Nano: Egy Díjmentes Elosztott Kriptovaluta Hálózat

Colin LeMahieu
clemahieu@nano.co

***Kivonat*—Mostanában, a nagy igények és a korlátozott skálázhatóság megnövelték a népszerű kriptovaluták átlagos tranzakció idejét és költségét, amivel egy nemkívánt élményt okoznak. Bemutatjuk a Nano-t, ami egy újszerű blokkrács szerkezettel rendelkező kriptovaluta ahol minden egyes számla saját blokkláncal rendelkezik, és ezáltal egy közel villámgyors tranzakció sebességet és korlátlan skálázhatóságot biztosít. Minden felhasználó saját blokkláncal rendelkezik, mely azt teszi lehetővé, hogy azt aszinkron módon frissítsék a hálózat többi részével, ami gyors tranzakciókat eredményez minimális többletráfordítással. A tranzakciók a számlák egyenlegeit követik nyomon a tranzakciók összegével ellentétben, ezzel egy agresszív adatbázis metszést (pruning) tesznek lehetővé a biztonság veszélyeztetése nélkül. A mai napig a Nano hálózat 4.2 millió tranzakciót dolgozott fel egy metszetlen főkönyvvel, ami csak 1.7GB méretű. A Nano díjmentes, villámgyors tranzakciói azt elsőosztályú kriptovalutává teszik a fogyasztói ügyletekhez.**

***Index kifejezések*—kriptovaluta, blokklánc, nano, osztott főkönyv, digitális, tranzakciók**

I. Bevezetés

A Bitcoin 2009-ben történő bevezetése óta egy fokozatos áttérés indult el a tradicionális, kormányhatóságok által támogatott valuták és pénzügyi rendszerekről a modernebb, kriptográfián alapuló fizetési rendszerek felé, melyek lehetőséget ajánlanak a pénzeszközök tárolására és forgalmazására egy közvetítőmentes (trustless) és biztonságos módon [1]. A növekvő tranzakció idők, magas díjak és a kétes hálózati skálázhatóság kérdéseket vetett fel a Bitcoin, mint mindennapos pénzként való praktikusságával kapcsolatban.

Ebben a könyvben bemutatjuk a Nano-t, ami egy innovatív blokkrács adat struktúrára épülő, rövid késési idejű (low-latency) kriptovaluta, mely korlátlan skálázhatóságot és tranzakció díjmentességet kínál. A Nano alapvetően egy egyszerű protokoll, azzal az egyedüli céllal, hogy egy csúcsteljesítményű kriptovaluta legyen. A Nano protokoll kisteljesítményű hardveren is futtatható, ami egy praktikus és decentralizált kriptovalutává teszi azt a mindennapi használathoz.

Ebben a könyvben kimutatott kriptovaluta statisztika a közzététel dátumában pontos.

II. Háttér

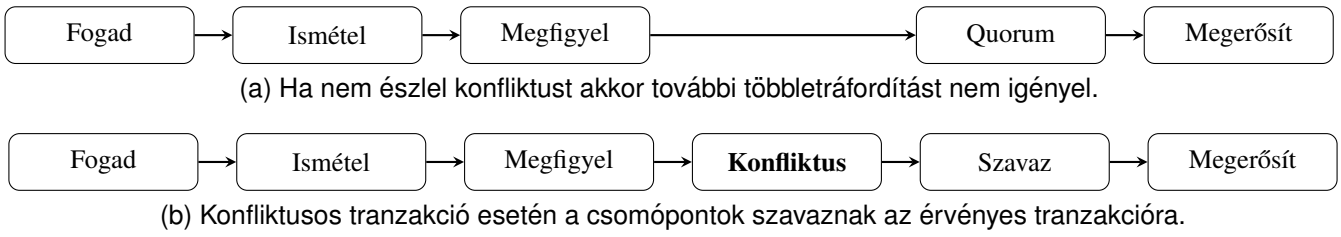
2008-ban, egy ismeretlen személy, Satoshi Nakamoto álnéven kiadott egy fehér könyvet, amiben felvázolta a világ első decentralizált kriptovalutáját, a Bitcoin-t [1]. Egy kulcsfontosságú innováció, melyet a Bitcoin hozott létre, az a blokklánc, ami egy publikus, megváltozhatatlan és decentralizált adatstruktúra, melyet a valuta tranzakcióinak a főkönyveként használnak. Sajnos a Bitcoin fejlődése során, a protokollban lévő különféle problémák lekorlátozták a Bitcoin-t számos alkalmazás esetében:

1) Gyenge skálázhatóság: Minden egyes blokkláncban lévő blokk korlátozott mennyiségű adatot tud eltárolni, ami azt jelenti, hogy a rendszer csak valahány tranzakciót tud feldolgozni egy másodperc alatt, ezzel foltokat, mint árucikket hagy a blokkban. Jelenleg egy medián tranzakció díja $10.38 [2].
2) Magas késési idő: Az átlagos visszaigazolási idő 164 perc [3].
3) Energia hatástalanság: A Bitcoin hálózat a becslések szerint 27.28TWh-ot fogyaszt egy évben, átlagosan 260KWh-ot használ tranzakcióként [4].

A Bitcoin és más kriptovaluták úgy működnek, hogy egy konszenzust teremtenek meg a globális főkönyvükben azzal a céllal, hogy hitelesítsék a szabályos tranzakciókat, míg ellen állnak a rosszhiszemű szereplőknek. A Bitcoin a konszenzust egy gazdasági intézkedés által teremti meg, amit Proof of Work-nek (POW), vagyis munkabizonyítéknak hívnak. A POW rendszerben a résztvevők azon versengenek, hogy kiszámoljanak egy értéket, amit úgy hívnak, hogy *nonce*, úgy, hogy a teljes blokk hash-e egy céltartományon belül legyen. Ez az érvényes tartomány fordítva arányos a teljes Bitcoin hálózat felhalmozott számítógépes teljesítményével azért, hogy fenntartson egy konzisztens átlagos időt, ami egy érvényes nonce megtalálásába telik. Az érvényes nonce megtalálóját arra engedélyezik, hogy a blokkot hozzáadja a blokklánchoz, ezért azok játszanak nagyobb szerepet a blokklánc státuszában, akik minél több számítógépes erőforrást használnak fel a nonce kiszámításához. A POW ellenáll a Sybil támadásnak, ahol egy entitás többszörös entitásként mutatkozik abból a célból, hogy nagyobb befolyásra tegyen szert a decentralizált rendszerben, és hogy nagyban csökkentse a verseny kondícióit is, melyek önmagukban léteznek a globális adat struktúra hozzáférésekor.

Egy alternatív konszenzus protokoll, a Proof Of Stake (POS) vagyis kockázati bizonyíték először 2012-ben mutatkozott be a Peercoin által [5]. A POS rendszerben a résztvevők szavaznak, melynek súlya megegyezik az adott kriptovaluta tulajdonukban lévő vagyonuk összegével. Ezzel a megállapodással azoknak van nagyobb befolyásuk, akik nagyobb pénzügyi befektetéssel rendelkeznek, és eredendően arra vannak motiválva, hogy a rendszer őszinteségét betartsák, különben a befektetéseik elveszítését teszik kockára. A POS eltávolítja a pazarló számítógépes teljesítmény versenyét, mivel csak egy kisteljesítményű hardveren futó könnyű súlyú szoftvert igényel.

Az eredeti Nano könyv és az első béta kivitelezés 2014 decemberében jelent meg, ami által az első Directed Acyclic Graph (DAG) azaz Irányított Körmentes Gráfon alapuló kriptovaluták egyikévé vált [6]. Nemsokkal azután más DAG kriptovaluták is elkezdtek kifejlődni, nevezetesen a DagCo-



(a) Ha nem észlel konfliktust akkor további többletráfordítást nem igényel.

(b) Konfliktusos tranzakció esetén a csomópontok szavaznak az érvényes tranzakcióra.

1. ábra. A Nano nem igényel további többletráfordítást a tipikus tranzakciókhoz. Konfliktusos tranzakció esetén a csomópontok a tranzakció megőrzésére szavaznak.

in/Byteball és az IOTA [7], [8]. Ezek a DAG-on alapuló kriptovaluták megtörték a blokklánc formát, javítva a rendszer teljesítményt és biztonságot. A Byteball úgy teremt konszenzust, hogy egy "main-chain"-ben bízik meg, ami őszinte, tisztességes és a felhasználó által megbízható "szemtanúk"-ból áll, míg az IOTA a konszenzust a stacked tranzakciók felhalmozott munkabizonyítékán keresztül éri el.

A Nano a konszenzust egy egyenleg súlyán alapuló választáson keresztül teremti meg a konfliktusos tranzakciók esetében. Ez a konszenzus rendszer gyorsabb és sokkal determinisztikusabb tranzakciókat biztosít miközben fenntart egy erős, decentralizált rendszert. A Nano folytatja ezt a fejlődést, és helyét már elfoglalta, mint egyike a legjobban teljesítő kriptovalutáknak.

III. A Nano összetevői

Mielőtt leírjuk a Nano átfogó szerkezetét, meghatározzuk az egyes összetevőket, amik megalkotják a rendszert.

III-A. Számla

A számla, a digitális aláírás kulcspárosának a nyilvános kulcs része. A nyilvános kulcs, úgyis nevezik, mint cím, meg van osztva a többi hálózati résztvevővel, míg a privát kulcsot titokban tartják. A digitálisan aláírt adatcsomag azt biztosítja, hogy a tartalom jóvá van hagyva a privát kulcs tulajdonosa által. Egy felhasználó több számlát is vezethet, egy számlának viszont csak egy publikus címe lehet.

III-B. Blokk/Tranzakció

A "blokk" és "tranzakció" kifejezéseket gyakran szinonimaként használják, ahol a blokk egyetlen egy tranzakciót tartalmaz. A tranzakció kifejezetten a tevékenységre, míg a blokk a tranzakció digitális kódolására utal. A tranzakciókat a privát kulccsal írják alá, ami ahhoz a számlához tartozik melyben a tranzakciókat lebonyolították.

III-C. Főkönyv

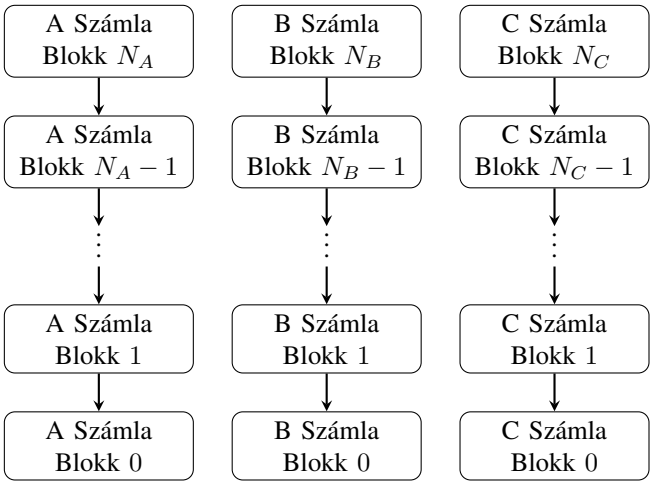
A főkönyv az egy globális számlagyűjtemény (könyvelés), melyben minden egyes számla saját tranzakciós lánccal rendelkezik (2. ábra). Ez egy kulcsfontosságú dizájn komponens, ami azon kategória alá esik, ahol az egy futási időn alapuló megállapodást helyettesíti egy dizájn cikluson alapuló megállapodással; az aláírás ellenőrzés által pedig mindenki egyetért, hogy csakis a számlatulajdonos képes módosítani a saját láncát. Ezáltal a látszólagosan megosztott adatstruktúrát és főkönyvet egy megosztatlanra alakítja át.

III-D. Csomópont

A *csomópont* az egy szoftver, ami egy olyan komputeren fut mely a Nano protokollhoz illeszkedik, és ami a Nano hálózatban részt vesz. A szoftver kezeli a főkönyvet és minden olyan számlát, amit a csomópont ellenőrizhet, ha létezik olyan. Egy csomópont eltárolhatja akár a teljes főkönyvet vagy annak csak a metszett (pruned) történetét, ami minden egyes számla blokkláncának az utolsó néhány blokkját tartalmazza. Új csomópont létrehozásakor ajánlott, hogy a teljes történetet és a metszést helyileg ellenőrizzük.

IV. Rendszer áttekintés

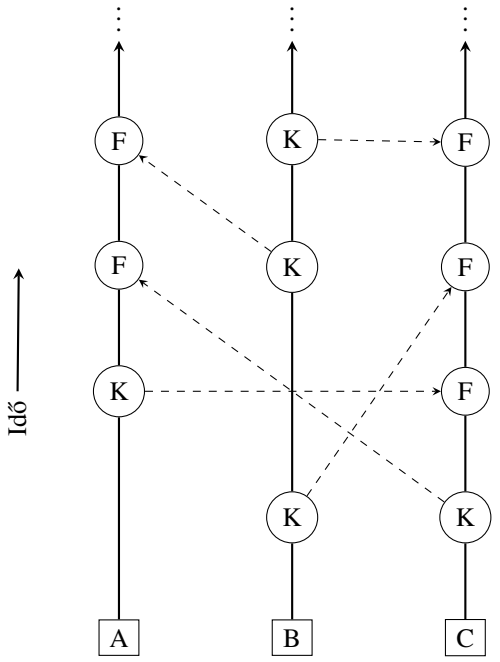
Ellentétben a blokklánccal, amit sok más kriptovaluta használ, a Nano egy *blokkrács* szerkezetet alkalmaz. Minden számla saját blokklánccal (számla-lánc) rendelkezik, ami megegyezik a számla tranzakció/egyenleg történetével (2. ábra). Minden egyes számla-lánc csakis a számlatulajdonos által frissíthető; ezzel lehetővé teszi azt, hogy ezen számla-láncok azonnal és aszinkron módon frissíthetők legyenek a blokkrács többi részével, ami gyors tranzakciókat eredményez. A Nano protokollja rendkívül könnyű súlyú; minden tranzakció belefér a minimálisan előírt UDP csomagméretbe ahhoz, hogy az interneten keresztül közvetíthető legyen. A csomópontok hardver igényei szintén minimálisak, mivel a csomópontoknak a blokkokat csak rögzíteniük és újra közvetíteniük kell a legtöbb tranzakció esetén (1. ábra).



2. ábra. Minden egyes számla saját blokklánccal rendelkezik, ami a számlaegyenleg történetét tartalmazza. A Blokk 0 egy nyitott tranzakció kell hogy legyen (IV-B bekezdés)

A rendszer egy *genesis számlával* van inicializálva, ami a *genesis egyenleget* tartalmazza. A genesis egyenleg egy fix mennyiség, ami soha nem növelhető. Ez a genesis egyenleg fel van osztva és el van küldve a többi számlának a küldési tranzakciók által, melyek a genesis számla-láncon vannak regisztrálva. Az összes számla egyenlegének végösszege soha nem fogja túllépni az eredeti genesis egyenleget, ami ad a rendszernek egy mennyiségi felső határt, melyet az nem fog tudni megnövelni.

Ez a bekezdés bemutatja azt, hogy a különböző típusú tranzakciók hogyan készülnek és terjednek szét a teljes hálózaton.



3. ábra. A blokkrács ábrázolása. Minden pénz transzfer igényel egy küldő (K) és egy fogadó (F) blokkot, melyeket a saját számla-lánc tulajdonosuk aláírt (A,B,C)

IV-A. Tranzakciók

A pénzküldés az egyik számláról a másikra két tranzakciót vesz igénybe: a *küldés* leemeli az összeget a küldő egyenlegéről, a *fogadás* pedig hozzáadja az összeget a fogadó számla egyenlegéhez (3. ábra).

Az összegek transzfere, úgy, mint különálló tranzakciók a küldő és a fogadó számlákon, néhány fontos célt szolgálnak:

1) Sorrendbe állítja a bejövő transzfereket melyek alapvetően aszinkronok.
2) A tranzakciók méretét kicsinek tartja, hogy beleférjenek az UDP csomagokba.
3) Megkönnyíti a főkönyv metszését az adat lábnyom minimalizálásával.
4) Elszigeteli a kiegyenlített tranzakciókat a kiegyenlítetlenektől.

Ha több mint egy számla küld ugyanannak a cél számlának, akkor az egy aszinkron művelet; a hálózati késés és az, hogy a küldő számlák nem feltétlenül kommunikálnak egymással azt jelenti, hogy nincs egy általánosan elfogadott módszer, amiből tudnák, hogy melyik tranzakció történt meg először. Mivel a hozzáadás asszociációs, az input sorrendbe állítása nem számít, így egyszerűen szükség van egy globális megállapodásra. Ez egy kulcsfontosságú dizájn komponens, ami átalakítja az egy futási időn alapuló megállapodást egy dizájn cikluson alapuló megállapodásra. A fogadó számla ellenőrzést gyakorol afelett, hogy eldöntse melyik tranzakció érkezett be először és hogy az a bejövő blokkok megjelölt sorrendjében ki van-e fejezve.

Ha egy számla egy nagyobb transzfert akar lebonyolítani, ami már fogadva lett több kisebb transzferként, akkor azt olyan módon akarjuk prezentálni, hogy az beleférjen egy UDP csomagba. Amikor a fogadó számla sorrendbe teszi a bejövő transzfereket, akkor az nyilván tartja a folytatólagos számla egyenlegének összegét, így bármikor képes lesz átküldeni bármekkora összeget egy fix méretű tranzakcióval. Ez különbözik a Bitcoin és más kriptovaluták kimenő/bejövő tranzakciós modelljétől.

Vannak csomópontok melyek nem érdekeltek abban, hogy forrásaikat egy számla teljes tranzakció történetének tárolására használják fel; csakis az egyes számlák jelenlegi egyenlege érdekli őket. Amikor egy számla létrehoz egy tranzakciót, akkor az összevont egyenlegét lekódolja, így az ilyen csomópontoknak csakis az utolsó blokkot kell nyomon követniük, ami lehetővé teszi, hogy elvessék a régi történeti adatokat míg a helyességről gondoskodnak.

Még úgyis, hogy a fókusz a dizájn ciklusos megállapodásokon van, létezik egy késési időkeret, melyben a tranzakciók érvényesítése folyik abból az okból kifolyólag, hogy azonosítsák és elbánjanak a hálózatban lévő rosszhiszemű egyénekkel. Mivel az egyetértés gyorsan megtörténik a Nano-ban, századmásodpercek-másodpercek nagyságrendjében mérhető, a bejövő tranzakciók kétféle ismerős kategóriáját mutathatjuk be a felhasználónak: kiegyenlített és kiegyenlítetlen. A kiegyenlített tranzakciók olyan tranzakciók, melyekben a számla már generált fogadó blokkokat. A kiegyenlítetlen tranzakciók nincsenek még a fogadó összesített egyenlegébe foglalva. Ez helyettesíti a más kriptovaluták sokkal összetettebb és kevésbé ismert visszaigazolási metrikáját.

IV-B. Számla létrehozása

Egy számla létrehozásához egy *nyitott* (open) tranzakció kezdeményezése szükséges (4. ábra). A nyitott tranzakció mindig az első tranzakciója lesz minden egyes számla-láncnak, ami a pénz első fogadásakor hozható létre. A *számla* (account) mező tárolja a nyilvános kulcsot (címet) ami az aláírásra használatos privát kulcsból van deriválva. A *forrás* (source) mezőben található a tranzakció hash-e, amely a pénzt küldte át. A számla létrehozása során egy képviselőt választanak, aki a nevükben szavaz; ez a későbbiekben megváltoztatható (IV-F bekezdés). A számla kinyilváníthatja magát saját képviselőjeként.

IV-C. Számlaegyenleg

A számla egyenlege magában a főkönyvben van rögzítve. A tranzakció összegének rögzítésével ellentétben, a megerősítés

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

4. ábra. Egy nyílt tranzakció anatomiája

(IV-I bekezdés) a küldő blokk, valamint az azt megelőző blokk egyenlege közötti különbség ellenőrzését szükségelteti. A fogadó számla aztán megnövelheti az előző egyenlegét mivel az bele van számítva az új fogadó blokkban megadott végső egyenlegbe. Ezt azért teszi, hogy javítsa a feldolgozási időt a nagy volumenű blokkok letöltésekor. A számlatörténet lekérdezésekor az összegek már adottak.

IV-D. Számláról küldés

Számláról való küldéskor a címnek már rendelkeznie kell egy meglévő nyitott blokkal és ezáltal egy egyenleggel is (5. ábra). Az *előző* (previous) mező tartalmazza a számla-láncban lévő előző blokk hash-ét. A *cél* (destination) mező tartalmazza a számlát, ahova a pénzt küldik. A küldő blokk megváltozhatatlan amint jóvá hagyják. A pénzt azonnal levonják a küldő számlaegyenlegéről amint kiközvetítik a hálózatra, és az addig várakozik *(pending)* míg a fogadó fél aláírja a blokkot a pénz elfogadásához. A függőben lévő pénz nem tekinthető visszaigazolásra várakozónak mivel az a küldő számlájáról el lett költve és a küldő már nem tudja visszavonni a tranzakciót.

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

5. ábra. Egy küldő tranzakció anatomiája

IV-E. Tranzakció fogadása

A tranzakció befejezéséhez az elküldött pénz fogadójának létre kell hoznia egy fogadói blokkot a saját számla-láncában (6. ábra). A forrás mező a kapcsolódó elküldött tranzakció hash-ére hivatkozik. Amint ezt a blokkot létrehozzák és kiközvetítik, a számla egyenlege frissítve lesz és a pénz hivatalosan is átmegy a számlára.

IV-F. A képviselő kinevezése

A számlatulajdonosok azon képessége, hogy megválasszanak egy képviselőt, aki a nevükben szavaz,

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

6. ábra. Egy fogadó tranzakció anatomiája

egy igen hatékony decentralizált eszköz, aminek nincs szilárd megfelelője a Proof of Work (munkabizonyíték) vagy a Proof of Stake (kockázati bizonyíték) protokollokban. A hagyományos PoS rendszerekben, a számlatulajdonosoknak egy csomópontot kell futtatniuk ahhoz, hogy részt vehessenek a szavazásban. Egy csomópont folyamatos futtatása sok felhasználó számára nem praktikus; a képviselőnek adott jog a számla nevében történő szavazásra enyhíti ezt az elvárást. A számlatulajdonosok azzal a képességgel rendelkeznek, hogy a konszenzust bármikor bármilyen számlára áthelyezhetik. A *módosító* (change) tranzakció megváltoztatja a számla képviselőjét úgy, hogy elveszi a szavazás súlyát a régi képviselőtől és azt hozzáadja az új képviselőhöz (7. ábra). Az ilyen tranzakciók során nem történik pénzmozgás, a képviselőnek pedig nincs felhatalmazása arra, hogy a számlán lévő pénzt elköltse.

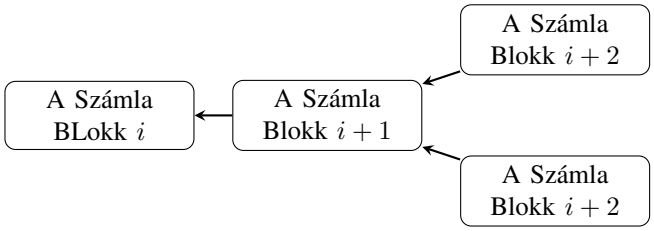
```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

7. ábra. Egy módosító tranzakció anatomiája

IV-G. A Fork-ok (elágazás) és a Szavazás

Egy fork akkor történik meg, amikor egy j számú beírt blokk $b_1, b_2, \ldots, b_j$ ugyanarra a blokkra, mint saját elődjeként hivatkozik (8. ábra). Ezek a blokkok egy konfliktusos képet mutatnak a számla állapotáról, ami megoldásra szorul. Csakis a számla tulajdonosa képes a blokkokat beírni a számla-láncba, tehát a fork valószínű, hogy a számla tulajdonosa által végzett gyatra programozás vagy rossz szándék (dupla költekezés) eredménye.

Amint ez kiderül, a képviselő egy szavazást kezdeményez a főkönyvében lévő blokk b_i-re hivatkozva, melyet aztán kiküld a hálózatra. A csomópont szavazásának súlya, w_i, azon összes számlák összefogó egyenlegével felel meg, melyek azt a saját képviselőjüknek nevezték meg. A csomópont figyeli az M számú online résztvevők bejövő szavazatait és egy összesített listát vezet 4 szavazási periódusról, ami összesen 1 perces, és hitelesíti a nyertes blokkot (1. egyenlet).



8. ábra. Egy fork akkor jön létre amikor két (vagy több) beírt blokk ugyanarra a megelőző blokkra hivatkozik. Régebbi blokkok a bal oldalon; újabb blokkok a jobb oldalon

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

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

A legnépszerűbb blokk b^* kapja a szavazatok többségét, amit a csomópont főkönyve bejegyez (2. egyenlet). A vesztes blokkot/blokkokat kiselejtezik. Ha egy képviselő felcserél egy blokkot a főkönyvében, akkor azzal elindít egy új szavazást egy magasabb sorrendű számmal, amit aztán kiközvetít a hálózatra. Ez az **egyetlen** olyan eset amikor a képviselők szavaznak.

Bizonyos körülményekben, a pillanatnyi hálózati kapcsolódási hibák azt okozhatják, hogy a már kiközvetített blokkot nem minden fél fogadja el. Ezen a számlán lévő mindent azt követő blokk, érvénytelenként, figyelmen kívül lesz hagyva azon felek által, akik nem látták az eredeti kiközvetítést. Ezen blokk újra közvetítését a további felek elfogadják és az azt követő blokkokat pedig automatikusan lehívják. Még akkor is amikor egy fork történik meg vagy egy blokk hiányzik, az csakis a tranzakcióban hivatkozott számlákat befolyásolja; a hálózat többi része ugyanúgy folytatja tovább a tranzakciók feldolgozását az összes többi számla részére.

IV-H. Proof of Work (munkabizonyíték)

Mind a négy típusú tranzakció rendelkezik egy munka mezővel, amit helyesen kell kitölteni. A munka mező által tudja a tranzakció létrehozója kiszámítani a nonce-ot úgy, hogy a nonce hash-e össze van fűzve az előző mezővel a fogadó/küldő/módosító tranzakciók esetében vagy a számla mezővel a nyitott tranzakció esetén, ami egy bizonyos határérték alatt van. A Bitcoin-nal ellentétben a munkabizonyítékot a Nano egy anti-spam eszközként használja, mely hasonló a Hashcash-hez, és ami másodpercek alatt kiszámítható [9]. Ahogy a tranzakciót elküldik, a következő blokk munkabizonyítéka előre kiszámítható mivel az előző blokk mező már ismert; ezzel a végfelhasználók számára a tranzakciók villámgyorsnak fognak tűnni egészen addig amíg a tranzakciók között eltelt idő nagyobb a POW kiszámításához igényelt időnél.

IV-I. Tranzakció ellenőrzés

Ahhoz, hogy egy blokkot érvényesnek nyilvánítsanak, a következő jellemzőkkel kell rendelkeznie:

1) A blokk még nem szerepelhet a főkönyvben (dupla tranzakció).
2)) A számla tulajdonosa által aláírva kell lennie.
3) Az előző blokk a számla-lánc fő blokkja. Ha már létezik de nem a fő blokk akkor az egy fork.
4) A számlának egy nyitott blokkal kell rendelkeznie.
5) A kiszámított hash megfelel a POW határérték követelményének.

Ha ez egy fogadó blokk, akkor ellenőrzi, hogy az eredeti blokk hash-e függőben van-e, ami azt jelenti, hogy még nem lett kiegyenlítve. Ha ez egy küldő blokk, akkor annak egyenlege kisebbnek kell lennie az előző egyenlegnél.

V. Támadási vektorok

A Nano-t, mint minden más decentralizált kriptovalutát rosszhiszemű egyének megtámadhatják, pénzügyi nyereség vagy a rendszer felfüggesztése céljából. Ebben a bekezdésben leírunk néhány lehetséges támadási szituációt, azok következményeit és azt, hogy a Nano protokollja milyen megelőző intézkedések tesz ezek ellen.

V-A. Blokk Rés Szinkronizáció

A IV-G bekezdésben kifejtettük azt a szituációt amikor egy blokkot nem helyesen közvetítenek ki ami azt okozza, hogy a hálózat figyelmen kívül hagyja az azt követő blokkokat. Amikor egy csomópont észreveszi azt, hogy a blokk nem hivatkozik egy megelőző blokkra akkor két lehetősége van:

1) Figyelmen kívül hagyja a blokkot mivel lehetséges, hogy ez egy rosszhiszemű érvénytelen blokk.
2) Újrasync-et kérvényez egy másik csomóponttól.

Újrasync esetén a TCP kapcsolatot egy üzembeállító (bootstrapping) csomóponttal kell kialakítani azért, hogy a resynchez szükséges, a hálózatban megnövekvő forgalom ki legyen szolgálva. Viszont, ha a blokk valóban egy hibás blokk, akkor az újrasync szükségtelen volna, és az a hálózati forgalmat fölöslegesen növelné meg. Ez a Hálózat Felerősítő Támadás a szolgáltatások felfüggesztését eredményezi.

Hogy elkerüljék a szükségtelen újrasync-et, a csomópontok addig várnak míg a potenciális rosszhiszemű blokkra történő szavazatok egy bizonyos határértékét megfigyelték mielőtt elkezdik a kapcsolatfelvételt az üzembeállító csomóponttal a szinkronizáláshoz. Ha egy blokk nem kap elég szavazatot akkor azt limlom adatnak nyilvánítják.

V-B. Tranzakció elárasztás

Egy rosszhiszemű egyén számos szükségtelen bár érvényes tranzakciót küldözgethet az olyan számlák között, melyeket ő kezel, ezzel megkísérli megtelíteni a hálózatot. A tranzakciós díjak nélkül ezt a támadást akár a végtelenségig folytathatná. Mivel a POW minden egyes tranzakcióhoz szükséges, ez limitálja a tranzakciók gyakoriságát, amit a rosszindulatú egyén generálhat, a számítógépes erőforrásokba történő jelentős befektetése nélkül. Még az ilyen támadás alatt is amikor megkísérlik felduzzasztani a főkönyvet, azon csomópontok, amik nem teljes történeti csomópontok, képesek lemetszeni (prune) a régi tranzakciókat a láncról; ezzel kordában tartják a tárhely használatot az ilyen típusú támadás ellen szinte minden felhasználónál.

V-C. Sybil támadás

Egy egyén akár százasával is létrehozhat Nano csomópontokat egy különálló gépen; bár míg a szavazati rendszer súlyozott, ami a számla egyenlegén alapul, az extra csomópontok hálózathoz hozzáadásával nem fog a támadó extra szavazatokat nyerni. Tehát semmiféle előnyre nem tesz szert a Sybil támadás által.

V-D. Fillér-költős támadás

A fillér-költős támadás az, amikor a támadó elenyésző mennyiséget költ nagyon sok számlán azzal a céllal, hogy elpazarolja a csomópont tárolási erőforrásait. A blokk publikáció a POW által gyakoriság-limitált, ami a számlák és tranzakciók létrehozását egy bizonyos mértékben korlátozza. Azon csomópontok melyek nem teljes történeti csomópontok képesek a számlák egy statisztikai mérték alá való metszésére, ahol a számla nagy valószínűséggel nem érvényes. Befejezésül, a Nano-t úgy hangolták, hogy minimális állandó tárhelyet használjon, így az a hely, ami egy újabb számla tárolásához szükséges arányos a nyitott blokk+indexelés méretével $(\text{open block} + \text{indexing}) = 96\text{B} + 32\text{B} = 128\text{B}$. Ez annyit jelent, hogy 1GB képes 8 millió fillér-költős számlát tárolni. Ha a csomópontok sokkal agresszívabban akarnának metszeni akkor kidolgozhatnak egy olyan kiosztást, ami a hozzáférés gyakoriságán alapul és a ritkán használatos számlákat a lassabb tárhelyekhez delegálhatnák.

V-E. Előre kalkulált PoW támadás

Mivel a számlatulajdonos az egyetlen személy, aki blokkokat ad a számla-lánchoz, a szekvenciális blokkok kiszámíthatók a munkabizonyítékaikkal együtt még mielőtt ki lennének közvetítve a hálózatra. Ilyenkor a támadó egy hosszabb időtartam alatt számtalan sorozatos minimális értékű blokkokat generál. Egy bizonyos ponton, a támadó véghezvisz egy DoS-t olyan módon, hogy a hálózatot rengeteg érvényes tranzakcióval árassza el, amit más csomópontok feldolgoznak és visszajeleznek amilyen gyorsan csak lehet. Ez egy fejlettebb verziója annak a tranzakció elárasztásnak, amit a V-B bekezdésben tárgyaltunk. Egy ilyen támadás csak rövid ideig működne de más típusú támadásokkal együtt is használhatják, úgy, mint a >50%-os támadás (V-F bekezdés), hogy növeljék a hatékonyságot. A tranzakció gyakoriság-korlátozás és más módszerek jelenleg tanulmányozás alatt állnak a támadások mérséklésének érdekében.

V-F. >50%-os támadás

A Nano konszenzus metrikája egy egyenleg súlyos szavazati rendszer. Ha a támadónak sikerül 50%-nál nagyobb szavazati többséget nyernie akkor az a hálózat kilengését okozhatja, mellyel a rendszert sérültté teszi. A támadó képes csökkenteni az egyenleg összegét, amit fel kell áldoznia azáltal, hogy megakadályozza az érvényes csomópontok szavazását a hálózat DoS-ekor. A Nano a következő intézkedéseket alkalmazza az ilyen támadások megelőzése érdekében:

1) Az elsődleges védekezés az ilyen típusú támadás ellen a szavazat súlyának a rendszerbe való befektetéshez kötése. A számlatulajdonos alapvetően arra van motiválva, hogy a rendszer őszinteségét betartsa hogy ezzel megvédje a befektetését. Megkísérelni a főkönyv hamisítását ártalmas lenne az egész rendszerre és tönkre tenné a befektetéseiket.
2) Az ilyen támadás ára a Nano piaci kapitalizációjával arányos. A POW rendszerekben kifejleszthető egy olyan technológia, ami aránytalan kontrollt ad a pénzügyi befektetéshez viszonyítva ha a támadás is sikeres, ez a technológia más célra is felhasználható lehetne amint a támadás befejeződött. A Nano-ban a rendszer támadásának ára arányban van magával a rendszerrel, tehát ha egy támadás sikeres akkor a támadásban lévő befektetés nem nyerhető vissza.
3) Abból a célból, hogy megőrizzék a szavazók maximális határozatképességét, a védelem következő vonala a képviseleti szavazás. Azon számlatulajdonosok, melyek kapcsolódási okok miatt nem tudnak a szavazásban megbízható módon részt venni kijelölhetnek egy képviselőt, aki az ő egyenlegük súlyának egyenértékével szavazhat. A képviselők számának maximalizálása, illetve diverzitása növeli a hálózat rugalmasságát.
4) A Nano-ban a forkok soha nem véletlenek, tehát a csomópontok szabályzati döntést hozhatnak arról, hogy hogyan működjenek együtt a forkolt blokkokkal. A nem támadó számlák csak akkor veszélyeztettek a blokk forkokkal szemben, ha egyenleget fogadnak egy támadó számlától. A blokk forkoktól biztonságban maradni kívánó számlák kis ideig vagy akár sokáig is várakozhatnak azelőtt, hogy attól a számlától fogadjanak, ami a forkokat generálta vagy dönthetnek úgy is hogy attól egyáltalán ne fogadjanak. A fogadók szintén képesek külön számlákat generálni és azokat használni amikor pénzt fogadnak gyanús számláktól, abból a célból, hogy a többi számlát elszigeteljék.
5) A végső védekezési vonal, amit egyelőre még nem vezettek be, a *blokk megerősítés*. A Nano hatalmas erőfeszítéseket tesz annak érdekében, hogy a blokk forkokat szavazás által gyorsan elintézze. A csomópontok konfigurálhatók a blokkok megerősítésére, ami megelőzné azt, hogy azok egy bizonyos idő után visszaállíthatók legyenek. A hálózat megfelelően védve van azáltal, hogy a fókusz a gyors intézési időn van mellyel a kétes forkokat megakadályozza.

Az $> 50\%$-os támadás egy sokkal kifinomultabb verziója a 9. ábrán van részletezve. Az „Offline” az a képviselők azon százaléka, akiket kineveztek viszont nincsenek online-ban a szavazáshoz. A „Stake” az a befektetés összege, amivel a támadó szavaz. Az „Active” az azok a képviselők, akik online-ban vannak és a protokollnak megfelelve szavaznak. A támadó kompenzálni tudja azt a stake-et (részesedés) amit fel kell áldoznia úgy, hogy a más szavazókat kilöki offline-ba a hálózat DoS támadásával. Ha ez a támadás hosszantartó akkor a támadott képviselők szinkronizáltalanná válnak, amit az „Unsync” demonstrál. Végül, a támadó szert tehet egy rövid löketre a relatív szavazás erejében azáltal, hogy a szolgáltatás

támadását áthelyezi az új képviselőkre míg a régiek újra szinkronizálják a főkönyvüket, ezt az „Attack” demonstrálja.

Offline	Unsync	**Attack**	Active	Stake

9. ábra. egy lehetséges szavazati megállapodás, ami csökkentheti az 51%-os támadás igényeit

Ha a támadó képes a Stake>Active-ot elérni ezen körülmények kombinációja által akkor a részesedései árán az sikeresen meg tudná hamisítani a szavazatokat a főkönyvben. Meg tudjuk becsülni azt, hogy mennyibe kerülhet az ilyen típusú támadás azzal, hogy megvizsgáljuk a más rendszerek piaci kapitalizációját. Ha azt becsüljük, hogy a képviselők 33%-a offline vagy DoS támadás alatt áll akkor a támadónak a piaci kapitalizáció 33%-át kéne megvásárolnia ahhoz, hogy a rendszert a szavazással támadja.

V-G. Bootstrap mérgezés

Ha a támadó minél hosszabb ideig képes birtokolni egy régi privát kulcsot, aminek egyenlege van, annál nagyobb az esélye annak, hogy az egyenlegeknek, melyek annak idején léteztek, nincsenek résztvevő képviselői, mivel azok egyenlegei vagy képviselői már újabb számlákra transzfereltek át. Ez azt jelenti, hogy ha egy csomópont bootstrapped a hálózat egy régebbi értelmezéséhez, ahol a támadó egy bizonyos időben egy határozatképes létszámban lévő szavazati részesedéssel rendelkezik a képviselőkkel szemben, akkor az képes lenne megingatni az azon csomópontra vonatozó szavazati döntéseket. Ha ez az új felhasználó kapcsolatba akarna lépni bárkivel a támadó csomóponton kívül, akkor minden tranzakcióját elutasítják mivel azok fő blokkjai különböznek. Ennek a végső eredménye az, hogy a csomópontok elpazarolják a hálózaton lévő új csomópontok idejét azzal, hogy helytelen információt szolgáltatnak. Ennek megelőzésére, a csomópontok párosíthatók a számlák és jónak tudott blokk fejek eredeti adatbázisával; ez helyettesíti az adatbázis letöltését a genesis blokkig visszamenőleg. Minél közelebb áll a letöltés az aktuális állapothoz annál magasabb az esélye a precíz védekezésnek az ilyen támadás ellen. Elvégre, az ilyen támadás valószínűleg nem rosszabb, mint a bootstrapping alatt álló csomópontok részére történő limlom adatszolgáltatás mivel olyankor nem tudnának tranzakcióba lépni senki olyannal, aki egy jelen időben létező adatbázissal rendelkezik.

VI. Kivitelezés

Jelenleg a referencia kivitelezés C++-ban van megvalósítva és 2014 óta release-eket produkál a Github-on [10].

VI-A. Dizájn tulajdonságok

A Nano kivitelezés megfelel a szerkezeti szabványnak, amit ebben a könyvben felvázoltunk. A további specifikációkat itt írjuk le.

VI-A1. Aláíró algoritmus: A Nano egy módosított ED25519 elliptikus görbés Blake2b hash-es algoritmust használ az összes digitális aláíráshoz [11]. Az ED25519-et a gyors aláírás, gyors verifikáció és a magas biztonság miatt választották.

VI-A2. Hashing algoritmus: Mivel a hashing algoritmust csak a hálózati spam megelőzésére használják, az algoritmus választása kevésbé fontos mikor azt a bányászaton alapuló kriptovalutákhoz viszonyítjuk. Kivitelezésünk a Blake2b-t használja, mint digest (kivonatoló) algoritmus, a blokk tartalmak ellen [12].

VI-A3. Kulcs származási funkció: a referencia pénztárcában lévő kulcsok le vannak kódolva egy jelszóval, ami egy kulcs származási funkción megy át, hogy azt megvédje az ASIC cracking próbálkozások ellen. Jelenleg az Argon2 [13] a nyertes abban az egyetlen publikus versenyben, aminek az a célja, hogy egy rugalmas kulcs származási funkciót hozzon létre.

VI-A4. Blokk intervallum: Mivel minden számlának van egy saját blokklánca, a frissítések a hálózat állapotában aszinkron módon végezhetők. Így a blokk intervallumok nem léteznek és a tranzakciók pedig azonnal publikálhatók.

VI-A5. UDP üzenet protokoll: A rendszerünk úgy van dizájnolva, hogy korlátlanul működjön, lehetőleg minimális számítógépes erőforrásokat használva. Minden rendszerben lévő üzenet úgy lett dizájnolva hogy állapot nélküli legyen és beleférjen egy egyetlen UDP csomagba. Ez szintén megkönnyíti a váltakozó kapcsolattal rendelkező lite-os felek számára, hogy részt vegyenek a hálózatban a TCP kapcsolatok rövid idejű újra létesítése nélkül. A TCP csak az új felek esetében használatos amikor azok nagy mennyiségben akarják bootstrappelni a blokkláncokat.

A csomópont úgy bizonyosodik meg arról, hogy a hálózat fogadta a tranzakcióit, hogy figyeli a más csomópontok tranzakciós közvetítő forgalmát, ami által látnia kéne több példány magához történő visszajelzését.

VI-B. IPv6 and Multicast (többes-adás)

A kapcsolat nélküli UDP-re való építés lehetővé teszi, hogy a jövőbeni kivitelezések IPV6 multicast-ot használjanak a tradicionális tranzakció elárasztás és szavazati kiközvetítés helyett. Ez lecsökkenti majd a hálózati sávszélesség fogyasztást és több szabályozási rugalmasságot nyújt a csomópontoknak a jövőben.

VI-C. Teljesítmény

Ezen írás idejében, a Nano hálózat már 4.2 millió tranzakciót dolgozott fel, amivel egy 1.7GB méretű blokkláncra gyarapodott. A tranzakció idők a másodpercek nagyságrendjében mérhetők. A jelenlegi referencia kivitelezés, ami általános SSD tárolókon üzemel, több mint 10,000 tranzakciót képes feldolgozni másodpercenként, elsődlegesen IO bound-ként működtetve.

VII. Erőforrás kihasználás

Ez egy összefoglalás a Nano csomópont által használt erőforrásokról. Továbbá, megvizsgálunk néhány ötletet az erőforrások használatának lecsökkentésére speciális használati esetekben. A csökkentett csomópontokat tipikusan könnyű, metszett vagy egyszerűsített fizetési ellenőrző (SPV) csomópontoknak hívnak.

VII-A. Hálózat

Egy csomópont hálózati aktivitása függ attól, hogy a csomópont mennyire járul hozzá a hálózat egészségéhez.

VII-A1. Képviselő: a képviselői csomópont maximális hálózati erőforrásokat igényel mivel az figyeli a többi képviselők szavazati forgalmát és közzéteszi a saját szavazatait.

VII-A2. Trustless: A trustless csomópont hasonló a képviselői csomóponthoz, viszont az csak egy megfigyelő, ami nem tartalmaz egy képviselői számla privát kulcsot és nem teszi közzé a saját szavazatait.

VII-A3. Trusting: A trusting csomópont figyeli az egy azon képviselő szavazati forgalmát, akiben megbízik, azért, hogy helyesen bonyolítsa le a konszenzust. Ez lecsökkenti a bejövő szavazati forgalom mennyiségét az e csomóponthoz menő képviselőktől.

VII-A4. Light: A light (könnyű) csomópont szintén egy trusting csomópont, ami a forgalmat azon számlák részére figyeli meg, amikben az érdekelt, a hálózat minimális használata révén.

VII-A5. Bootstrap: A bootstrap csomópont felszolgálja a teljes főkönyvet vagy annak részeit azon csomópontok számára, melyek online-ba hozzák magukat. Ez a TCP csatlakozás által történik meg és nem UDP-vel, mivel hatalmas mennyiségű adatról van szó, ami fejlett forgalomszabályozást igényel.

VII-B. Merevlemez kapacitás

A felhasználói igényektől függően a különféle csomóponti konfigurációk különböző tárolási igényeket követelnek.

VII-B1. Történelmi: Az a csomópont, ami az összes tranzakció teljes történelmét kívánja megőrizni, maximum mennyiségű kapacitást igényel.

VII-B2. Aktuális: A felhalmozott egyenlegek blokkokkal együttesen történő megőrzésének a dizájnja miatt, a csomópontoknak csak az utolsó vagy a fő blokkokat kell megőrizniük ahhoz, hogy részt vegyenek a konszenzusban. Ha egy csomópontnak nincs érdekében az, hogy a teljes történelmet megtartsa akkor dönthet úgy, hogy csak a fő blokkokat őrzi meg.

VII-B3. Light: a light (könnyű) csomópont nem őriz meg semmilyen helyi főkönyvi adatot és csak azért vesz részt a hálózatban, hogy figyelje az aktivitást azokon a számlákon, amikben érdekelt, vagy hogy opcionálisan új tranzakciókat hozzon létre a birtokában lévő privát kulcsokkal.

VII-C. CPU

VII-C1. Tranzakció generálás: Annak a csomópontnak, ami új tranzakciók létrehozásában érdekelt, fel kell mutatnia egy Proof of Work nonce-t ahhoz, hogy átmenjen a Nano throttling mechanizmusán. A különféle hardverek számítási teljesítményének összehasonlítása az A függelékben található meg.

VII-C2. Képviselő: A képviselő köteles az aláírásokat ellenőrizni a blokkok és szavazatok esetében, valamint a saját aláírásait előállítani, azért hogy a konszenzusban részt vehessen. A CPU erőforrások mennyisége a képviselői csomópont számára jelentősen kisebb, mint a tranzakciót létrehozóé, aminek működnie kell egy jelen időben létező számítógép bármilyen szimpla CPU-jával.

VII-C3. Megfigyelő: A megfigyelő csomópont nem generálja a saját szavazatait. Mivel az aláírás generálási többletráfordítás minimális, a CPU igények majdnem azonosak a képviselő csomópont futtatásáéval.

VIII. Befejezés

Ebben a könyvben bemutattuk a szerkezetét az egy trustless, díjmentes, alacsony késési idejű kriptovalutának, ami egy újszerű blokkrács szerkezetet és delegált Proof of Stake szavazást hasznosít. A hálózat minimális erőforrásokat vesz igénybe, magas teljesítményű bányászati hardvert nem igényel és képes a magas tranzakció forgalom feldolgozására. Mindezt az által éri el, hogy minden egyes számla saját blokklánccal rendelkezik, ami kiküszöböli a hozzáférési problémákat és a globális adatstruktúra elégtelenségeit. Beazonosítottuk az esetleges rendszertámadó vektorokat és bizonyítékokat mutattunk fel arra, hogy a Nano hogyan áll ellen az ilyen formájú támadásoknak.

függelék A
POW hardver teljesítménymérés

Mint ahogy korábban említettük, a Nano-ban a POW azért van, hogy a hálózati spam-et csökkentse. A csomópont implementációnk gyorsítást biztosít, ami kihasználja az OpenCL kompatibilis GPU-kat. Az I táblázat a különféle hardverek valós tejesítménymérésének összehasonlítását mutatja. A POW küszöb jelenleg fixált viszont egy adaptív küszöb bevezethető az átlagos számítógépes teljesítmények fejlődésével.

I. táblázat
Hardver POW teljesítmény

Eszköz	Tranzakció per másodperc
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

Köszönetnyilvánítás

Szeretnénk megköszönni Brian Pugh-nak ezen könyv összeállítását és formázását.

Hivatkozások

[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