Bitcoin: Un Sistem Peer-to-Peer De Plată Electronică

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

Traducere în română realizată de Gazeta Bitcoin
Corectură text și adaptare diagrame realizate de NeuroticFish

Abstract. O versiune de bani electronici complet peer-to-peer ar permite ca plățile online să fie trimise în mod direct de la o parte către alta fără a apela la o instituție financiară. Semnăturile digitale oferă o parte a soluției, dar beneficiile principale sunt pierdute dacă este necesar în continuare un intermediar de încredere pentru prevenirea dublei cheltuiri. Propunem o soluție pentru problema dublei cheltuiri prin utilizarea unei rețele peer-to-peer. Prin hashing, rețeaua marchează temporal tranzacțiile într-un lanț continuu de dovezi de lucru bazate pe hash, formând un registru care nu poate fi modificat decât prin reefectuarea dovezii de lucru. Cel mai lung lanț nu servește doar ca dovadă a secvenței de evenimente la care a asistat, ci este și o dovadă că a provenit de la cea mai mare parte a puterii de procesare. Cât timp majoritatea puterii de procesare este controlată de noduri care nu cooperează spre a ataca rețeaua, acestea vor genera cel mai lung lanț și vor depăși atacatorii. Rețeaua în sine necesită o structură minimă. Mesajele sunt propagate pe baza efortului optim depus, iar nodurile pot părăsi rețeaua și pot reintra după cum doresc, acceptând cel mai lung lanț de dovezi de lucru ca fiind dovada pentru ce s-a întâmplat în lipsa lor.

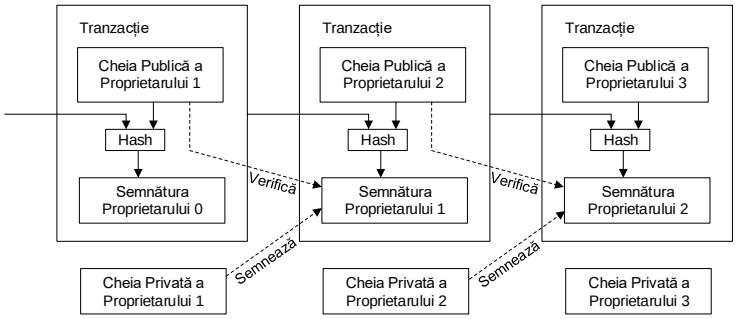
1. Introducere

Comerțul pe Internet a ajuns să se bazeze aproape în întregime pe instituții financiare pe post de intermediari de încredere pentru a procesa plățile electronice. În timp ce sistemul funcționează bine pentru majoritatea tranzacțiilor, el suferă din cauza slăbiciunii inerente a modelului bazat pe încredere. Tranzacțiile complet ireversibile nu sunt tocmai posibile, întrucât instituțiile financiare nu pot evita să medieze disputele. Costul medierii crește costurile de tranzacționare, limitând dimensiunea minimă practică a unei tranzacții și eliminând posibilitatea micilor tranzacții uzuale, și apare un cost mai mare prin pierderea abilității de a realiza plăți ireversibile pentru servicii ireversibile. Din cauza posibilității de reversibilitate crește și nevoia de încredere. Comercianții trebuie să fie precauți cu clienții, stresându-i pentru mai multe informații decât ar avea nevoie. Un anumit procent de fraudare este acceptat ca fiind inevitabil. Aceste costuri și incertitudini legate de plăți pot fi evitate personal prin folosirea banilor fizici, însă nu există niciun mecanism care să efectueze plăți printr-un canal de comunicații fără a apela la un intermediar de încredere.

Avem nevoie de un sistem de plată electronic bazat pe dovezi criptografice în detrimentul încrederii, care să permită oricăror două părți să tranzacționeze între ele în mod direct, fără a fi necesară o terță parte în care să aibă încredere. Tranzacțiile care sunt practic ireversibile din punct de vedere computațional ar proteja vânzătorii de fraude, iar mecanismele escrow de rutină ar putea fi implementate cu ușurință pentru a proteja cumpărătorii. În această lucrare propunem o soluție pentru problema dublei cheltuiri folosind un server distribuit peer-to-peer de marcaje temporale care să genereze dovada computațională a ordinii cronologice a tranzacțiilor. Sistemul este securizat cât timp nodurile oneste controlează în mod colectiv mai multă putere de procesare decât orice grup de noduri care ar coopera în cadrul unui atac.

2. Tranzacţii

O monedă electronică o definim ca un lanț de semnături digitale. Fiecare deținător transferă moneda către altul prin semnarea digitală a unui hash al tranzacției anterioare și cheia publică a următorului posesor și adăugând acestea la finalul monedei. Un beneficiar poate verifica semnăturile pentru a verifica lanțul de posesiune.

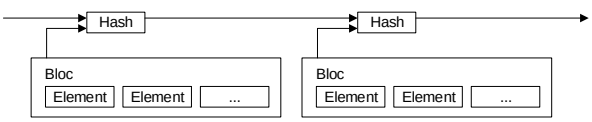


Desigur, problema este aceea că beneficiarul nu poate verifica dacă vreunul dintre proprietari a efectuat o dublă cheltuire. O soluție comună constă în introducerea unei autorități centrale de încredere, sau a unei monetării, care să verifice fiecare tranzacție pentru dublă cheltuire. După fiecare tranzacție, moneda trebuie returnată monetăriei pentru emiterea unei noi monede, și doar monedele emise direct de monetărie sunt considerate ca nefiind dublu cheltuite. Problema acestei soluții este că soarta întregului sistem monetar depinde de compania din spatele monetăriei, întrucât fiecare tranzacție trece prin aceasta, ca în cazul unei bănci.

Avem nevoie ca beneficiarul să știe dacă proprietarii anteriori nu au semnat niciuna din tranzacțiile anterioare. Pentru scopul nostru, cea mai veche tranzacție este cea care contează, deci nu ne interesează de tentativele ulterioare de dublă cheltuire. Singura cale de a confirma absența unei tranzacții este prin a cunoaște toate tranzacțiile. În cadrul modelului bazat pe monetărie, aceasta cunoștea toate tranzacțiile și decidea care a fost prima. Pentru a reuși acest lucru fără un intermediar de încredere, trebuie ca tranzacțiile să fie anunțate public [1] și avem nevoie de un sistem prin care participanții să se pună de acord asupra unui istoric unic al ordinii în care au primit tranzacțiile. Beneficiarul are nevoie de o dovadă că pentru fiecare tranzacție majoritatea nodurilor au agreat care a fost prima primită.

3. Server cu Marcaj Temporal

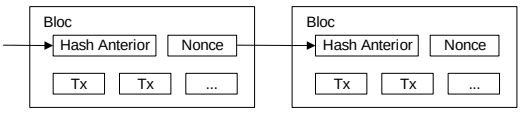
Soluția pe care o propunem începe cu un server cu marcaj temporal. Un server cu marcaj temporal lucrează prin preluarea unui hash al unui bloc de elemente pentru a fi marcat temporal și prin publicarea acestui hash pe scară largă, cum ar fi într-un ziar sau într-o postare pe Usenet [2-5]. Marcajul temporal demonstrează că datele respective au existat la momentul respectiv, evident, pentru a putea fi incluse în hash. Fiecare marcaj temporal include marcajul anterior în hash-ul său, alcătuind un lanț, fiecare marcaj temporal întărindu-le pe cele anterioare.



4. Dovadă de Lucru

Pentru a implementa un server distribuit de marcaje temporale bazat pe un model peer-to-peer, vom avea nevoie mai degrabă să utilizăm un sistem de dovezi de lucru similar cu Hashcash-ul lui Adam Back [6], nu un ziar sau postări de pe Usenet. Dovada de lucru implică scanarea după o valoare care, atunci când este trecută printr-un hash, cum ar fi SHA-256, are un hash care începe cu un număr de biți pe zero. Munca medie necesară crește exponențial cu numărul de biți pe zero solicitați și poate fi verificată prin executarea unui singur hash.

Pentru rețeaua noastră de marcaje temporale, implementăm dovada de lucru prin incrementarea unui nonce din bloc până când este găsită o valoare care conferă hash-ului blocului numărul necesar de biți pe zero. Odată ce puterea de procesare a fost extinsă pentru a satisface dovada de lucru, blocul nu poate fi modificat fără reefectuarea muncii. Și cum blocurile ulterioare sunt înlănțuite, munca de a modifica blocul ar presupune reefectuarea tuturor blocurilor ulterioare.



Dovada de lucru rezolvă de asemenea și problema determinării reprezentării în impunerea deciziilor majorității. Dacă majoritatea ar fi calculată sub forma o-adresă-IP-un-vot, deciziile ar putea fi subminate de oricine ar fi capabil să aloce mai multe adrese IP. În esență, dovada de lucru este un-CPU-un-vot. Decizia majoritară este reprezentată de cel mai lung lanț, care deține cel mai mare efort depus pentru dovada de lucru. Dacă o majoritate a puterii de calcul este deținută de noduri oneste, lanțul onest va crește cel mai rapid și va întrece lanțurile rivale. Pentru a modifica un bloc din trecut, un atacator ar trebui să reefectueze atât dovada de lucru a acestui bloc cât și a celorlalte blocuri ulterioare, și apoi să ajungă din urmă și să depășească munca nodurilor oneste. Vom arăta ulterior că probabilitatea ca un atacator lent să ajungă din urmă celelalte noduri scade exponențial pe măsură ce se adaugă noi blocuri.

Pentru a compensa creșterea vitezei echipamentelor hardware și variațiile în timp ale interesului de a rula noduri, dificultatea dovezii de lucru este determinată de o medie mobilă a numărului mediu de blocuri pe oră. Dacă acestea sunt generate prea repede, dificultatea crește.

5. Rețea

Pașii pentru rularea rețelei sunt următorii:

1) Tranzacțiile noi sunt propagate către toate nodurile.
2) Fiecare nod colectează noi tranzacții într-un bloc.
3) Fiecare nod lucrează pentru a găsi o dovadă de lucru dificilă pentru blocul său.
4) Când un nod găsește o dovadă de lucru, propagă blocul către toate nodurile.
5) Nodurile acceptă blocul doar dacă toate tranzacțiile din cadrul său sunt valide și nu sunt deja cheltuite.
6) Nodurile își exprimă acceptarea blocului lucrând la următorul bloc al lanțului, folosind hash-ul blocului acceptat pe post de hash anterior.

Nodurile consideră mereu cel mai lung lanț ca fiind cel corect și vor continua să lucreze pentru a îl extinde. Dacă două noduri propagă simultan versiuni diferite ale următorului bloc, unele noduri vor primi o versiune ca fiind primul bloc, iar alte noduri vor primi cealaltă versiune. În acest caz, nodurile vor lucra asupra primului bloc primit, dar vor salva cealaltă ramură în caz că aceasta devine mai lungă. Egalitatea va dispărea la găsirea următoarei dovezi de lucru și una dintre ramuri devine mai lungă; nodurile care lucrau la cealaltă ramură se vor muta pe cea mai lungă.

Nu este neapărat necesar ca noile tranzacții propagate să ajungă la toate nodurile. Cât timp ajung la mai multe noduri, ele vor fi incluse în scurt timp într-un bloc. Propagarea blocurilor este de asemenea tolerantă în mesajele omise. Dacă un nod nu primește un bloc, el îl va solicita când îl primește pe următorul și realizează că îi lipsește unul dintre blocuri.

6. Stimulent

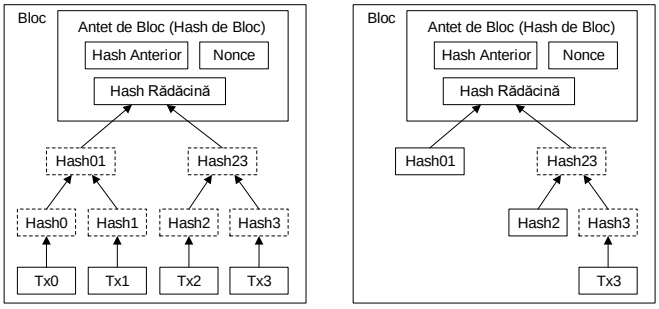
Prin convenție, prima tranzacție dintr-un bloc este o tranzacție specială care generează o nouă monedă deținută de creatorul blocului. Aceasta reprezintă un stimulent pentru ca nodurile să susțină rețeaua, și oferă o metodă pentru distribuția inițială în circulație a monedelor, întrucât nu există o autoritate centrală care să le emită. Adăugarea permanentă a unui număr constant de monede noi este analogă cu minerii de aur care cheltuie resurse pentru a adăuga aur nou în circulație. În cazul nostru, cheltuiala constă în timp de procesare și electricitate.

De asemenea, stimulentul poate fi finanțat cu comisioane de tranzacționare. Dacă valoarea de ieșire a unei tranzacții este mai mică decât valoarea de intrare, diferența este un comision de tranzacționare care este adăugat valorii stimulentului blocului care conține tranzacția. Odată ce un număr predeterminat de monede a intrat în circulație, stimulentul se poate transforma total în comisioane de tranzacționare și poate fi complet lipsit de inflație.

Stimulentul poate ajuta la încurajarea nodurilor spre a rămâne oneste. Dacă un atacator avid este capabil să acumuleze o putere de procesare mai mare decât toate nodurile oneste, el ar trebui să aleagă dacă o folosește pentru a frauda oamenii, furându-le plățile, sau pentru a genera monede noi. Ar trebui să ajungă la concluzia că este mai profitabil să urmeze regulile jocului, reguli care îl favorizează pe el cu mai multe monede decât ale tuturor celorlalți, în loc să submineze sistemul și validitatea propriei averi.

7. Revendicarea Spațiului de pe Disc

Odată ce ultima tranzacție dintr-o monedă este îngropată sub destule blocuri, tranzacțiile cheltuite anterior pot fi înlăturate pentru economisirea spațiului de pe disc. Pentru a facilita această operațiune fără a distruge hash-ul blocului, tranzacțiile sunt transpuse prin intermediul hashing-ului într-un Arbore Merkle [7][2][5], în care doar rădăcina este inclusă în hash-ul blocului. Blocurile vechi pot fi apoi compactate prin îndepărtarea ramurilor arborelui. Nu este nevoie ca hash-urile de la interior să fie stocate.

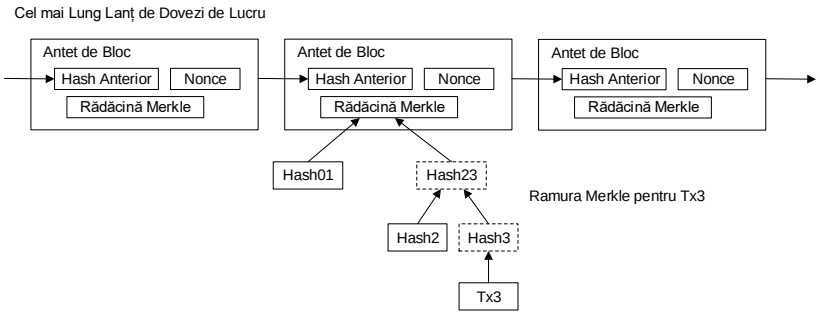


Tranzacțiile Procesate într-un Arbore Merkle — După Tăierea TX0-2 din Bloc

Antetul unui bloc fără tranzacții poate ocupa aproximativ 80 octeți. Dacă presupunem că blocurile sunt generate la fiecare 10 minute, 80 octeți * 6 * 24 * 365 = 4,2MB pe an. Ținând cont că, în mare, PC-urile se vând cu 2GB de RAM începând cu 2008, și de Legea lui Moore, care estimează o creștere curentă de 1,2GB pe an, stocarea nu ar trebui să fie o problemă chiar dacă antetele blocurilor ar trebui să fie păstrate în memorie.

8. Verificare Simplificată a Plăţii

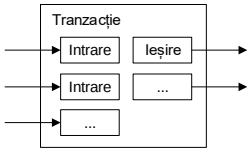
Este posibilă verificarea plăților fără rularea unui nod întreg de rețea. Un utilizator are nevoie doar să păstreze o copie a antetelor blocurilor din cel mai lung lanț de dovezi de lucru, copie pe care o poate obține prin interogarea nodurilor din rețea până când este convins că deține cel mai lung lanț, și până când obține ramura Merkle care corelează o tranzacție cu blocul în care este marcată temporal. Utilizatorul nu poate verifica el însuși tranzacția, dar prin corelarea acesteia cu un loc din lanț, el poate vedea că un nod al rețelei a acceptat-o, și că blocurile adăugate ulterior confirmă de asemenea faptul că rețeaua a acceptat-o.



Astfel, verificarea este de încredere cât timp nodurile oneste controlează rețeaua, dar este mai vulnerabilă dacă rețeaua este învinsă de un atacator. În timp ce nodurile rețelei pot verifica ele însele tranzacțiile, metoda simplificată poate fi păcălită de tranzacțiile fabricate ale unui atacator, cât timp atacatorul continuă să învingă rețeaua. O strategie de protecție împotriva acestei situații ar consta în acceptarea alertelor de la nodurile rețelei atunci când acestea detectează un bloc invalid, solicitându-se soft-ului utilizatorului să descarce întregul bloc și tranzacțiile pentru care au venit alerte, ca să confirme inconsistența. Afacerile care primesc plăți în mod frecvent vor dori probabil să ruleze propriile noduri pentru o securitate mai independentă și verificare mai rapidă.

9. Combinare şi Divizare a Valorii

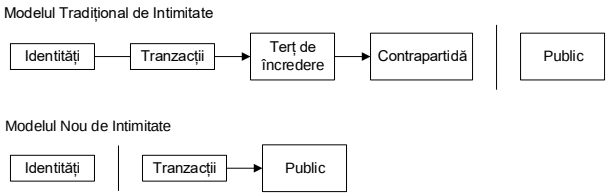
Chiar dacă ar fi posibilă manipularea inidividuală a monedelor, ar fi dificilă realizarea unei tranzacții separate pentru fiecare cent dintr-un transfer. Pentru a permite valorilor să fie divizate și combinate, tranzacțiile conțin multiple intrări și ieșiri. În mod normal, va exista o intrare unică de la o tranzacție anterioară mai mare sau mai multe intrări care combină sume mai mici, și cel mult două ieșiri: una pentru plată și una care returnează restul, dacă există, expeditorului.



De reținut, fan-out-ul, atunci când o tranzacție depinde de mai multe tranzacții, iar aceste tranzacții depind de multe altele, nu reprezintă o problemă. Nu există niciodată nevoia de a extrage o copie de sine stătătoare completă a istoricului tranzacției.

10. Intimitate

Modelul bancar tradițional atinge un anumit nivel de intimitate prin faptul că limitează accesul la informație părților implicate și intermediarilor de încredere. Necesitatea de a anunța public toate tranzacțiile este opusul aceastei metode, dar intimitatea este menținută în continuare prin ruperea fluxului de informație într-un alt punct: prin păstrarea cheilor publice în anonimitate. Publicul poate vedea când cineva trimite o sumă altcuiva, dar fără nicio informație care să coreleze tranzacția cu cineva. Este ceva similar nivelului de informație disponibil la bursele de valori, unde ora și cantitatea tranzacționată sunt publice, dar nimeni nu știe cine sunt părțile implicate.



Pentru un plus de intimitate, o nouă pereche de chei ar trebui folosită la fiecare tranzacție, pentru ca acestea să nu poată fi corelate cu un proprietar comun. Uneori, corelarea este inevitabilă în cazul tranzacțiilor cu multe intrări, care, din necesitate, dezvăluie că intrările lor au aparținut aceluiași proprietar. Riscul este acela că, dacă este dezvăluit proprietarul unei chei, corelarea ar putea dezvălui și alte tranzacții care au aparținut aceluiași proprietar.

11. Calcule

Considerăm scenariul în care un atacator încearcă să genereze un lanț alternativ mai rapid decât lanțul onest. Chiar dacă s-ar reuși acest lucru, sistemul nu ar fi expus schimbărilor arbitrare, cum ar fi crearea de valoare din nimic sau obținerea unor bani care nu au aparținut niciodată atacatorului. Nodurile nu vor accepta o tranzacție invalidă pe post de plată, iar nodurile oneste nu vor accepta niciodată un bloc care ar conține astfel de tranzacții. Un atacator poate doar să încerce să modifice una din tranzacțiile sale pentru a lua înapoi banii pe care i-a cheltuit recent.

Cursa dintre lanțul onest și un lanț atacator poate fi descrisă ca o Deviație Aleatorie Binomială. Evenimentul de succes este ca lanțul onest să fie extins cu un bloc, crescându-i avansul cu +1, iar evenimentul de eșec este ca lanțul atacator să fie extins cu un bloc, reducând diferența cu -1.

Probabilitatea ca un atacator să ajungă din urmă pornind de la un anumit deficit este analogă cu problema Ruinării Jucătorului. Să ne gândim la un jucător de jocuri de noroc, cu un număr nelimitat de credite, care începe cu un deficit și are potențialul de a juca un număr infinit de încercări pentru a încerca să atingă pragul de rentabilitate. Putem calcula probabilitatea pentru a atinge vreodată pragul de rentabilitate, sau probabilitatea ca un atacator să reușească vreodată să ajungă din urmă lanțul onest, astfel [8]:

p = probabilitatea ca un nod onest să găsească următorul bloc

q = probabilitatea ca atacatorul să găsească următorul bloc

q_z = probabilitatea ca atacatorul să ajungă vreodată din urmă, pornind de la z blocuri mai în spate

$$q_z = \begin{cases} 1 & \text{dacă } p \leq q \\ (q/p)^2 & \text{dacă } p > q \end{cases}$$

Dată fiind presupunerea noastră că p > q, probabilitatea scade exponențial pe măsură ce crește numărul de blocuri pe care atacatorul trebuie să le ajungă din urmă. Având șansele împotriva sa, dacă atacatorul nu reușește un salt norocos înainte, șansele sale devin aproape nule, pe măsură ce rămâne din ce în ce mai mult în urmă.

Acum luăm în considerare cât timp trebuie să aștepte destinatarul unei noi tranzacții înainte de a fi suficient de convins că expeditorul nu poate modifica tranzacția. Presupunem că expeditorul este un atacator care dorește ca destinatarul să creadă un anumit timp că a fost plătit, apoi să întoarcă plata către el, după trecerea unui interval de timp. Destinatarul va fi alertat când acest fapt are loc, însă expeditorul speră să fie prea târziu.

Destinatarul generează o nouă pereche de chei și oferă cheia publică expeditorului cu puțin timp înainte de semnare. Aceasta împiedică expeditorul să pregătească dinainte un lanț de blocuri lucrând la el încontinuu până când are norocul de a lua un avans suficient de mare, apoi executând tranzacția la acel moment. Odată ce tranzacția este trimisă, expeditorul necinstit începe să lucreze în secret la un lanț paralel care conține versiunea alterată a tranzacției sale.

Destinatarul așteaptă până când tranzacția a fost adăugată într-un bloc și z blocuri au fost înlănțuite în continuare. El nu știe precis care este progresul atacatorului, dar presupunând că blocurile oneste au apărut la timpul mediu de așteptare al fiecărui bloc, progresul potențial al atacatorului va fi o distribuție Poisson cu o valoare așteptată:

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

Pentru a obține probabilitatea ca atacatorul să reușească acum să ajungă din urmă, multiplicăm densitatea Poisson pentru fiecare parte a progresului pe care l-ar fi putut face cu probabilitatea de a ajunge din urmă de la acel punct:

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

Rearanjând pentru a evita să adunăm o coadă infinită de distribuție...

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

Convertind în cod 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;
}
```

Rulând unele rezultate, observăm că probabilitatea scade exponențial cu 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
```

Rezolvând pentru P mai mic de 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. Concluzie

Am propus un sistem pentru tranzacții electronice fără a ne baza pe încredere. Am început cu cadrul de lucru uzual al monedelor realizate din semnături digitale, care oferă control puternic al proprietății, dar care este incomplet fără o cale de a preveni dubla cheltuire. Pentru a rezolva aceasta, am propus o rețea peer-to-peer care folosește dovada de lucru pentru a înregistra istoricul public al tranzacțiilor într-un mod care devine rapid nepractic din punct de vedere computațional pentru un atacator care ar dori să le modifice dacă nodurile oneste controlează o majoritate a puterii de procesare. Rețeaua este robustă în simplitatea sa nestructurată. Nodurile lucrează concomitent, cu o coordonare minimă. Nu au nevoie să fie identificate, întrucât mesajele nu sunt rutate către un loc anume, ci trebuie doar să fie transmise pe baza unui efort optim. Nodurile pot părăsi rețeaua și pot reintra după cum doresc, acceptând lanțul de dovezi de lucru ca fiind dovada pentru ce s-a întâmplat în lipsa lor. Ele votează prin intermediul puterii de procesare, exprimându-și acceptarea blocurilor valide lucrând la extinderea acestora și respingând blocurile invalide refuzând să lucreze asupra lor. Orice reguli și stimulente necesare pot fi puse în aplicare cu acest mecanism de consens.

Referințe

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

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

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," *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," *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," *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," *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.