Bitcoin: tarpusavio atsiskaitymų elektroninių pinigų sistema

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Išversta LBCA.LT

Anotacija. Elektroniniai pinigai, kuriuos būtų galima siųsti internetu tiesiogiai iš vienos šalies kitai šaliai, nesikišant jokiai finansų įstaigai. Skaitmeniniai parašai yra dalis sprendimo, tačiau pagrindinė nauda prarandama, jei siekiant išvengti dvigubų išlaidų vis dar reikia patikimos trečiosios šalies. Siūlome dvigubo išlaidavimo problemos sprendimą, naudojant tarpusavio tinklą. Tinklas žymi sandorius laiko žymomis, įrašydamas juos į tęstinę hash pagrįsta darbo įrodymų grandine, taip suformuojant įrašą, kurio negalima pakeisti neatlikus pakartotinio darbo (PoW) įrodymo. Ilgiausia grandinė ne tik įrodo užfiksuotų įvykių seką, bet ir įrodo, kad ji atsirado iš didžiausio procesoriaus galios valdytojo. Kol didžiąją dalį procesoriaus galios kontroliuoja mazgai, nebendradarbiaujantys atakuojant tinklą, jie sukurs ilgiausią grandinę ir užkirs kelią įsilaužėliams. Pačiam tinklui reikia minimalios struktūros. Sandoriai tvirtinami įrodymais, o mazgai gali palikti tinklą ir vėl prie jo prisijungti savo nuožiūra, priimdami ilgiausią darbo įrodymo grandinę kaip įrodymą, kas įvyko, kol jų nebuvo.

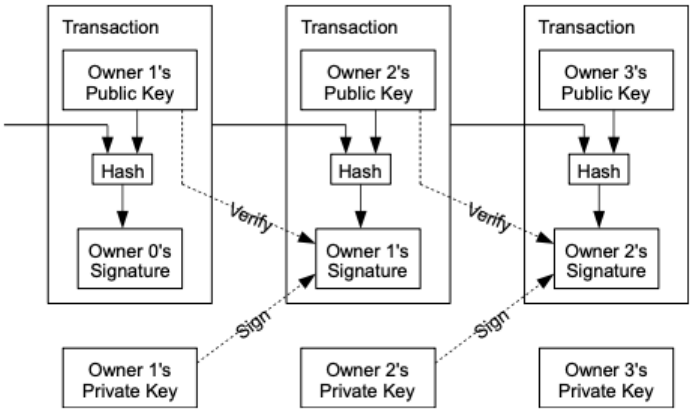
1. Įvadas

Prekyba internetu beveik išimtinai priklauso nuo finansų įstaigų, kurios yra patikimos trečiosios šalys, vykdančios elektroninius mokėjimus. Nors ši sistema veikia pakankamai gerai daugeliu atvejų, ji vis dar kenčia nuo pasitikėjimu grindžiamam modeliui būdingų trūkumų. Visiškai negrįžtami sandoriai iš tikrųjų neįmanomi, nes finansų įstaigos negali išvengti tarpininkavimo ginčuose. Tarpininkavimo išlaidos didina sandorių sąnaudas, todėl apribojamas mažiausias praktinis sandorio dydis ir nutraukiama galimybė sudaryti nedidelius atsitiktinius sandorius; be to, prarandama galimybė atlikti negrįžtamuosius mokėjimus už negrįžtamąsias paslaugas. Atsiradus grįžtamojo ryšio galimybei, didėja pasitikėjimo poreikis. Prekybininkai turi saugotis savo klientų, reikalauti iš jų daugiau informacijos, nei jiems reikėtų kitu atveju. Tam tikras sukčiavimo procentas priimamas kaip neišvengiamas. Šių išlaidų ir mokėjimo netikrumo galima išvengti asmeniškai naudojant fizinę valiutą, tačiau nėra mechanizmo, kaip atlikti mokėjimus ryšio kanalu be patikimos šalies.

Reikalinga ne pasitikėjimu, o kriptografiniais įrodymais pagrįsta elektroninių mokėjimų sistema, kuri leistų bet kurioms dviem norinčioms šalims sudaryti tiesioginius tarpusavio sandorius, nereikalaujant patikimos trečiosios šalies. Sandoriai, kurių neįmanoma pakeisti skaičiavimo būdu, apsaugotų pardavėjus nuo sukčiavimo, o pirkėjams apsaugoti būtų galima lengvai įdiegti įprastus sąlyginio deponavimo mechanizmus. Šiame darbe siūlome dvigubo išlaidavimo problemos sprendimą naudodami tarpusavio paskirstytų laiko žymų serverį, kad sukurtume chronologinės sandorių eilės skaičiavimo įrodymą. Sistema yra saugi tol, kol sąžiningi mazgai kolektyviai valdo daugiau procesoriaus galios nei bet kuri bendradarbiaujanti užpuolikų mazgų grupė.

2. Sandoriai

Elektroninę monetą apibrėžiame kaip skaitmeninių parašų grandinę. Kiekvienas savininkas perduoda monetą kitam savininkui skaitmeniniu parašu pasirašydamas ankstesnio sandorio hash ir kito savininko viešąjį raktą ir pridėdamas juos prie monetos pervedimo. Gavėjas gali patikrinti parašus, kad patikrintų nuosavybės grandinę.

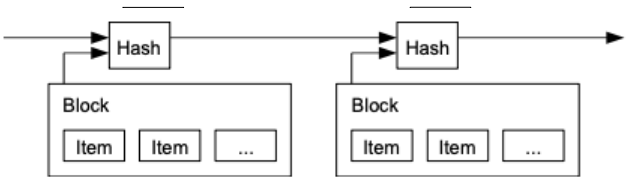


Problema, žinoma, yra ta, kad gavėjas negali patikrinti, ar vienas iš savininkų neišleido monetos du kartus. Įprastas sprendimas - įvesti patikimą centrinę instituciją arba monetų kalyklą, kuri tikrina, ar kiekviena operacija nėra dvigubai išleista. Po kiekvienos operacijos moneta turi būti grąžinama į monetų kalyklą, kad būtų išleista nauja moneta, ir tik tiesiogiai iš monetų kalyklos išleistos monetos yra patikimos, kad nebuvo išleistos du kartus. Šio sprendimo problema yra ta, kad visos pinigų sistemos likimas priklauso nuo monetų kalyklą valdančios įmonės, nes kiekvienas sandoris turi vykti per ją, kaip ir per banką.

Reikia, kad gavėjas žinotų, jog ankstesni savininkai nepasirašė jokių ankstesnių sandorių. Mūsų tikslais skaičiuojama anksčiausia operacija, todėl mums nerūpi vėlesni bandymai dvigubai išleisti pinigus. Vienintelis būdas patvirtinti, kad sandorio nėra, yra žinoti apie visus sandorius. Monetomis pagrįstame modelyje monetų kalykla žinojo apie visus sandorius ir sprendė, kurie iš jų atėjo pirmieji. Kad tai būtų galima padaryti be patikimos šalies, apie sandorius turi būti skelbiama viešai [1], ir mums reikia sistemos, kad dalyviai galėtų susitarti dėl vienos istorijos, kurioje būtų nurodyta jų gavimo tvarka. Gavėjui reikia įrodyti, kad kiekvieno sandorio metu dauguma mazgų sutarė, kad jis buvo gautas pirmas.

3. Laiko žymų serveris

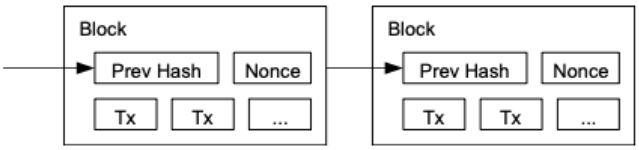
Mūsų siūlomas sprendimas prasideda nuo laiko žymų serverio. Laiko žymų serveris veikia imdamas laiko žymą turinčių elementų bloko hash ir plačiai skelbdamas šį hash, pavyzdžiui, laikraštyje ar Usenet žinutėje [2-5]. Laiko žyma įrodo, kad duomenys, akivaizdu, turėjo egzistuoti tuo metu, kad patektų į hash. Kiekviena laiko žyma į savo hash įtraukia ankstesnę laiko žymą, sudarydama grandinę, kurioje kiekviena papildoma laiko žyma sustiprina prieš tai buvusias.



4. Darbo įrodymas (PoW)

Norint įgyvendinti paskirstytą laiko žymų serverį tarpusavio pagrindu, mums reikės naudoti ne laikraščių ar Useneto pranešimus, o darbo įrodymų sistemą, panašią į Adamo Bako "Hashcash" [6]. Darbo įrodymas apima vertės, kurią išskleidus, pavyzdžiui, naudojant SHA-256, hash prasideda nulio bitų skaičiumi, nuskaitymą. Vidutinis reikalingas darbas yra eksponentiškas reikiamų nulinių bitų skaičiui ir gali būti patikrintas atliekant vieną hash.

Mūsų laiko žymų tinkle darbo įrodymą įgyvendiname didindami bloko nonce, kol randama vertė, kuri bloko hash suteikia reikiamus nulinius bitus. Kai procesoriaus pastangos buvo panaudotos tam, kad blokas atitiktų darbo įrodymo reikalavimą, bloko negalima pakeisti neatlikus pakartotinio darbo. Kadangi vėlesni blokai yra grandininiu būdu sujungti po bloko, norint pakeisti bloką, reikėtų iš naujo atlikti visus po jo einančius blokus.



Darbo įrodymas taip pat išsprendžia atstovavimo daugumos sprendimų priėmimo procese nustatymo problemą. Jei dauguma būtų grindžiama principu "vienas IP adresas - vienas balsas", ją galėtų nuversti bet kas, galintis paskirti daug IP adresų. Darbo įrodymas iš esmės yra "vienas procesorius - vienas balsas". Daugumos sprendimui atstovauja ilgiausia grandinė, į kurią investuota daugiausia darbo įrodymo pastangų. Jei daugumą procesoriaus galios kontroliuoja sąžiningi mazgai, sąžininga grandinė augs sparčiausiai ir aplenks visas konkuruojančias grandines. Norėdamas pakeisti praeitą bloką, užpuolikas turėtų iš naujo atlikti bloko ir visų po jo einančių blokų darbo įrodymą ir tada pasivyti ir aplenkti sąžiningų mazgų darbą. Vėliau parodysime, kad tikimybė, jog lėtesnis užpuolikas pasivys kitus blokus, mažėja eksponentiškai.

Siekiant kompensuoti didėjantį aparatinės įrangos greitį ir kintantį susidomėjimą veikiančiais mazgais laikui bėgant, darbo įrodymo sudėtingumas nustatomas pagal slenkantį vidurkį, kuriuo siekiama nustatyti vidutinį blokų skaičių per valandą. Jei jie generuojami per greitai, sudėtingumas didėja.

5. Tinklas

Tinklo paleidimo veiksmai:

1) Nauji sandoriai perduodami visiems mazgams.
2) Kiekvienas mazgas surenka naujus sandorius į bloką.
3) Kiekvienas mazgas stengiasi rasti sudėtingą darbo įrodymą savo blokui.
4) Kai mazgas randa darbo įrodymą, jis išsiunčia bloką visiems mazgams.
5) Mazgai priima bloką tik tuo atveju, jei visi jame esantys sandoriai yra galiojantys ir dar nepanaudoti.
6) Priėmę bloką mazgai išreiškia savo pritarimą blokui, kurdami kitą grandinės bloką, naudodami priimto bloko hash kaip ankstesnį hash.

Mazgai visada laiko ilgiausią grandinę teisinga ir toliau ją pratęsia. Jei du mazgai vienu metu transliuoja skirtingas kito bloko versijas, ir kai kurie mazgai gali pirmiau gauti vieną ar kitą. Tokiu atveju jie dirba su pirmąja gautąja, bet išsaugo kitą atšaką, jei ji taptų ilgesnė. Varžybos kuri tikra bus nutrauktos, kai bus rastas kitas darbo įrodymas ir viena šaka taps ilgesnė; tada mazgai, kurie dirbo su kita šaka, pereis prie ilgesnės šakos.

Naujų sandorių transliacijos nebūtinai turi pasiekti visus mazgus. Jei jos pasieks daug mazgų, netrukus pateks į bloką. Blokų transliacijos taip pat toleruoja nutrūkusius pranešimus. Jei mazgas negauna bloko, jis jo paprašys, kai gaus kitą bloką ir supras, kad jį praleido.

6. Skatinimas

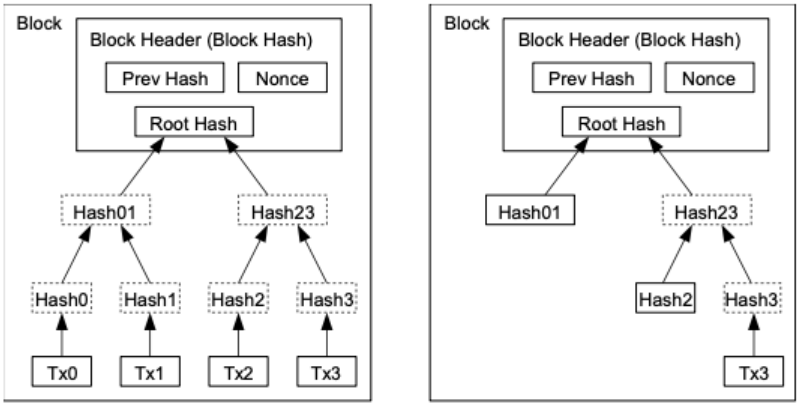
Pagal susitarimą pirmoji bloko transakcija yra speciali transakcija, kuria pradedama kurti nauja moneta, priklausanti bloko kūrėjui. Tai skatina mazgus palaikyti tinklą ir suteikia galimybę iš pradžių paskirstyti monetas apyvartoje, nes nėra centrinės institucijos, kuri jas išleistų. Nuolatinis pastovaus naujų monetų kiekio pridėjimas yra analogiškas aukso kasėjams, kurie eikvoja išteklius, kad į apyvartą įtrauktų auksą. Mūsų atveju tai yra procesoriaus laikas ir elektros energija.

Ši paskata taip pat gali būti finansuojama iš sandorių mokesčių. Jei sandorio išvesties vertė yra mažesnė už jo įvesties vertę, skirtumas yra sandorio mokestis, kuris pridedamas prie bloko, kuriame yra sandoris, skatinamosios vertės. Kai į apyvartą patenka iš anksto nustatytas monetų skaičius, skatinamoji priemonė gali visiškai pereiti prie sandorių mokesčių ir būti visiškai be infliacijos.

Ši paskata gali padėti paskatinti mazgus išlikti sąžiningus. Jei užpuolikas gali surinkti daugiau procesoriaus galios nei visi sąžiningi mazgai, jam tektų rinktis, ar naudoti ją žmonių apgaudinėjimui, vagiant jų mokėjimus, ar naudoti ją naujoms monetoms generuoti. Jam turėtų būti naudingiau žaisti pagal taisykles, tokias taisykles, kurios jam palankios ir suteikia daugiau naujų monetų nei visiems kitiems kartu sudėjus, nei kenkti sistemai ir savo turto pagrįstumui.

7. Diskinės vietos atkūrimas

Kai naujausia monetos operacija yra paslėpta po pakankamu skaičiumi blokų, prieš tai buvusias operacijas galima išmesti, kad būtų sutaupyta vietos diske. Kad tai būtų lengviau padaryti nesugadinant bloko hash, sandoriai yra hashuojami Merkle medžiu [7][2][5], į bloko hash įtraukiama tik šaknis. Tada senus blokus galima sutankinti, atjungiant medžio šakas. Vidinių hešų saugoti nereikia.

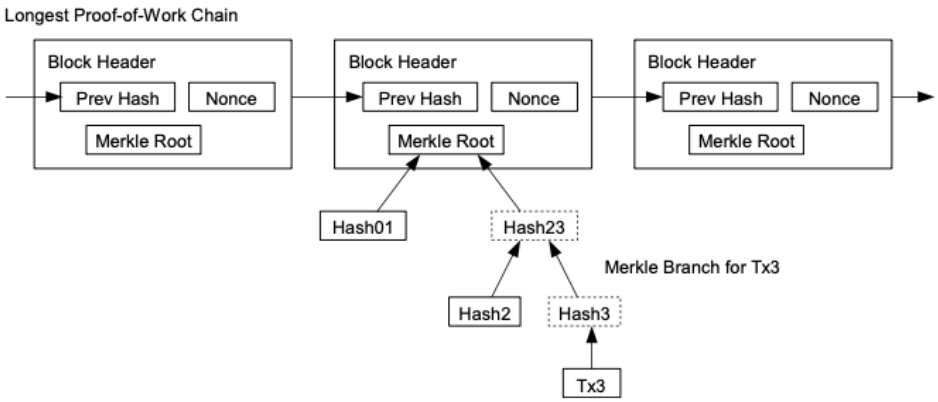


Transactions Hashed in a Merkle Tree — After Pruning Tx0-2 from the Block

Bloko antraštė be operacijų sudarytų apie 80 baitų. Jei manome, kad blokai generuojami kas 10 minučių, 80 baitų * 6 * 24 * 365 = 4,2 MB per metus. Nuo 2008 m. kompiuterių sistemose paprastai parduodama 2 GB operatyviosios atminties, o pagal Mūro dėsnį prognozuojamas dabartinis augimas
1,2 GB per metus, saugojimas neturėtų kelti problemų, net jei blokų antraštės turi būti saugomos atmintyje.

8. Supaprastintas mokėjimų tikrinimas

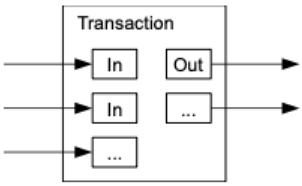
Mokėjimus galima patikrinti neįjungus viso tinklo mazgo. Vartotojui tereikia turėti ilgiausios darbo įrodymo grandinės blokų antraščių kopiją, kurią jis gali gauti užklausdamas tinklo mazgus, kol įsitikins, kad turi ilgiausią grandinę, ir gauti Merkle'io atšaką, susiejančią sandorį su bloku, kuriame jis pažymėtas laiko žyma. Jis negali pats patikrinti sandorio, bet susiejęs jį su vieta grandinėje, gali matyti, kad tinklo mazgas jį priėmė, o po jo pridėti blokai toliau patvirtina, kad tinklas jį priėmė.



Todėl patikra yra patikima tol, kol tinklą kontroliuoja sąžiningi mazgai, tačiau ji yra labiau pažeidžiama, jei tinklą užvaldo užpuolikas. Nors tinklo mazgai gali patys tikrinti sandorius, supaprastintą metodą gali apgauti užpuoliko sufabrikuoti sandoriai tol, kol užpuolikas gali užvaldyti tinklą. Viena iš strategijų, kaip nuo to apsisaugoti, būtų priimti tinklo mazgų perspėjimai, kai jie aptinka negaliojantį bloką, ir paskatinti naudotojo programinę įrangą atsisiųsti visą bloką ir perspėjamąsias operacijas, kad būtų patvirtintas neatitikimas. Dažnai mokėjimus gaunantys klientai tikriausiai vis tiek norės paleisti savo mazgus, kad būtų užtikrintas nepriklausomas saugumas ir greitesnis patikrinimas.

9. Vertės derinimas ir skaidymas

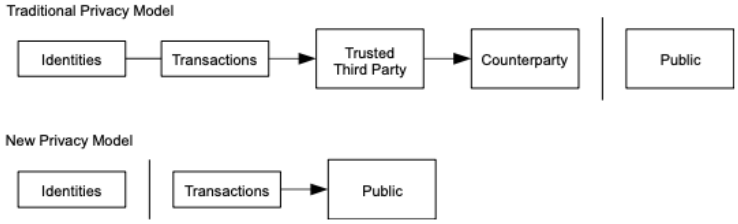
Nors monetas būtų galima tvarkyti atskirai, būtų nepatogu atlikti atskirą operaciją dėl kiekvieno pervedimo cento. Kad būtų galima padalyti ir sujungti vertę, sandoriuose yra kelios įvestys ir išvestys. Paprastai yra vienas įėjimas iš didesnio ankstesnio sandorio arba keli įėjimai, jungiantys mažesnes sumas, ir ne daugiau kaip du išėjimai: vienas skirtas mokėjimui, o kitas grąžina grąžą, jei tokia yra, atgal siuntėjui.



Reikėtų pažymėti, kad "fan-out", kai sandoris priklauso nuo kelių sandorių, o tie sandoriai priklauso nuo daugelio kitų, čia nėra problema. Niekada nereikia išgauti visos atskiros sandorio istorijos kopijos.

10. Privatumas

Tradicinis bankininkystės modelis užtikrina tam tikrą privatumo lygį, nes prieiga prie informacijos suteikiama tik dalyvaujančioms šalims ir patikimai trečiajai šaliai. Būtinybė viešai skelbti apie visas operacijas neleidžia taikyti šio metodo, tačiau privatumas vis tiek gali būti išlaikytas informacijos srautą nutraukiant kitoje vietoje: viešus raktus laikant anonimiškais. Visuomenė gali matyti, kad kažkas siunčia tam tikrą sumą kitam asmeniui, tačiau be informacijos, siejančios sandorį su bet kuriuo asmeniu. Tai panašu į akcijų biržų skelbiamos informacijos lygį, kai viešai skelbiamas atskirų sandorių laikas ir dydis, "juosta", tačiau nenurodoma, kas buvo sandorio šalys.



Kaip papildoma užkarda kiekvienai operacijai turėtų būti naudojama nauja raktų pora, kad jos nebūtų susietos su bendru savininku. Tam tikro susiejimo vis tiek neišvengiama, kai atliekamos kelių įėjimų operacijos, kurios būtinai atskleidžia, kad jų įėjimai priklausė tam pačiam savininkui. Rizika yra ta, kad atskleidus rakto savininką, susiejimas gali atskleisti kitas operacijas, kurios priklausė tam pačiam savininkui.

11. Skaičiavimai

Nagrinėjame scenarijų, kai užpuolikas bando sugeneruoti alternatyvią grandinę greičiau nei sąžininga grandinė. Net jei tai pavyksta, tai nesudaro prielaidų savavališkiems sistemos pokyčiams, pavyzdžiui, vertės sukūrimui iš oro arba pinigų, kurie niekada nepriklausė užpuolikui, paėmimui. Mazgai nepriims negaliojančio sandorio kaip mokėjimo, o sąžiningi mazgai niekada nepriims bloko su jais. Užpuolikas gali tik bandyti pakeisti vieną iš savo paties sandorių, kad atsiimtų neseniai išleistus pinigus.

Varžybas tarp sąžiningos grandinės ir užpuoliko grandinės galima apibūdinti kaip Binominį atsitiktinį ėjimą. Sėkmės įvykis yra sąžiningos grandinės pratęsimas vienu bloku, padidinant jos pranašumą +1, o nesėkmės įvykis yra užpuoliko grandinės pratęsimas vienu bloku, sumažinant atotrūkį -1.

Tikimybė, kad užpuolikas pasivys tam tikrą deficitą, yra analogiška lošėjo žlugimo problemai. Tarkime, kad neribotą kreditą turintis lošėjas pradeda nuo deficito ir atlieka potencialiai begalinį skaičių bandymų, kad pasiektų nuostolį. Tikimybę, kad jis kada nors pasieks pelną arba kad užpuolikas kada nors pasivys sąžiningą grandinę, galime apskaičiuoti taip [8]:

p = tikimybė, kad sąžiningas mazgas ras kitą bloką
q = tikimybė, kad užpuolikas ras kitą bloką
q_z = tikimybė, kad užpuolikas kada nors pasivys atsilikusį nuo z blokų

$$q_z = \begin{cases} 1 & \text{jei } p \le q \\ (q/p)^z & \text{jei } p > q \end{cases}$$

Atsižvelgiant į mūsų prielaidą, kad $p > q$, tikimybė mažėja eksponentiškai, nes didėja blokų, kuriuos užpuolikas turi pasivyti, skaičius. Jei puolėjas, turėdamas tokius šansus, anksti nesugebės laimingai išsiveržti į priekį, jo šansai vis labiau atsiliekant tampa nykstamai maži.

Dabar svarstome, kiek laiko naujo sandorio gavėjas turi laukti, kad įsitikintų, jog siuntėjas negali pakeisti sandorio. Darome prielaidą, kad siuntėjas yra užpuolikas, kuris nori priversti gavėją kurį laiką manyti, kad jis jam sumokėjo, o praėjus tam tikram laikui pakeisti jį ir sumokėti sau. Gavėjas bus įspėtas, kai tai įvyks, tačiau siuntėjas tikisi, kad bus per vėlu.

Gavėjas sukuria naują raktų porą ir viešąjį raktą perduoda siuntėjui prieš pat pasirašymą. Tai neleidžia siuntėjui iš anksto pasiruošti blokų grandinės, nuolat dirbant su ja tol, kol jam pasiseka nueiti pakankamai toli į priekį, ir tuomet įvykdyti sandorį. Išsiuntęs sandorį, nesąžiningas siuntėjas pradeda slapta dirbti su lygiagrečia grandine, kurioje yra alternatyvi jo sandorio versija.

Gavėjas laukia, kol sandoris bus įtrauktas į bloką ir po jo bus susieti z blokai. Jis nežino tikslaus užpuoliko padarytos pažangos dydžio, tačiau darant prielaidą, kad sąžiningi blokai užtruko vidutiniškai tiek laiko, kiek tikimasi vienam blokui, galima užpuoliko pažanga bus Poissono pasiskirstymas su tikėtina verte:

$$\lambda = z\frac{q}{p}$$

Kad gautume tikimybę, jog užpuolikas vis dar gali pasivyti dabar, padauginsime Poissono tankį kiekvienam jo galimai padarytos pažangos dydžiui iš tikimybės, kad jis gali pasivyti nuo to taško:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

Pertvarkykite, kad išvengtumėte begalinės pasiskirstymo uodegos sumavimo...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Konvertavimas į C kodą...

```c
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Atlikę tam tikrus rezultatus matome, kad tikimybė mažėja eksponentiškai su z.

```
q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012

q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

Sprendžiant, kai P mažesnis nei 0,1 %...

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

12. Išvada

Mes pasiūlėme elektroninių sandorių sistemą, kuri nesiremia pasitikėjimu. Pradėjome nuo įprastos monetų, pagamintų iš skaitmeninių parašų, sistemos, kuri užtikrina stiprią nuosavybės kontrolę, tačiau yra neišbaigta be būdo užkirsti kelią dvigubam išlaidavimui. Kad tai išspręstume, pasiūlėme lygiaverčių vartotojų tinklą, kuriame naudojamas darbo įrodymas (angl. proof-of-work), kad būtų įrašoma vieša sandorių istorija, kurią pakeisti užpuolikui greitai tampa skaičiavimo požiūriu nepraktiška, jei sąžiningi mazgai kontroliuoja didžiąją dalį procesoriaus galios. Tinklas yra patikimas dėl savo nestruktūrizuoto paprastumo. Mazgai dirba visi vienu metu ir mažai koordinuojami. Jų nereikia identifikuoti, nes pranešimai nėra nukreipiami į konkrečią vietą ir turi būti pristatomi tik dedant geriausias pastangas. Mazgai gali palikti tinklą ir vėl prie jo prisijungti savo nuožiūra, priimdami darbo įrodymo grandinę kaip įrodymą, kas įvyko, kol jų nebuvo. Jie balsuoja savo procesoriaus galia, išreikšdami savo pritarimą galiojantiems blokams, dirbdami prie jų pratęsimo, ir atmesdami negaliojančius blokus, atsisakydami prie jų dirbti. Naudojant šį konsensuso mechanizmą galima įgyvendinti bet kokias reikiamas taisykles ir paskatas.

Nuorodos

[1] W. Dai, "b-money", http://www.weidai.com/bmoney.txt, 1998 m.

[2] H. Massias, X.S. Avila ir J.-J. Quisquater, "Design of a secure timestamping service with minimum trust requirements", In *20th Symposium on Information Theory in the Benelux,* May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document", In *Journal of Cryptology*, vol 3, no 2, p. 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Skaitmeninio laiko žymėjimo efektyvumo ir patikimumo didinimas", In *Sequences II: Methods in Communication, Security and Computer Science*, p. 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Saugūs bitų eilučių vardai", In *Proceedings of the 4th ACM Conference on Computer and Communications Security,* p. 28-35, 1997 m. balandis.

[6] A. Back, "Hashcash - kovos su paslaugų atsisakymu priemonė", http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems", In *Proc. 1980 Symposium on Security and Privacy,* IEEE Computer Society, p. 122-133, 1980 m. balandis.

[8] W. Feller, "Įvadas į tikimybių teoriją ir jos taikymą", 1957 m.