Nano: Et gebyrfrit, distribueret kryptovalutanetværk

Colin LeMahieu
clemahieu@nano.co

***Resumé*—høj efterspørgsel og begrænset skalering har for nyligt forøget den gennemsnitlige transaktionstid og gebyrerne i populære kryptovalutaer, hvilket giver en utilfredsstillende oplevelse. Vi introducerer hermed Nano: En kryptovaluta med en ny blokgitterarkitektur, hvor hver konto har dens egen blockchain, som byder på nærmest øjeblikkelig transaktionshastighed og ubegrænset skalering. Hver bruger har deres egen blockchain, som tillader dem at opdatere asynkront med resten af netværket, hvilket resulterer i hurtige transaktioner med minimale omkostninger. Transaktioner kontrollerer ens kontobalance, i stedet for transaktionens mængde og tillader aggressiv databasebeskæring, uden at gå på kompromis med sikkerhed. Til dato har Nanos netværk behandlet 4.2 millioner transaktioner, med en ubeskåret ledger-størrelse på kun 1.7GB. Nanos gebyrfrie, splitsekund-transaktioner formår at udkåre den som lederen af kryptovaluta for handelstransaktioner.**

***Indeksbegreber*—kryptovaluta, blockchain, Nano, distribueret ledger, digital, transaktioner**

I. Introduktion

SIDEN implementeringen af Bitcoin i 2009 har der været et voksende skred væk fra traditionelle, regeringsstøttede valutaer og finansielle systemer til moderne betalingssystemer baseret på kryptografi, som tilbyder muligheden for at opbevare og overføre midler i en decentral og tryg manér [1]. For at kunne fungere effektivt må en valuta være let at overføre, uigenkaldelig og have begrænsede (eller ingen) gebyrer. De forøgede transaktionstider, store gebyrer og den tvivlsomme netværksskalering har stillet spørgsmåltegn ved Bitcoins praktiske egenskaber som en hverdagsvaluta. I dette whitepaper introducerer vi Nano: En lavlatens kryptovaluta bygget på en innovativ blokgitterdatastruktur, der tilbyder ubegrænset skalering og ingen transaktionsgebyrer. Nanos design er en simpel protokol, hvis eneste formål er at fungere som en højtydende kryptovaluta. Nano-protokollen kan køre på hardware med lavt strømforbrug, hvilket tillader den at være en praktisk, decentraliseret kryptovaluta til hverdagsbrug.

Kryptovalutastatistik, viderebragt i dette whitepaper, er præcise fra og med publiceringsdatoen.

II. Baggrund

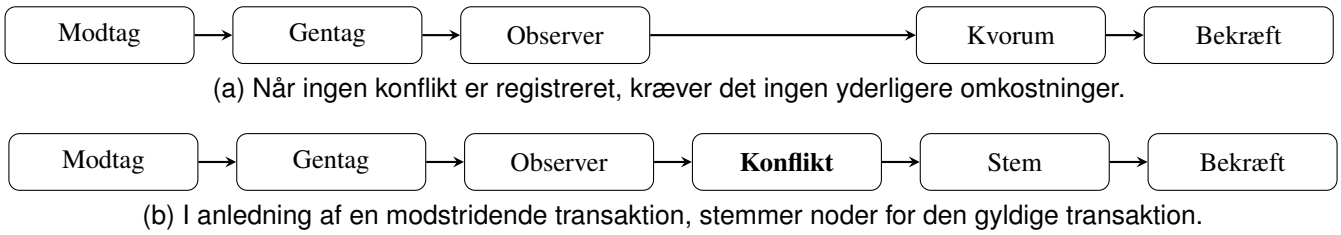
I 2008 publicerede et anonymt individ under pseudonymet Satoshi et whitepaper, som beskrev verdens første decentraliseret kryptovaluta ved navnet Bitcoin [1]. En essentiel innovation medført af Bitcoin var blockchainen: En offentlig, uforanderlig og decentraliseret kryptovalutadatastruktur, hvilket anvendes som en *ledger* (regnskabsbog) for valutaens transaktioner. Da Bitcoin modnede, opstod der desværre problemer, og Bitcoin blev uoverkommelig for mange applikationer:

1) Ringe skalering: Hver blok i blockchainen kan opbevare en begrænset mængde af data, hvilket betyder, at systemet kun kan behandle en begrænset mængde transaktioner hvert sekund, hvilket gør pladser i blokken til en handelsvare. I øjeblikket er medianen for transaktionsafgifter $10.38 [2].
2) Høj latens: Den gennemsnitlige bekræftelsestid er 164 minutter [3].
3) Ineffektivt strømforbrug: Bitcoins netværk forbruger et estimat på 27.28TWh hvert år, hvilket er et gennemsnitligt forbrug på 260KWh per transaktion. [4].

Bitcoin, og andre kryptovalutaer, fungerer vha. konsensus på deres globale ledger, for at verificere legitime transaktioner samt modstå skadelige aktører. Bitcoin opnår konsensus via en økonomisk foranstaltning, som hedder *Proof of Work* (PoW). I et PoW-system konkurrerer deltagere i at beregne et nummer, som kaldes en *nonce*, sådan at hele blokkens hash er i en bestemt rækkevidde. Rækkevidden er omvendt proportional med den samlede regnekraft i hele Bitcoins netværk for at kunne være i stand til at vedligeholde et vedvarende gennemsnit i tid, som det tager for at finde en gyldig nonce. Finderen af en gyldig nonce tillades derefter at tilføje blokken til deres blockchain. Derfor spiller dem, som opbruger flere databehandlingsresurser til at beregne en nonce, en større rolle i blockchainens tilstand. PoW sørger for modstand mod et Sybil-angreb, hvor en enhed opfører sig som adskillige enheder for at få ekstra magt i et decentraliseret system, og reducerer også betydeligt race-vilkårene, som i sagens natur eksisterer, mens man får adgang til en global datastruktur.

En alternativ konsensusprotokol ved navn *Proof of Stake* (PoS) blev oprindeligt introduceret af Peercoin, i 2012 [5]. I et PoS-system stemmer deltagerne med en vægt tilsvarende mængden af værdi, de er i besiddelse af, i den givne kryptovaluta. Med denne organisering får dem, som har en højere finansiel investering, tildelt mere magt, samt i sagens natur tilskyndelsen til at opretholde ærligheden af systemet, eller risikere tab af deres investering. PoS fjerner den ineffektive regnekraftskonkurrence, da det kun kræver let software, der køres på hardware med lavt strømforbrug.

Nanos oprindelige whitepaper og første implementering af beta blev publiceret december 2014, hvilket gør den til en af de første DAG-baserede (Directed Acrylic Graph) kryptovalutaer [6]. Kort efter begyndte andre DAG-kryptovalutaer at blive udviklet, navnlig DagCoin/Byteball og IOTA [7], [8]. Disse DAG-baserede kryptovalutaer knækkede blockchain-formen med deres forbedring af systemydeevne samt sikkerhed. Byteball opnår konsensus ved at stole på en "hovedkæde" bestående af ærlige, velanskrevne og bruger-betroede "vidner", mens IOTA opnår konsensus via det samlede PoW af stablede transaktioner. Nano opnår konsensus via en balancevægtet stemme om modstridende transaktioner. Dette konsensussystem tilbyder hurtigere, mere deterministiske transaktioner,



(a) Når ingen konflikt er registreret, kræver det ingen yderligere omkostninger.

(b) I anledning af en modstridende transaktion, stemmer noder for den gyldige transaktion.

Figur 1. Nano kræver ingen yderligere omkostninger for typiske transaktioner. I anledningen af modstridende transaktioner, stemmer noder om de beholdes.

samt vedligeholdelse af et stærkt, decentraliseret system. Nano fortsætter denne udvikling og har positioneret sig selv som en af de højestydende kryptovalutaer til dato.

III. Nanos Elementer

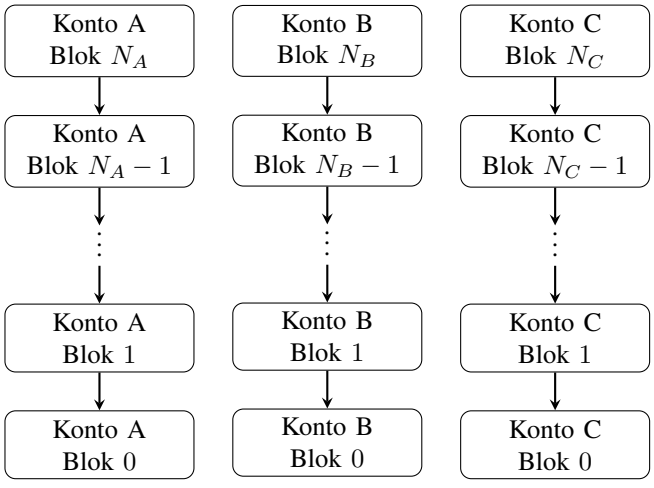
Inden vi beskriver Nanos arkitektur, overordnet set, definerer vi systemets individuelle elementer.

A. Konto

En konto er public-key (offentlignøgle) delen af den digitale key-pair signatur. Ens public-key, også kaldet adressen, er delt med andre deltagere i netværket, mens en private-key (privatnøgle) er skjult. En digitalt signeret pakke af data forsikrer, at indholdet var godkendt af den, som bærer kontoens private-key. En bruger kan kontrollere mange konti, men kun én offentlig adresse må eksistere per konto.

B. Blok/Transaktion

Begrebet "blok" og "transaktion" er ofte anvendt omskifteligt, hvor en blok indeholder en enkel transaktion. Transaktion refererer specifikt til handlingen, mens blok refererer til den digitale indkodning, af transaktionen. Transaktioner er signeret af kontoens private-key, hvor transaktionen bliver udført.



Figur 2. Hver konto har dens egen blockchain, som indeholder kontoens balancehistorik. Blok 0 skal være en åbentransaktion. (Sektion IV-B)

C. Ledger

En ledger er den globale regnskabsbog, hvor hver konto har dens egen transaktionskæde. (Figur 2). Dette er et central element i designet, som hører under kategorien erstattelse af driftidsoverenskomst med designtidsoverenskomst: Alle enes via signaturkontrol om, at kontoejeren er den eneste, som kan modificere sin egen kæde. Dette konverterer en tilsyneladende delt datastruktur (en fordelt ledger) til et sæt af udelte.

D. Node

En *node* er et stykke software, som kører på en computer, der indretter sig efter Nanos protokol og deltager i Nanos netværk. Softwaren styrer ledgeren og konti, noden kan kontrollere, hvis der er nogen. En node kan enten opbevare hele ledgeren eller en beskåret historie, som kun indeholder de sidste par blokke for hver konto blockchain. Når man opsætter en ny node, er det anbefalet at verificere hele historien og beskære lokalt.

IV. Systemoversigt

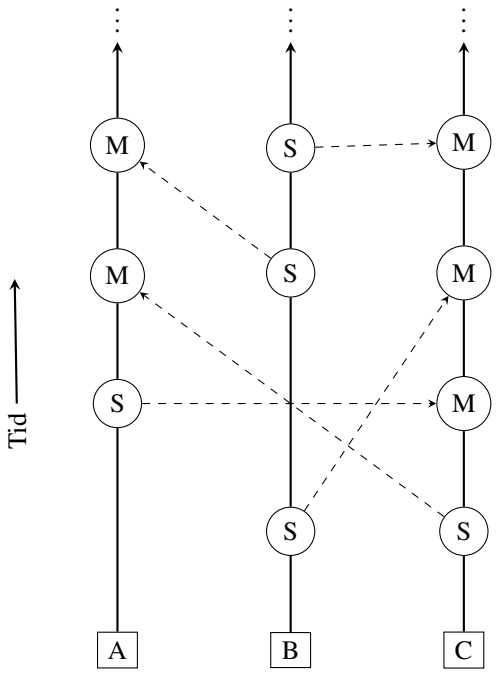
I modsætning til blockchains anvendt i mange andre kryptovalutaer, anvender Nano en *blokgitterstruktur*. Hver konto har dens egen blockchain (kontokæde) ækvivalent til kontoens balance/transaktionshistorik (Figur 2). Hver kontokæde kan kun blive opdateret af kontoens ejer; dette tillader hver kontokæde at blive opdateret øjeblikkeligt og asynkront med resten af blokgitteret, som resulterer i hurtige transaktioner. Nanos protokol er ekstremt let; hver transaktion passer inden for den påkrævede minimum UDP-pakkestørrelse, der kræves for at blive sendt over internettet. Hardwarekrav for noder er også minimale, da noder kun skal bogføre og genudsende blokke for de fleste transaktioner (Figur 1).

Systemet igangsættes med en *genesis-konto*, som indeholder *genesis-balancen*. Genesis-balancen har en fastlagt kvantitet og kan aldrig forøges. Genesis-balancen er opdelt og sendt til andre konti via sendtransaktioner registreret på genesis' kontokæde. Summen af alle konti balancer vil aldrig overgå den oprindelige genesis-balance hvilket giver systemet en øvre grænse for kvantitet og ingen evne til at forøge.

Denne sektion vil gennemgå, hvordan forskellige typer af transaktioner er opbygget og formeret op igennem netværket.

A. Transaktioner

At overføre midler kræver to transaktioner: En *send*, som fratrækker mængden fra send-balance, og en *modtag*, som



Figur 3. Visualisering, af blokgitterets struktur. Hver transaktion, af midler kræver en sendblok (S) og modtagblok (M), som er signeret, af kontokædens ejer.

tilføjer mængden til modtagerens balance (Figur 3).

At overføre mængder som særskilte transaktioner i senderens og modtagerens konti har et par vigtige formål:

1) Sekvensere indkommende overførsler som i sagens natur er asynkron.
2) Bevare transaktioner små for at passe ind i UDP-pakker.
3) Fremme ledger-beskæring ved at minimere data-fodspor.
4) Isolere afklarede transaktioner fra uafklarede.

Mere end én konto, der overfører til den samme destination, er en asynkron operation: Netværkslatens og de sendende konti, som ikke nødvendigvis er i kommunikation med hinanden, betyder at der ikke er nogen forenelig måde til at vide, hvilken transaktion fandt sted først. Eftersom sammentælling er associativ, er den orden, hvert input er sekvenseret efter, irrelevant, derfor mangler vi simpelthen en global overenskomst. Dette er et centralt designelement, der konverterer en driftstidsoverenskomst ind til en designtidsoverenskomst. Den modtagende konto har ret til at beslutte, hvilken transaktion ankom først, og er udtrykt af de indkommende blokkes signerede orden.

Hvis en konto ønsker at udføre en stor transaktion, som var modtaget som et sæt af mange små overførsler, ville vi helst repræsentere dette på en manér, som passer inden i en UDP-pakke. Når en modtagende konto sekvenserer input-overførsler, fører den løbende regnskab med summen af dens kontobalance, således har den altid evnen til at overføre enhver mængde med en transaktion af fastlagt størrelse.

Nogle noder er uinteresseret i at lægge resurser ud til opbevaring af en kontos fulde transaktionshistorik: De er kun interesseret i hver kontos nuværende balance. Når en konto udfører en transaktion, indkoder kontoen dens indsamlede balance, og disse noder behøver kun at holde styr på den seneste blok, hvilket tillader dem at kassere historisk data, mens den vedligeholder korrekthed.

Selv med et fokus på designtidsoverenskomst er der et tidsrum med forsinkelse ved valideringen af transaktioner pga. identifikation og håndtering af ringe aktører i netværket. Da overenskomster bliver opnået hurtigt i Nano, på få millisekunder til sekunder, kan vi præsentere brugeren med to bekendte kategorier af indkommende transaktioner: Afklaret og uafklaret. Afklarede transaktioner er transaktioner, hvor en konto har genereret modtagblokke. Uafklarede transaktioner er endnu ikke integreret ind til modtagerens samlede balance. Dette er en erstatning fra de mere komplekse og ubekendte bekræftelsesmetrikker i andre kryptovalutaer.

B. oprettelse af konto

For at oprette en konto skal du udsende en *åben* transaktion (Figur 4). En åbentransaktion er altid den første transaktion i enhver kontokæde og kan blive skabt ved første modtagelse af midler. Feltet *account* opbevarer ens public-key (adresse), som stammer fra ens private-key, der er anvendt til signering. Feltet *source* indeholder hashet af transaktionen, som sendte midlerne. Ved oprettelse af konto må én repræsentant blive udvalgt til at stemme på dine vegne: Dette kan blive ændret senere (Sektion IV-F). Kontoen kan erklære sig selv som dens egen repræsentant.

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

Figur 4. en åbentransaktions anatomi

C. Kontobalance

Kontoens balance er bogført inde i selve ledgeren. I stedet for at bogføre mængden af transaktionen, kræver verificering (Sektion IV-I) kontrollering af forskellen mellem balancen, på sendeblokken og balancen, af den foregående blok. Den modtagende konto kan dermed forøge den tidligere balance, som målt i den endelige præsenteret i modtagblokken. Dette er blevet gjort for at forbedre proceshastigheden, når der bliver hentet en høj volume af blokke.

D. At sende fra en konto

For at kunne sende fra en adresse må adressen allerede have en eksisterende åbenblok og derfor en balance (Figur 5). Feltet *previous* indeholder hashet af den tidligere blok i kontokæden. Feltet *destination* indeholder kontoen, som midlerne skal sendes til. En sendeblok er uforanderlig ved bekræftelse. Når den er udsendt til netværket, bliver midlerne øjeblikkeligt fratrukket fra balancen på senderens konto og venter som

verserende, indtil det modtagende parti signerer en blok som accept af disse midler. Verserende midler burde ikke anses som afventende bekræftelse, da de er så godt som brugt fra senderens konto, og senderen kan således ikke tilbagekalde transaktionen.

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

Figur 5. En sendtransaktions anatomi

E. At modtage en transaktion

For at gennemføre en transaktion må modtageren af de sendte midler skabe en modtagblok på denne kontokæde (Figur 6). Source-feltet refererer til hashet af den associerede sendtransaktion. Når denne blok er skabt og udsendt, er kontoens balance opdateret, og midlerne er officielt rykket til deres konto.

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

Figur 6. En modtagtransaktions anatomi

F. Udpegelse af repræsentant

Kontoholderens evne til at vælge en repræsentant for at stemme på deres vegne er et kraftfuldt decentraliseringsværktøj, som ingen stærk analog har i PoW- eller PoS-protokoller. I konventionelle PoS-systemer må kontoejerens node køre for at deltage i afstemning. Kontinuerlig drift af en node er upraktisk for mange brugere: At give en repræsentant magten til at stemme på en kontos vegne, gør dette krav nemmere. Kontoholdere har evnen til at overflytte konsensus til en anden konto til enhver tid. En *ændre* transaktion ændrer repræsentanten af en konto ved at fratage stemmevægt og addere til den nye repræsentant (Figur 7). Ingen midler er flyttet i denne transaktion, og repræsentanten har ikke købekraft over kontoens midler.

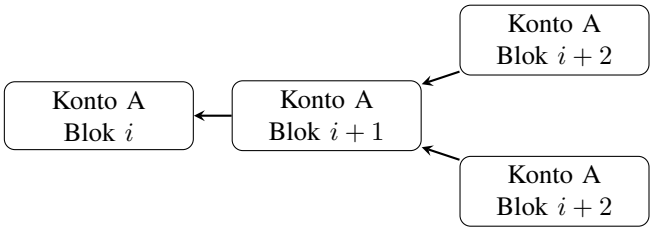
G. Forker og afstemning

En fork (skillevej) forekommer, når j-signerede blokke $b_1, b_2, \ldots, b_j$ hævder den samme blok, som dens forgænger (Figur 8). Disse blokke forårsager et modstridende syn på

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

Figur 7. En ændretransaktions anatomi

en konto og må blive løst. Kun kontoens ejer har evnen til at signere blokke ind til deres egen kontokæde, således må en fork være resultatet af ringe programmering eller skadelig hensigt (double-spend) af kontoens ejer.



Figur 8. En fork forekommer, når to (eller flere) signerede blokke refererer til den samme tidligere blok. Ældre blokke er på den venstre side; nyere blokke er til højre.

Ved opklaring vil en repræsentant skabe en stemme, som henviser til blok $\hat{b}_i$, i dens ledger og udsende det til netværket. Vægten af nodens stemme, w_i, er summen af alle balancerne på alle konti, som har udnævnt den som repræsentant. Noden vil observere indkommende stemmer fra de andre M online repræsentanter, beholde en samlet score for 4 stemmeperioder, 1 minut i alt og bekræfte den vindende blok (Regnestykke 1).

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

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

Den mest populære blok b^* vil have flertallet af stemmerne og blive opholdt i nodens ledger (Regnestykke 2). Blokken/blokkerne, som taber stemmen, vil blive kasseret. Hvis en repræsentant erstatter en blok i dens ledger, vil det skabe en ny stemme med et højere sekvensnummer og udsende den til netværket. Dette er det **eneste** scenarie, hvor repræsentanter stemmer.

I nogle omstændigheder kan korte problemer med tilslutning til netværket forårsage, at en udsendt blok ikke vil blive accepteret af alle ligemænd. Enhver efterfølgende blok på denne konto vil blive ignoreret som ugyldig af ligemænd, som ikke så den oprindelige udsendelse. En genundsendelse af denne blok vil blive accepteret af resterende ligemænd, og efterfølgende blokke vil blive hentet automatisk. Selv hvis en fork eller forsvundet blok forekommer, vil kun kontoerne refereret til i transaktionen blive påvirket; resten af netværket fortsætter med at behandle transaktioner for alle andre kontoer.

H. Proof of Work

Alle fire transaktionstyper har et arbejdsfelt, som skal anvendes aktivt. Arbejdsfeltet tillader skaberen af transaktionen at beregne en nonce, sådan at hashet af noncen, sammenkædet med det tidligere felt i modtag-/send-/ændretransaktioner eller kontofeltet i en åbentransaktion, er under en hvis tærskelværdi. I modsætning til Bitcoin er PoW i Nano simpelthen benyttet som et værktøj til anti-spam, ligesom Hashcash, som kan blive beregnet inden for sekunder [9]. Når en transaktion er sendt, kan PoW'en, for den efterfølgende blok blive forudregnet, da det tidligere blokfelt er kendt; dette vil få transaktioner til at fremstå øjeblikkelig for slutbrugeren, så længe tiden mellem transaktioner er større, end tiden krævet for at beregne PoW'en.

I. Verificering af transaktion

For at en blok skal anses gyldig, må den have følgende attributer:

1) Blokken skal allerede være i ledgeren.
2) Skal blive signeret af kontoens ejer (kopitransaktion).
3) Den tidligere blok er hovedblokken af kontokæden. Hvis den eksisterer, men ikke er hoved, er den således en fork.
4) Kontoen må have en åbenblok.
5) Det beregnede hash møder PoW'ens tærskelkrav.

Hvis det er en modtagblok, så kontrollér om kildeblokkens hash er verserende, hvilket betyder, den endnu ikke er indløst. Hvis det er en sendblok, må balancen være mindre end den tidligere balance.

V. Angrebsvektorer

Nano kan, ligesom alle andre decentraliserede kryptovalutaer, blive angrebet af skadelige partier, i et forsøg på finansiel gevinst eller nedlæggelse af system.

A. Synkronisering af blokmellemrum

I sektion IV-G behandlede vi scenariet, hvor en blok muligvis ikke kunne blive udsendt ordentligt, som forårsager, at netværket ignorerer efterfølgende blokke. Hvis en node observerer en blok, som ikke har refereret til den tidligere blok, har den to valgmuligheder:

1) Ignorer blokken, da den sandsynligvis kunne være en skadelig skrotblok.
2) Efterspørg en resynkronisering med an enden node.

I tilfældet af en resynkronisering skal TCP-forbindelsen dannes med en bootstrapping-node for at facilitere den forøgede mængde trafik, en resynkronisering kræver. Hvis blokken dog faktisk var en ringe blok, ville resynkroniseringen være unødvendig og unødigt forøge trafik på netværket.

For at undgå unødvendig resynkronisering vil noder vente, indtil en vis tærskel af stemmer for en potentielt skadelig blok er observeret, inden de påbegynder forbindelse til en bootstrapnode for at synkronisere. Hvis en blok ikke modtager nok stemmer kan den antages at være skrotdata.

B. Transaktionsoversvømmelse

En skadelig enhed kunne sende mange unødvendige, men gyldige transaktioner mellem konti under dens kontrol, i et forsøg på at mætte netværket. Uden transaktionsafgifter er de i stand til at fortsætte dette angreb på ubestemt tid. PoW'en kravet for hver transaktion begrænser dog raten af transaktioner, som den skadelige enhed kunne generere, uden at investere betydeligt i resurser til databehandling. Selv under sådan et angreb i et forsøg på, at oppuste ledgeren, er noder, som ikke er fuldhistoriske noder, i stand til at beskære gamle transaktioner fra deres kæde: Dette fastgør opbevaringsforbruget fra denne type af angreb for stort set alle brugere.

C. Sybil-angreb

En enhed kunne skabe hundredvis af Nano-noder på en enkel maskine: Da stemmesystemet dog er vægtet baseret på ens kontobalance, vil tilføjelse af ekstra noder ind til netværket ikke gavne hackeren med ekstra stemmer.

D. Penny-Spend angreb

Et penny-spend angreb er, når en hacker bruger uendelig lille mængder til et stort antal konti for at spilde lagringsresurser af noder. Publicering af blokke er ratebegrænset af PoW'en. Således begrænser dette skabelsen af konti og transaktioner til et vist omfang. Noder, som ikke er komplette historiske noder, kan beskære konti under en statistisk metrik, hvor kontoen højst sandsynlig ikke er en gyldig konto. Endelig, Nano er indstillet til brug af den minimale permanente lagringsplads, således er opbevaringspladsen påkrævet for én ekstra konto forholdsmæssig til størrelsen af en åbenblok + indeksering = 96B + 32B = 128B. Dette svarer til, at 1GB er i stand til at opbevare 8 millioner penny-spend konti. Hvis noder ønskede at beskære mere aggressivt, kan de udregne en fordeling baseret på adgangsfrekvens og uddelegere sjældent brugte konti langsommere opbevaring.

E. Forudregnet PoW-angreb

Da ejeren af en konto vil være den eneste enhed, som tilføjer blokke til kontokæden, kan sekventielle blokke blive beregnet fælles med deres PoW, før de bliver udsendt til netværket. Hackeren genererer her et utal af sekventielle blokke, hver med en minimal værdi over en omfattende periode af tid. Ved et bestemt punkt udfører hackeren et Denial of Service (DoS) ved at oversvømme netværket med masser af gyldige transaktioner, som andre noder vil behandle og gengælde så hurtigt som muligt. Dette er en avanceret transaktionsoversvømmelse beskrevet i Sektion V-B. Sådan et angreb ville kun virke kortvarigt, men kunne bruges forbundet med andre angreb, som >50% Angreb (Sektion V-F) for at fremme effektivitet. Ratebegrænsning af transaktioner og andre teknikker bliver i øjeblikket undersøgt for at dæmpe angreb.

F. >50% Angreb

Metrikken af konsensus for Nano er et balancevægtet stemmesystem. Hvis en hacker er i stand til at opnå 50% af

stemmestyrken, kan de forårsage et netværk til at oscillere konsensus, hvilket gør systemet defekt. En hacker er i stand til at sænke mængden af balance de må forspilde, ved at forhindre gode noder i at stemme vha. DoS af netværket. Nano træffer følgende foranstaltninger for at forhindre sådan et angreb:

1) Den primære beskyttelse mod denne type angreb er stemmevægt, som er bundet til investering i systemet. En kontoholder er, i sagens natur, tilskyndet til at vedligeholde ærlighed i systemet for at beskytte deres investering. Et forsøg på at vende ledgeren ville være destruktivt for systemet generelt, og således ville det ødelægge deres investering.
2) Prisen for dette angreb er proportional til markedsværdien af Nano. I PoW-systemer kan teknologi blive opfundet, som giver uforholdsmæssig kontrol sammenlignet til monetær investering, og hvis angrebet er vellykket, kunne denne teknologi blive lavet om efter angrebet er gennemført. Med Nano ville kosten af, at angribe systemet skalere med selve systemet, og hvis et angreb skulle være vellykket, kan investeringen i angrebet ikke gendannes.
3) For at kunne vedligeholde det maksimale kvorum af stemmer er den næste række af forsvar repræsentantsafstemning. Kontoholdere, som ikke er i stand til driftsikkert at deltage i afstemning af forbindelsesmæssige årsager, kan udnævne en repræsentant, som kan stemme med vægten af deres balance. Maksimering af nummeret og diversitet af repræsentanter forøger netværksresilens.
4) Forker i Nano er aldrig tilfældige, således kan noder træffe beslutninger angående politikken om. hvordan de interagerer med forkede blokke. Det eneste tidspunkt ikke-hacker konti er sårbare over for blokforke er, hvis de modtager balance fra en angribende konto. Konti, som ønsker at være sikret fra blokforke, kan vente lidt eller meget længere, før de modtager fra en konto, som genererede forke, eller vælge aldrig at modtage.
5) En endelig række af forsvar, som ikke endnu er blevet implementeret, er *blokcementering*. Nano går langt for at afregne blokforke vha. afstemning. Noder kunne blive konfigureret til at cementere blokke, hvilket ville forhindre dem fra at blive rullet tilbage efter et vis stykke tid. Netværket er tilstrækkeligt sikret gennem fokusering af hurtig afregningstid for at forhindre tvetydige blokke.

En mere sofistikeret udgave af $> 50\%$ angrebet er illustreret i Figur 9. "Offline" er procentdelen af repræsentanter, som er blevet udnævnt, men ikke er online til at stemme. "Andel" er mængden af investering, hackeren stemmer med. "Aktiv" er repræsentanter, som er online og stemmer ifølge protokollen. En hacker kan opveje mængden, de må forspilde, ved at banke andre stemmere offline vha. et DoS af netværket. Hvis dette angreb kan opretholdes, vil de angrebne repræsentanter blive asynkroniseret, og dette er påvist ved "Unsync." Endelig kan en hacker vinde et kort udbrud i relativ afstemningskraft ved at skifte deres DoS-angreb til et nyt sæt af repræsentanter, mens det gamle sæt resynkroniserer ledgeren. Dette er påvist af "Angreb."

Hvis en hacker er i stand til at forårsage Andel>Aktiv ved kombination af disse omstændigheder, ville de være i stand til at vellykket vende stemmer på ledgeren på bekostningen af deres egen andel. Vi kan vurdere, hvor meget denne type angreb ville koste, ved at undersøge markedsværdien af andre systemer. Hvis vi vurderer, at 33% af repræsentanter er offline eller angrebet af DoS, ville hackeren være nødt til at købe 33% af markedsværdien for at kunne angribe systemet vha. afstemning.

Offline	Unsync	**Angreb**	Aktiv	Andel

Figur 9. En potentiel organisering af afstemning, som kunne reducere 51% angrebskrav.

G. Bootstrap-forgiftning

Desto længere en hacker er i stand til at holde en gammel private-key med en balance, desto højere er sandsynligheden for, at balancer som eksisterede på dette tidspunkt, ikke vil have deltagende repræsentanter pga. deres balancer eller repræsentanter har overført til nyere konti. Dette betyder, at hvis en node er bootstrappet til en ældre repræsentation af netværket, hvor hackeren har et kvorum af afstemningsandelden i forhold til repræsentanter på dette tidspunkt, ville de være i stand til at oscillere afstemningsbeslutninger til denne node. Hvis denne nye bruger ønsker at interagere med nogen, ud over den angribende node, ville alle deres transaktioner blive benægtet, da de har forskellige hovedblokke. Resultatet for netværket er, at noder kan spilde nye noders tid i netværket ved at fodre dem med dårlig information. For at forhindre dette kan noder blive parret med en oprindelig database af konti og anerkendte, gode hovedblokke: Dette er en erstatning af at hente databasen hele vejen tilbage til genesis-blokken. Desto tættere hentningen er til at være aktuel, desto højere er sandsynligheden for korrekt at forsvare mod dette angreb. Når alt kommer til alt er dette angreb ikke værre, end at fodre databras til noder, mens de bootstrapper, da de ikke ville være i stand til at afregne med nogen som helst, som har en moderne database.

VI. Implementering

I øjeblikket, er refereimplementeringen implementeret i C++, og har produceret udgivelser siden 2014 på Github [10].

A. Designfunktioner

Nanos implementering er knyttet til arkitekturstandarden, beskrevet i whitepaperet. Yderligere specifikationer er beskrevet her.

1) Signeringsalgoritme: Nano anvender en modificeret ED25519 elliptisk kurvealgoritme, med Blake2b hashing for samtlige digitale signaturer [11]. ED25519 var valgt for hurtig signering, hurtig verificering og høj sikkerhed.

2) Hashingalgoritme: Da hashing-algoritmen kun er anvendt for forhindring af netværk-spam, er valget af algoritme mindre vigtigt i forhold til mining-baserede kryptovalutaer. Vores implementering anvender Blake2b som en fordøjelsesalgoritme mod indhold af blokke. [12].

3) Nøgleafledningsfunktion: I vores reference-wallet er nøgler krypteret med en adgangskode, og adgangskoden er næret gennem en nøgleafldeningsfunktion for at beskytte mod ASIC-cracking. For tiden er Argon2 [13] vinderen af den eneste offentlige konkurrence med det formål at skabe en bøjelig nøgleafledningsfunktion.

4) Blokinterval: Da hver konto har dens egen blockchain, kan opdateringer blive udført asynkront til netværkets tilstand. Derfor er der ingen blokintervaller, og transaktioner kan publiceres øjeblikkeligt.

5) UDP-beskedsprotokol: Vores system er designet til at operere på ubestemt tid vha. den minimale mængde af resurser til beregning som muligt. Alle beskeder i systemet er designet til at være statsløse og passer ind i en enkel UDP-pakke. Dette gør det også lettere for light ligemænd med intermitterende konnektivitet at deltage i netværket, uden at genetablere kortvarige TCP-forbindelser. TCP er kun anvendt for nye ligemænd, når de ønsker at bootstrappe blockchains i en størrelsesmæssig facon.

Noder kan være sikre på, at deres transaktioner er modtaget af netværket, ved at observere traffik af transaktionsudsendelse fra andre noder, da den burde se adskillige kopier genlydt tilbage til sig selv.

B. IPv6 og Multicast

Konstruering ovenpå forbindelsesløse UDP'er tillader implementeringer af anvendelse af IPv6 multicast som en erstatning for traditionel transaktionsoversvømmelse samt stemmeudsendelse. Dette vil reducere netværkets bredbåndsforbrug samt give mere fleksibilitet hvad angår noders retningslinjer fremadrettet.

Ydeevne I skrivende stund er 4.2 millioner transaktioner blevet behandlet af Nanos netværk, som i afkast giver en blockchain-størrelse på 1.7GB. Transaktionstider er målt i sekunder. En aktuel referenceimplementering kan operere 10.000 transaktioner i sekundet på handelsvare-SSD'er, som primært er IO-bundne.

VII. Resurser

Dette er en oversigt af resurser anvendt af en Nano-node. Derudover gennemgår vi idéer til reducering af resurseforbrug under specifikke tilfælde af brug. Reducerede noder er typisk kaldet light, beskåret eller simplificeret-verificering-af-betaling-noder.

A. Netværk

Mængden af netværksaktivitet afhænger af, hvor meget netværket bidrager til helbredet af et netværk.

1) Repræsentant: En repræsentantnode kræver maksimale netværksresurser, da den observerer stemmetrafik fra andre repræsentanter og publicerer dens egne stemmer.

2) Decentral: En decentral node er lig en repræsentantnode, men er kun en observant, som ikke indeholder en private-key for en repræsentantkonto, og kan ikke selv publicere stemmer.

3) Tillidsfuld: En tillidsfuld node observerer stemmetrafik fra én repræsentant, som den stoler på kan udføre konsensus korrekt. Dette reducerer mængden af indbunden stemmetrafik fra repræsentanter, som går til denne node.

4) Light: en light-node er også en tillidsfuld node, som kun observerer traffik for konti, som den er interesseret i at tillade minimal netværksforbrug.

5) Bootstrap: en bootstrap-node tjener forskellige dele, eller alle af noders ledgere, som bringer dem selv online. Dette er gjort over en TCP-forbindelse fremfor UDP, da det involverer store mængder data, som kræver avanceret kontrol af strømning.

B. Diskkapacitet

Afhængig af brugerkrav kan forskellige node-konfigurationer kræve forskellige opbevaringskrav.

1) Historisk: En node, som er interesseret i at beholde en fuldhistorisk optegnelse af alle transaktioner, vil kræve den maksimale mængde af opbevaring.

2) Strømstyrke: Pga. designet af at beholde indsamlede balancer med blokke, kræver det kun noder at beholde de seneste eller hovedblokke for hver konto for at være i stand til at deltage i konsensus. Hvis en node er uinteresseret i at beholde en fuld historik, kan den vælge kun at beholde hovedblokkene.

3) Light: en light-node beholder ingen lokale ledger-data og deltager kun i netværket for at observere aktivitet på konti, hvor den er interesseret eller eventuelt skaber nye transaktioner med private-keys, som den holder.

C. CPU

1) Generering af transaktioner: En node, som er interesseret i at skabe nye transaktioner, må producere en PoW-nonce for at kunne bestå Nanos drosleringsmekanisme. Beregning af adskillige hardware er benchmarket i bilag A.

2) Repræsentant: En repræsentant må verificere signaturer for blokke, noder og også producere deres egen signaturer for at kunne deltage i konsensus. Mængden af CPU-resurser for en repræsentantnode er betydeligt mindre end generering af transaktioner og burde virke virke med en enkelt CPU i en moderne computer.

3) Observant: En observantnode genererer ikke dens egne stemmer. Da omkostninger for genering af signatur er minimale, er CPU-kravene næsten identiske, med en repræsentant-node.

VIII. Konklusion

I dette whitepaper præsenterede vi konstruktionen af en decentral, afgiftsløs, lavlatens kryptovaluta, som benytter en ny struktur kaldet blokgitter og afstemning vha. Delegated Proof of Stake. Netværket kræver minimale resurser, ingen minedrift med hardware, som har et højt strømforbrug og kan behandle høj transaktionsgennemløb. Alt dette er opnået vha. individuelle blockchains for hver konti, som eliminerer adgangsproblemer og ineffektiviteter i en global datastruktur. Vi identificerede mulige angrebsvektorer på systemet og præsenterede argumenter for, hvordan Nano er modstandsdygtig mod disse slags angreb.

BILAG A
POW-HARDWAREUDGANGSPUNKTER

Som nævnt tidligere er PoW'en i Nano at reducere spam af netværk. Vores node-implementering tilbyder acceleration, som kan udnytte OpenCL-kompatible GPU'er. Tabel 1 tilbyder en virkelig sammenligning af udgangspunkt for adskillige hardware. I øjeblikket er PoW-tærsklen fastlåst, men en adaptiv tærskel kan blive implementeret, som den gennemsnitlige regnekraft skrider frem.

Tabel I
POW-YDEEVNE AF HARDWARE

Enheder	Transaktioner per sekund.
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

BEKENDELSE

Vi vil gerne takke Brian Pugh for kompileringen og formateringen af whitepaperet.

LITTERATUR

[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