Bitcoin: Lygiarangių Tinklų (P2P) Piniginė Sistema

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

Translated in Lithuanian from bitcoin.com/bitcoin.pdf by DDomas

Santrauka. Lygiarangiais tinklais paremti virtualūs pinigai leistų atlikti mokėjimus tiesiogiai tarp dviejų šalių be jokių finansinių institucijų. Skaitmeniniai parašai tik dalinis sprendimas, tačiau pagrindiniai privalumai tampa nereikšmingi, jeigu trečioji šalis reikalinga norint išvengti dvigubų išlaidų. Mūsų turimas sprendimas išsprendžia dvigubų išlaidų problemą naudojant lygiarangių tinklų sistemą. Laiku ženklintos transakcijos išsaugomos grandinėje, naudojančioje šifruotą darbo įrodymą (*ang.* proof-of-work), tokiu būdu suformuojančios įrašą, kuris negali būti pakeistas neperdarant proof-of-work. Ilgiausia grandinė yra ne tik įvykių įrodymas, bet ir įrodymas jog jiems įrašyti buvo naudojami didžiausi kompiuteriniai resursai (*ang.* CPU). Kol didžiausi kompiuteriniai resursai naudojami tų mazgų (*ang.* nodes), kurių tikslai nėra atakuoti tinklą, tol jie prisidės kuriant ilgiausią grandinę ir pralūgs kenkėjus. Pačiam tinklui reikalinga minimali struktūra. Pranešimai transliuojami geriausių pastangų būdu (*ang.* best effort) ir mazgai gali betkuriuo metu palikti ir vėl prisijungti prie tinklo priimdami grandinę su ilgiausiu proof-of-work kaip įrodymą, kas nutiko mazgui esant atsijungus.

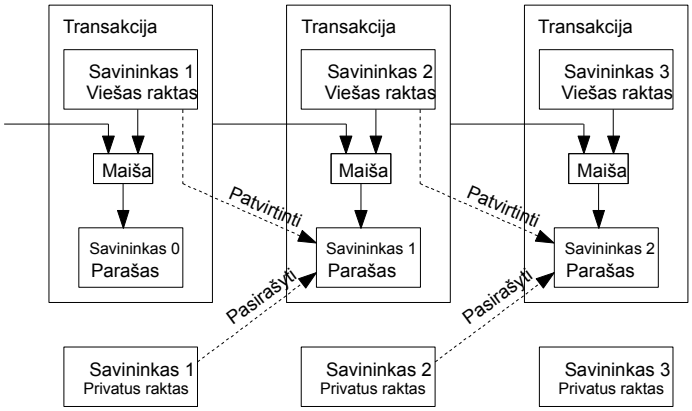
1. Įžanga

Internetinė prekyba beveik tapo visiškai priklausoma nuo finansinių institucijų, kuriomis tenka pasitikėti norint atlikti elektroninius mokėjimus. Kol ši sistema veikia daugumai transakcijų, ji neišvengiamai kenčia nuo silpnybių, kurios randamos pasitikėjimo modeliuose. Negrįžtamos transakcijos neįmanomos, kadangi finansinės institucijoms tenka įsiterpti nesusipratimų atvejais. Tarpininkavimas didina transakcijų mokesčius, mažina minimalų praktišką transakcijos dydį ir daro įprastas mažo dydžio transakcijas neįmanomas. Dar didesni mokesčiai nusimato prarandant galimybę atlikti negrįžtamą mokėjimą už negrįžtamas paslaugas. Norint grįžtamumo, didėja reikalas pasitikėti. Pardavėjai nepasikliaudami savo klientais, reikalauja daugiau informacijos nei jiems išties reikia. Priimta, jog tam tikra dalis siukčiavimų yra neišvengiama. Šių išlaidų ir mokėjimų neaiškumai išvengiami tik naudojam fizinę valiutą, tačiau neegzistuoja mechanizmas, kuriuo galima atlikti internetinius mokėjimus be tarpininkų, kuriais reikėtų pasitikėti.

Todėl reikalinga elektroninė mokėjimų sistema, naudojanti kriptografinius skaičiavimus vietoj pasitikėjimo, kas leidžia dviems šalims atlikti mokėjimus be tarpininkų. Reikalaujant didelių kompiuterinių resursų norint pakeisti transakcijas, nuo sukčiavimų būtų apsaugomi pardavėjai, o naudojant salyginio deponavimo sutarčių mechanizmus būtų galima apsaugoti ir pirkėjus. Šiame dokumente dalinamės sprendimu dviguboms išlaidoms, naudojant lygiarangių tinklų serverį su laiko atžymomis tam, kad sukurti kompiuterinį chronologišką transakcijų įrodymą. Ši sistema yra saugi, kol sąžiningi mazgai bendrai turi daugiau skaičiavimo galios negu gruopuotė kenkėjų.

2. Transakcijos

Skaitmeninę monetą nusakome skaitmeninių parašų grandine. Monetos savininkas persiunčia ją kitam skaitmeniniu būdu pasirašydamas ankstesnės transakcijos maišą (*ang.* hash) su viešu gavėjo raktu ir pridedamas juos prie monetos galo. Gavėjas gali patikrinti parašus norėdamas patikrinti visą anktesnių savininkų grandinę.

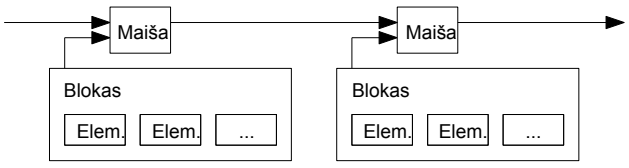


Visgi problema išlieka, jog gavėjas negali patikrinti, kad kažkuris iš ankstensių savininkų nepanaudojo monetos dukart. Galimas sprendimas yra naudoti patikimą centrinę instituciją, kalyklą, kuri tikrina kiekvieną transakciją, jog ji nebūtų atlikta dukart. Po kiekvienos transakcijos, moneta privalo būti sugrąžinta į kalyklą tam, kad nauja moneta būtų išduota ir galima užtikrinti, jog jos nėra naudotos dukart. Visgi ta pati problema egzistuoja, jog visa piniginė sistema priklauso nuo kalykos, kurią naudojant atliekama kiekviena transakcija, visai kaip naudojantis bankais.

Mums reikalingas būdas kuriuo gavėjas galėtų būti užtikrintas jog ankstesni monetos savininkai nepasirašė jokių ankstesnių transakcijų. Mums aktuali tik ankčiausia transakcija, todėl mums nerūpi vėlyvesni dvigubų išlaidų bandymai. Vienintelis būdas patikrinti trūkstamą transakciją yra žinant visas transakcijas. Kalyklos atveju, kalykla žino apie kiekvieną transakciją ir gali nuspręsti, kuri moneta atvyko pirmoji. Norint tai atlikti be tarpininkų, transakcijos privalo būti paviešinamos [1] ir mums reikalinga sistema, kurioje visi dalyviai sutinka dėl vienos versijos eiliškumo, kuriuo buvo vykdomos transakcijos. Gavėjui reikalingas įrodymas, kad kiekvienos transakcijos metu dauguma mazgų sutiko, jog tai pirmoji gauta transakcija.

3. Laiko žymų Serveris

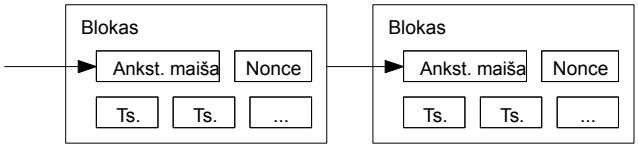
Mūsų siūlomas sprendimas prasideda nuo laiko žymų serverio. Laiko žymų serveris naudoja bloko ir jo elementų maišą ir publikuoja šią maišą, taip, kaip daroma laikraščiuose arba Usenet post[2-5] funkcija. Laiko žyma įrodo, jog informacija egzistavo tuo konkrečiu metu tam, kad galėtų pakliūti į maišą. Kiekviena laiko žyma turinti ankstesnę laiko žymą savo maišoje, tokiu būdu formuoja grandinę, kurioje kiekviena nauja laiko žyma sustiprina anksesniąsias.



4. Proof-of-Work

Tam kad sukurti paskirstytą laiko žymų serverį lygiarangių tinklų sistemoje, mums reikalinga proof-of-work sistema, kaip Adam Back Hashcash, vietoj laikraščių ar Usenet post. Į proof-of-work įeina reikšmės skenavimas, kurios maiša naudojant SHA-256, prasideda nuo skaičiaus su nuliniais bitais. Vidutinis reikalingas darbas auga eksponentiškai priklausomai nuo nulinių bitų skaičiaus ir gali būti patikrinamas įvykdant vieną maišą.

Mūsų laiko žymos tinklui sukuriame proof-of-work padidindami kintamąjį (*ang.* nonce) bloke iki kol reikšmė surasta, kurios bloko maiša pateikia reikalingą kiekį nulinių bitų. Kai CPU resursai naudojami atliekant proof-of-work, šis blokas negali būti pakeistas nedarant viso darbo iš naujo. Kadangi visi blokai yra sujungiami, norint pakeisti vieną bloką, reikia atlikti darbą visiems sekantiems blokams.



Proof-of-work taip pat padeda nustatant atstovavimą daugumos sprendimo metu. Jeigu dauguma susideda iš visų balsų, kur vienas IP adresas turi vieną balsą, tokia sistema gali būti apeinama kiekvieno galinčio sukurti daugiau IP adresų vienam įrenginiui. Iš esmės tai proof-of-work yra vienas CPU – vienas balsas. Daugumos sprendimas matomas ilgiausioje grandinėje, kuri turi didžiausią kiekį proof-of-work, kuris buvo naudojamas ją kuriant. Jeigu dauguma CPU galios valdoma sąžiningų mazgų, tuomet ši grandinė augs greičiausiai ir pralenks visas konkuruojančias grandines. Norint pakeisti ankstenį bloką, užpuolikas turėtų perdaryti proof-of-work tam blokui ir visiems sekantiems blokams, o tada prisivyti ir pralenkti darbą, atliekamą sąžiningų mazgų. Vėliau pademonstruosime, jog tikimybė, kad lėtesnis kenkėjas prisivys mažėja eksponentiškai, kadangi nauji blokai yra pridedami.

Tam, kad kompensuotume greitėjančią kompiuterinę įrangą ir besikeičiantį susidomėjimą mazgo dalyvavimu tinkle, proof-of-work sudėtingumas nustatomas keičiant vidutinį objektą ir vidutinį blokų skaičių per valandą. Jeigu jie sukuriami per greitai, tuomet sudėtingumas didėja.

5. Tinklas

Tinklo dalyviai atlieka šiuos veiksmus:

1) Naujos transakcijos transliuojamos visiems mazgams.
2) Kiekvienas mazgas sugrupuoja naujas transakcijas į bloką.
3) Kiekvienas mazgas atlieka proof-of-work jo blokui.
4) Kai mazgas galiausiai suranda proof-of-work, jis transliuoja bloką kitiems mazgams.
5) Mazgai priima bloką tik tuo atveju, jeigu visos transakcijos jame yra galiojančios ir neišleistos.
6) Priimdami galiojantį bloką, mazgai pradeda darbą sekančiam blokui grandinėje, pastarojo bloko maiša yra naudojama sekančiame bloke.

Mazgai laiko ilgiausią grandinę teisinga ir deda pastangas ją prailginant. Jeigu du mazgai transliuos skirtingas sekančio bloko versijas tuo pat metu, kai kurie mazgai gali gauti betkurį iš jų. Tuo atveju jie dirba su tuo, kurį gavo, tačiau išsaugo ir kitą tuo atveju, jeigu jis bus pratęsiamas. Lygiosios nutraukiamos, kai sekantis proof-of-work surastas ir viena iš atšakų prailginama; mazgai kurie sekė kitą atšaką prisijungs prie tos, kuri ilgesnė.

Naujų transakcijų transliacijos neprivalo pasiekti visus mazgus. Kol jos pasiekia bent kiek mazgų, anksčiau ar vėliau jos atsidurs bloke. Bloko transliacijos toleruoja žinučių praradimą. Jeigu mazgas negavo bloko, jis pareikalaus jo kai gaus sekantį bloką, suprasdamas jog trūksta vieno bloko.

6. Paskata

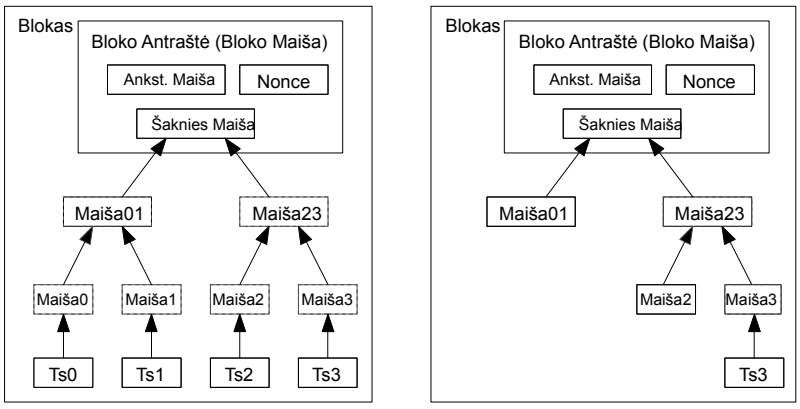
Pirma transakcija bloke yra svarbi ir ji sukuria naują monetą, kuri priklauso to bloko kūrėjui. Tai motyvuoja mazgus palaikyti tinklą ir tokiu būdu monetos įtraukiamos į apyvartą, kadangi nėra jokios centrinės institucijos joms išduoti. Pastovi įnaša stabilaus kiekio naujų monetų yra analogija aukso kasėjų resursų naudojimui norint įtraukti daugiau aukso į apyvartą. Šiuo atveju eikvojami CPU laikas ir elektra.

Paskata taip pat yra transakcijų mokesčių formos. Jeigu transakcijos išvesties suma mažesnė nei įvesties, šis skirtumas yra transakcijos mokestis, kuris pridedamas prie bloko atlygio. Kai visos numatytosios monetos pateks į apyvartą, paskata nukryps tik į transakcijos mokesčius ir infliacija taps nežymi.

Paskatinimai turėtų motyvuoti mazgus išlikti sąžiningais. Jeigu šykštus kenkėjas galėtų surinkti daugiau CPU galios nei visi sąžiningi mazgai, tuo atveju jis galėtų rintis tarp susigrąžinimo visų savo mokėjimų arba naudoti šiuos resursus naujų monetų kūrimui. Jam labiau naudinga žaisti pagal taisykles, kurios parankios jam dėl daugiau naujų monetų nei visų kitų kartu sudėjus, negu pakenkti sistemos ir savo paties gerovei.

7. Disko Vietos Susigrąžinimas

Kai paskutinė transakcija yra pakankamai apsupta kitų blokų, panaudotos transakcijos gali būti atmestos norint sutaupyti vietos diske. Norint tai padaryti nepakeičiant bloko maišos, transakcijos šifruojamos naudojant Merkle Tree [7][2][5], į bloko maišą įtraukiant tik šaknį. Seni blokai taip suspaudžiami apleidžiant medžio šakas. Vidinės maišos neprivalo būti saugomos.

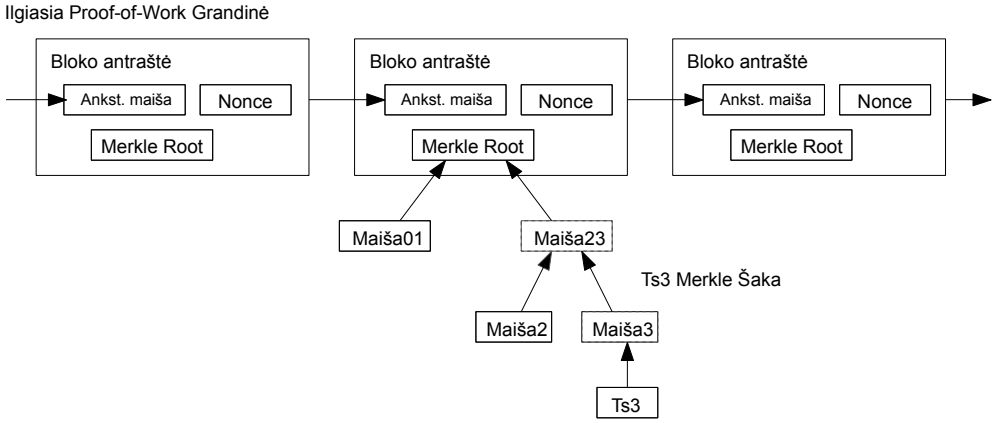


Merkle Tree Tranzakcijų Maiša Po Ts2 iškirpimo iš bloko

Be jokių transakcijų bloko antraštė užimtų 80 baitų. Tarkime, jog blokai sukuriami 10 minučių, 80 baitų * 6 * 24 * 365 = 4.2MB per metus. Naudojant 2008 metų kompiuterius su 2GB darbine atmintimi (RAM) ir remiantis Moore's Law apskaičiavimais, pasak kurių 1.2GB augimo per metus tempu, neturėtų kilti problemų dėl atminties net jeigu ir antraštės būtų saugojamos atmintyje.

8. Supaprastintas Mokėjimų Patvirtinimas

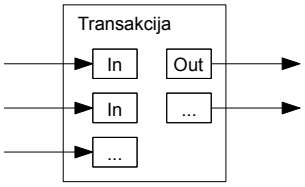
Patvirtinti mokėjimus galima ir neturint viso mazgo savo įrenginyje. Vartotojui tereikia turėti ilgiausios proof-of-work grandinės blokų antraščių kopiją, kurias galima gauti tikrinant su tinklo mazgais iki kol yra saugu manyti jog vartotojas turi ilgiausią grandinę ir gauti Merkle šaką kuri nurodo į tranzakciją kurios bloke ji pažymėta laiko žyma. Jis pats negali patikrinti transakcijos, tačiau nurodant į tinkamą vietą grandinėje, jis gali matyti, jog tinklo mazgas priėmė transakciją ir blokai po jo tik dar kartą patvirtina, jog transakcija buvo priimta.



Kol tinklas yra valdomas sąžiningų mazgų, tol patikrinimas yra patikimas, tačiau jis labiau pažeidžiamas, jeigu tinklas perimtas kenkėjų. Nors tinklo mazgai ir gali patikrinti kiekvieną transakciją patys, tačiau šis metodas gali būti padirbtas jeigu kenkėjai gali kontroliuoti didžiąją tinklo dalį. Vienas iš būdų kaip apsisaugoti nuo tokių nesklandumų yra gaunant perspėjimus iš kitų mazgų, kurie gauna negaliojantį bloką, kas taip pat skatintų parsisiųstį visą bloką ir visas transakcijas, kurios gali būti negaliojančios. Įmonės, kurios dažnai daro mokėjimus turėtų saugoti pilną tinklo mazgą tam, kad užtikrintų saugumą ir greitesnį transakcijų ir blokų patvirtinimą.

9. Sumos Sujungimas ir Išskirstymas

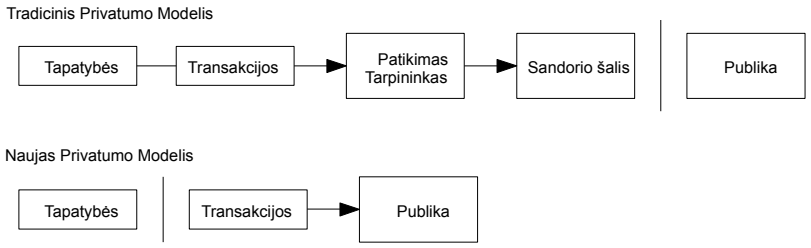
Net jei ir įmanoma apdirbti kiekvieną monetą individualiai, atliekant atskirą transakciją kiekvienam centui būtų per daug sudėtinga. Norint išskirstyti ir sudėti visą sumą, transakcijos turi keletą įvesčių (*ang.* in) ir išvesčių (*ang.* out). Įprastai tai būtų arba viena įvestis iš didesnės transakcijos atliktos anksčiau arba keletas įvesčių iš mažesnių kiekių ir ne daugiau nei dvi išvestys: pirma tai pats mokėjimas ir antra – jeigu liko, graža siuntėjui.



Tai visiškai įprasta, jog transakcija gali priklausyti nuo keleto kitų transakcijų, kurios taip pat priklauso nuo kitų transakcijų. Niekad nereikia išgauti galutinės transakcijos kopijos.

10. Privatumas

Naudojant tradicinį bankininkavimo modelį, tam tikras privatumo lygis pasiekiamas ribojant tarpininkų ir dalyvaujančių šalių prieigą prie informacijos. Šis būdas nereikalingas, jeigu visos transakcijos prieinamos viešai, bet privatumas išlaikomas, jeigu vieši raktai lieka anonimiškais. Visi gali sekti, jog vyksta persiuntimas, tačiau jokia informacija apie siuntėjus ir gavėjus nėra prieinama. Visai kaip informacija pateikiama vertybinių popierių biržoje (*ang.* stock market), kurioje kiekvienas sandėrio laikas ir dydis pateikiamas viešai „juostoje“, tačiau nepateikiant, kas yra jo dalyviai.



Kaip papildoma ugniasienė, kiekviena kartą atliekant transakciją, nauja raktų pora turėtų būti naudojama. Kai kuriais atvejais susiejimas neišvengiamas, kai transakcija susideda iš keleto įvesčių, kas parodo jog jos kyla iš to paties savininko. Rizika yra tame, jog jeigu savininkas yra paviešintas, susiejimas gali atskleisti ir kitas transakcijas atliktas to paties savininko.

11. Skaičiavimai

Tarkim, jog kenkėjas bando sukurti atskirą grandinę greičiau, negu tikrąją grandinę. Net jeigu jam pasiseka, tai nereiškia, jog sistema pasikeis ir pradės kurti naudą iš oro arba gauti pinigus, kurie niekad nepriklausė kenkėjui. Mazgai niekad nepriims negaliojančios transakcijos kaip mokėjimo ir sąžiningi mazgai niekad nepriimtų bloko su tokiomis transakcijomis. Kenkėjas tik gali pabandyti pakeisti betkurią iš savo transakcijų, bandydamas atgauti išleistus pinigus.

Šios lenktynės tarp sąžiningų mazgų ir kenkėjo vadinamas Binomial Random Walk. Tikimybė, jog tikroji grandinė bus prailginta vienu bloku auga +1, o tikimybė jog kenkėjo grandinė bus prailginta vienu bloku mažina skirtumą -1.

Tikimybė jog kenkėjas išnaikints atsilikimą yra analogiškas Gambler's Ruin problemai. Tarkim, kad žaidėjas su neribotu kiekiu kreditų pradeda žaidimą su deficitu ir žaidžia neribotą skaičių žaidimų bandydamas išlyginti rezultatą. Galime apskaičiuoti tikimybę, kad jis kada nors išlygins rezultatą arba išvis prisivys su tikrąja grandine šiuo būdu [8]:

p = tikimybė, kad sąžiningas mazgas suras sekantį bloką
q = tikimybė, kad kenkėjas suras sekantį bloką
q_z = tikimybė, kad kenkėjas prisivys atsilikdamas z skaičių blokų

$$q_z = \begin{cases} 1 & if\ p \leq q \\ (q/p)^z & if\ p > q \end{cases}$$

Sprendžiant pagal tai, kad p > q, tikimybė eksponentiškai mažėja, kadangi blokų kiekis, kurį kenkėjui reikia pasivyti, pastoviai auga. Kadangi tikimybės yra ne kenkėjo naudai, jeigu jis nepradeda atakos iš anksto, sėkmės atvejis praktiškai visiškai išnyksta.

Pažiūrėkime kiek laiko gavėjui reikia pralaukti, jog jis būtų tikras, kas siuntėjas nepakeis transakcijos. Tarkim, kad siuntėjas yra kenkėjas, kuris bando įtikinti gavėją, kad jis atliko mokėjimą prieš kurį laiką, o tada apsukti mokėjimą atgal sau pačiam. Gavėjas bus perspėtas, kai tai nutinka, tačiau siuntėjas viliasi, jog tai jau bus per vėlu.

Gavėjas sukuria naują raktų porą ir pasidalina viešu raktu su siuntėju prieš pat pasirašydamas. Tokiu būdu esame tikri, jog siuntėjas nepasiruoš blokų grandinės iš anksto iki kol jam pasiseks pažengti pakankamai toli ir įvykdyti transakciją būtent tuo metu. Kai transakcija išsiūsta, siukčiaujantis siuntėjas paslapčiomis dirba su skirtinga tos pačios transakcijos versija.

Gavėjas laukia kol transakcija įtraukta į bloką ir z kiekis naujų blokų buvo prijungta prie jo. Jis tiksliai nežino kiek toli kenkėjas pažengė, tačiau praėjo vidutinis sąžiningų mazgų laikas kiekvienam blokui, kenkėjo sėkmės tikimybė paskaičiuojama kaip Poisson distribution formulę:

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

Norint surasti tikimybę, kad kenkėjas vis dar gali prisivyti, galime padauginti Poisson tankį kiekvienam proceso žingsniui su tikimybe, kad jis gali pasivyti nuo būtent tos vietos:

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

Pertvarkome, kad išvengtume begalinio paskirstymo sumavimo...

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

Išreiškiant į C kodą…

```
#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;
}
```

Atliekant bandymus matome, jog su z, tikimybė mažėja eksponentiškai.

```
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
```

Sprendimai, kur P mažiau 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 pateikėme elektroninių mokėjimų sistemą, kuri nereikalauja pasitikėjimo. Pradėjome nuo įprastos monetų su skaitmeniniais parašais struktūros, kuri suteikią kontrolę, tačiau nėra išbaigta, kadangi ji negali apsisaugoti nuo dvigubų išlaidų. Kad tai išspręstume, pasiūlėme lygiarangių tinklų sistemą, kuri naudoja proof-of-work tam, jog visų transakcijų vieša istorija būtų įregistruojama ir lengvai tampa per daug sudėtinga kenkėjų atakoms, kol didžioji dauguma CPU energijos kontroliuojama sąžiningų mazgų. Tinklas yra stiprus savo padrikame paprastume. Mazgai bendradarbiauja visi kartu ir beveik nereikalauja koordinacijos. Jiems nebūtinos tapatybės, kadangi žinutės nėra transliuojamos konkretiems gavėjams, bet pristatomos ten, kur įmanoma. Mazgai gali betkuriuo metu išeiti ir prisijungti, priimdami proof-of-work grandinę kaip įrodymą įvykių, kurie nutiko kol jie buvo neprisijungę. Jie išreiškia savo balsą naudodami savo CPU resursus, priimdami galiojančius blokus ir juos prailgindami ir atsisakydami prisidėti prie negaliojančių blokų - jie juos atmeta. Betkurios iš šių taisyklių ir paskatinimai taikomi naudojant bendro susitarimo (*ang.* consensus) mechanizmą.

Nuorodos

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

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal 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, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," *In Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," 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, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.