Bitcoin: Een Peer-to-Peer Elektronisch Geldsysteem

Satoshi Nakamoto

satoshin@gmx.com

www.bitcoin.org

Nederlandse vertaling: 14tE9fUuvSgZ7NyoMJBn2PKH4ApngWPQUG

Abstract. Een volledig peer-to-peer versie van elektronisch geld staat toe om online betalingen direct te versturen van de ene partij naar de andere zonder langs een financiële instantie te gaan. Digitale handtekeningen zorgen voor een deel van de oplossing, maar de grootste voordelen gaan verloren als er nog steeds een vertrouwde derde partij vereist is om dubbele uitgaves te voorkomen. Wij stellen een oplossing voor om probleem van dubbele uitgaven voor via het gebruik van een peer-to-peer netwerk. Dit netwerk plaatst tijdstempels op transacties door ze te "hashen" in een constant groeiende keten van op hashes gebaseerde bewijs-van-werk (proof of work). De langste keten dient hierbij niet louter als bewijs van de volgorde van waargenomen gebeurtenissen, maar bewijst dat deze kwam van de grootste poule van CPU kracht. Zolang de meerderheid van die CPU kracht gecontroleerd wordt door knooppunten in het netwerk die niet samenwerken om dit netwerk aan te vallen, zullen zij de langste keten genereren en de aanvallers te snel af zijn. Het netwerk zelf vergt een minimale structuur. Boodschappen worden verstuurd op een best-effort-basis, en knooppunten kunnen zich op het netwerk aansluiten en verlaten op elk moment dat zij dat willen, zolang ze de langste proof-of-work keten aanvaarden als bewijs van wat er gebeurd is tijdens hun afwezigheid.

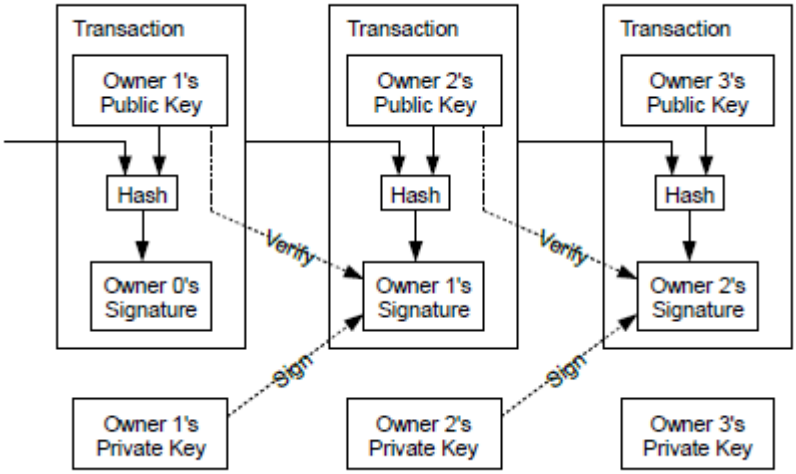
1. Introductie

Handel op het internet vertrouwt vandaag de dag bijna uitsluitend op financiële instellingen die dienst doen als vertrouwde derde partijen bij het verwerken van elektronische betalingen. Terwijl dit systeem goed genoeg werkt voor de meeste transacties, lijdt het nog steeds onder de inherente zwakke plekken van het op vertrouwen gebaseerde model. Compleet onomkeerbare transacties zijn niet echt mogelijk, sinds de financiële instituties het niet kunnen vermijden om te bemiddelen bij disputen. De kostprijs van deze tussenkomsten verhoogt transactiekosten, beperkt de minimum praktische transactiegrootte en de mogelijkheid om kleine losse transacties te plegen. Daarnaast is er een bredere kost bij het verliezen van het maken van onomkeerbare betalingen en onomkeerbare diensten. Met de mogelijkheid voor een terugboeking, vergroot de nood voor de vertrouwensfactor. Verkopers moeten op hun hoede zijn voor hun klanten, en hen lastigvallen voor meer informatie dan ze anders zouden nodig hebben. Een zeker percentage aan fraude wordt aanvaard als onvermijdelijk. Deze kosten en onzekerheden bij betalingen kunnen worden vermeden wanneer iemand in persoon fysieke valuta gebruikt, maar er bestaat geen enkel mechanisme om betalingen te maken via een communicatiekanaal zonder een vertrouwde partij.

Er is behoefte aan een elektronisch betalingssysteem gebaseerd op cryptografisch bewijs in de plaats van vertrouwen, dat aan twee daartoe bereide partijen toestaat rechtstreeks met elkaar te handelen zonder de nood aan een vertrouwde derde partij. Transacties die computationeel onpraktisch zijn om terug te boeken zouden verkopers beschermen van fraude, en routine escrows of derdenrekeningen zouden gemakkelijk kunnen worden geïmplementeerd om kopers te beschermen. In deze paper, stellen we een oplossing voor die het dubbele uitgave probleem aankaart via een peer-to-peer verdeelde tijdstempel server om rekenkundig bewijs te leveren van de chronologische volgorde van transacties. Dit systeem is veilig zolang eerlijke knooppunten collectief meer CPU kracht leveren dan die van een collaborerende groep van aanvallende knooppunten.

2. Transacties

We definiëren een elektronische munt als een keten van digitale handtekeningen. Elke eigenaar transfereert de munt naar de volgende door een digitale handtekening te plaatsen op een hash van de voorgaande transactie en de publieke sleutel van de volgende eigenaar en deze toe te voegen aan het einde van de munt. Een te betalen persoon kan deze handtekeningen verifiëren om de keten van eigenaarschap te verifiëren.



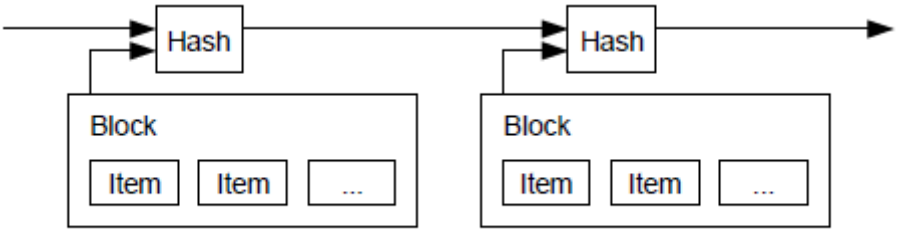
Het probleem is natuurlijk dat de te betalen persoon niet kan verifiëren dat een van de eigenaars de munt niet twee keer heeft uitgegeven. Een veel voorkomende oplossing is de introductie van een vertrouwde centrale autoriteit, of munter, die elke transactie controleert op dubbele uitgaves. Na elke transactie, dient de munt terug te keren naar de munterij, om een nieuwe munt uit te giften, en enkel munten die zo rechtstreeks in circulatie gebracht zijn kunnen vertrouwd worden om niet dubbel uitgegeven te worden. Het probleem met deze oplossing is dat het lot van het hele monetaire systeem afhangt van het bedrijf achter deze muntslag, omdat elke transactie langs hen moet gaan, net als bij een bank.

We hebben een manier nodig voor de te betalen persoon om te weten dat de vorige eigenaars geen eerdere transacties had getekend. Voor onze doeleindes, de vroegste transactie is degene die telt, zodat we ons dan geen zorgen meer maken om latere pogingen tot een dubbele uitgave. De enige manier om de afwezigheid van transacties te bevestigen is op de hoogte zijn van alle transacties. In dit munterij gebaseerde model, was de munterij bewust van elke transacties om te beslissen welke er het eerste arriveerde. Om dit te verwezenlijken zonder gebruik van een vertrouwde partij, moeten transacties plaatsvinden in de volgorde van ontvangst. De ontvanger heeft bewijs nodig dat op het tijdstip van elke transactie, de meerderheid van de knooppunten akkoord ging dat het de eerst ontvangen transactie was.

3. Tijdstempel server

De oplossing die wij voorstellen, begint met een tijdstempel server. Een tijdstempel server werkt door het nemen van een hash van een block items die een tijdstempel moeten krijgen en het ruim publiceren van die hash, zoals bijvoorbeeld in een krant of een Usenet bericht [2-5]. De tijdstempel bewijst dat de

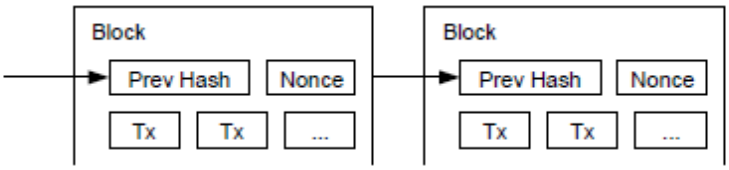
data bestaan moet hebben op dat moment, natuurlijk, om in de hash te kunnen geraken. Elke tijdstempel bevat de vorige tijdstempel in zijn hash, en vormt een keten, waarin elke vorige tijdstempel de stempel daarvoor versterkt.



4. Proof-of-Work

Om een gedistribueerde tijdstempel server op een peer-to-per basis te implementeren, zullen we een proof-of-work systeem vergelijkbaar met Adam Blacks Hashcash moeten gebruiken, in de plaats van krantenartikels of Usenet berichten. Deze proof-of-work omvat het scannen naar een waarde die wanneer gehasht, zoals met SHA-256, de hash begint met een getal van nul bits. Het gemiddelde benodigde werk hiervoor is exponentieel in het aantal van nul bits vereist en kan geverifieerd worden door het uitvoeren van een enkele hash.

Voor ons tijdstempelnetwerk implementeren we het proof-of-work systeem door het verhogen van een nonce (gelegenheidsgegevens) in het blok tot er een waarde wordt gevonden die de hash in het blok de vereiste nul bits geeft. Eens de CPU inspanning besteed is om te voldoen aan het proof-of-work systeem, kan het blok niet meer veranderd worden zonder al het werk te herdoen. Wanneer later blokken aan elkaar geketend worden, omvat het werk dat nodig is om het huidige blok te veranderen, het opnieuw uitvoeren van het werk van alle blokken die erna komen.



Het proof-of-work systeem lost daarnaast ook het probleem op van het bepalen van vertegenwoordiging in besluitvorming. Wanneer de meerderheid gebaseerd zou zijn op basis van één IP-adres-per-stem, zou dat systeem omzeild kunnen worden door gelijk wie dat meerdere IP's kan toewijzen. Proof-of-work is in essentie één CPU = één stem. De meerderheidsbeslissing is vertegenwoordigd door de langste keten, die de grootste proof-of-work inspanning in zich geïnvesteerd heeft. Wanneer een meerderheid van de CPU kracht onder controle is van eerlijke knooppunten, zal de eerlijke keten het snelst groeien en alle concurrerende ketens voorbijstreven. Om een voorgaande blok aan te passen zou een aanvaller al het werk van de proof-of-work van het blok en alle blokken erna moeten opnieuw doen en dan het werk van de eerlijke knooppunten moeten inhalen en overtreffen. We zullen later aantonen dat de waarschijnlijkheid dat een tragere aanvaller kan inhalen exponentieel verkleint naarmate er meer blokken toegevoegd worden.

Om te compenseren voor een verhoging in hardware snelheid en met de tijd afwisselende interesse om knooppunten te draaien, is de moeilijkheidsgraad van het proof-of-work systeem bepaald door een voortschrijdend gemiddelde dat gebaseerd is op het aantal blokken per uur. Wanneer de blokken te snel worden aangemaakt, verhoogt de moeilijkheidsgraad.

5. Netwerk

De stappen om het netwerk te onderhouden zijn als volgt:

1) Nieuwe transacties worden uitgezonden naar alle knooppunten
2) Elk knooppunt verzamelt nieuwe transacties in een blok
3) Elk knooppunt werkt aan het vinden van een moeilijke proof-of-work voor zijn blok
4) Wanneer een knooppunt een proof-of-work vindt, zendt die dat uit naar alle knooppunten
5) Knooppunten accepteren een blok enkel als alle transacties erin geldig zijn en niet reeds uitgegeven zijn.
6) Knooppunten drukken hun aanvaarding van een blok uit door te werken aan de creatie van het volgende blok in de ketting, door de hash van het vorige aanvaarde blok te gebruiken.

Knooppunten zien de langste keten telkens als de correcte en zullen verder werken aan de verlenging ervan. Wanneer twee knooppunten gelijktijdig verschillende versies van het volgende blok uitzenden, zullen sommige knooppunten de een of de andere het eerst ontvangen. In dat geval, werken zij verder met de eerste die zij ontvangen hebben, maar bewaren ze de andere tak in het geval deze langer zou worden. Deze band wordt gebroken wanneer de volgende proof-of-work gevonden is en één tak langer wordt. De knooppunten die aan de andere tak aan het werken waren, schakelen dan over naar de langere tak.

Nieuwe transacties hoeven niet noodzakelijk alle knooppunten te bereiken. Zolang zij vele knooppunten bereiken, zal het niet te lang duren vooraleer ze opgenomen worden in een blok. Blok transacties zijn daarnaast ook tolerant voor verloren boodschappen. Wanneer een kooppunten geen blok ontvangt, zal hij die aanvragen bij ontvangst van de volgende blok en beseffen dat hij er één had gemist.

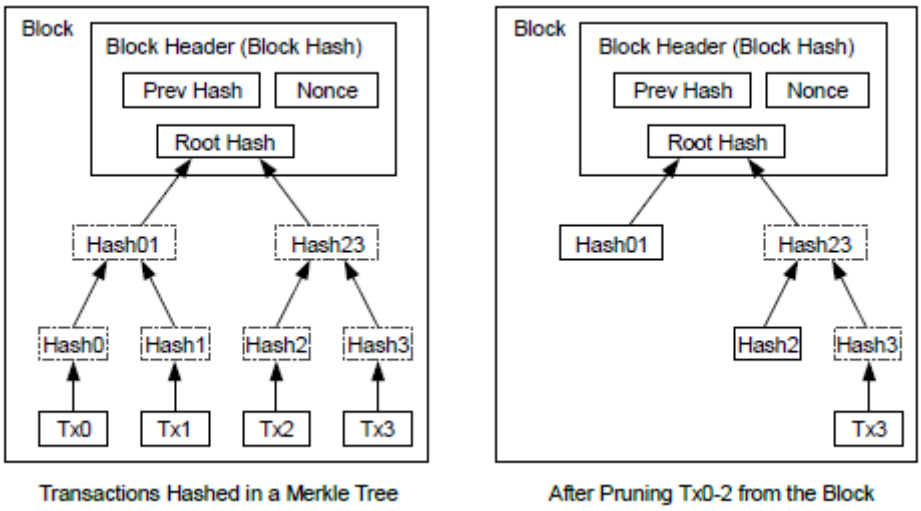
6. Stimulansen

Volgens afspraak is de eerste transactie in een blok een speciale transactie die een munt creëert die eigendom wordt van de maker van het blok. Dit zorgt voor een stimulans voor knooppunten om het netwerk te ondersteunen, en schenkt hen een manier om allereerst munten in omloop te brengen, gezien er geen centrale autoriteit is die ze munt. Deze gestage toevoer van een constante aantal nieuwe munten is analoog aan gouddelvers die grondstoffen verbruiken om goud in omloop te brengen. In ons geval, wordt CPU tijd en elektriciteit verbruikt.
Deze prikkel kan ook verder gevoed worden door transactiekosten. Wanneer de output waarde minder is dan zijn input waarde, is het verschil een transactiekost die wordt toegevoegd aan de stimulanswaarde van het blok die de transactie bevat. Zodra een vooraf bepaald aantal munten in omloop gebracht is, kan de stimulans volledig overgaan naar transactiekosten en volledig inflatievrij zijn.
Deze stimulans kan helpen om knooppunten eerlijk te houden. Wanneer een hebberige aanvaller de mogelijkheid heeft om meer CPU kracht te verzamelen dan alle eerlijke knooppunten samen, zal hij moeten kiezen tussen het frauderen van mensen door hun betalingen terug te stelen, of om het te

gebruiken om nieuwe munten aan te maken. Hij zou het meer winstgevend moeten vinden om volgens de regels te spelen, wanneer deze regels hem meer gunst doen met meer munten dan iedereen gecombineerd, dan door het systeem te ondermijnen samen met de geldigheid van zijn eigen rijkdommen.

7. Opslagruimte terugeisen

Zodra de laatste transactie in een munt begraven is onder genoeg blokken, kunnen de bestede transacties weggegooid worden om schijfruimte te besparen. Om dit gemakkelijker te maken zonder de hash van een blok te breken, worden transacties gehasht in een *Merkle Boom* [7] [2] [5], met enkel de wortel (root) opgenomen in de hash van het blok. Oude blokken kunnen dan compacter gemaakt worden door het kappen van de takken van de boom. De binnenste hashes hoeven niet opgeslagen worden.

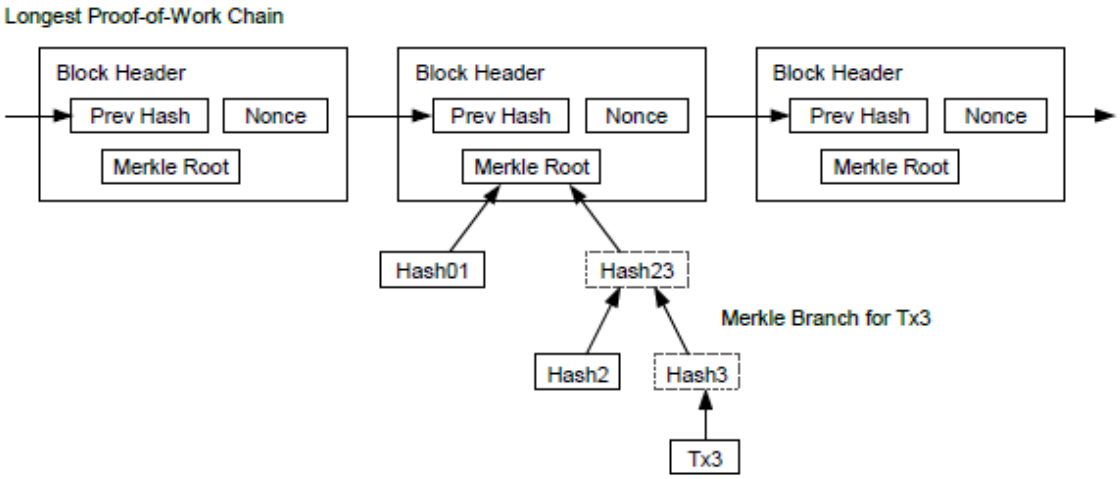


(Transacties gehasht in een Merkle Boom) (Na Tx0-5 te snoeien van het blok)

Een blok header zonder transacties zou rond de 80 bytes zijn. Als we veronderstellen dat blokken elke tien minuten gegenereerd worden, dan is 80 bytes * 6 * 24 * 365 = 4.2 MB per jaar. Met computersystemen die doorgaans 2 GB RAM hebben sinds 2008 en de Wet van Moore die een huidige groei van 1.2 GB per jaar voorspelt, zou de opslag geen probleem mogen zijn zelfs als de blok headers in het geheugen zouden moeten worden bewaard.

8. Vereenvoudigde betalingsverificatie

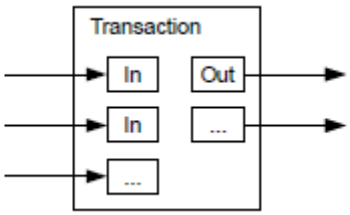
Het is mogelijk om betaling te verifiëren zonder een volledig netwerkknooppunt te draaien. Een gebruiker dient enkel een kopie van de blok headers van de langste proof-of-work keten te bewaren, die hij kan bekomen door ze op te vragen bij netwerkknooppunten tot hij ervan overtuigd is dat hij de langste keten heeft, en de Merkle tak kan bekomen die de transactie verbindt aan het blok waarin het een tijdstempel heeft. Hij kan de transactie zelf niet controleren, maar door het te linken aan een plaats in de ketting, kan hij zien dat het knooppunt het aanvaard heeft, en blokken erna bevestigen dat het netwerk de transactie aanvaard heeft.



Zodoende, is de verificatie betrouwbaar zo lang de eerlijke knooppunten controle hebben over het netwerk, maar is het meer kwetsbaar als het netwerk overmeesterd is door een aanvaller. Terwijl netwerkknooppunten op zichzelf transacties kunnen bevestigen, kan de versimpelde methode om de tuin geleid worden door een aanvaller zijn vervalste transacties zo lang de aanvaller het netwerk erin blijft slagen het netwerk te overmeesteren. Een strategie om het netwerk te beschermen zou kunnen zijn om alarmsignalen van knooppunten te aanvaarden wanneer deze een ongeldig blok detecteren, en de gebruiker zijn software er toe aanzetten om het volledige blok te downloaden en de gesignaleerde transacties om de inconsistentie te bevestigen. Bedrijven die regelmatig betalingen ontvangen willen waarschijnlijk hun eigen knooppunten draaien voor een meer onafhankelijke veiligheid en snellere verificatie.

9. Combineren en Opsplitsen van Waarde

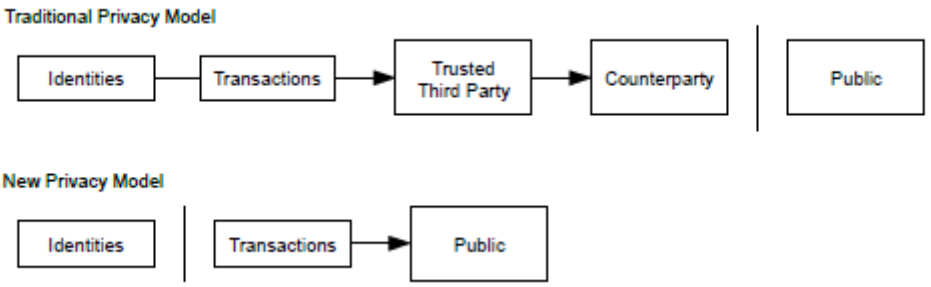
Hoewel het mogelijk zou zijn de munten individueel te behandelen, zou het onhandelbaar worden om een aparte transactie te maken voor elke cent in een overdracht. Om toe te staan dat waarde opgedeeld en gecombineerd wordt, bevatten transacties meerdere inputs en outputs. Normaal gezien al er ofwel een enkele input van een grotere eerdere transactie zijn, of meerdere inputs die kleinere bedragen combineren, en ten hoogste twee outputs: een voor de betaling, en een om wisselgeld te geven, wanneer toepasselijk, terug naar de verzender.



Het moet vermeld worden dat bij een fan-out, waarbij een transactie afhangt van verscheidene transacties, en deze transacties nog eens afhangen van nog vele anderen, er hier geen probleem is. Er is nooit een nood om een compleet op zichzelf staande kopie van een transactie zijn geschiedenis op te vragen.

10. Privacy

Het traditionele bankmodel behaalt een zekere graad van privacy door toegang tot de informatie van betrokken partijen en de vertrouwde derde partij te beperken. De noodzaak om alle transacties publiekelijk aan te kondigen, sluit deze methode uit, maar privacy kan nog steeds behouden worden door de informatiestroom op een andere plaats te breken: door publieke sleutels anoniem te houden. Het publiek kan zien wanneer iemand een bedrag naar iemand ander stuurt, maar zonder dat deze informatie aan een persoon gelinkt wordt. Dit is gelijkaardig aan het niveau van informatie op aandelenbeurzen, waar de tijd en omvang van individuele trades, de "tape", publiek gemaakt wordt, maar zonder iemand te vertellen wie de handelende partijen waren.



Als een bijkomende firewall, zou een nieuw paar sleutels gebruikt moeten worden voor elke transactie die niet aan een gemeenschappelijke eigenaar gelinkt wil worden. Een deel van dat linken is nog steeds onvermijdelijk bij transacties met meerdere inputs, omdat het bij deze noodzakelijk is te onthullen dan de inputs eigendom waren van dezelfde eigenaar. Het risico bestaat dat wanneer de eigenaar van een sleutel onthuld wordt, er via linken onthuld zou kunnen worden welke andere transacties aan diezelfde eigenaar toebehoren.

11.Berekeningen

We houden rekening met een scenario waarin een aanvaller een alternatieve keten sneller probeert te genereren dan de eerlijke keten. Zelfs wanneer dit verwezenlijkt wordt, stelt dit het systeem niet bloot aan willekeurige veranderingen, zoals het scheppen van waarde uit het ijle of om geld te pakken die nooit eigendom was van de aanvaller. Knooppunten zullen geen ongeldige transacties aanvaarden als betaling en eerlijke knooppunten zullen nooit een blok aanvaarden met deze erin. Een aanvaller kan enkel proberen om een van zijn eigen transacties te veranderen of geld terug te nemen die hij reeds heeft uitgegeven.

De race tussen een eerlijke keten en een aanvaller zijn keten kan gekenmerkt worden door een Binomiale Random Walk (willekeurige wandel). In het geval van succes wordt de keten verlengd met een blok, waardoor zijn voorsprong vergroot met +1, en in het geval van mislukking wordt de aanvaller zijn keten verlengd met een blok, waardoor de kloof verkleind wordt met -1. De waarschijnlijkheid dat een aanvaller kan inhalen vanaf een gegeven tekort is analoog aan het *Gambler's Ruin* probleem (ondergang van de gokker). Veronderstel dat een gokker met oneindig krediet begint met een achterstand en een oneindig aantal beurten kan spelen in een poging om break-even te draaien. We kunnen de kans berekenen dat hij ooit break-even draait, of dat een aanvaller ooit de eerlijke keten inhaalt, als volgt [8]:

p= kans dat een eerlijk knooppunt het volgende blok vindt

q= kans dat de aanvaller het volgende blok vindt

qz= kans dat de aanvaller ooit kan inhalen vanaf z blokken achterstand

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

Onder onze veronderstelling dat $p > q$, verkleint de waarschijnlijkheid exponentieel naarmate het aantal blokken dat de aanvaller moet inhalen, vergroot. Met de kansen tegen hem gekeerd, als hij er niet in slaagt om vroeg in het proces met veel geluk vooruit te springen, worden zijn kansen astronomisch klein naarmate hij verder achter geraakt.
We houden nu rekening met hoe lang de ontvanger van een nieuwe transactie moet wachten alvorens hij voldoende zeker kan zijn dat de verzender de transactie niet meer kan veranderen. We veronderstellen dat de verzender een aanvaller is die de ontvanger eventjes wil doen geloven dat hij hem betaald heeft, om vervolgens zichzelf na een tijdje terug te betalen. De ontvanger zal op de hoogte gebracht worden wanneer dat gebeurt, maar de verzender hoopt dat dit te laat zal zijn.

De ontvanger genereert een nieuw sleutelpaar en geeft de publieke sleutel aan de verzender kort voor het tekenen. Dit verhindert dat de verzender een blokketen die voorloopt voor te bereiden door daar voortdurend aan te werken tot hij genoeg geluk heeft om ver genoeg voorop te geraken, en dan de transactie op dat moment uit te voeren. Eens de transactie verzonden is, begint de oneerlijke verzender in het geheim te werken aan een parallelle keten die een alternatieve versie van zijn transactie bevat. De ontvanger wacht tot de transactie toegevoegd is aan een blok en z blokken erna eraan gelinkt zijn. Hij kan de exacte voorsprong van een aanvaller niet weten, maar veronderstellende dat de eerlijke blokken het gemiddelde aantal tijd per blok hebben genomen, kan de aanvaller zijn mogelijke voorsprong uitgedrukt worden in een Poisson verdeling met de verwachte waarde:

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

Om de waarschijnlijkheid dat de aanvaller nog steeds kan inhalen te berekenen, vermenigvuldigen we Poisson densiteit met elke verwachte hoeveelheid voorsprong hij zou kunnen gemaakt hebben met de waarschijnlijkheid dat hij vanaf dat punt zou kunnen inhalen:

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

Herschikt om de oneindige staart van de distributie te voorkomen…

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

Geconverteerd 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;
}
```

Wanneer we enkele resultaten bekijken, zien we de waarschijnlijkheid exponentieel dalen 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
```

Oplossen 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 zonder beroep te moeten doen op vertrouwen. We zijn gestart met het gewoonlijke raamwerk van munten gemaakt van digitale handtekeningen, dat een sterke controle over eigenaarschap schenkt, maar dat is incompleet zonder een manier om dubbele uitgaves te voorkomen. Om dit op te lossen, stellen we een peer-to-peer netwerk voor die gebruik maakt van proof-of-work om een publieke geschiedenis van transacties te loggen dat snel rekenkundig onpraktisch wordt voor een aanvaller om te veranderen zo lang dat eerlijke knooppunten de meerderheid van de CPU kracht controleren. Het netwerk is robuust in zijn ongestructureerde eenvoud. Knooppunten werken allemaal tegelijk met weinig coördinatie. Ze moeten niet geïdentificeerd worden, sinds boodschappen niet verstuurd worden langs een specifieke plaats en enkel ontvangen moeten worden op een best-effort basis. Knooppunten kunnen weg gaan en het netwerk en zich terug bij het netwerk voegen wanneer zij maar willen, zo lang dat ze de proof-of-work keten als bewijs van wat er gebeurt is tijdens hun afwezigheid aanvaarden. Ze stemmen via hun CPU kracht, drukken hun aanvaarding van geldige blokken uit door te werken aan de verlening ervan en verwerpen ongeldige blokken door te weigeren aan deze te werken. Enige benodigde regels en stimulansen kunnen gehandhaafd worden via dit overeenstemmingsmechanisme.

Bronvermelding

[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.