa i ne efikasnosti globalnih baza podataka. Identificirali smo moguće vrste napada na sustav i prezentirali na koji način je Railblocks otporan na te napade.

Dodatak A
PoW hardver testovi

Kao što je spomenuto, PoW u RaiBlocks služi smanjenju nepoželjnog spam-a. Naša implementacija noda pruža mogućnost ubrzanja korištenjem OpenCL i kompatibilnih GPU-a. Tablica I pruža stvarne testove i usporedbe različitog hardvera. Trenutno PoW granica je fiksna, a promjenjiva granica može biti implementirana sa napretkom procesorske snage prosječnog računala.

Zahvale

Zahvaljujemo Brian Pugh-u na stvaranju i formatiranje ovog dokumenta.

Tablica I
HARDVERSKE POW PERFORMANSE

Uređaj	Transakcije u sekundi
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

LITERATURA

[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