Bitmono: Samtavola Elektrona Kontantsistemo

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

Tradukis
Marco Gazzetta
dev@mrgazz.com

Resumo. Pure samtavola versio de cifereca kontanto ebligus retan pagadon de unu partio al alia sen bezoni peradon de financa institucio. Ciferecaj subskriboj estas parto de la solvo, sed ĝiaj ĉefaj avantaĝoj perdiĝas se oni daŭre bezonas fidindan trian partion por malhelpi duoblan elspezon. Ni proponas solvon de la duoblelspeza problemo pere de samtavola reto. La reto datigas ĉiujn transakciojn per haketado en adan ĉenon da haketa laborpruvo, kiu estigas registron neŝanĝeblan sen refardo de la laborpruvo. La plej longa ĉeno ne nur estas pruvo de la sinsekvo da okazaĵoj atestitaj, sed ankaŭ de ĝia elveno el la plej granda komunaĵo da procesora potenco. Dum plimulto da procesora potenco estas kontrolata de nodoj, kiuj ne kunlaboras atakante la reton, ili estigos la plej longan ĉenon kaj antaŭpasos tiajn atakantojn. La reto mem nur bezonas minimuman strukturon. La mesaĝoj estas elsendataj kiel eble plej rapide, kaj nodoj povas forlasi kaj reeniri la reton laŭplaĉe, akceptante la plej longan laborpruvan ĉenon kiel pruvon de okazinto dum ilia malĉeesto.

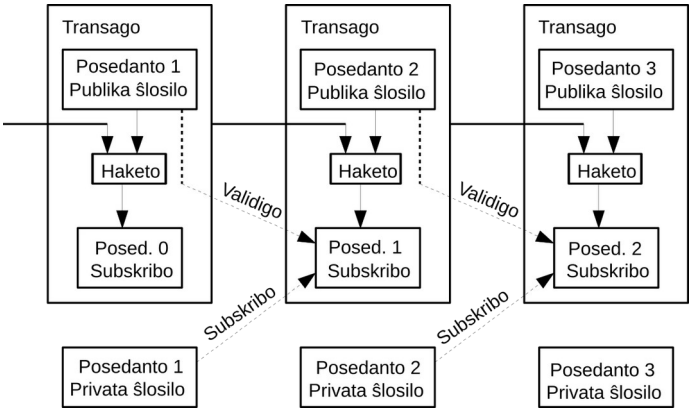
1. Enkonduko

Interreta komerco dependas preskaŭ plene de financaj institucioj kiel fidata triaj partioj por procedado de ciferecaj pagoj. Kvankam tiu sistemo funkcias sufiĉe bone por plimulto da transagoj, ĝi tamen suferas pro la ena malforteco de la fidecbaza modelo. Tute malretroigeblaj transagoj ne vere eblas, ĉar financaj institucioj ne povas malhelpi peri disputojn. Perada kosto pligrandigas transagajn kostojn, limigante la grandon de minimuma praktika transago kaj malebligante malgrandajn ĉiutagajn transagojn, kaj estas eĉ pli granda kosto en perdi la eblecon pagi malretroigeble por malretroigeblaj servoj. Ju pli probabla retroigado, des pli oni bezonas fidon. Komercistoj devas malfidi pri iliaj klientoj, postulante pli da informoj pri ili ol necesas. Oni akceptas kelkan procentaĵon da trompado kiel neeviteblan. Ĉi tiujn kostojn kaj malcertecojn pri pagado oni povas eviti per senpera uzado de tuŝebla kontanto, sed neniu mekanismo haveblas por pagi tra komunika kanalo sen fidenda partio.

Oni bezonas ciferecan pagsistemon bazitan sur kriptografia pruvo anstataŭ sur fido, ebligantan transagon inter iuj du interkonstentantaj partioj rekte unu kun la alia sen bezoni fidatan trian partion. Transagoj kiuj procesorpotence maleblas esti retroigataj protektus vendantojn de trompo, kaj rutinaj garantiaj mekanismoj facile ekstariĝus protekte de aĉetantoj. En ĉi tiu dokumento ni proponas solvon pri la duoblelspeza problemo uzantan samtavolan distribuan datigilan servon kiu kreas komputikan pruvon de la kronologia sinsekvo de transagoj. Ĉi tiu sistemo estas sekura dum konfideblaj nodoj kontrolas pli da procesorpotenco ol ajna alia grupo da kunlabore atakantaj nodoj.

2. Transagoj

Ni difinas ciferecan moneron kiel ĉenon da ciferecaj subskriboj. Posedanto transigas sian moneron al sekva per cifereca subskribado de la haketo de la antaŭa transago kaj de la publika ŝlosilo de la tuj posedonto, kaj per aldono de ĉi-tiuj fine de la monero. Pagito povas validigi la subskribojn por certigi la posedantaran ĉenon.

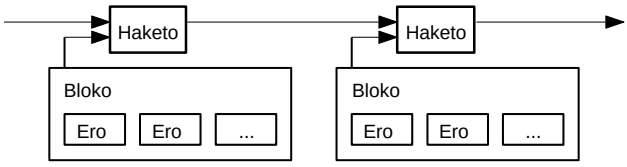


La problemo pri tiu sistemo estas ke la pagato ne povas certigi ke paganto ne duoblelspezis la moneron. Ofta solvo konsistas el la enkonduko de fidenda tria partio, nomata monfarejo, kiu kontrolas ĉu iu monero duoblelspeziĝis. Post ĉiu transago, monero devas sendiĝi al la monfarejo, kiu eldonos novan moneron. Oni nur povas fidi ke tiuj moneroj kiuj estis eldonataj rekte de la monfarejo ne estas doublelspezataj. La problemo pri ĉi tiu solvo estas, ke la tuta monsistemo dependas de la monfarejo, kaj ĉiu transago devas validiĝi per ĝi, kvazaŭ banko.

Ni bezonas metodon por ke pagato sciu, ke paganto ne subskribis aliajn transagojn antaŭe. En nia konsiderado, la plej frua transago el ĉiuj estas la sole valida, do ni ne devas konsideri postajn kaj duoblelspezajn transagojn. La ununura maniero konfirmi maleston de transago estas scii pri ĉiuj transagoj. En la modelo monfareja, la monfarejo scias pri ĉiuj transagoj kaj decidas, kiu venis unua. Por atingi tion sen fidata tria partio, transagoj devas esti publike eldonataj [1] kaj ni bezonas sistemon por ke partoprenantoj povu akordiĝi pri ununura historio de la sinsekvo de ricevado. Pagato bezonas pruvon, ke date de ĉiu transago, plimulto da nodoj akordiĝis pri ĝia unueco.

3. Datiga servilo

Nia proponata solvo komenciĝas per datiga servilo. Tia servilo funkcias per kreado de haketo pri datigaĵaro kaj disvastigado publika de la haketo, ekzemple en ĵurnalo aŭ afiŝo Usenet [2-5]. La datigo pruvas ke la datumoj ekzistis je la dato, memvide, tiel ke ili enmetiĝis en la datigaĵaro. Ĉiu datigo inkluzivas la antaŭan datigon en sia haketo, tiel kreante ĉenon en kiu ĉiu aldona datigo plicertigas tiujn antaŭ ĝi.

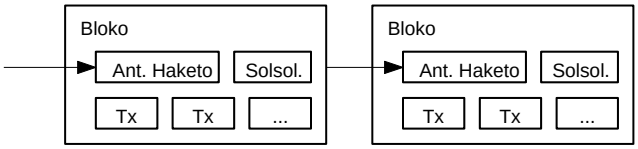


4. Laborpruvo

Por realigi distribuan datigan servilon per samtavola interago ni bezonas laborpruvon similan al

Hashcash de Adam Back [6] anstataŭ ĵurnalo aŭ afiŝo Usenet. La laborpruvo konsistos el trovi valoron kies haketo, ekz. kreita per algoritmo SHA-256, komenciĝas per decidita nombro da nulaj bitoj. La averaĝa kvanto da laboro bezonata kreskiĝas eksponente laŭ la nombro da nulaj bitoj bezonataj, sed estas validigata per unu haketado.

Por nia datigila retservo, ni starigas la laborpruvon per alkrementado de solsolaĵo en la bloko ĝis kiam valoro atingiĝas, kun kiu la haketo de la bloko havas la bezonatajn nulajn bitojn. Post procesorpotenco estis uzata por trovi la solvon de la laborpruvo, oni ne plu povas ŝanĝi la blokon sen refari la laboron. Kiam postaj blokoj enĉeniĝas post ĝi, la laboro bezonata por ŝanĝi la unuan blokon inkluzivus relaborpruvi ĉiujn ĉenerojn post ĝi.



Ĉi tiu laborpruvo ankaŭ solvas la problemon decidi pri la naturo de plimulteco. Se plimulto baziĝus sur «unu voto po IP-adreso», ĉiu kiu povas ekhavi multajn IP-adresojn povus subfosi ĝin. Laborpruvo esence estas «unu voto po procesoro». La plimulta decido reprezentiĝas per la plej longa ĉeno, kiu entenas la plej grandan laborpruvan investon. Se plimulto da procesorpotenco estas kontrolata de honestaj nodoj, la honesta ĉeno kreskiĝos plej rapide kaj antaŭpasos ĉiujn konkurencajn ĉenojn. Ni pruvos poste ke la probableco de pli malrapida atakanto sampasanta kun la plimulto malkreskiĝas eksponente dum ĉeneraj blokoj aldoniĝas.

Kompense de kreska komputilrapideco kaj varia emo estigi nodojn, la malfacileco de la laborpruvo decidiĝas per kuranta averaĝo, kiu celas averaĝan nombron da blokoj po horo. Se oni estigas tiujn blokojn tro rapide, la malfacileco kreskiĝos.

5. Reto

La paŝoj al kurado de la reto estas la sekvaj :

1) Novaj transagoj elsendiĝas al ĉiuj nodoj.
2) Ĉiu nodo kunmetas novajn transagojn en blokon.
3) Ĉiu nodo laboras por trovi la malfacilan laborpruvon por kunmetita bloko.
4) Kiam nodo trovas la laborpruvon, ĝi elsendas ĝin al ĉiuj nodoj.
5) La nodoj akceptas la blokon nur se ĉiuj transagoj en ĝi estas validaj kaj ne jam elspezitaj.
6) La nodoj validigas sian akceptadon de la bloko per eklaborado pri la sekva bloko en la ĉeno, uzante la haketon de la akceptita bloko kiel antaŭa haketo de la sekva bloko.

La nodoj ĉiam konsideras la plej longan ĉenon la ĝustan kaj daŭros labori por plilongigi ĝin. Se du nodoj elsendas malsamajn versiojn de la sekva bloko samtempe, kelkaj nodoj ricevos unu aŭ la alian unue. Tiukaze, tiuj nodoj laboros per la unue ricevitan, sed savas la alian branĉon kaze de ĝia plejlongiĝo. Ĉi tiu situacio daŭros ĝis kiam la sekva laborpruvo troviĝas surbaze de unu branĉo el la du kaj tiu branĉo iĝas la plej longa; la nodoj prilaborantaj la alian branĉon devos transiĝi al la alia.

Novaj elsendoj de transagoj ne necese devas atingi ĉiujn nodojn. Se ili atingas multajn nodojn, ili inkluziviĝos en bloko post mallonge. Ankaŭ blokaj elsendoj toleremas mankatajn mesaĝojn. Se nodo ne ricevas blokon, ĝi petos ĝin kiam ĝi ricevas la sekvan kaj ekscias pri la mankantan.

6. Stimulo

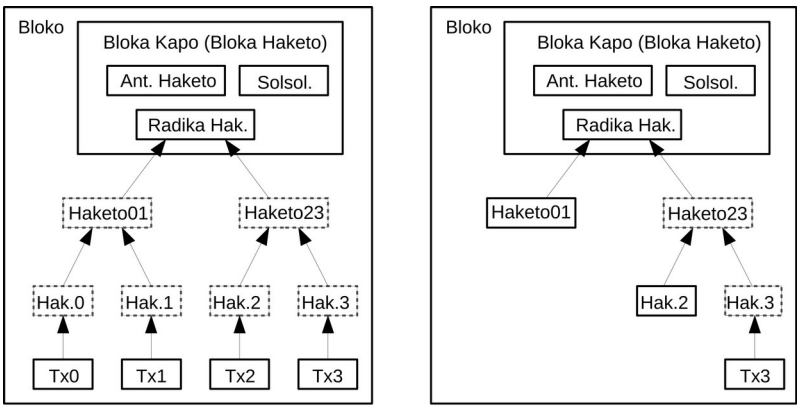
Laŭ konvencio, la unua transago en ĉiu bloko estas speciala transago kiu ekstarigas novan moneron posedatan de la blokkreanto. Ĉi tio aldonas stimulon por ke nodoj subtenu la reton kaj estigas manieron por distribui novajn monerojn en cirkulado, ĉar ne ekzistas centra aŭtoritato kiu eldonus tiujn. La ada aldono de konstanta kvanto da novaj moneroj estas analoga al orfosistoj kiuj elspezas rimedojn por aldoni oron en cirkuladon. En nia kazo, la rimedoj konsistas el procesorpotenco kaj elektro.

Tiun stimulon oni povas monigi per transagaj kotizoj. Se la elflua valoro de iu transago estas malpli ol la enflua, la diferenco estas transaga kotizo aldonata al la stimula valoro de la bloko kiu entenas la transagon. Kiam antaŭdecidita nombro da moneroj eniris cirkuladon, la stimulo povas transigi tute al transagaj kotizoj kaj iĝis tute seninflacia.

La stimulo povas helpi honestigi nodojn. Se avida atakanto eblas kunmeti pli da procesorpotenco ol ĉiujn honestajn nodojn, li devus elekti ĉu fraŭdi aliajn per reŝtelado de siaj pagaĵoj, aŭ kreadi novajn monerojn. Li trovus ke li profitus pli per ludi laŭ la reguloj, kiuj favoras lin per pli da novaj moneroj ol ĉiuj aliaj kune, ol subfosi la sistemon kaj la validecon de siaj riĉoj.

7. Reakiri Diskospacon

Kiam nova transago en monero estas fosita sub sufiĉe da aliaj blokoj, ĝiaj elspezataj transagoj povas esti forigitaj por savi diskospacon. Por plifaciligi tion kaj ne malĝustigi la blokan haketon, transagoj estas haketataj en Merkle Arbo [7][2][5], kaj nur la radiko inkluzivas en la bloka haketo. Malnovaj blokoj kompakteblas per elmovado de branĉoj de tiu arbo. Haketoj interne de transagoj ne bezonas savadon.



Transagoj Haketitaj En Merkle Arbo

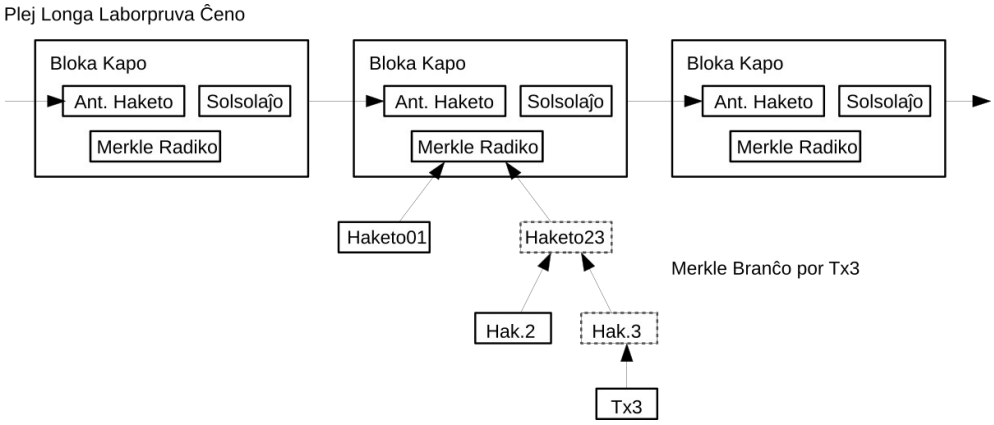
Post Forpreni Tx0-2 El La Bloko

Bloka kapo sen transagoj larĝas proksimume 80 bitokojn. Se ni supozas, ke blokoj estas kreataj po dek minutoj, 80 bitokoj * 6 * 24 * 365 = 4.2MB po jaro. Konsidere ke komputilsistemoj tipe vendiĝas kun 2GB de RAM en 2008, kaj la leĝo de Moore antaŭvidas aktualan kreskon de 1.2GB po jaro, spaco probable ne iĝos problemo eĉ se la blokaj kapoj devas resti en ĉefmemoro.

8. Simpligita Validigo de Pagoj

Eblas validigi pagojn sen funkciigi plenan retan nodon. Uzanto nur devas teni kopion de la blokaj kapoj de la plej longa laborpruva ĉeno, kiun oni povas ekhavi petante retnodojn ĝis

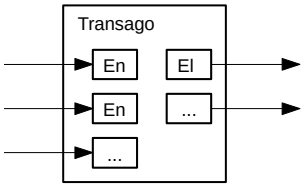
konvinkiĝo havi la plej longan ĉenon, kaj ekhavi la Merkle branĉon kiu ligas la transagon al la bloko en kiu ĝi datiĝis. Oni mem ne povas validigi la transagon, sed ĉar ekzistas ligo al ĝi ie en la ĉeno, oni povas vidi ke iu retnodo akceptis ĝin, kaj nodoj aldonitaj post ĝi plie konfirmas, ke la reto akceptis ĝin.



Sekve, tiu validigo estas fidebla dum honestaj nodoj kontrolas la reton, sed estas pli vundebla se la reto superfortiĝas fare de atakanto. Dum retnodoj povas validigi transagojn memfare, la simpligita metodo povas esti trompata de transagoj artefaritaj de atakanto dum kiam tiu atakanto povas superfortigi la reton. Ebla strategio malhelpi tion estus akcepti alarmojn de aliaj retnodoj kiam ili detektas malvalidan blokon, igante la softvaron elŝuti la tutan blokon kaj alarmigajn transagojn por konfirmi malvalidecon. Entreprenoj kiuj ricevas pagojn ofte emus daŭre funkciigi siajn plenajn retnodojn por pli sendependa sekureco kaj pli rapida validigo.

9. Kunigi kaj Disigi Valorojn

Kvankam eblus uzi monerojn unuope, estus neregeble krei disan transagon por ĉiu ero en monelsendo. Por ebligi ke valoroj povu kuniĝi kaj disiĝi, transagoj entenas plurajn envalorojn kaj elvalorojn. Normale ĉiu transago entenos aŭ unuopan envaloron el pli granda antaŭa transago, aŭ plurajn envalorojn kunigante pli malgrandajn kvantojn. Ĝi ankaŭ havos pleje du elvalorojn : unu por pagado, kaj alia retroigante ŝanĝmonon al sendinto kie necese.

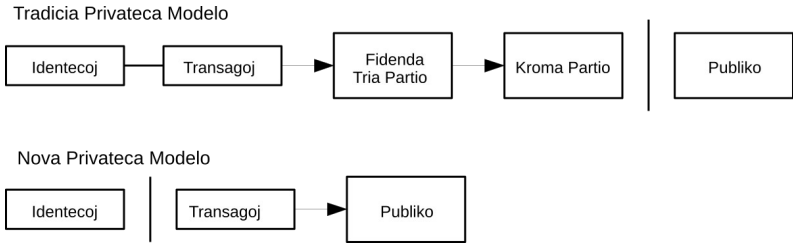


Rimarku ke sinsekva disiĝo, kie transago dependas de pluraj transagoj, kiuj mem povus dependi de pliaj pluraj, ne estas problemo ĉi-kaze. Oni neniam bezonas eltiri tutan memstaran version de la historio de iu transago.

10. Privateco

La tradicia banksistemo efektivigas kvanton da privateco per limigo de informoj pri la partoprenantaj partioj kaj la fidata tria partio. La bezono anonci ĉiujn transagojn publike malhelpas ĉí tiun metodon, sed oni tamen povas efektivigi privatecon per limigo de informfluo

aliloke: per anonimigo de publikaj ŝlosiloj. Ĉiuj povas vidi ke iu elsendas monerkvanton al iu alia, sed sen la informoj necesaj por ligi la transagon al iu specifa. Tio similas al la informkvanto eldonata de borsoj, kiuj publikigas daton kaj kvanton de unuopaj transagoj sen mencii la partiojn.



Kiel aldona protektilo, oni uzu novan ŝlosilparon por ĉiu transago por malebligi ilian ligadon al komuna posedanto. Ioma ligado tamen ne estas malhelpebla pro transagoj plur-envalorajn, kiuj memdire malkaŝas komunan posedadon. Oni riskas ke se posedanto de iu ŝlosilo malkaŝiĝas, tia ligado povus malkaŝi aliajn transagojn fare de la sama posedanto.

11. Kalkuladoj

Ni konsideru la kazon de atakanto kiu provas krei alternativan ĉenon pli rapide ol kreiĝo de la honesta ĉeno. Eĉ se tio okazus, ĝi ne ebligus laŭvolajn ŝanĝojn en la sistemo, ekz. krei valoron el nenio aŭ forpreni monon kiu ne apartenis al la atakanto. Nodoj ne akceptos malvalidan transagon kiel pagon, kaj honestaj nodoj neniam akceptas blokon kiu entenas tiajn. Atakanto nur povus provi ŝanĝi siajn transagojn por repreni monon ĵus elspezatan.

La konkurso inter la honesta ĉeno kaj ajna atakanta ĉeno estas karakterizita kiel Binoma Hazarda Promenado. La sukcesa okazo konsistas el la plilongigo de la honesta ĉeno per unu bloko, etendigante ĝian antaŭiĝon je +1, kaj la malsukcesa okazo konsistas el la plilongigo de la atakanta ĉeno per unu bloko, maletendigante la diferencon je -1.

La probableco ke atakanta ĉeno samlongiĝu ekde difinita malplia longeco estas analoga al problemo pri Vetludanta Pereado. Supozu ke vetludanto kun senfina kredito komencas malvenke kaj ludas teorie senfinan nombron da provoj por atingi egalecon. Ni povas kalkuli la probablecon, ke li iam atingos egalecon, aŭ ke atakanta ĉeno samlongiĝas kun honesta, sekve [8] :

p = probableco, ke honesta nodo trovos la sekvan blokon
q = probableco, ke la atakanto trovos la sekvan blokon
q_z = probableco, ke la atakanto iam atingos samlongecon eke de manko de z blokoj.

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

Konsidere de nia antaŭkondiĉo, ke p > q, la probableco malkreskas eksponenciale laŭ la nombro da blokoj de malplilongeco. Ĉar probableco estas malfavora, se la atakanto ne antaŭeniras frue pro bonŝanco, lia ebleco venki malkreskas rapidege dum li malantaŭeniĝas plu.

Ni nun konsideras kiom longe ricevinto de nova transago devas atendi antaŭ esti sufiĉe sekura, ke la sendinto ne povas ŝanĝi la transagon. Ni antaŭsupozas, ke la sendinto estas

atakanto, kiu volas kredigi la ricevinton, ke li estis pagita dum kelka tempo, sed poste ŝanĝas tion al pago de si mem post plia tempo. La ricevinto estos atentigata pri tia okazo, sed la sendinto esperas ke tro malfruos.

La ricevonto kreas novan ŝlosilparon kaj donas la publikan ŝlosilon al sendonto tuj antaŭ subskribado. Tio malhelpas, ke la sendonto preparu ĉenon da blokoj antaŭ subskribado per daŭra laborado ĝis bonŝanca ebleco antaŭeniĝi sufiĉe, kaj ke li tiam efektivigu la transagon. Kiam tiu transago elsendiĝas, la malhonesta sendinto eklaboras kaŝe pri paralela ĉeno kiu entenas la alternativan version de la transago.

La ricevinto atendas ĝis la transago aldoniĝis al iu bloko kaj *z* blokoj ligiĝis post ĝi. Li ne scias la precizan kvanton da progreso faritan de la atakanto, sed supozante ke la honestaj blokoj haviĝis post la averaĝe atendata tempo po bloko, la ebla progreso de la atakanto estos *Poisson* distribuo kun la atendita valoro:

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

Por akiri la probablecon, ke la atakanto ankoraŭ povus samlongigi sian ĉenon, ni multiplikas la *Poisson* densecon por ĉiu kvanto da progreso ebla per la probableco de samlongiĝo ekde tiu punkto:

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

Rearanĝante pro evitado de senfina sumo de la distribua vosto...

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

Reskribante en kodon C...

```
#include <math.h>
double AtakantSukcesProbableco(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;
}
```

Rulante kelkajn rezultojn, ni vidas ke la probableco malkreskas eksponenciale laŭ *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
```

Solvante por P malpli ol 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. Konkludo

Ni proponis sistemon por ciferecaj transagoj senbezone de fido. Ni komencis per la ofte uzata kadro de moneroj faritaj el ciferecaj subskriboj, kiu havigas striktan kontrolon pri posedo, sed ne estas kompleta sen maniero malhelpi duoblelspezojn. Por solvi tion, ni proponis samtavolan reton uzantan laborpruvojn por registri la publikan historion de transagoj, kiu rapide iĝas praktike kompute neŝanĝebla fare de atakanto, se la plimulto da procesorpotenco estas kontrolata de honestaj nodoj. La reto estas fortika per ĝia senstrukturiĝa simpleco. Nodoj kunlaboras ade kun minimuma kunordigo. Ili ne bezonas identigon, ĉar mesaĝoj ne bezonas elsendon al ajna preciza lokiĝo kaj nur bezonas plejpenadan ricevadon. Ĉiuj nodoj povas forlasi kaj eniri la reton laŭvole, akceptante la laborpruvan ĉenon kiel pruvo de tio, kio okazis dum ilia malĉeesto. Ili voĉdonas per ilia procesorpotenco, esprimante sian akceptadon de validaj blokoj per provo etendigi ilin, kaj malakceptante malvalidajn blokojn per rifuzado prilabori ilin. Ĉiuj reguloj kaj stimuloj povas esti starigataj per ĉi tiu tutinterkonsenta mekanismo.

Referencoj

[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," En *20th Symposium on Information Theory in the Benelux*, majo 1999.

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

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

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," En *Proceedings of the 4th ACM Conference on Computer and Communications Security*, paĝoj 28-35, aprilo 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," En Proc. *1980 Symposium on Security and Privacy*, IEEE Computer Society, paĝoj 122-133, aprilo 1980.

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