Naar het Nederlands vertaald vanuit https://www.bitcoin.com/bitcoin.pdf
Door Testeriknl

Bitcoin: Een Peer-to-Peer Elektronisch Cash Systeem

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

Samenvatting. Een zuivere peer-to-peer versie van elektronisch geld zou het mogelijk maken om zonder tussenkomst van financiële instellingen online-betalingen rechtstreeks van de ene naar de andere partij te verrichten. Digitale handtekeningen bieden een deel van de oplossing, maar de belangrijkste voordelen gaan verloren als er nog steeds een vertrouwde derde partij nodig is om dubbele besteding te voorkomen. Wij bieden een oplossing voor het probleem van de dubbele uitgaven met behulp van een peer-to-peer-netwerk. Het netwerk tijdstempelt transacties door deze aan de groeiende keten gebaseerd op hash-proof-of-work te schakelen, een grootboek vormend dat niet kan worden gewijzigd zonder het proof-of-work opnieuw uit te voeren. De langste keten dient niet alleen als bewijs voor de volgorde van de waargenomen transacties plaatsvonden, maar ook dat deze afkomstig was van de pool met de meeste CPU-rekenkracht. Zolang de meerderheid aan CPU-rekenkracht wordt bestuurd door knooppunten die niet samenwerken om het netwerk aan te vallen, genereert zij de langste keten en blijft zij aanvallers altijd een stap voor. Het netwerk vereist een minimale structuur. Berichten worden uitgezonden op basis van de ‘best effort’ en knooppunten kunnen naar believen het netwerk verlaten en betreden, waarbij zij de keten met het langste proof-of-work accepteren als bewijs van wat er gebeurde nadat zij het netwerk verlieten.

1. Inleiding

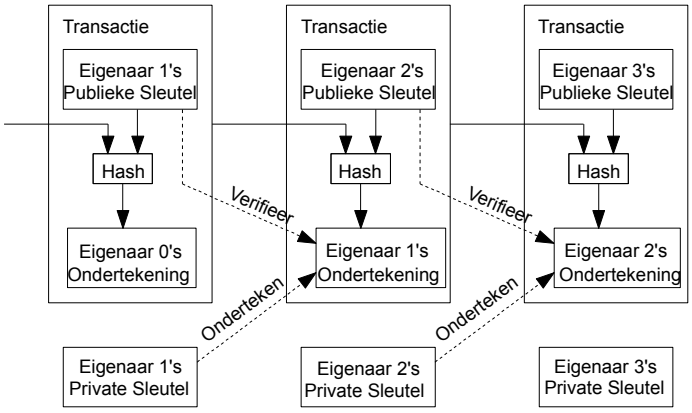
Handel op internet is bijna volledig afhankelijk geworden van financiële instellingen die als vertrouwde derde partijen optreden om elektronische betalingen te verwerken. Hoewel het systeem goed genoeg werkt voor de meeste transacties, heeft het nog altijd te lijden onder de inherente zwakheden van het op vertrouwen gebaseerde model. Volledig onherroepelijke transacties zijn praktisch onmogelijk, omdat financiële instellingen bemiddeling bij geschillen niet kunnen vermijden. Bemiddelingskosten verhogen de transactiekosten, stellen daarmee eisen aan het minimum praktische bestedingsbedrag, waarmee de toepassing voor kleine dagelijkse transacties onbruikbaar is. En er gaan kosten in algemene zin mee gemoeid door het onvermogen onherroepelijke betalingen voor onherroepelijke diensten te verrichten. Met de mogelijkheid van herroepelijke transacties, groeit de behoefte aan vertrouwen. Ondernemers moeten op hun hoede zijn en zijn klanten lastig vallen door meer informatie te vragen dan men in normaal zou doen. Een bepaald percentage van fraude wordt als onvermijdelijk geaccepteerd. Deze kosten en betalingsonzekerheden kunnen worden vermeden door onderling fysieke valuta te gebruiken, maar er bestaat geen mechanisme om betalingen te verrichten via een communicatiekanaal zonder vertrouwde partij.

Wat ontbreekt, is een elektronisch betalingssysteem gebaseerd op cryptografisch bewijs in plaats van vertrouwen, waardoor twee bereidwillige partijen rechtstreeks met elkaar transacties kunnen verrichten zonder de noodzaak van een vertrouwde derde partij. Transacties die rekenkundig onmogelijk zijn om terug te draaien zouden verkopers beschermen tegen fraude,

routinematige escrow-mechanismen kunnen eenvoudig worden ingebouwd om kopers te beschermen. In dit paper stellen wij een oplossing voor het probleem van de dubbele uitgaven voor, middels een peer-to-peer gedistribueerde tijdstempelserver die het rekenkundig bewijs van de chronologische van transactievolgorde genereert. Het systeem is beveiligd zolang de eerlijke knooppunten gezamenlijk over meer CPU-rekenkracht beschikken dan de samenwerkende aanvallende groep knooppunten.

2. Transacties

Wij definiëren een elektronische munt als een schakel van digitale handtekeningen. Iedere eigenaar verzendt de munt naar de volgende door digitale ondertekening van de vorige hash en de publieke sleutel van de volgende eigenaar die aan het einde van de munt wordt toegevoegd. Aan de hand van de handtekeningen kan de begunstigde de huidige eigenaar en vorige eigenaren controleren.



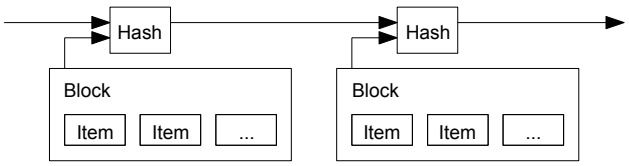
Het probleem is natuurlijk dat de begunstigde niet kan verifiëren dat een van de eigenaren dezelfde munt niet twee keer heeft uitgegeven. Een veel voorkomende oplossing is het introduceren van een vertrouwde centrale autoriteit, of munt, die elke transactie controleert op dubbel uitgeven. Na iedere transactie moet de munt naar de centrale autoriteit worden teruggebracht om een nieuwe munt uit te geven, en enkel de munten die rechtstreeks door de centrale autoriteit zijn uitgegeven, worden geacht niet als dubbel te zijn uitgegeven. Het probleem van deze oplossing is dat het lot van het hele geldsysteem afhangt van de centrale autoriteit die de munt beheert, waarbij iedere transactie langs die autoriteit moet, als ware het een bank.

Het probleem is natuurlijk dat de begunstigde niet kan verifiëren dat een van de eigenaren dezelfde munt niet twee keer heeft uitgegeven. Een veel voorkomende oplossing is het introduceren van een vertrouwde centrale autoriteit, of munt, die elke transactie controleert op dubbel uitgeven. Na iedere transactie moet de munt naar de centrale autoriteit worden teruggebracht om een nieuwe munt uit te geven, en enkel de munten die rechtstreeks door de centrale autoriteit zijn uitgegeven, worden geacht niet als dubbel te zijn uitgegeven. Het probleem van deze oplossing is dat het lot van het hele geldsysteem afhangt van de centrale autoriteit die de munt beheert, waarbij iedere transactie langs die autoriteit moet, als ware het een bank.

3. Timestamp Server

De oplossing die wij voorstellen begint met een tijdstempelserver. Een tijdstempelserver werkt door de hash te nemen van een blok met items die van een tijdstempel moeten worden voorzien en deze hash op grote schaal te publiceren, zoals in een krant of Usenet-bericht [2-5]. Het

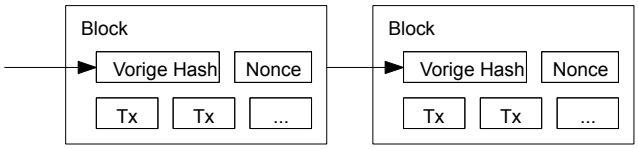
tijdstempel bewijst dat de gegevens op dat moment bestonden, hoogstwaarschijnlijk, om in de hash opgenomen te worden. Iedere tijdstempel bevat de vorige tijdstempel in zijn hash, waardoor een keten wordt gevormd, waarbij elke toegevoegde tijdstempel de tijdstempels uit het verleden herbevestigd.



4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

Om ons timestamp-netwerk implementeren we de proof-of-work door een nonce in het blok te verhogen totdat de waarde wordt gevonden die de hash van het blok de vereiste nul bits geeft. Als de CPU-inspanning eenmaal is verricht om het te laten voldoen aan de proof-of-work, kan het blok niet worden gewijzigd zonder de berekening opnieuw uit te voeren. Aangezien latere blokken daarachter worden geketend, omvat het werk om het desbetreffende 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 zou worden gebaseerd op één-IP-adres-één stem, dan zou deze kunnen worden ondermijnd door iemand die veel IP-adressen tot zijn beschikking heeft. Proof-of-work is in wezen één-CPU-één-stem. De meerderheidsbeslissing wordt vertegenwoordigd door de langste keten waarin het meeste proof-of-work is geïnvesteerd. Als de meerderheid aan CPU-rekenkracht wordt gecontroleerd door eerlijke knooppunten, dan zal deze eerlijke keten het snelst groeien en de concurrerende ketens overtreffen. Om een blok uit het verleden te wijzigen, moet een aanvaller het bewijs van werk van het blok en alle blokken erna opnieuw uitvoeren en vervolgens het werk van de eerlijke knooppunten inhalen en overtreffen. We zullen later aantonen dat de kans dat een langzame aanvaller de eerlijke knooppunten inhaalt, exponentieel afneemt zodra volgende blokken worden toegevoegd.

Ter compensatie voor toenemende hardwaresnelheid en wisselende interesse in het runnen van knooppunten in verloop van de tijd, wordt de moeilijkheidsgraad voor het proof-of-work bepaald door een voortschrijdend gemiddelde dat gericht is op het gemiddeld aantal blokken per uur dat wordt gevonden. Als blokken te snel worden gegenereerd, dan neemt de moeilijkheidsgraad toe.

5. Netwerk

De stappen om het netwerk te laten opereren 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 het proof-of-work voor zijn blok.
4) Wanneer een knooppunt het proof-of-work vindt, verzendt het knooppunt het blok naar alle knooppunten.
5) Knooppunten accepteren het blok alleen als alle transacties daarin geldig zijn en nog niet zijn uitgegeven.
6) Knooppunten drukken hun acceptatie van het blok uit door te werken aan de creatie van het volgende blok in de keten, door de hash van het geaccepteerde blok te gebruiken als de vorige hash.

Knooppunten beschouwen de langste keten altijd als de enige waarheid en blijven werken aan de uitbreiding daarvan. Als twee knooppunten tegelijkertijd verschillende versies van het volgende blok uitzenden, kunnen andere knooppunten verschillen in volgorde van ontvangst. In dat geval werken de knooppunten aan het blok dat zij als eerste hebben ontvangen, maar bewaren zij de andere tak voor het geval die langer wordt. De schakel wordt gebroken wanneer het volgende proof-of-work is gevonden en één van de afsplitsingen langer is geworden; de knooppunten die op de kortere keten werkten verplaatsen zich vervolgens naar de langste keten.

Nieuwe transactie-uitzendingen hoeven niet noodzakelijkerwijs alle knooppunten te bereiken. Zolang zij meerdere knooppunten bereiken, worden transacties al snel in een blok opgenomen. Blokuitzendingen zijn ook tolerant ten aanzien van vervallen berichten. Als een knooppunt een blok mist, dan vraagt het knooppunt dit blok aan bij ontvangst van het volgende blok en neemt het knooppunt het gemiste blok alsnog op in de keten.

6. Incentives

Volgens afspraak is de eerste transactie in een blok een speciale transactie waarbij een nieuwe munt wordt gecreëerd die eigendom is van de maker van het blok. Dit geeft knooppunten een positieve stimulans 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 de wijze waarop goudzoekers middelen besteden om goud in de circulatie te brengen. In ons geval zijn de middelen de verbruikte CPU-tijd en elektriciteit.

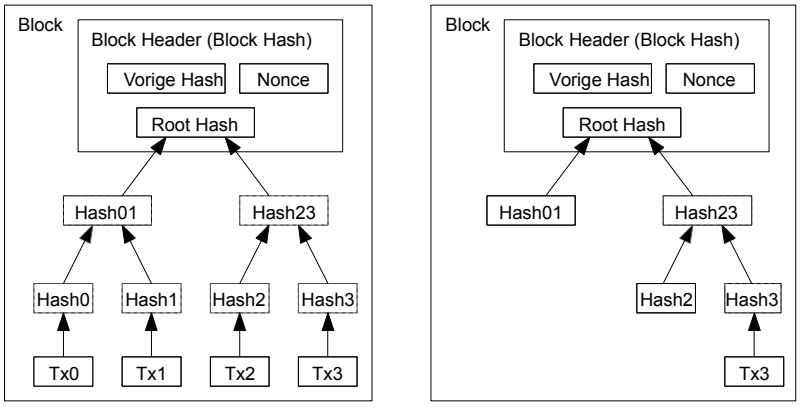
De positieve stimulans kan ook worden bekostigd door transactie-fooien. [Verzenders van een transactie kunnen een bedrag als fooi bestemd voor knooppunten aan hun transactie toevoegen.] Als de uitvoerwaarde van een transactie kleiner is dan de invoerwaarde, dan is het verschil de transactie-fooi [bedoeld voor de knooppunten]. Deze dient als stimulans om in het eerstvolgende blok met transacties te worden opgenomen. Zodra een vooraf bepaald aantal munten in omloop is gekomen, kan dienen transactiekosten als enige stimulans en is het netwerk volledig inflatie-vrij.

De stimulansen kunnen knooppunten ertoe bewegen om eerlijk te blijven. Als een hebzuchtige aanvaller in staat is gebleken om meer CPU-rekenkracht van het netwerk te verwerven dan alle eerlijke knooppunten bij elkaar, dan zou hij moeten kiezen tussen gebruik van de rekenkracht voor het oplichten van mensen door zijn uitgaven ongedaan te maken, of ter gebruik voor het genereren van nieuwe munten. Het zou voor hem winstgevender moeten zijn om zich aan de regels te houden – regels die zodanig in zijn voordeel werken dat hij meer munten verdient dan de rest van de knooppunten bij elkaar, dan te proberen het systeem en de validiteit van zijn eigen verdiensten te ondermijnen.

7. Vrijmaken Opslagruimte

Als eenmaal de laatste transactie van een munt onder genoeg andere blokken is verstopt, kunnen de vorige transacties van de munt uit de keten worden verwijderd om ruimte te besparen. Om dit mogelijk te maken zonder de hash van het blok te wijzigen, worden transacties gehasht in de

vorm van een Merkle Tree, [7] [2] [5], waarbij enkel de root is opgenomen in de hash van het blok. Oude blokken kunnen vervolgens worden gecomprimeerd gelijk het het snoeien van de takken van een boom. De hashes uit de gecomprimeerde blokken hoeven niet te worden opgeslagen.

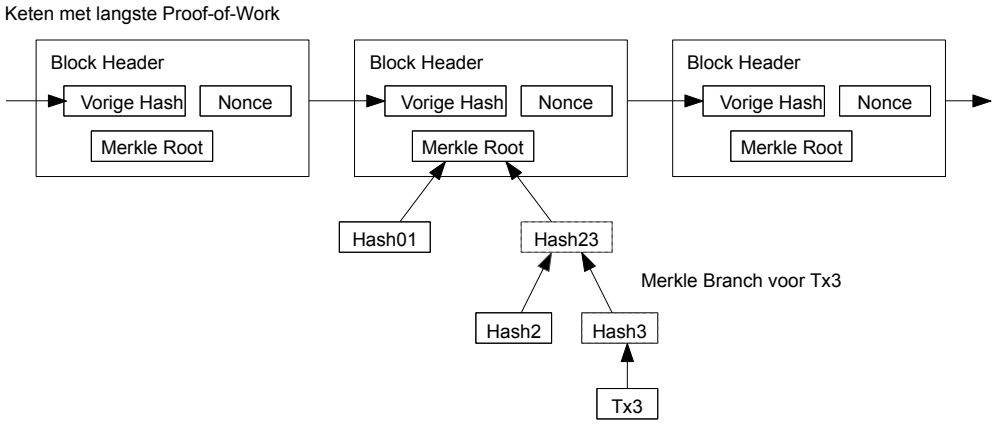


Ge-hashte transacties in een Merkle Tree Na snoeien van Tx0-2 in het Block

Een blok header zonder transacties zou ongeveer 80 bytes zijn. Als we aannemen dat blokken iedere 10 minuten gegenereerd worden, 80 bytes * 6 * 24 * 365 = 4,2 MB per jaar. Aangezien computersystemen verkocht worden met 2GB aan RAM in het jaar 2008, en Moore's Law een groei van 1,2GB per jaar voorspelt, zou opslag niet het probleem zijn zelfs in het geval de blok headers wel geheugenopgeslag vereisen.

8. Vereenvoudigde Verificatie van Betaling

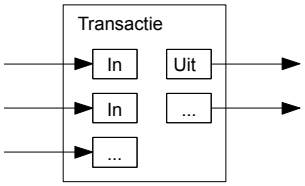
Het is mogelijk betalingen te verifiëren zonder beschikking te hebben over de volledige keten. Een gebruiker hoeft slechts te beschikken over een kopie van de blok headers van de langste keten van proof-of-work. Die kan hij verkrijgen door de gegevens bij de knooppunten op te vragen tot hij ervan overtuigd is dat hij de langste keten heeft gevonden, zich de Merkle Tree-tak verschaft die de transactie naar het blok leidt waarin het is getimetampt. Hij kan de transactie niet voor zichzelf controleren, maar door de transactie in de keten te plaatsen kan hij zien of een knooppunt de transactie heeft geaccepteerd, en de toevoeging van latere blokken bovenop de transactie bevestigen dat het netwerk de betaling heeft geaccepteerd.



Zoals beschreven, de verificatie is betrouwbaar zolang de eerlijke nodes het netwerk controleren, maar is kwetsbaarder als het netwerk door een aanvaller wordt overschaduwd. Daar waar knooppunten de transacties zelfstandig kunnen verifiëren, kan de versimpelde methode voor de gek worden gehouden door gekunstelde transacties van een aanvaller voor zolang de aanvaller de het netwerk overschaduwd. Een strategie gebruikers tegen gekunstelde transacties te beschermen zou het accepteren van waarschuwingsmeldingen vanuit de netwerk knooppunten zijn, die worden gegenereerd bij de dectectie van een ongeldig blok. Daarop wordt de software van de gebruiker aangezet tot het downloaden van het volledig blok met de inconsistente transacties die de waarschuwing veroorzaakten. Ondernemingen die frequente betalingen ontvangen willen waarschijnlijk hun eigen knooppunt draaien voor een meer zelfstandiger beveiliging en snellere verificatie.

9. Combining and Splitting Value

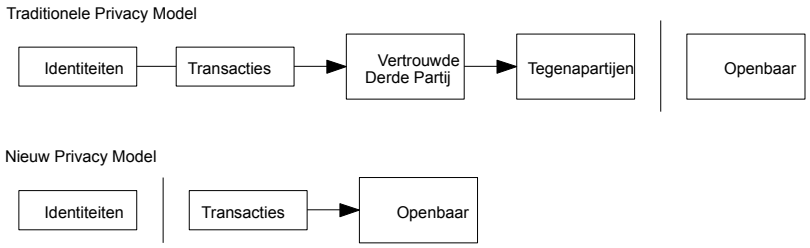
Hoewel het mogelijk is coins apart te volgen, zou het omslachtig worden te maken een afzonderlijke transactie voor iedere cent betrokken in een vermogensoverdracht. Om het mogelijk te maken om waarde te delen en samen te voegen, bevatten transacties meerdere inputs en outputs. Normaal gesproken is er, of een enkele input afkomstig uit een vorige grotere transactie, of meerdere inputs samengesteld uit kleinere bedragen, en op zijn hoogst twee outputs: de ene output geldt als betaling en de andere output geldt, indien aanwezig, als wisselgeld bestemt voor de verzender.



Er moet opgemerkt worden dat het aantal inputs om een specifieke output te voeden – een transactie afhankelijk is van meerdere transacties, en die op hun beurt afhankelijk zijn van vele andere, niet het probleem is. De noodzaak is nooit aanwezig om een volledige standalone kopie van de transactiegeschiedenis te verkrijgen.

10. Privacy

Het traditionele bankmodel bewerkstelligt een zekere mate van privacy door beperkte toegang tot informatie te verlenen aan betrokken partijen en vertrouwelijke derde partij. De vereiste alle transacties publiekelijk te maken is onverenigbaar met deze methode, evenwel kan privacy nog steeds geborgd worden door de informatiestroom op een andere plaats te onderbreken: Door de publieke sleutels te anonimiseren. Men kan zien dat iemand een betaling aan iemand anders overmaakt, zonder dat deze transactie informatie naar iemand herleidbaar is. Dit is net als de informatie die wordt vrijgegeven op aandelenmarkten, waar tijd en omvang van de individuele transacties, de “tape’, openbaar wordt gemaakt, zonder daarbij de namen van de betrokkenen te vermelden.



Als toegevoegde bescherming, moet voor iedere transactie een nieuw sleutelpaar gebruikt worden zodat de ontvanger niet te herleiden is. Herleiding is tot op zekere hoogte onvermijdelijk bij multi-input transacties, die noodzakerlijkerwijs openbaren dat hun inputs tot een en dezelfde eigenaar behoorden. Het risico is het geval waarin de eigenaar van een sleutel bekend is geraakt, hij aan andere transacties verbonden kan worden.

11. Berekeningen

We nemen het scenario waarin een aanvaller sneller dan eerlijke keten een plaatsvervangende keten probeert te genereren. Zelfs als hij hierin slaagt, wordt het systeem niet volledig blootgesteld aan willekeurige wijzigingen, als het creëren van munten uit ‘thin air’, of geld onttrekken dat niet aan de aanvaller toebehoorde. Knooppunten zullen ongeldige transacties als betaling weigeren en eerlijke knooppunten zullen blokken die ongeldige transacties bevatten nooit accepteren. Een aanvaller kan enkel proberen een van zijn eigen transacties te wijzigen en zijn recent uitgegeven geld terugpakken.

De wedloop tussen de eerlijke keten en die van de aanvaller kan gekarakteriseerd worden als een Binominal Random Walk. Een succes event wordt geteld als de eerlijke keten met een blok wordt uitgebreid, waarmee zijn voorsprong met +1 toeneemt, een failure wordt geteld als een blok aan de keten van de aanvaller wordt toegevoegd, waardoor de voorsprong met -1 afneemt.

De mogelijkheid dat een aanvaller vanuit een achterstandspositie op gelijke hoogte komt is analoog aan een Gambler’s Ruin probleem. Stel je een gokker voor met onbeperkte creditcard op achterstand, die tot in het oneindige zal proberen zijn verlies in te lopen. We kunnen de kans of hij ooit zijn achterstand kan wegwerken berekenen, of op gelijke hoogte kan komen met de eerlijke keten is als volgt [8]:

p = kans dat een eerlijk knooppunt het volgende blok vindt
q = kans dat een aanvaller het volgende blok vindt
q_z= kans dat de aanvaller ooit een achterstand van z blokken kan inlopen

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

Vanuit onze assumptie dat p > q, neemt de kans van slagen van de aanval exponentioneel af naarmate het aantal blokken dat de aanvaller moet inhalen toeneemt. Met de verwachtingen tegen hem, als hij vanaf het begin door geluk geen voorsprong opbouwt, nemen zijn kansen af naarmate hij mij achterop geraakt.

We bekijken nu hoe lang een ontvanger van een nieuwe transactie moeten wachten op voldoende bevestiging dat de verzender zijn transactie niet kan wijzigen. Daarbij nemen we aan dat de verzender de aanvaller is, die doet voorkomen alsof hij een tijdje geleden al heeft betaald en de betaling ongedaan maakt door het bedrag binnen enige tijd aan zichzelf te betalen. Als dat geschiedt wordt de ontvanger hiervan op de hoogte gesteld, maar de aanvaller hoopt dat dit te laat

zal gebeuren.

De ontvanger genereert een nieuwe sleutelset en geeft de publieke sleutel aan de verzender kort voor ondertekening. Dit belemmert de mogelijkheid voor de verzender net zo lang aan een keten van blokken vooruit in de tijd voor te bereiden tot hij het geluk heeft om een voorsprong op te bouwen die groot genoeg is en uiteindelijk de transactie uitvoert. Als de transactie eenmaal is verzonden, begint de oneerlijke verzender heimelijk aan een parallelle keten te werken met daarin de gewijzigde transactie van zijn transactie.

De ontvanger wacht totdat de transactie in een blok is gezet en daarop een x aantal blokken is toegevoegd. De ontvanger weet niet welke voortgang de aanvaller precies heeft gemaakt, maar uitgaande dat de verwerking van eerlijke blokken per blok de gemiddelde tijd in beslag neemt, dan is de mogelijke voortgang van de aanvaller een Poisson-verdeling met een verwachte waarde van:

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

Om de kans te berekenen dat de aanvaller nog steeds kan aanhaken, vermenigvuldigen we de Poisson dichtheid voor elke mate van voortgang die hij gemaakt zou kunnen hebben met de kans die hij heeft om vanuit de bewuste achterstand opgelijke hoogte te komen:

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

Herschikt ter vermijding van sommering van de oneindige staart van de kansverdeling…

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

Naar C code vertaald…

```
#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 enige resultaten berekenen, zien we de kans exponentieel verkleinen bij toename van *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
```

Berekend voor P is kleiner 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 elektronisch betalingsverkeer gecreëerd waarbij geen onderling vertrouwen nodig is. We begonnen met het gebruikelijke framework waar munten worden gecreëerd met behulp van digitale handtekeningen die de gebruiker volledig eigendom verschaft, maar dat niet compleet is zonder de bescherming tegen dubbele bestedingen. Om dit probleem op te lossen stelden wij het gebruik van een peer-to-peer netwerk voor dat door middel van proof-of-work de publieke historie van transacties bijhoudt. Als eerlijke knooppunten over de meerderheid aan CPU-rekenkracht beschikken is de historie binnen afzienbare tijd niet meer door een aanvaller rekenkundig te wijzigen. Het netwerk is robuust door zijn ongestructureerde eenvoud. Knooppunten werken meteen samen waarbij weinig coördinatie wordt gevraagd. Zij

hoeven niet te worden geïdentificeerd aangezien berichten niet naar een bepaalde plaats worden gestuurd, en berichten slechts op basis van ‘best effort’ worden verzonden. Knooppunten kunnen het netwerk betreden en verlaten wanneer zij willen, waarbij zij het proof-of-work accepteren dat is verricht in de tijd dat zij niet op het netwerk waren. Knooppunten stemmen met hun CPU-rekenkracht, waarbij zij geldige blokken accepteren door hier op verder te bouwen en ongeldige blokken afwijzen door daar niet op verder te bouwen. Alle benodigde regels en stimuli kunnen door dit consensus mechanisme worden afgedwongen.

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.