Bitcoin: Ett peer-to-peer elektroniskt kontantsystem

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

Translated into Swedish from bitcoin.org/bitcoin.pdf
by @hanspandeya

Sammanfattning. En renodlad peer-to-peer version av ett elektroniskt kontant-system skulle möjliggöra direktbetalning från en part till en annan utan att behöva passera genom en bank eller finansiell institution. Digitala signaturer utgör en del av lösningen, men de huvudsakliga fördelarna med direktbetalning går förlorade om en betrodd tredje part alltjämt behövs för att förhindra dubbelbetalning. Vi föreslår en lösning på dubbelbetalningsproblemet som använder ett P2P-nätverk. Nätverket tidsstämplar transaktioner genom hashberäkning i en löpande kedja av hash-baserad proof-of-work; och formar en journal som inte kan ändras utan att proof-of-work arbetet görs om. Den längsta kedjan tjänar inte blott som bevis på ordningsföljden av händelser som iakttagits, utan fungerar även som bevis på att kedjan härrör från den största samlingen av CPU-kraft på nätverket. Så länge majoriteten av CPU-kraft styrs av noder, som inte samarbetar för att angripa nätverket, kommer den att generera den längsta kedjan och utkonkurrera angripare. Nätverket självt behöver minimal uppbyggnad. Meddelanden är utsända efter bästa förmåga, och noder kan lämna och återansluta till nätverket som de vill, samt godtar den längsta proof-of-work kedjan som bevis på händelser som inträffar när de är bortkopplade.

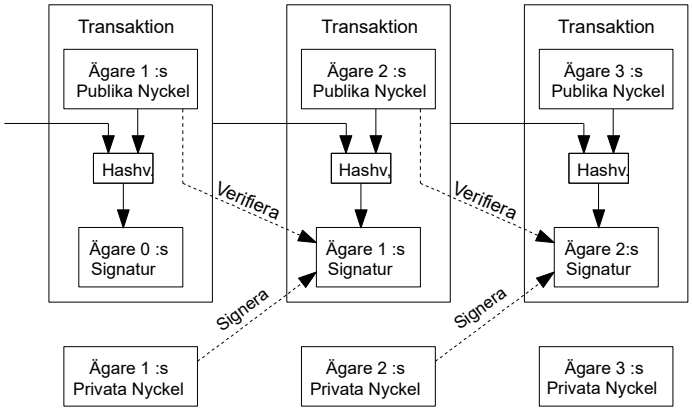
1. Introduktion

Handel på internet har kommit att förlita sig nästan uteslutande på finansiella institutioner som betjänar som betrodd tredje part vid elektronisk betalning. Medan systemet fungerar tillräckligt bra för de flesta transaktioner, lider det av inneboende svagheter hos den tillitsbaserade betalningsmodellen. Helt oåterkalleliga transaktioner är inte möjliga eftersom finansiella institutioner inte kan komma ifrån medling vid tvister. Medlingskostnader ökar transaktionskostnader; begränsar den minsta möjliga transaktionen; och beskär möjligheten att utföra små tillfälliga transaktioner. Därtill finns en större kostnad till följd av förlusten av möjligheten att göra irreversibla betalningar för tjänster som inte är uppsägningsbara. Med möjligheten att återkalla transaktioner ökar behovet av tillit. Handlare måste vara måna om sina kunders säkerhet och besvära dem för mer information än vad annars skulle behövas. En viss procent av bedrägeri godtas som oundviklig. Dessa kostnader och osäkerheter kan undvikas om betalning görs personligen med en fysisk valuta men det finns inte någon mekanism för att göra betalningar över en kommunikationskanal utan inblandning av en betrodd part.

Vad som behövs är ett elektroniskt betalsystem som bygger på kryptografiska bevis i stället för på tillit som tillåter två villiga parter att genomföra en transaktion direkt med varandra utan en betrodd tredje parts inblandning. Transaktioner som är beräkningsmässigt opraktiska att återkalla skulle skydda säljare mot bedrägeri, och rutinmässig deponering kan enkelt införas för att skydda köpare. I föreliggande artikel, föreslår vi en lösning på dubbelbetalningsproblemet som utnyttjar en P2P-distribuerad tidserver för att generera beräkningsbevis på transaktioners kronologiska ordning. Systemet är säkert så länge tillförlitliga noder har kontrollen över mer CPU-kraft på nätverket än någon grupp av sammarbetande angripare har.

2. Transaktioner

Vi definierar ett elektronisk mynt som en kedja av digitala signaturer. Varje ägare överför ett mynt till nästa ägare genom att digitalt signera föregående transaktionens hashvärde samt nästa ägares publika nyckel och lägga till dessa i slutet av myntet. En betalningsmottagare kan verifiera signaturer och verifiera kedjan av ägare.

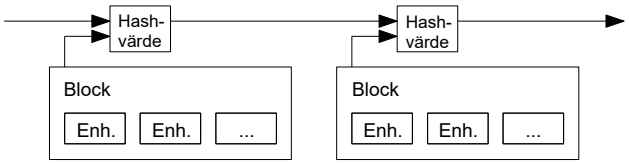


Problemet är givetvis att betalningsmottagaren inte kan kontrollera att en av ägarna inte dubbelbetalt. En vanlig lösning är att införa ett centralt betrodd organ eller utgivare som granskar varje transaktion för förekomst av dubbelbetalning. Efter varje transaktion måste således myntet lämnas tilllbaka till utgivaren som får utge ett nytt mynt, och endast mynt som kommer direkt från utgivaren kan förlitas på att inte vara dubbelbetalda. Problemet med denna lösning är att hela penningsystemets öde hänger på företaget som utger mynt och att varje transaktion måste passera genom det, precis som genom en bank.

Vi behöver utforma en lösning för att betalningsmottagaren med säkerhet ska veta att föregående ägare inte har signerat några tidigare transaktioner. För våra syften, är det den tidigaste transaktionen som räknas. Vi bryr oss inte om senare försök att dubbelbetala. Det enda sättet att säkerställa att en transaktion inte utförts är att ha kännedom om samtliga transaktioner som har ägt rum. I myntmodellen med en central utgivare, hade utgivaren kännedom om samtliga transaktioner och kunde avgöra vilken transaktion som anlänt först. För att åstadkomma motsvarande utan inblandning av en betrodd part måste transaktionerna offentligöras [1], och vi behöver införa ett förfarande så att deltagarna kan komma överens om transaktionshistorikens ordningsföljd. Betalningsmottagaren behöver bevis på att vid tidpunkten för varje transaktion, var majoriteten överens om att transaktionen var den första.

3. Tidserver

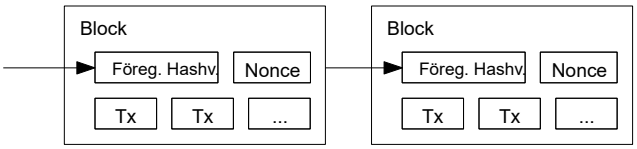
Lösningen vi föreslår inleds med en tidserver. En tidserver fungerar genom att beräkna hashvärdet av ett block av enheter som ska tidstämplas; och offentliggör resultatet med bred publicering såsom genom en tidning ett eller Usenet inlägg [2-5]. Tidsstämpeln är bevis på att datan måste ha existerat vid tidpunkten, uppenbart nödvändigt för att inkluderas i hashberäkningen. Varje tidsstämpel inkluderar föregående tidsstämpel i dess hashberäkning, och bildar på så sätt en kedja, där varje tillkommande tidsstämpel stärker tidigare tidsstämplars giltighet.



4. Proof-of-Work

För att implementera en distribuerad tidserver i ett P2P-nätverk, kommer vi att behöva använda ett proof-of-work metod liknande Adam Backs Hashcash [6], snarare än tidnings- eller Usenet-inlägg. Proof-of-work, bevis på utfört arbete, innefattar avsökning efter ett värde som vid hashberäkning, såsom med SHA-256, ger blockets hashvärde en insekvens av ett fixt antal nollbitar. Genomsnittsarbetet ökar exponentiellt med antalet nollor som krävs, och att arbetet verkligen utförts kan verifieras genom en enda hashberäkning.

För vårt tidsstämpelnätverk utför vi proof-of-work genom att stegvist öka nonce-värdet i blocket tills ett värde hittas som ger blockets hashvärde erforderligt antal nollbitar. När väl CPU-insatsen för att uppfylla proof-of-work investerats, kan inte blocket ändras utan att arbetet görs om igen. Då tillkommande block länkas efter det, kommer arbetet att ändra ett block även innefatta att göra om arbetet för samtliga block efter det.



Proof-of-work löser också problemet med att välja principen för representation av majoritetsbeslut. Om majoriteten bestäms enligt principen en IP-adress - en röst skulle den kunna överröstas av någon som lyckas samla ihop många IP-adresser. Proof-of-work bygger huvudsakligen på principen en CPU - en röst. Majoritetsbeslutet är således representerad av den längsta blockkedjan som den största insatsen för proof-of-work investerats i. Om en majoritet av CPU-kraften är styrd av tillförlitliga noder, kommer den tillförlitliga kedjan att växa snabbast och utkonkurrera varje annan konkurrerande kedja. För att modifiera ett avslutat block, skulle en angripare bli tvungen att göra om proof-of-work för blocket och för alla block efter det och därutöver komma ikapp samt överträffa de tillförlitliga noderna. Vi kommer senare att visa att sannolikheten att en långsammare angripare kommer ikapp avtar exponentiellt med antalet block som tilläggs.

För att kompensera för utvecklingen av allt snabbare hårdvara och för att intresset för drift av noder varierar över tid, regleras svårighetsgraden av proof-of-work av ett rörligt medelvärde som siktar på ett konstant antal nyskapade block per timme. Om block genereras för snabbt, ökar svårighetsgraden.

5. Nätverk

Stegen vid körning av nätverket är som följer:

1) Nya transaktioner sänds ut till alla noder.
2) Varje nod samlar ihop nya transaktioner i ett block.
3) Varje nod arbetar på en svårlöslig proof-of-work uppgift för sitt block.
4) När en nod lyckas hitta ett proof-of-work, utsänder den blocket till alla noder.
5) Noder godkänner blocket endast om alla transaktioner i det är giltiga och inte redan har utförts.

6) Noder uttrycker sitt godkännande av blocket genom att arbeta på att skapa nästa block i kedjan och använder det godkända blockets hashvärde som föregående hashvärde.

Noder betraktar alltid den längsta kedjan som den rätta och kommer att fortsätta att arbeta på dess förlängning. Om två noder utsänder olika versioner av nästa blocket samtidigt, kan några noder få den ena eller den andra först. I så fall, arbetar de på den första versionen som mottagits, men sparar den andra grenen utifall den skulle bli längre. Det oavgjorda läget kommer att brytas när nästa proof-of-work hittas och en gren blir längre; noder som arbetar på den andra grenen kommer då att övergå till att arbeta på den längre.

Sändningar av nya transaktioner behöver inte nödvändigtvis nå alla noder. Så länge de når många noder, kommer de att inkluderas i ett block inom kort. Block-utsändningar är också toleranta mot förlorade meddelanden. Om inte en nod mottagit ett utsänt block, kommer noden att begära det när den mottar nästa block och inser att den missat ett.

6. Incitament

Enligt seden är den första transaktionen i ett block en speciell transaktion som skapar ett mynt ägd av skaparen av blocket. Härigenom stärks incitamentet för noder att stödja nätverket och en väg skapas för att sätta nya mynt i cirkulation eftersom det inte finns någon central utgivare av mynt. Den stadiga tillförseln av ett konstant antal nya mynt är analogt med guldbrytares förbrukning av resurser för att sätta guld i omlopp. I vårt fall är det CPU-tid och el som är resurserna som förbrukas.

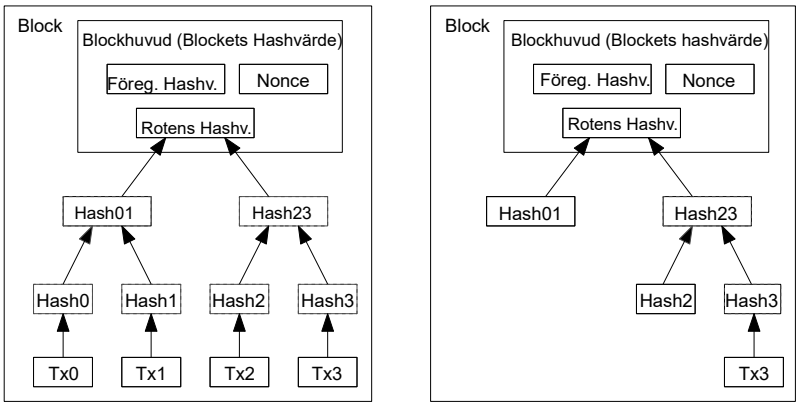
Incitamentet kan också finansieras av transaktionsavgifter. Om utdatavärdet av en transaktion är lägre än dess indatavärdet, utgör mellanskillnaden en transaktionsavgift som läggs till incitamentet för blocket. När ett förutbestämt antal mynt har trätt i cirkulation, kan incitamentet övergå till att bestå helt av transaktionsavgifter och vara helt inflationsfri.

Incitamentet kan bidra till att uppmuntra noder att förbli hederliga. Om en girig angripare lyckas samla ihop mer CPU-kraft än samtliga tillförlitliga noder tillsammans, blir han tvungen att välja mellan att använda CPU-kraften för att lura människor genom att stjäla tillbaka betalningar han har gjort till dem, eller använda kraften för att generera nya mynt. Han borde finna det mer lönande att spela enligt reglerna, sådana regler som belönar honom med fler nya mynt än alla andra tillsammans, istället för att underminera systemet och giltigheten av sin egen förmögenhet.

7. Återvinning av diskutrymme

När väl den senaste transaktionen med ett mynt är begravd under tillräckligt många block, kan transaktioner som utförts tidigare, raderas för att frigöra diskutrymme. För att underlätta detta utan att förstöra blockets hashvärde, hashberäknas transaktioner i ett Merkle-träd [7] [2] [5] med endast roten inkluderad i beräkningen. Gamla block kan därefter göras kompakta genom borttagande av grenar från trädet. Hashvärdena inuti behöver inte lagras.

Ett blockhuvud utan transaktioner tar upp ca. 80 bytes av minnet. Om vi antar att block genereras var tionde minut, krävs 80 bytes * 6 * 24 * 365 = 4.2MB av lagringsutrymme per år. Då datorsystem vanligtvis säljs med 2 GB RAM i 2008, och Moores lag förutspår att nuvarande tillväxt är 1,2 GB per år, bör lagring inte vara ett problem även om blockhuvudena måste lagras I internminnet.

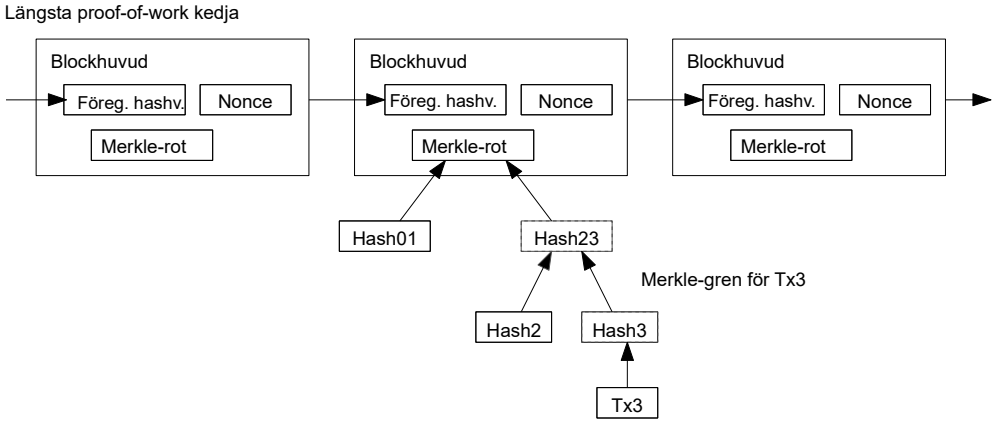


Transaktioner hashade i ett Merkle Träd

Efter borttagning av Tx0-2 från blocket

8. Förenklad betalningsverifikation

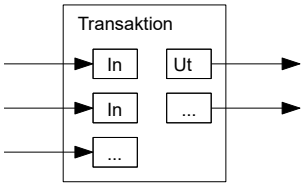
Det är möjligt att verifiera betalningar utan att ha en fullständig nätverksnod igång. En användare behöver endast behålla en kopia av blockhuvudena från den längsta proof-of-work kedjan, villket han kan erhålla genom förfrågningar till nätverksnoder tills han är övertygad om att han har den längsta kedjan, och komma åt Merkle-grenen som länkar transaktionen till blocket den är tidsstämplad i. Han kan inte verifiera transaktionen själv, men genom att länka den till en plats i kedjan, kan han se att en nätverksnod har accepterat den, varvid block som tillfogas efter den, ytterligare bekräftar att nätverket har godkänt transaktionen.



Som sådan, är verifikationen tillförlitlig så länge ärliga noder styr över nätverket, men blir mer osäker om nätverket tas över av en angripare. Medan nätverksnoder kan verifiera betalningar själva, kan den förenklade metoden luras av en angripares fabricerade transaktioner så länge angriparen har kontrollen över nätverket. En strategi för att skydda mot detta skulle vara att acceptera varningsmeddelanden från nätverksnoder när de detekterar ett ogiltigt block, vilket skulle igångsätta användarens programvarans nedladdning av det fullständiga blocket samt transaktioner som varnats för, så att inkonsistensen kan bekräftas. Företag som ofta mottar betalningar kommer förmodligen ändå vilja ha egna noder i drift för mer säkerhetsoberoende och snabbare verifikation de får i gengäld.

9. Kombinering och delning av värden

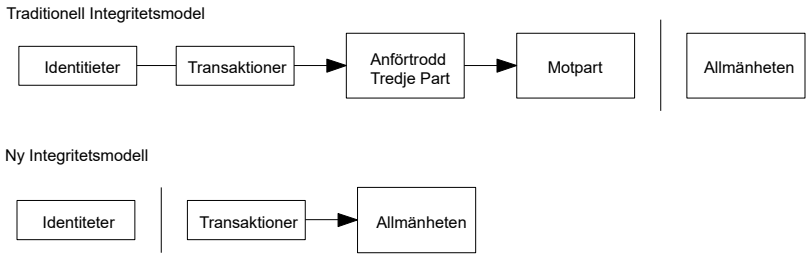
Trots att det skulle vara möjligt att behandla varje mynt för sig, skulle det vara kostsamt och ohanterligt att utföra en separat transaktion för varenda cent i en överföring. För att möjliggöra delning och kombinering av värden, innehåller transaktioner flera in- och utdata. Normalt kommer det finnas antingen enkel indata från en större föregående transaktion eller flera indata som kombinerar mindre värden, och som högst två utdata: en för betalningen, och en för att lämna tillbaka växel, om något blir över, till avsändaren.



Det bör noteras att fan-out, där en transaktion är beroende av många transaktioner, vilka i sin tur är beroende av många andra, inte är ett problem här. Det finns aldrig behov av att dra ut en fullständig fristående kopia av transaktionshistoriken.

10. Integritet

Den traditionella bankmodellen uppnår ett visst integritetsskydd genom begränsning av tillgång till information till inblandade parter samt betrodd tredje part. Behovet av att offentliggöra alla transaktioner utesluter emellertid ett sådant förfarande, men integritetsskydd kan ändå upprätthållas genom strypning av informationsflödet på ett annat sätt: nämligen genom att hålla publika nycklar anonyma. Allmänheten kan se att någon skickar en summa till någon annan, men inte personliga uppgifter som kopplar transaktionen till någon. Detta är jämförbart med skyddsnivån på information som släpps av börser, med tid och storlek på enskilda avslut, det s.k. "bandet", utan att avslöja vilka parterna är.



Som en ytterligare brandvägg, bör ett nytt nyckelpar användas för varje transaktion för att undvika att nyckelpar förknippas med en gemensam ägare. Viss koppling är emellertid oundviklig vid flerindata-transaktioner, som av nödvändighet avslöjar att deras indata tillhört en och samma ägare. Risken är att om innehavaren av en nyckel avslöjas, kan kopplingen avslöja andra transaktioner som tillhör innehavaren.

11. Beräkningar

Vi betraktar scenariot med en angripare som försöker att generera en alternativ kedja snabbare än den tillförlitliga kedjan. Även om detta uppnås, slås inte systemet upp och blottar det för godtyckliga ändringar, såsom värdeskapande ur tomma luften eller uttagande av pengar som aldrig tillhört angriparen. Noder kommer nämligen inte att godkänna en ogiltig transaktion som

betalning, och tillförlitliga noder kommer aldrig att godkänna ett block som innehåller sådana transaktioner. En angripare kan endast försöka ändra en av sina egna transaktioner för att få tillbaka pengar han nyligen använt.

Kapplöpningen mellan den tillförlitliga kedjan och en angripares kedja kan modelleras av en *Binomial slumpvandring*. Det lyckade utfallet är att den tillförlitliga kedjan blir förlängd med ett block, och ökar dess försprång med +1, medan det misslyckade utfallet är att angriparens kedja blir förlängd med ett block, och minskar gapet med -1.

Sannolikheten att en angripare kommer ikapp från ett givet underläge är analogt med *Spelarens ruinproblemet*. Antag att en spelare med obegränsad kredit startar med ett underskott och spelar potentiellt ett oändligt antal spelomgånger för att försöka nå break-even. Vi kan beräkna sannolikheten att han någonsin når break-even, eller att en angripare kommer ikapp den tillförlitliga kedjan, enligt följande [8]:

p = sannolikheten att en tillförlitlig nod hittar nästa block
q = sannolikheten att angriparen hittar nästa block
q_z = sannolikheten att angriparen någonsin kommer ikapp från att ligga z block efter.

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

Givet vårt antagande att p > q, faller sannolikheten exponentiellt när antalet block som angriparen måste komma ikapp ökar. Med oddsen mot sig, om inte angriparen har tur och gör ett lyckat språng framåt i ett tidigt skede, blir hans chanser försvinnande små allteftersom han hamnar alltmer på efterkälken. Vi betraktar nu hur länge mottagaren av en betalning måste vänta innan han är tillräckligt säker på att avsändaren inte kan ändra betalningen. Vi antar att avsändaren är en angripare som vill få mottagaren att tro ett tag att han betalt honom, och efter en stund återkallar betalningen för att betala tillbaka till sig själv. Mottagaren kommer att varnas när detta sker, men avsändaren hoppas att det kommer att vara för sent.

Mottagaren genererar ett nytt nyckelpar och ger den publika nyckeln till avsändaren strax innan signering. Härigenom förhindras avsändaren från att förbereda en kedja av block i förväg genom att arbeta oavbrutet på det tills han har tur nog att få ett tillräckligt försprång, och utför sedan transaktionen i det ögonblicket. När betalningen väl har skickats, börjar den ohederliga avsändaren arbeta i hemlighet på en parallell kedja innehållande en alternativ version av transaktionen.

Mottagaren väntar tills transaktionen har blivit inkluderat i ett block och z block har blivit länkade efter det. Han vet inte exakt hur stora framsteg angriparen har gjort, men om det antas att sammanlänkning av de hederliga blocken tagit den förväntade genomsnittstiden per block, kommer angriparens möjliga framsteg att vara fördelade enligt en Poissonfördelning med väntevärde:

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

För att erhålla sannolikheten att angriparen fortfarande nu kan komma ikapp, multiplicerar vi Poissonfördelningens täthetsfunktion för varje framsteg han har gjort med sannolikheten att han kan komma ikapp från den punkten:

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

Skriver om för att undvika att summera fördelningens oändliga svans ...

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

Implementerar med C-kod...

```
#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;
}
```

Vid testkörning av ett antal resultat, ser vi att sannolikheten faller exponentiellt med 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
```

Vi löser för P mindre än 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. Slutsats

Vi har föreslagit ett system för elektroniska transaktioner som inte är avhängig av tillit. Vi började med det vanliga ramverket för mynt som skapas med digitala signaturer, vilket ger stark kontroll över ägandet men är alltjämnt ofullständigt, utan någon mekanism som förhindrar dubbelbetalning. För att lösa problemet, föreslog vi ett P2P-nätverk som använder proof-of-work för att skapa en offentlig historik över transaktioner, som snabbt blir beräkningmässigt opraktiskt för en angripare att ändra, om tillförlitliga noder styr över majoriteten av CPU-kraft i nätverket. Nätverket är robust i sin ostrukturerad enkelhet. Noder arbetar alla samtidigt med lite samordning. De behöver inte vara identifierade, eftersom meddelanden inte vidarebefordras till någon särskild plats utan behöver endast levereras efter bästa förmåga. Noder kan lämna och återansluta till nätverket som de vill, samt accepterar proof-of-work kedjan som bevis på händelser som inträffar när de är bortkopplade. De röstar med sin CPU-kraft, och uttrycker sitt godkännande av giltiga block genom att arbeta på att förlänga dem, och sitt avvisande genom sin vägran att arbeta på dem. Nödvändiga regler och incitament kan genomföras med denna konsensusmekanism.

Referenser

[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.