Bitcoin: Een Peer-to-Peer Electronisch geld Systeem

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

Translated in Dutch from bitcoin.org/bitcoin.pdf by GiftbitNL

Inhoud. Een pure Peer-to-Peer versie van electronisch geld wat online betalingen mogelijk maakt om van de ene naar de andere partij over te maken zonder inmenging van een financiële instelling. Digitale handtekeningen zijn een deel van de oplossing, maar de belangrijkste voordelen gaan verloren als er alsnog een vertrouwde derde partij wordt vereist om dubbele betalingen te voorkomen. We bieden een oplossing voor het probleem met behulp van een Peer-to-Peer netwerk. Het netwerk timestampt transacties door ze in een doorlopende keten (chain) van op hash-gebaseerde Proof-of-Work te plaatsen wat een log vormt dat niet kan worden gewijzigd zonder de Proof-of-Work opnieuw uit te voeren. De langste chain dient niet alleen als bewijs van de volgorde van gebeurtenissen, maar ook als bewijs dat deze afkomstig was van de grootste pool van CPU-kracht. Zolang een meerderheid van de CPU-kracht wordt bestuurd door nodes die niet samenwerken om het netwerk aan te vallen, genereren ze de langste chain en laten ze aanvallers achterwege. Het netwerk zelf vereist een minimale structuur. Berichten worden op de best mogelijke manier uitgezonden en nodes kunnen wanneer dan ook het netwerk verlaten en weer meedoen door de langste Proof-of-Work-chain te accepteren als bewijs van wat er gebeurde toen ze weg waren.

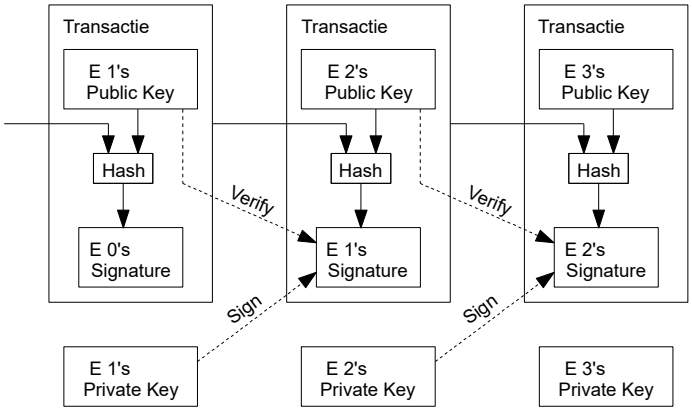
1. Introductie

Handel op het internet is bijna volledig afhankelijk geworden van financiële instellingen die als vertrouwde derde partijen dienen om elektronische betalingen te verwerken. Hoewel het systeem voor de meeste transacties goed genoeg werkt, heeft het nog steeds te lijden van de inherente zwakheden van het op vertrouwen gebaseerde model. Volledig niet-omkeerbare transacties zijn niet echt mogelijk omdat financiële instellingen er niet omheen kunnen bemiddelen bij disputen. De kosten van bemiddelingen verhogen de transactiekosten, beperken de minimale praktische transactiegrootte en sluiten de mogelijkheid voor kleine casuele transacties af, en er is een bredere kost in het verlies van het vermogen om niet-omkeerbare betalingen te doen voor niet-omkeerbare diensten. Met de mogelijkheid van omkeerbare betalingen spreidt de behoefte aan vertrouwen zich uit. Handelaren moeten op hun hoede zijn voor hun klanten, ze lastigvallen voor meer informatie dan ze anders nodig zouden hebben. Een bepaald percentage van fraude wordt als onvermijdelijk geaccepteerd. Deze kosten en betalingsonzekerheden kunnen persoonlijk worden vermeden door fysieke valuta te gebruiken, maar er bestaat geen mechanisme om betalingen via een communicatiekanaal uit te voeren zonder een vertrouwde partij.

Wat nodig is, is een elektronisch betalingssysteem gebaseerd op cryptografisch bewijs in plaats van vertrouwen, waardoor twee bereidwillige partijen rechtstreeks met elkaar transacties kunnen maken zonder de noodzaak van een vertrouwde derde partij. Transacties die computationeel onpraktisch zijn om terug te draaien beschermen verkopers tegen fraude, en routinematige escrowmechanismen kunnen eenvoudig worden geïmplementeerd om kopers te beschermen. In dit artikel bieden we een oplossing voor het probleem van de dubbele uitgaven met behulp van een Peer-to-Peer gedistribueerde timestampserver om computationeel bewijs van de chronologische volgorde van transacties te genereren. Het systeem is beveiligd zolang eerlijke nodes gezamenlijk meer CPU-kracht besturen dan elke samenwerkende groep aanvallende nodes.

2. Transacties

We definiëren een electronische coin als een chain van digitale handtekeningen.
Elke eigenaar (E) draagt de munt over naar de volgende door een hash van de vorige transactie en de publieke sleutel (Public Key) van de volgende eigenaar digitaal te ondertekenen en toe te voegen aan het einde van de coin.

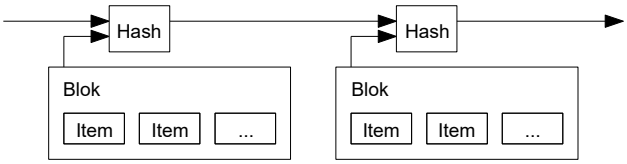


Het probleem is natuurlijk dat de ontvanger niet kan verifiëren dat een van de eigenaren de munt niet dubbel heeft uitgegeven. Een veelvoorkomende oplossing is om een vertrouwde centrale autoriteit of munt te introduceren die elke transactie controleert op dubbele uitgaven. Na elke transactie moet de munt naar de mint worden teruggebracht om een nieuwe munt uit te geven, en alleen munten die rechtstreeks uit de mint zijn uitgegeven worden niet dubbel uitgegeven. Het probleem met deze oplossing is dat het lot van het hele geldsysteem afhangt van het bedrijf dat de munt beheert, waarbij elke transactie net als een bank door hen heen moet gaan. We hebben een manier nodig voor de ontvanger om te weten dat de vorige eigenaren geen eerdere transacties hebben ondertekend. Voor onze doeleinden is de vroegste transactie de transactie die telt, dus we geven niet om latere pogingen om een dubbele besteding uit te voeren.

De enige manier om de afwezigheid van een transactie te bevestigen, is om op de hoogte te zijn van alle transacties. In het mintgebaseerde model was de mint op de hoogte van alle transacties en besliste welke als eerste arriveerde. Om dit zonder een vertrouwde partij te bereiken, moeten transacties openbaar worden aangekondigd [1], en we hebben een systeem nodig voor deelnemers om overeenstemming te bereiken over een enkele geschiedenis van de volgorde waarin ze zijn ontvangen. De ontvanger moet bewijzen dat op het tijdstip van elke transactie de meerderheid van de nodes was overeengekomen dat het de eerste was die werd ontvangen.

3. Timestamp Server

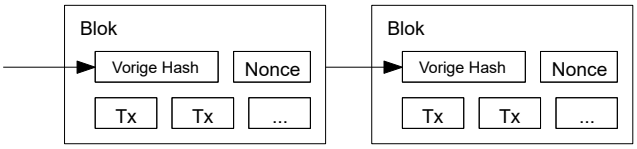
De oplossing die we bieden begint met een tijdstempelserver (timestamp server). Een timestamp server werkt door een hash te nemen van een blok met items om vervolgens een timestamp te krijgen en de hash op grote schaal te publiceren (zoals een krant of Usenet-bericht [2-5]). De timestamp bewijst dat de gegevens op dat moment bestonden, uiteraard om in de hash te komen. Elke timestamp bevat de vorige timestamp in zijn hash waardoor een chain wordt gevormd, waarbij elke extra timestamp de eerdere tijdstip versterkt.



4. Proof of Work (PoW)

Om een gedistribueerde timestampserver op een Peer-to-Peer-basis te implementeren zullen we een Proof-of-Work-systeem moeten gebruiken dat vergelijkbaar is met Adam Back's Hashcash [6] in plaats van kranten- of Usenet-berichten. Proof-of-Work betreft het zoeken naar een waarde dat wanneer hashed, zoals bij SHA-256, de hash begint met een aantal nul-bits. Het vereiste gemiddelde werk is exponentieel in het aantal vereiste nul bits en kan worden geverifieerd door het uitvoeren van een enkele hash.

Voor onze timestamp-netwerk implementeren we de Proof-of-Work door een nonce in het blok toe te voegen totdat een waarde wordt gevonden die de hash van het blok de vereiste nul bits geeft. Zodra de inspanning van de CPU is besteed om deze te laten voldoen aan de Proof-of-Work kan het blok niet worden gewijzigd zonder het werk opnieuw uit te voeren. Naarmate latere blokken erachter worden geketend, omvat het werk om het blok te veranderen ook het opnieuw doen van alle blokken erna.



Het proof-of-work lost ook het probleem op van het bepalen van representatie bij het nemen van meerderheidsbeslissingen. Als de meerderheid was gebaseerd op één-IP-adres-één stem zou het kunnen worden ondermijnd door iemand die veel IP's kan toewijzen. Proof-of-work is in essentie één-CPU-één-stem. De meerderheidsbeslissing wordt vertegenwoordigd door de langste chain die de grootste Proof of Work heeft geïnvesteerd. Als een meerderheid van de CPU-energie wordt gecontroleerd door eerlijke nodes, zullen de eerlijke nodes het snelst groeien en de concurrerende nodes overtreffen. Om een blok uit het verleden te wijzigen moet een aanvaller de Proof of Work van het blok en alle blokken erna opnieuw uitvoeren en vervolgens het werk van de eerlijke nodes inhalen en overtreffen. We zullen later laten zien dat de kans dat een langzamere aanvaller inhaalt exponentieel afneemt naarmate volgende blokken worden toegevoegd.

Om de toenemende hardwaresnelheid en de variërende interesse in lopende nodes in de loop van de tijd te compenseren, wordt de moeilijkheidsgraad van het werk bepaald door een voortschrijdend gemiddelde dat is gericht op een gemiddeld aantal blokken per uur. Als ze te snel worden gegenereerd, neemt de moeilijkheidsgraad toe.

5. Netwerk

De stappen voor het uitvoeren van het netwerk zijn als volgt:

1) Nieuwe transacties worden uitgezonden naar alle nodes.
2) Elke node verzamelt nieuwe transacties in een blok.
3) Elke node werkt aan het vinden van een moeilijk Proof-of-Work voor zijn blok.
4) Wanneer een node een Proof-of-Work vindt, zendt het het blok naar alle nodes.
5) Nodes accepteren het blok alleen als alle transacties daarin geldig zijn en nog niet zijn uitgegeven.
6) Nodes drukken hun acceptatie van het blok uit door te werken aan het creëren van het volgende blok in de chain met behulp van de hash van het geaccepteerde blok als de vorige hash.

Nodes beschouwen de langste chain altijd als de juiste en blijven werken aan de uitbreiding ervan. Als twee nodes tegelijkertijd verschillende versies van het volgende blok uitzenden, kunnen sommige nodes het een of ander eerst ontvangen. In dat geval werken ze aan de eerste die ze hebben ontvangen, maar bewaren ze de andere tak voor het geval het langer wordt. De das wordt verbroken wanneer de volgende Proof-of-Work wordt gevonden en een tak langer wordt; de nodes die aan de andere tak werkten schakelen dan over naar de langere tak.

Nieuwe transactie-uitzendingen hoeven niet noodzakelijk alle nodes te bereiken. Zolang ze vele nodes bereiken komen ze al snel in een blok. Blokuitzendingen zijn ook tolerant ten aanzien van gevallen berichten. Als een node geen blok ontvangt, vraagt het dit aan wanneer het de volgende blok ontvangt en realiseert dat het een heeft gemist.

6. Incentive

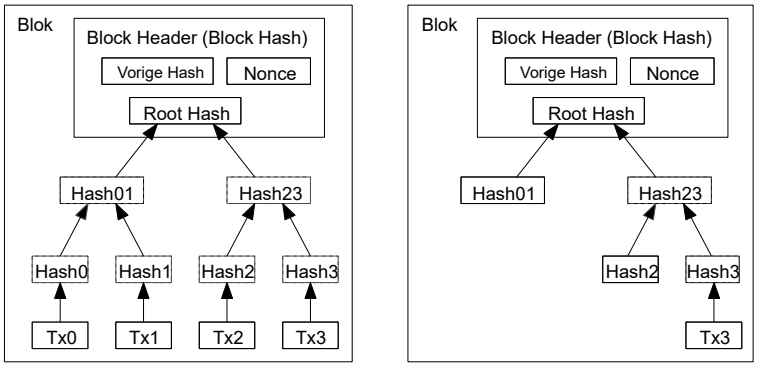
Volgens afspraak is de eerste transactie in een blok een speciale transactie waarbij een nieuwe munt wordt gestart die eigendom is van de maker van het blok. Dit voegt een stimulans toe voor nodes om het netwerk te ondersteunen en biedt een manier om eerst munten in omloop te brengen, aangezien er geen centrale autoriteit is om ze uit te geven. De gestage toevoeging van een constante hoeveelheid nieuwe munten is analoog aan goudzoekers die middelen besteden om goud aan de circulatie toe te voegen. In ons geval is het CPU-tijd en elektriciteit die wordt verbruikt.

De incentive kan ook worden gefinancieerd met transactiekosten. Als de output van een transactie kleiner is dan de input, is het verschil de transactiekosten die worden toegevoegd aan de stimuleringswaarde van het blok dat de transactie bevat. Zodra een vooraf bepaald aantal munten in omloop zijn gekomen, kan de stimulans volledig overgaan op transactiekosten en volledig inflatie-vrij zijn.

De incentive kan helpen om nodes aan te moedigen om eerlijk te blijven. Als een hebzuchtige aanvaller in staat is meer CPU-kracht te verzamelen dan alle eerlijke nodes, zou hij moeten kiezen tussen het gebruik ervan om mensen te bedriegen door zijn betalingen te stelen of het te gebruiken om nieuwe munten te genereren. Hij zou het winstgevender moeten vinden om volgens de regels te spelen, zoals regels die hem bevoordelen met meer nieuwe munten dan alle anderen gecombineerd, dan om het systeem en de geldigheid van zijn eigen rijkdom te ondermijnen.

7. Schijfruimte terugwinnen

Zodra de laatste transactie in een munt onder voldoende blokken is begraven, kunnen de gebruikte transacties daarvoor weggegooid worden om schijfruimte te besparen. Om dit te vergemakkelijken zonder de hash van het blok te verbreken worden transacties gehasht in een Merkle Tree [7] [2] [5] met alleen de root in de hash van het blok. Oude blokken kunnen vervolgens worden samengeperst door takken van de boom af te stoten. De binnenbehuizingen hoeven niet te worden opgeslagen.



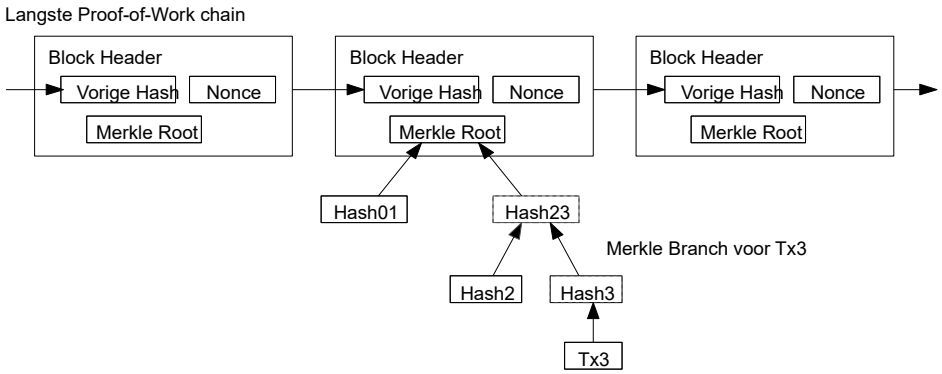
Transacties Hashed in een Merkle Tree

Na het snoeien van Tx0-2 van het blok

Een block-header zonder transacties zou ongeveer 80 bytes zijn. Als we er van uit gaan dat blokken elke 10 minuten worden gegenereerd, 80 bytes * 6 * 24 * 365 = 4,2 MB per jaar. Omdat computersystemen gewoonlijk vanaf 2008 met 2 GB RAM verkopen en de wet van Moore een huidige groei van 1,2 GB per jaar voorspelt, zou opslag geen probleem moeten zijn, zelfs als de blokheaders in het geheugen moeten worden bewaard.

8. Vereenvoudigde betalingsverificatie

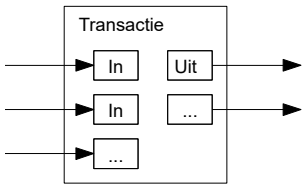
Het is mogelijk om betalingen te verifiëren zonder een volledige node te gebruiken. Een gebruiker hoeft alleen een kopie te bewaren van de block-headers van de langste proof-of-work-ketting, die hij kan krijgen door netwerknodes te ondervragen totdat hij overtuigd is dat hij de langste ketting heeft, en de Merkle-tak verkrijgt die de transactie aan het blok koppelt die voorzien is van een tijdstempel. Hij kan de transactie niet voor zichzelf controleren, maar door hem te koppelen aan een plaats in de chain kan hij zien dat een netwerknode het heeft geaccepteerd, en blokken die zijn toegevoegd nadat deze hebben bevestigd dat het netwerk het heeft geaccepteerd.



Als zodanig is de verificatie betrouwbaar zolang eerlijke nodes het netwerk besturen, maar kwetsbaarder is als het netwerk wordt overmeesterd door een aanvaller. Hoewel netwerknodes transacties voor zichzelf kunnen verifiëren, kan de vereenvoudigde methode worden misleid door de verzonnen transacties van een aanvaller zolang de aanvaller het netwerk kan blijven overmeesteren. Eén strategie om hiertegen te beschermen zou zijn om waarschuwingen van netwerknodes te accepteren wanneer ze een ongeldig blok detecteren, waarbij de software van de gebruiker wordt gevraagd om het volledige blok en gealigneerde transacties te downloaden om de inconsistentie te bevestigen. Bedrijven die veelvuldig worden betaald willen waarschijnlijk nog steeds hun eigen nodes gebruiken voor meer onafhankelijke beveiliging en snellere verificatie.

9. Waarde combineren en splitsen

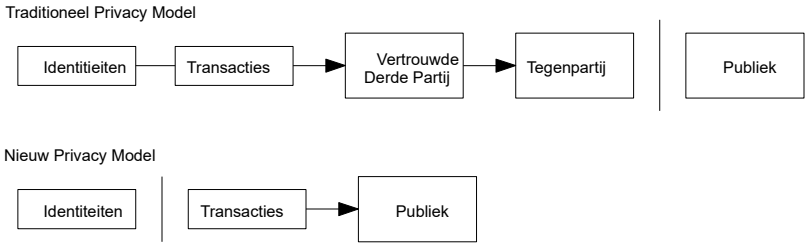
Hoewel het mogelijk zou zijn om munten afzonderlijk te behandelen, zou het onpraktisch zijn om voor elke cent in een overdracht een afzonderlijke transactie te doen. Om de waarde te kunnen splitsen en combineren bevatten transacties meerdere inputs en outputs. Normaal gesproken zal er of een enkele invoer zijn van een grotere eerdere transactie of meerdere inputs die kleinere bedragen combineren, en maximaal twee outputs: één voor de betaling en één die de wijziging, indien aanwezig, terugstuurt naar de afzender.



Het moet opgemerkt worden dat fan-out, waarbij een transactie afhankelijk is van verschillende transacties, en die transacties afhankelijk zijn van veel meer, hier geen probleem is. Het is nooit nodig om een volledig op zichzelf staande kopie van de geschiedenis van een transactie te extraheren.

10. Privacy

Het traditionele bankmodel bereikt een niveau van privacy door de toegang tot informatie te beperken tot de betrokken partijen en de vertrouwde derde partij. De noodzaak om alle transacties aan te kondigen sluit publiekelijk deze methode uit, maar privacy kan nog steeds worden gehandhaafd door de informatiestroom op een andere plaats te doorbreken: door openbare sleutels anoniem te houden. Het publiek kan zien dat iemand een bedrag naar iemand anders stuurt, maar zonder informatie die de transactie aan iemand linkt. Dit is vergelijkbaar met het niveau van informatie vrijgegeven door beurzen, waar de tijd en omvang van individuele transacties (de "tape") openbaar wordt gemaakt, maar zonder te vertellen wie de partijen waren.



Als extra firewall moet voor elke transactie een nieuw sleutelpaar worden gebruikt om te voorkomen dat ze aan een gemeenschappelijke eigenaar worden gekoppeld. Sommige koppelingen zijn nog steeds onvermijdelijk bij transacties met meerdere inputs die noodzakelijkerwijs onthullen dat hun invoer eigendom was van dezelfde eigenaar. Het risico is dat als de eigenaar van een sleutel wordt onthuld, het linken zou andere transacties kunnen onthullen die toebehoorden aan dezelfde eigenaar.

11. Berekeningen

We beschouwen het scenario van een aanvaller die sneller een alternatieve chain probeert te genereren dan de eerlijke chain. Zelfs als dit wordt bereikt, gooit het het systeem niet open voor arbitraire veranderingen, zoals het creëren van waarde uit het niets of het nemen van geld dat nooit aan de aanvaller toebehoorde. Nodes accepteren geen ongeldige transactie als betaling en eerlijke nodes accepteren nooit een blok dat deze bevat. Een aanvaller kan alleen proberen om een van zijn eigen transacties te wijzigen om geld terug te nemen dat hij onlangs heeft uitgegeven.

De race tussen de eerlijke chain en een aanvallende chain kan worden gekarakteriseerd als een binominale willekeurige wandeling. De succesgebeurtenis is dat de eerlijke chain wordt verlengd met één blok, waardoor de voorsprong met +1 toeneemt, en de foutgebeurtenis is de chain van de aanvaller die wordt uitgebreid met één blok, waardoor de kloof met -1 wordt verkleind.

De kans dat een aanvaller een bepaald tekort inhaalt is analoog aan het probleem van een Gambler's Ruin. Stel dat een gokker met onbeperkt krediet begint met een tekort en mogelijk een oneindig aantal pogingen doet om break-even te worden. We kunnen de waarschijnlijkheid berekenen dat hij ooit breakeven bereikt, of dat een aanvaller ooit de eerlijke chain inhaalt. Het gaat als volgt [8]:

p = kans dat een eerlijke node het volgende blok vindt
q = kans dat de aanvaller het volgende blok vindt
q_z = kans dat de aanvaller de achterliggende z blokkeert

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

Gegeven onze aanname dat p> q, daalt de kans exponentieel als het aantal blokken dat de aanvaller moet inhalen stijgt. Met de tegenvaller bij hem, als hij geen geluk maakt om vroeg naar voren te komen, worden zijn kansen verdwijnend klein naarmate hij verder achterop raakt.

We gaan nu na hoe lang de ontvanger op een nieuwe transactie moet wachten voordat hij voldoende zeker is dat de verzender de transactie niet kan wijzigen. We gaan ervan uit dat de afzender een aanvaller is die de ontvanger wil laten geloven dat hij een tijdje heeft betaald, en vervolgens overschakelt om zichzelf na enige tijd terug te betalen. De ontvanger zal worden gewaarschuwd wanneer dat gebeurt, maar de afzender hoopt dat het te laat zal zijn.

De ontvanger genereert een nieuw sleutelpaar en geeft de public key aan de afzender kort voordat hij ondertekent. Dit voorkomt dat de zender van tevoren een reeks blokken voorbereidt door er continu aan te werken totdat hij genoeg geluk heeft om ver genoeg vooruit te komen en vervolgens de transactie op dat moment uit te voeren. Zodra de transactie is verzonden, begint de oneerlijke afzender in het geheim te werken aan een parallelle ketting met een alternatieve versie van zijn transactie.

De ontvanger wacht totdat de transactie aan een blok is toegevoegd en er z-blokken achter zijn gekoppeld. Hij weet niet precies hoeveel vooruitgang de aanvaller heeft geboekt, maar gaat er van uit dat de eerlijke blokken de gemiddelde verwachte tijd per blok hebben genomen, is de potentiële voortgang van de aanvaller een Poisson-verdeling met verwachte waarde:

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

Om de waarschijnlijkheid te krijgen dat de aanvaller nu nog kan inhalen, vermenigvuldigen we de Poisson-dichtheid voor elke vooruitgang die hij gemaakt zou kunnen hebben door de waarschijnlijkheid dat hij vanaf dat punt kon inhalen:

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

Herinrichten om te voorkomen dat de oneindige staart van de verdeling wordt opgeteld ...

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

Converteren naar C code...

```
#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;
}
```

Als we wat resultaten uitvoeren, zien we dat de kans exponentieel afneemt met 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
```

Oplossing voor P minder dan 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. Conclusie

We hebben een systeem voor elektronische transacties voorgesteld die trouweloos is. We zijn begonnen met het gebruikelijke kader van munten die bestaat uit digitale handtekeningen, die sterke controle over eigendom biedt, maar onvolledig is zonder een manier om dubbele uitgaven te voorkomen. Om dit op te lossen hebben we een Peer-to-Peer-netwerk met gebruik van Proof-of-Work aangeboden om een openbare geschiedenis van transacties vast te leggen die snel rekenkundig onpraktisch wordt voor een aanvaller om te veranderen als eerlijke nodes de meerderheid van de CPU-kracht besturen. Het netwerk is robuust in zijn ongestructureerde eenvoud. Nodes werken allemaal tegelijk met weinig coördinatie. Ze hoeven niet te worden geïdentificeerd omdat berichten niet naar een bepaalde plaats worden gerouteerd en alleen op de best mogelijke manier moeten worden afgeleverd. Nodes kunnen vertrekken en naar believen weer samenkomen in het netwerk, en accepteren de Proof of Work als bewijs van wat er gebeurde toen ze weg waren. Ze stemmen met hun CPU-kracht en drukken hun acceptatie van geldige blokken uit door eraan te werken, deze uit te breiden en ongeldige blokken te verwerpen door te weigeren om eraan te werken. Alle benodigde regels en incentives kunnen worden afgedwongen met dit consensusmechanisme.

Referenties

[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.