Bitcoin: Vastastikuline elektrooniline sularahasüsteem

Satoshi Nakamoto
satoshin@gmx.com
https://bitcoin.org/en/bitcoin-paper

Eesti keelde tõlkinud:
@Ekux, www.faval.eu
ekux.faval@protonmail.com

Üldistus. Täielikult vastastikuline (*peer-to-peer*) variant elektroonilisest sularahast võimaldaks *online*-maksete saatmise otse ühelt osapoolelt teisele, ilma, et need peaksid finantsasutusi läbima. Digiallkirjade näol on tegemist osalise lahendusega, kuid peamised hüved kaovad, sest topeltkulutamise ära hoidmiseks on siiski vaja usaldatavat kolmandat osapoolt.
Vastastikulisusel põhineva võrgustiku näol pakume lahendust topeltkulutamise probleemile. Võrgustik määrab tehingud ajaliselt, sättides need jätkuvasse räsitud töö-tõestuse ahelasse, moodustades registri, mida ei saa ilma tehtud töö uuesti tegemiseta muuta. Pikim ahel mitte üksnes ei toimi aset leidnud sündmustejada tõestusena vaid ka tõestusena, et see ahel on loodud kõrgeima CPU-võimsuse abil. Nii kaua kui suurema osa CPU-võimsuse haldajatest (ühenduspunktidest) ei tee võrgustiku ründamiseks koostööd, tekitavad nemad kõige pikema ahela ning edendavad ründajaid. Võrgustiku enda algkonstruktsioon on minimaalne. Andmetel puudub järelevalve ning ühenduspunktid võivad soovi korral igal ajal võrgustikust lahkuda ja sellega uuesti liituda, käsitledes pikimat töö-tõestuse ahelat kui tõestust selle kohta, mis on aset leidnud aja vältel, mil nad eemal viibisid.

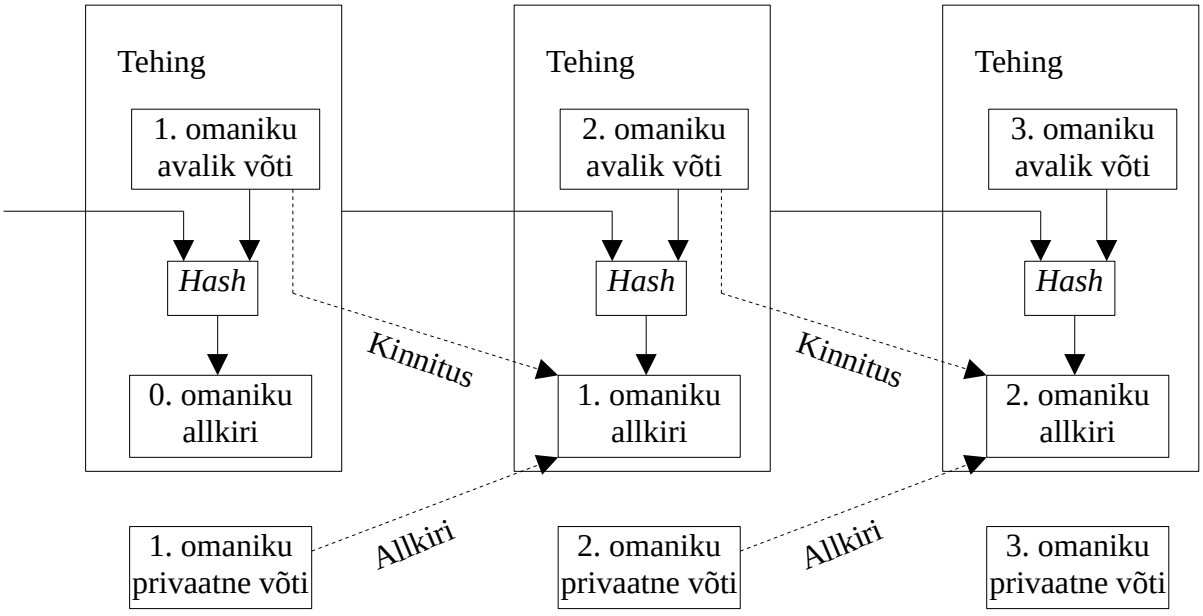
1. Sissejuhatus

Internetikaubandus on elektrooniliste maksete töötlemisel asunud peaaegu eranditult toetuma usaldatava kolmanda osapoole rolli täitvatele finantsasutustele. Olles

enamiku tehingute puhul piisavalt heaks lahenduseks, kannatab see süsteem siiski usaldussüsteemi mudelile omaste nõrkuste küüsis. Lõplikult pöördumatuid tehinguid ei ole võimalik teostada, sest finantsasutustel ei ole võimalik vaidlustamisi vältida. Vaidlustustele kuluvast ressursist tulenevalt suurenevad tehingutasud, piirates minimaalset praktilise tehingu suurust ja lõigates ära väikeste juhutehingute võimaluse ning pöördumatute teenuste eest pöördumatute maksetega tasumise võimekuse puudumisega kaasnevad samuti laiemad kulud. Taganemise võimalusega kaasneb usaldusvajaduse laialivalgumine. Kauplejad peavad oma klientide suhtes teadlikud olema, kiusates neid rohkemate teabenõuetega kui neil muul juhul vaja oleks. Teatud hulka pettust käsitletakse vältimatuna. Neid kulusid ja ebakindlusi on võimalik füüsilist sularaha kasutades vältida, kuid puudub mehhanism, mis võimaldaks teostada elektroonilisi makseid ilma usaldatava kolmanda osapooleta.
Vaja on süsteemi, mis põhineb krüptograafilisel tõestusel, mitte usaldusel, andes võimaluse kahel suvalisel osapoolel tehinguid sõlmida, vajamata selleks usaldatavat kolmandat osapoolt. Arvutuslikult ebapraktilised taganemisvõimalused kaitseksid kauplejaid pettuse eest ning rutiinne tingdeponeerimise (*escrow*) mehhanisme saaks lihtsasti ostjate kaitseks rakendada. Kasutades jagatud aegmääramise serverit, mis toodab tehingute kronoloogilise järjekorra arvutuslikku tõestust, pakume käesoleva paberiga lahendust topeltkulutamise probleemile. Süsteem on turvaline nii kaua kui ausad ühenduspunktid omavad kollektiivselt kontrolli mistahes koostöötavast ründegrupeeringust suurema CPU võimsuse üle.

2. Tehingud

Elektroonilist münti (*coin*) defineerime digitaalsete allkirjade ahelana. Iga omanik kannab mündi üle järgmisele omanikule allkirjastades digitaalselt sellega seotud oleva eelneva tehingu räsitud kujul (*hash*) ja järgmise omaniku avaliku võtme ning lisades selle rahaühiku lõppu. Makse saaja saab omandiahela õigsust tõendada läbi allkirjade kontrollimise.

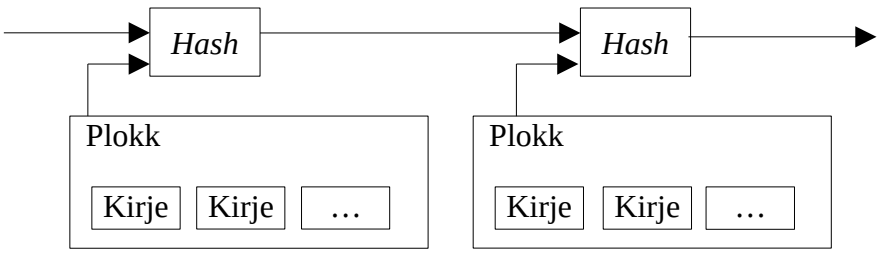


Probleem seisneb loomulikult selles, et makse saaja ei saa veenduda, et keegi eelnevatest omanikest poleks münti topeltkulutanud. Tavapärane lahendus sellele on sisse tuua keskne autoriteet, või rahapada, mis kontrollib kõiki tehinguid, et neis ei esineks topeltkulutamist. Iga tehingu järel tuleb münt rahapatta tagastada, et see uue mündi looks ning topeltkulutamise kindlaks peetakse üksnes neid münte, mis on sellest rahapajast pärit. Selle lahenduse probleem seisneb selles, et kuna kõik tehingud käivad panga kombel läbi selle rahapaja, siis on kogu rahasüsteemi saatus seda rahapada omava institutsiooni kätes.

Vaja on meetodit, mille abil saaks makse saaja veenduda, et eelnevad omanikud ei oleks ühtegi varasemat tehingut allkirjastanud. Kuna meie eesmärgipäraselt on just varasem tehing see, mida loetakse õigeks, siis me ei tunne huvi tuleviku topeltkulutamise katsete vastu. Ainus viis ühe tehingu puudumise tõestamiseks on olla teadlik kõikidest tehingutest. Rahapaja-tüüpi mudelis oli rahapada teadlik kõikidest tehingutest ning otsustas, millised toimusid esimesena. Et seda ilma keskse usalduspooleta saavutada, peavad kõik tehingud olema avalikult välja kuulutatud[1] ning vaja on süteemi, mis lubaks osalistel kokku leppida üks kindel tehingute ajalugu. Makse saajal on vaja tõestust, et tehingu toimumise hetkel nõustub enamus ühenduspunkte, et just see tehing toimus esimesena.

3. Aegmääramise server

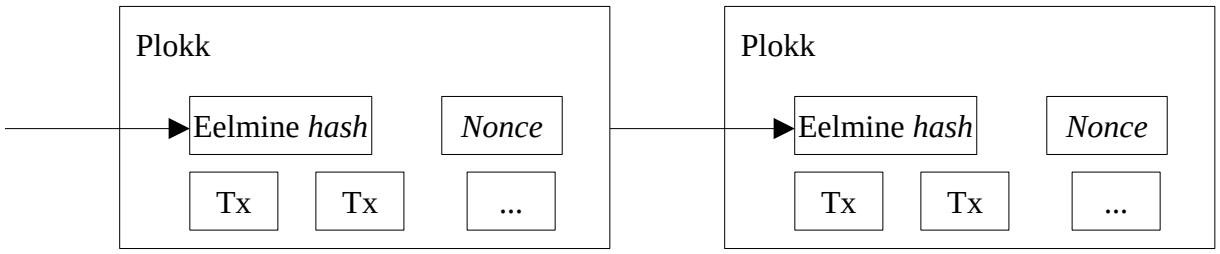
Meie pakutav lahendus algab aegmääramise serveriga. See võtab räsitud plokitäie kirjeid ajas määramiseks ning avalikustab need räsitud kujul, sarnaselt ajalehe või Usenet postitusega [2-5]. Ajatempel tõestab, et andmed pidid aja määramise hetkel eksisteerima, ilmselgelt, selleks, et neil oleks üldse võimalik olla osaks sellest räsist (*hashist*). Iga järgnev ajatempel sisaldab enda räsitud kujus eelnevat ajatemplit, moodustades ahela, kus iga järgnev ajatempel tugevab endale eelnevaid.



4. Töö-tõestus

Vastastikulisusel baseeruva jagatud aegmääramise serveri teostamiseks on meil erinevalt ajalehest või *Usenet* postitustest tarvis kasutusele võtta Adam Backi *Hashcash*ile[6] sarnanev töö-tõestuse süsteem. Niisugune töö-tõestuse süsteem hõlmab endas väärtuse skaneerimist, mis näiteks SHA-256 meetodil räsituna algab null-biti hulgaga. Keskmine vajalik tehtav töö on null-biti nõutavas hulgas eksponentsiaalne ja seda saab ühe ainsa räsi esitamisel kindlaks teha.

Aegmääramise võrgu jaoks rakendame töö-tõestust, kasvatades plokis sisalduva ühekordselt kasutatava elemendi(*nonce*) väärtust, kuni ploki *hash* sisaldab nõutud null-bittide hulka. Hetkel, mil CPU jõupingutust on töö-tõestuse saavutamiseks piisavalt laiendatud, ei ole ilma tehtud töö uuesti tegemiseta ploki sisu enam võimalik muuta. Kuna kõik järgnevad plokid on eelmiste külge aheldatud, tähendaks ühe ploki sisu muutmine vajadust muuta ka kõiki järgnevaid plokke.



Läbi enamuse otsustavuse lahendab töö-tõestus ka esitamise määramise probleemi. Kui enamuslikkus baseeruks üks-IP-üks-hääl mudelil, oleks seda võimalik õõnestada igaühel, kelle käsutuses on palju IP-sid. Töö-tõestus on sisuliselt üks-CPU-üks-hääl. Enamuse otsus esitatakse pikima ahelana, mis sisaldab kõige suuremat hulka töö-tõestuseks kulutatud jõupingutust. Kui enamus CPU võimsusest on legitiimsete ühenduspunktide käsutuses, siis kasvab legitiimne ahel kõige kiiremini ja edendab kõiki konkureerivaid ahelaid. Mistahes ploki sisu muutmiseks peaks ründaja uuesti tegema ploki ja kõikide sellele järgnevate plokkide loomiseks kulunud töö-tõestuse ning jõudma järele ja edendama ausate ühenduspunktide tehtavat tööd. Hiljem toome ka esile, kuidas aeglasema ründaja järele jõudmise tõenäosus uute plokkide lisamisega eksponentsiaalselt väheneb.

Kompenseerimaks kasvavat riistvara võimekust ja varieeruvat ühenduspunktide jooksutamise huvi, määratakse töö-tõestuse raskusaste liikuva muutuja alusel, mis tuletatakse keskmisest tunni jooksul loodavate uute plokkide arvust. Kui plokid tekivad liiga kiiresti, kasvab ka raskusaste.

5. Võrk

Võrgu jooksutamiseks vajalikud sammud on järgnevad:

1) Uued tehingud esitatakse kõikidele ühenduspunktidele.
2) Iga ühenduspunkt kogub uued tehingud plokki.
3) Iga ühenduspunkt töötab, et lahendada selle ploki keerukat töö-tõestuse ülesannet.
4) Kui ühenduspunkt lahendab töö-tõestuse ülesande, esitab see uue ploki kõikidele ühenduspunktidele.
5) Ühenduspunktid aksepteerivad uue ploki ainult siis, kui kõik selle sisalduvad tehingud on paikapidavad ja neid ei ole juba eelnevalt kulutatud.
6) Ühenduspunktid väljendavad uue ploki aksepteerimist läbi selle ploki *hashi* kasutamise eelmise ploki *hashina*, töötades seeläbi välja ahela järgmist plokki.

Ühenduspunktid käsitlevad tõese ahelana alati kõige pikemat ahelat ja töötavad selle pikendamise kallal. Kui kaks ühenduspunkti esitavad samast plokist samaaegselt kahte erinevat versiooni, võivad mõned ühenduspunktid saada ühe neist esimesena. Sellisel puhul hakatakse töötama selle ploki kallal, mis esimesena saadi, kuid jäetakse alles ka teine haru, juhuks kui see peaks pikemaks osutuma. Viik lõppeb siis, kui järgmine töö-tõestuse ülesanne lahendatakse ning ühest harust saab pikem; ühenduspunktid, mis töötasid teise haru kallal, lülituvad ümber pikemale harule.

Uued tehingud ei pea tingimata saama kõikidele ühenduspunktidele esitatud. Nii kaua kui nad jõuavad paljude ühenduspunktideni, saavad nad pikema viivituseta plokki lisatud. Ploki esitamised on vastuvõtlikud ka kaduma läinud teadete suhtes. Kui ühenduspunkt ei saa plokki kätte, esitab ta selle kohta päringu järgmist plokki vastu võttes ja saab aru, et tal on üks vahele jäänud.

6. Stiimul

Konventsionaalselt on ploki esimene tehing eriline tehing, mis tekitab uue mündi, mida omab ploki looja. See stimuleerib ühenduspunkte võrku toetama ja tekitab võimaluse müntide algse ringlusesse viimiseks, sest puudub keskne autoriteet, kes neid välja jagaks. Fikseeritud hulga uute müntide pidev lisandumine on võrreldav kullakaevandajatega, kes kulutavad ressurssi, et tuua ringlusesse uut kulda. Meie puhul on kulutatavaks ressursiks CPU tööaeg ja elektrienergia.

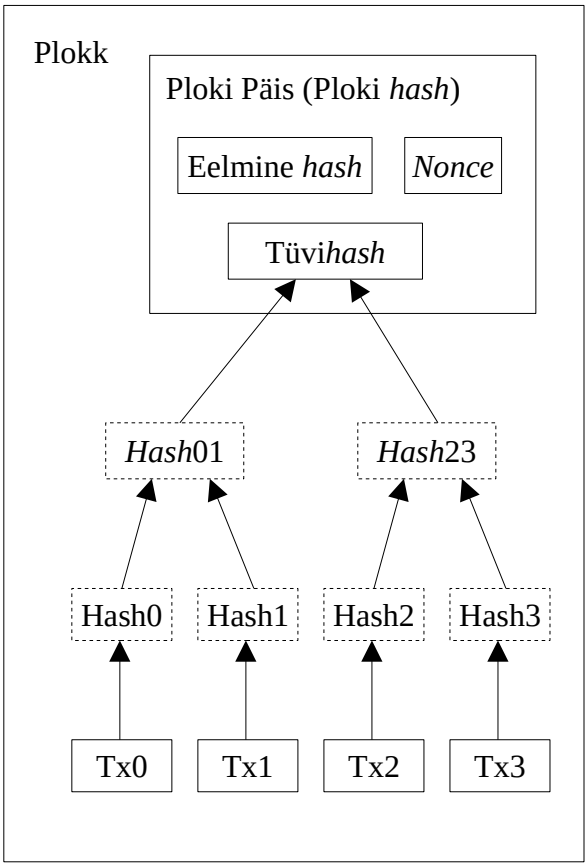
Stiimulit on võimalik rahastada ka läbi teenustasude. Kui tehingust väljuv väärtus on väiksem kui tehingusse sisenev väärtus, on tekkinud vahe teenustasu, mis lisatakse seda tehingut sisaldava ploki stiimuli väärtusesse. Kui ettemääratud müntide kogus on ringlusesse sisenenud, saab stiimul täielikult teenustasudele üle minna ja olla täielikult inflatsioonikindel.

Stiimul võib julgustada ühenduspunkte olema ausad. Kui ahne ründaja suudaks koondada rohkem CPU võimsust kui kõik ausad ühenduspunktid, peaks ta valima,

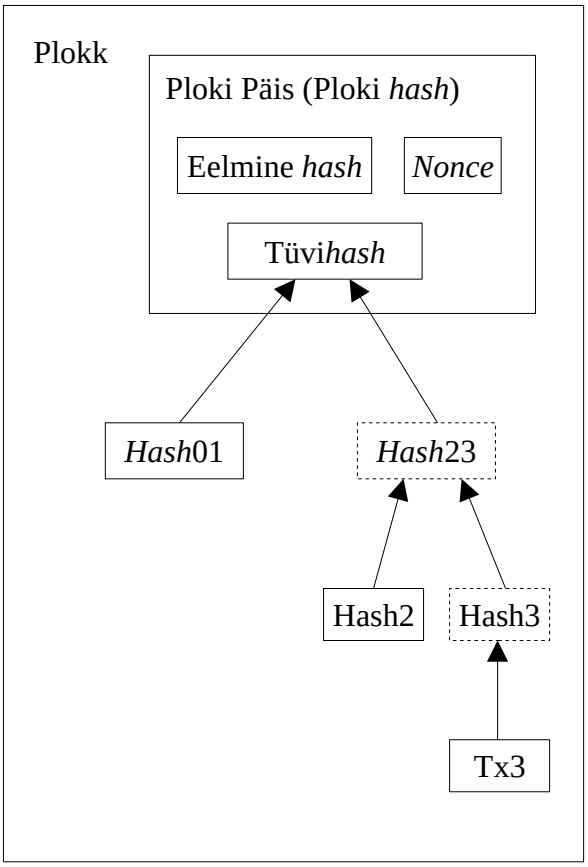
kas kasutada seda enda maksete tagasi võtmisega inimeste petmiseks või uute müntide genereerimiseks. Eeldatavasti leiab ta, et süsteemi ja tema enda rikkuse paikapidavuse õõnestamise asemel on kasumlikum mängida reeglite järgi, mis soosivad teda kõikidest teistest suurema hulga müntidega.

7. Ketta mahu taastamine

Kui mingi mündiga seonduv viimane tehing on ühel hetkel piisavalt paljude plokkide alla mattunud, võib ketta ruumi säästmiseks sellele eelnevad kulunud tehingud minema visata. Et sellele ilma ploki *hashi* lõhkumiseta kaasa aidata, on tehingud *hashitud* Merkle'i Puusse *(*ingl *Merkle Tree)*[7][2][5], kus ploki *hashi* kuulub üksnes selle puu tüvi. Vanu plokke saab selliselt, puu küljest harusid eemaldades, kokku pakkida. Sisemisi *hashe* ei ole vaja talletada.



Tehingud Merkle'i Puusse *hashituna*



Pärast Tx0-2 kärpimist Plokist

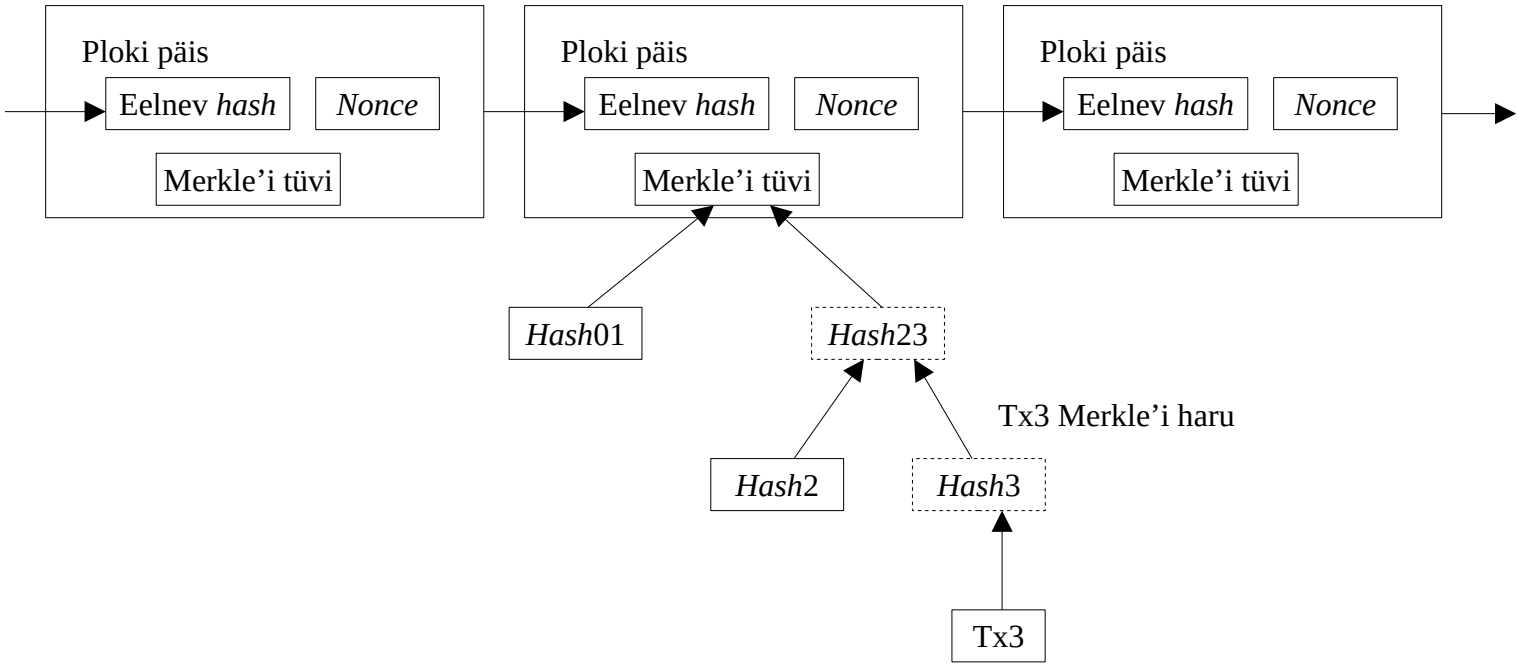
Ilma tehinguteta ploki päis oleks umbes 80 baiti suur. Kui eeldame, et iga 10 minuti järel luuakse uus plokk, 80 baiti * 6 * 24 * 365 = 4.2MB aasta kohta. 2008. aasta seisuga tüüpiliste arvutite puhul, mida müüakse 2GB RAM-ga ning Moore'i

Seadusega, mis ennustab jooksvat tõusu 1.2GB aasta kohta, ei tohiks mälumaht probleemiks osutuda isegi siis, kui ploki päiseid tuleks hoiustada vahemälus.

8. Lihtsustatud maksete kontrollimine

Makseid on võimalik kontrollida ka ilma täies mahus võrgu ühenduspunkti jooksutamiseta. Kasutajal pole vaja muud kui hoida pikima töö-tõestuse ahela plokkide päiste koopiat, mille ta saab, esitades päringuid võrgu ühenduspunktidele kuni ta on veendunud, et tema valduses on pikim ahel ning soetada Merkle'i haru, mis ühendab tehingu plokiga, milles see ajaliselt määratud on. Ta ei saa tehingut iseseisvalt kontrollida, kuid viies see kokku ahelas oleva kohaga, saab ta näha, et võrgus olev ühenduspunkt on selle aksepteerinud ning järgnevate plokkide olemasolu tõestab täiendavalt, et võrk on selle aksepteerinud.

Pikim töö-tõestuse ahel

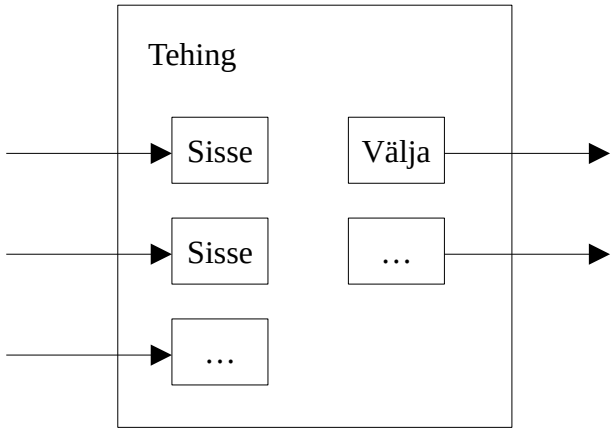


Selliselt on kontrollimine usaldusväärne nii kaua kui võrk on legitiimsete ühenduspunktide poolt kontrollitud, kuid on haavatavam siis kui võrgust käib üle ründaja jõud. Kuigi võrgu ühenduspunktid saavad tehinguid enda jaoks kinnitada, siis lihtsustatud meetodit on siiski võimalik ründaja fabritseeritud tehingutega petta nii kaua kui ründaja suudab jätkuvalt enda jõuga võrgust üle olla. Üks strateegia selle vastu kaitseks oleks võtta vastu võrgu ühenduspunktide teavitusi avastatud ebaõigete plokkide kohta, mis kutsub ebakõlade kinnitamiseks esile kasutaja tarkvara ploki ja kahtlusaluste tehingute täies mahus allalaadimise. Ettevõtted, mis võtavad vastu

sagedasi makseid, tahavad tõenäoliselt sõltumatuma turvalisuse ja kiirema kontrolli eesmärgil siiski ühenduspunkte jooksutada.

9. Väärtuste ühendamine ja jagamine

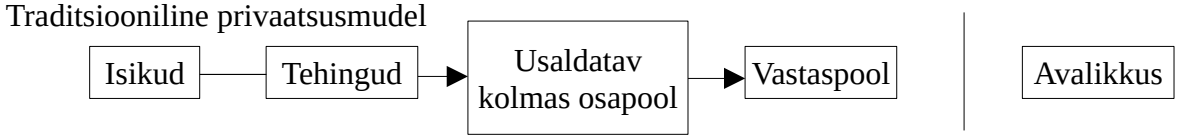
Kuigi müntide eraldi käsitsemine oleks võimalik, oleks iga ülekantava sendi jaoks eraldiseisva tehingu loomine kohmakas. Lubamaks väärtuste jagamist ja ühendamist, koosnevad tehingud mitmeist sisendeist ja väljundeist. Tavalisel puhul eksisteerib kas üksainus sisend, mis pärineb suuremast eelmisest tehingust või mitu sisendit, mis ühendab väiksemaid summasid ning mitte rohkem kui kaks väljundit: üks makse teostamiseks ja üks saatjale ülejäägi, kui jääb midagi üle, tagastamiseks.

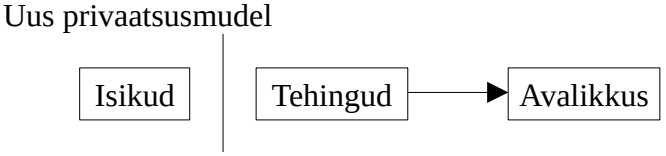


Tuleks ära märkida, et laialivalgumine, kus üks tehing sõltub mitmetest tehingutest ning need tehingud sõltuvad paljudest enamatest, pole siinkohal probleem. Ühe tehingu ajaloo täieliku eraldiseisva väljavõtte koopia järele ei teki kunagi vajadust.

10. Privaatsus

Traditsiooniline panganduse mudel saavutab privaatsuse taseme piirates osapoolte ja usaldatava kolmanda osapoole ligipääsu informatsioonile. Kõikide tehingute avalikustamise vajalikkus välistab selle meetodi, kuid privaatsust on siiski võimalik säilitada, murdes informatsiooni kulgu teise nurga alt: hoides avalikke võtmeid anonüümsetena. Avalikkus näeb, et keegi saadab kellelegi teatud summa, ilma, et need andmed tehingut kellegi isikuga seoks. See sarnaneb börsside poolt välja antava informatsiooniga, kus üksikute tehingute aeg ja maht, ehk “salvestis”, avalikustatakse, kuid jättes ütlemata, kes on osapooled.





11. Arvutused

Arvestame stsenaariumiga, kus ründaja püüab luua alternatiivset ahelat legitiimsest ahelast kiiremini. Isegi kui see saavutatakse, ei heitu süsteem seetõttu avatuks meelevaldsetele muudatustele nagu näiteks tühjast õhust väärtuse loomine või raha omistamine, mis ründajale ei kuulu. Ühenduspunktid ei aksepteeri kehtetuid tehinguid maksetena ning legitiimsed ühenduspunktid ei aksepteeri plokke, mis selliseid tehinguid sisaldavad. Ründaja saab ainult üritada muuta ühte tema enda poolt tehtud makset, eesmärgiga võtta tagasi raha, mille ta hiljuti kulutas.

Võidujooksu legitiimse ahela ja ründaja ahela vahel võib iseloomustada kui Binomiaalset Juhuslikku Käimist (ingl *Binomial Random Walk*). Legitiimse ahela pikenemist ühe ploki võrra loetakse õnnestunud juhtumiks, kasvatades selle edu +1 võrra ning ründaja ahela pikenemist ühe ploki võrra loetakse ebaõnnestunud juhtumiks, kahandades edumaad -1 võrra.

Ründaja järelejõudmise tõenäosus mingist madalseisust on võrreldav Mänguri Hävingu (ingl *Gambler's Ruin*) probleemiga. Kujutleme mängurit, kes alustab piiramatu krediidiga madalseisust mängimist ja kes mängib potentsiaalselt lõpmatu arvu mänge, eesmärgiga kahjumist vabaneda. Tõenäosust, et ta kahjumist vabaneb või et ründaja legitiimsele ahelale järgi jõuab, saab arvutada järgnevalt [8]:

p = tõenäosus, et legitiimne ühenduspunkt avastab järgmise ploki
q = tõenäosus, et ründaja avastab järgmise ploki
q_z = tõenäosus, et, alustades z-suurusest madalseisust, ründaja saavutab edu

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

Võttes arvesse eeldust, et $p > q$, langeb ründaja edu tõenäosus eksponentsiaalselt, sest nende plokkide arv, millele ründaja järgi peab jõudma, kasvab. Tõenäosus tema vastu töötamas tagab, et kui ründaja ei tee kohe alguses õnnelikku võidujooksu, muutuvad tema võimalused kaduvalt väikeseks ja ta jääb üha suuremasse madalseisu.

Järgnevalt arutame, kui kaua peab uue tehingu vastuvõtja ootama, enne kui ta saab piisavalt kindel olla, et saatja ei saa tehingut enam muuta. Eeldame, et saatja on ründaja, kes soovib panna vastuvõtjat uskuma, et ta maksis talle ning seejärel ümber lülituda ning maksta hoopis iseendale. Vastuvõtjat teavitatakse sellisel puhul, kuid saatja loodab, et see juhtub liiga hilja.

Vastuvõtja loob uue võtmepaari ja annab vahetult enne allkirjastamist avaliku võtme saatjale. See väldib olukorda, kus saatja valmistab aegsasti ette ahela jagu plokke, töötades nende kallal kuni teda saadab edu saavutamiseks piisavalt suur õnn

ning sellel hetkel tehingu teostab. Olles tehingu teostanud, alustab ebaaus saatja salamisi töötamist paralleelse ahela kallal, mis sisaldab tema tehingu alternatiivset versiooni.

Vastuvõtja ootab kuni tehing on plokki lisatud ja *z* arv plokke on selle järele aheldatud. Ta ei ole ründaja täpsete edusammudega kursis, kuid eeldusel, et legitiimsed plokid kasutasid keskmist ploki loomiseks kuluvat aega, saab ründaja potentsiaalsest arengust oodatava väärtusega Poissoni jaotus:

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

Selgitamaks tõenäosust, et ründaja ikkagi suudab edu saavutada, korrutame Poissoni tiheduse iga tehtava arengu koguse kohta tema järelejõudmise tõenäosusega:

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

Paigutame ümber, et vältida jaotise lõputu saba summeerimist…

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

C koodi teisendamine…

```
#include
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;
}
```

Tulemusi vaadates võime näha, et tõenäosus langeb *z* kasvuga eksponentsiaalselt.

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

Lahendused juhul, kui P on väiksem kui 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. Kokkuvõte

Oleme välja pakkunud elektrooniliste tehingute süsteemi, mis ei tugine usaldusele. Alustasime tavapärase raamistikuga, mis koosneb digiallkirjadest koosnevatest müntidest, pakkudes tugevat kontrolli omandi üle, kuid milles esineb puudujääke topeltkulutamise vältimises. Selle lahendamiseks pakkusime välja vastastikulise võrgu, mis kasutab töö-tõestust, salvestamaks tehingutest avalikku ajalugu, mille muutmine muutub ründaja jaoks arvutuslikult ebapraktiliseks kui enamus CPU võimsusest on legitiimsete ühenduspunktide käsutuses. Võrk on enda struktureerimata lihtsuses robustne. Ühenduspunktid töötavad vähese kooskõlastusega kõik samaaegselt. Neid ei ole tarvis tuvastada, sest sõnumeid ei saadeta ühessegi kindlasse asukohta ja nende väljastamine toimub ainult parima pingutuse alusel. Ühenduspunktid võivad soovi korral võrgust lahkuda ja sellega uuesti liituda, käsitledes töö-tõestuse pikimat ahelat tõestusena sellest, mis toimus sel ajal, kui nad eemal olid. Nad hääletavad enda CPU võimsusega, väljendades enda vastuvõtlikkust paikapidavate plokkide osas, töötades nende edasi arendamise kallal ning lükates tagasi kehtetud plokid nende kallal mitte töötades. Kõiki vajalikke reegleid ja stiimuleid saab selle konsensusmehhanismi abil täide viia.

Viited

[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.

[9] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” https://bitcoin.org/bitcoin.pdf