Bitcoin: Un sistema de diner electrònic d'igual a igual

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

Traduït al català de bitcoin.org/bitcoin.pdf
per Vicent Sus

Resum. Una versió purament d'igual a igual de diner electrònic permetria enviar pagaments en línia directament entre dues parts sense passar per una institució financera. Les signatures digitals proporcionen part de la solució, però els principals avantatges es perden si encara es requereix un tercer de confiança per evitar la doble despesa. Proposem una solució al problema de la doble despesa mitjançant una xarxa d'igual a igual. La xarxa segella en el temps les transaccions generant un hash d'aquestes i afegint-lo en una cadena contínua de prova de treball basada en hash, formant un registre que no es pot modificar sense refer la prova de treball. La cadena més llarga no només serveix com a prova de la seqüència d'esdeveniments presenciats, sinó com a prova que aquesta provenia del conjunt més gran de potència de CPU. Mentre la majoria de la potència de CPU estigui controlada per nodes que no cooperin per atacar la xarxa, aquests generaran la cadena més llarga i deixaran enrere els atacants. La xarxa en si mateixa requereix una estructura mínima. Els missatges es transmeten en funció del millor esforç, i els nodes poden sortir i tornar-se a unir a la xarxa a voluntat, acceptant la cadena de prova de treball més llarga com a prova del que ha passat mentre no hi eren.

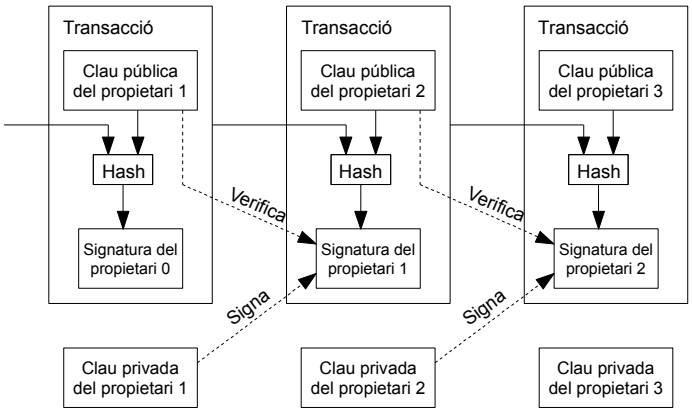
1. Introducció

El comerç a Internet s'ha basat gairebé exclusivament en institucions financeres que actuen com a tercers de confiança per processar pagaments electrònics. Tot i que el sistema funciona prou bé per a la majoria de transaccions, encara pateix les debilitats inherents del model basat en la confiança. Les transaccions completament irreversibles no són realment possibles, ja que les institucions financeres no poden evitar mediar conflictes. El cost de la mediació augmenta els costos de transacció, limitant la quantitat pràctica mínima de les transaccions i eliminant la possibilitat de petites transaccions ocasionals. També hi ha un elevat cost en la pèrdua de la capacitat de fer pagaments irreversibles per serveis irreversibles. Amb la possibilitat de la reversió, s'estén la necessitat de confiança. Els comerciants han de ser cautelosos amb els seus clients, importunant-los amb més consultes de les necessàries per obtenir informació. Un cert percentatge de frau s'accepta com a inevitable. Aquests costos i incerteses de pagament es poden evitar en persona mitjançant l'ús de diner físic, però no existeix cap mecanisme per fer pagaments a través d'un canal de comunicacions sense un tercer de confiança.

El que es necessita és un sistema de pagament electrònic basat en proves criptogràfiques en lloc de confiança, que permeti realitzar transaccions directament entre dues parts sense la necessitat d'un tercer de confiança. Les transaccions que són computacionalment poc pràctiques de revertir, protegirien els venedors del frau i fàcilment es podrien implementar mecanismes de dipòsit de garantia rutinari per protegir els compradors. En aquest document, proposem una solució al problema de la doble despesa mitjançant un servidor de segellat de temps distribuït d'igual a igual per generar una prova computacional de l'ordre cronològic de les transaccions. El sistema és segur sempre que els nodes honestos controlin col·lectivament més potència de CPU que qualsevol grup de nodes atacants.

2. Transaccions

Definim una moneda electrònica com una cadena de signatures digitals. Cada propietari transfereix la moneda al següent signant digitalment un hash de la transacció anterior i de la clau pública del següent propietari, i afegint-los al final de la moneda. Un beneficiari pot verificar les signatures per verificar la cadena de propietat.

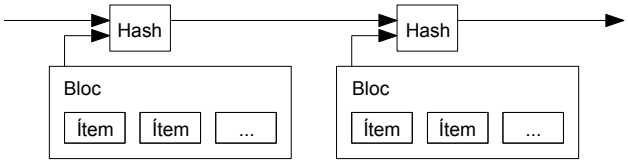


El problema, per descomptat, és que el beneficiari no pot verificar que un dels propietaris no hagi gastat la moneda dues vegades. Una solució habitual és introduir una autoritat central de confiança, o casa de la moneda, que comprova si hi ha una doble despesa en cada transacció. Després de cada transacció, la moneda s'ha de retornar a la casa de la moneda per emetre'n una de nova, i només les monedes emeses directament des de la casa de la moneda estan lliures de sospita de ser gastades dues vegades. El problema d'aquesta solució és que el destí de tot el sistema monetari depèn de la companyia que gestiona la casa de la moneda, on cada transacció ha de passar per ells, igual que un banc.

Necessitem una manera perquè el beneficiari sàpiga que els propietaris anteriors no han signat cap transacció prèvia. Per als nostres propòsits, la transacció realitzada més aviat és la que compta, de manera que no ens preocupen els intents posteriors de duplicar la despesa. L'única manera de confirmar l'absència d'una transacció és conéixer totes les transaccions. En el model basat en la casa de la moneda, aquesta coneixia totes les transaccions i decidia quina havia arribat primer. Per aconseguir això sense un tercer de confiança, les transaccions s'han d'anunciar públicament [1], i necessitem un sistema perquè els participants puguin acordar un historial únic de l'ordre en què les transaccions han estat rebudes. El beneficiari ha de demostrar que en el moment de cada transacció, la majoria de nodes havien acordat que aquella era la primera transacció rebuda.

3. Servidor de segellat de temps

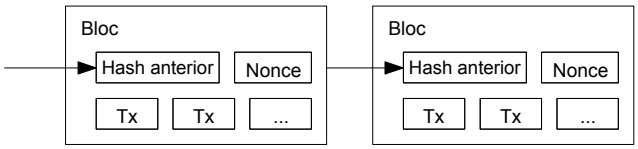
La sol·lució que proposem comença amb un servidor de segellat de temps. Un servidor de segellat de temps funciona agafant el hash d'un bloc d'elements que cal segellar en el temps i publicant àmpliament el hash, com en un diari o en una publicació d'Usenet [2-5]. El segellat de temps demostra que les dades havien d'haver existit en aquell moment per poder entrar al hash. Cada segellat de temps inclou l'anterior segellat de temps en el seu hash, formant una cadena, amb cada segellat de temps addicional reforçant els anteriors a aquest.



4. Prova de treball

Per implementar un servidor de segellat de temps distribuït d'igual a igual, haurem d'utilitzar un sistema de prova de treball similar a Hashcash d'Adam Back [6], en lloc de publicacions a diaris o a Usenet. La prova de treball consisteix en l'escaneig en busca d'un valor que quan se li aplica una funció hash, com per exemple amb SHA-256, el hash comenci amb un nombre de zero bits. El treball mitjà necessari és exponencial en el nombre de zero bits necessaris i es pot verificar executant un sol hash.

Per a la nostra xarxa de segellat de temps, implementem la prova de treball mitjançant l'increment d'un nonce en el bloc fins que es troba un valor que doni els zero bits necessaris al hash del bloc. Un cop esgotat l'esforç de CPU per satisfer la prova de treball, el bloc no es pot modificar sense refer el treball. Com els blocs posteriors són encadenats després d'aquest, el treball per modificar el bloc inclouria refer tots els blocs després d'aquest.



La prova de treball també resol el problema de determinar la representació en la presa de decisions majoritàries. Si la majoria es basés en un vot per cada adreça IP, qualsevol persona que pugui assignar moltes IP podria subvertir-la. La prova de treball és essencialment un vot per cada CPU. La decisió majoritària està representada per la cadena més llarga, ja que és on s'ha invertit el major esforç de prova de treball. Si la majoria de la potència de CPU està controlada per nodes honestos, la cadena honesta creixerà més ràpidament i superarà les cadenes competidores. Per modificar un bloc anterior, un atacant hauria de refer la prova de treball del bloc i de tots els blocs posteriors, i després hauria d'atrapar i superar el treball dels nodes honestos. Més endavant mostrarem que la probabilitat que un atacant més lent atrapi el treball disminueix exponencialment a mesura que s'afegeixen blocs posteriors.

Per compensar l'augment de la velocitat del hardware i l'interès variable en executar nodes al llarg del temps, la dificultat de la prova de treball ve determinada per una mitjana mòbil dirigida a un nombre mitjà de blocs per hora. Si es generen massa ràpid, la dificultat augmenta.

5. Xarxa

Els passos per executar la xarxa són els següents:

1) Les transaccions noves es transmeten a tots els nodes.
2) Cada node recopila noves transaccions en un bloc.
3) Cada node treballa per trobar una prova de treball difícil per al seu bloc.
4) Quan un node troba una prova de treball, transmet el bloc a tots els nodes.
5) Els nodes només accepten el bloc si totes les transaccions en ell són vàlides i no s'han gastat.
6) Els nodes expressen la seva acceptació del bloc treballant en la creació del següent bloc de la cadena, utilitzant el hash del bloc acceptat com a hash anterior.

Els nodes sempre consideren que la cadena més llarga és la correcta i seguiran treballant per ampliar-la. Si dos nodes transmeten simultàniament versions diferents del següent bloc, alguns nodes poden rebre primer una o l'altra. En aquest cas, els nodes treballen amb la primera versió que han rebut, però guarden l'altra ramificació en cas que aquesta es converteixi en la llarga. L'empat es trencarà quan es trobi la següent prova de treball i una ramificació s'allargui; els nodes que treballaven a l'altra ramificació canviaran a la més llarga.

La transmissió de transaccions noves no necessàriament ha d'arribar a tots els nodes. Mentre arribin a bastants nodes, entraran en un bloc al cap de poc temps. Les transmissions de blocs també són tolerants als missatges perduts. Si un node no rep un bloc, el sol·licitarà quan rebi el següent bloc i s'adoni que n'ha perdut un.

6. Incentiu

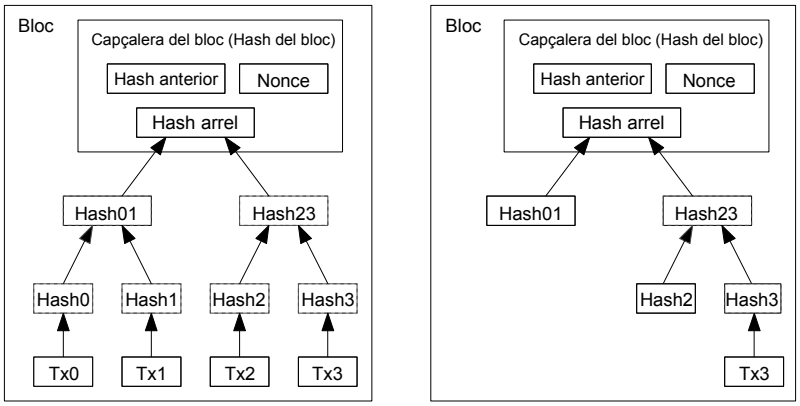
Per convenció, la primera transacció en un bloc és una transacció especial que inicia una nova moneda propietat del creador del bloc. Això afegeix un incentiu als nodes per donar suport a la xarxa i proporciona una manera de distribuir inicialment les monedes en circulació, ja que no hi ha cap autoritat central per emetre-les. L'addició contínua d'una quantitat constant de noves monedes és anàloga als miners d'or que gasten recursos per afegir or a la circulació. En el nostre cas, es consumeix temps de CPU i electricitat.

L'incentiu també es pot finançar amb les comissions de les transaccions. Si el valor de sortida d'una transacció és inferior al seu valor d'entrada, la diferència és una comissió de transacció que s'afegeix al valor de l'incentiu del bloc que conté la transacció. Un cop ha entrat en circulació un nombre predeterminat de monedes, l'incentiu pot passar completament a les comissions de transacció i estar lliure d'inflació.

L'incentiu pot ajudar a que els nodes es mantinguin honestos. Si un atacant avaricios és capaç de reunir més potència de CPU que tots els nodes honestos, hauria de triar entre utilitzar-la per defraudar persones robant els seus pagaments o utilitzar-la per generar noves monedes. L'atacant hauria de trobar més rendible complir amb les regles, unes regles que l'afavoreixen amb més monedes noves que a la resta de persones combinades, que no pas soscavar el sistema i la validesa de la seva pròpia riquesa.

7. Reclamant espai en disc

Un cop l'última transacció d'una moneda està enterrada a sota de suficients blocs, les transaccions gastades abans d'aquesta es poden descartar per estalviar espai en disc. Per facilitar-ho sense trencar el hash del bloc, s'aplica a les transaccions una funció hash amb l'estructura d'un arbre de Merkle [7][2][5], amb només l'arrel inclosa al hash del bloc. Els blocs vells es poden compactar tallant ramificacions de l'arbre. No cal emmagatzemar els hash interiors.



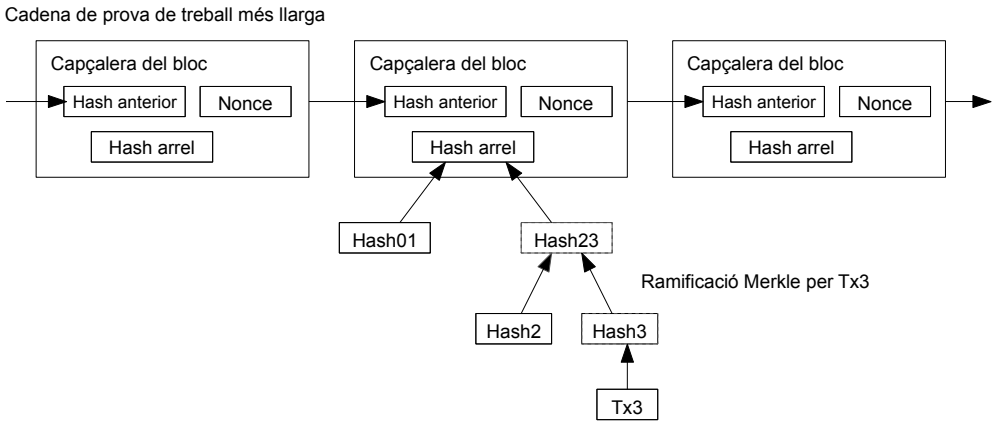
Transaccions després d'aplicar una funció Hash amb l'estructura d'un arbre de Merkle

Després de podar les transaccions 0, 1 i 2 del bloc

La capçalera d'un bloc sense transaccions seria de 80 bytes. Si suposem que els blocs es generen cada 10 minuts, 80 bytes * 6 * 24 * 365 = 4.2MB per any. Amb sistemes informàtics que normalment es venen amb 2GB de RAM a partir de 2008, i la llei de Moore que prediu un creixement de 1.2GB per any, l'emmagatzematge no hauria de ser un problema fins i tot si les capçaleres dels blocs s'han de guardar a la memòria.

8. Verificació de pagament simplificada

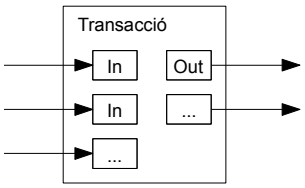
És possible verificar pagaments sense executar un node de xarxa complet. L'usuari només ha de conservar una còpia de les capçaleres dels blocs de la cadena de prova de treball més llarga, que pot obtenir consultant els nodes de la xarxa fins que estigui convençut que té la cadena més llarga, i obtindre la ramificació de Merkle que enllaça la transacció amb el bloc en què està segellat en el temps. L'usuari no pot comprovar la transacció per si mateix, però en vincular-la a un lloc de la cadena pot veure que un node de xarxa l'ha acceptada, i els blocs afegits després de la transacció confirmen aquesta acceptació.



La verificació és fiable sempre que els nodes honestos controlin la xarxa, però és més vulnerable si la xarxa és dominada per un atacant. Tot i que els nodes de la xarxa poden verificar les transaccions per si mateixos, el mètode simplificat pot ser enganyat per transaccions fabricades per un atacant durant el temps que aquest pugui seguir dominant la xarxa. Una de les estratègies per protegir-se d'això, seria acceptar alertes dels nodes de la xarxa quan detectin un bloc invàlid, demanant al software de l'usuari que es baixi el bloc complet i les transaccions alertades per confirmar la inconsistència. Les empreses que reben pagaments freqüents, probablement voldran executar els seus propis nodes per tindre més seguretat independent i una verificació més ràpida.

9. Combinant i dividint valor

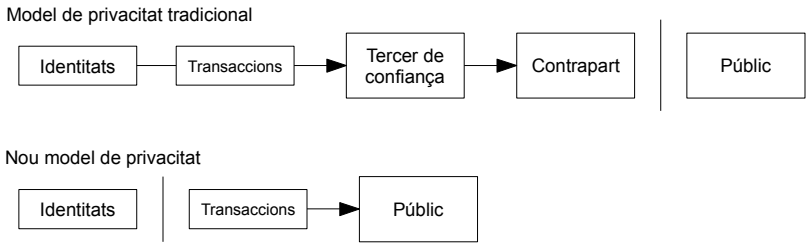
Tot i que seria possible gestionar monedes de manera individual, seria poc manejable fer una transacció independent per cada cèntim d'una transferència. Per permetre dividir i combinar el valor, les transaccions contenen múltiples entrades i sortides. Normalment hi haurà, o bé una única entrada d'una transacció anterior més gran, o múltiples entrades combinant quantitats més petites, i com a màxim dues sortides: una per al pagament i una altra que retornarà el canvi, si n'hi ha, al remitent.



Cal assenyalar que el fan-out, on una transacció depèn de diverses transaccions i aquestes depenen de moltes més, no és un problema. No hi ha mai la necessitat d'extreure una còpia independent i completa de l'historial d'una transacció.

10. Privacitat

El model bancari tradicional aconsegueix un nivell de privadesa limitant l'accés a la informació a les parts implicades i al tercer de confiança. La necessitat d'anunciar públicament totes les transaccions exclou aquest mètode, però la privadesa encara es pot conservar trencant el flux d'informació en un altre lloc: mantenint les claus públiques anònimes. El públic pot veure que algú envia un import a una altra persona, però sense informació que vinculi la transacció amb ningú. Això és similar al nivell d'informació que publiquen les borses, on es fa públic el temps i la mida de les operacions individuals, la "cinta", però sense dir qui són les parts.



Com a tallafoc addicional, s'hauria d'utilitzar un nou parell de claus per a cada transacció i així evitar que estiguin vinculades a un propietari comú. Una mica de vinculació és inevitable amb les transaccions multientrada, que necessàriament mostren que les seves entrades pertanyen al mateix propietari. El risc és que si es revela el propietari d'una clau, la vinculació pot revelar altres transaccions pertanyents al mateix propietari.

11. Càlculs

Considerem l'escenari d'un atacant que intenta generar una cadena alternativa més ràpid que la cadena honesta. Fins i tot si això s'aconsegueix, no obre el sistema a canvis arbitraris com ara crear valor de l'aire o agafar diners que mai no han estat de l'atacant. Els nodes no acceptaran una transacció invàlida com a pagament i els nodes honestos mai no acceptaran cap bloc que els contingui. Un atacant només pot intentar canviar una de les seves pròpies transaccions per recuperar els diners que ha gastat recentment.

La carrera entre la cadena honesta i una cadena atacant es pot veure com un passeig aleatori binomial. L'esdeveniment d'èxit és la cadena honesta sent ampliada un bloc, augmentant el seu avantatge en +1, i l'esdeveniment fallit és la cadena de l'atacant sent ampliada un bloc, reduint l'escletxa en -1.

La probabilitat que un atacant pugui atrapar la cadena honesta des d'un dèficit determinat és anàloga al problema de la ruïna del jugador. Suposem que un jugador amb un crèdit il·limitat comença amb dèficit i juga potencialment un nombre infinit de proves per intentar assolir el punt d'equilibri. Podem calcular la probabilitat que arribi a un punt d'equilibri o que un atacant atrapi la cadena honesta, de la següent manera [8]:

p = probabilitat que un node honest trobi el següent bloc
q = probabilitat que l'atacant trobi el següent bloc
q_z = probabilitat que l'atacant atrapi la cadena honesta des de z blocs enrere

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

Donat el nostre supòsit que $p > q$, la probabilitat disminueix exponencialment a mesura que augmenta el nombre de blocs que l'atacant ha de recuperar. Amb les probabilitats en contra, si no té un cop de sort des del principi, les seves possibilitats es van esfumant a mesura que es queda enrere.

Ara considerem quant temps ha d'esperar el destinatari d'una nova transacció per estar prou segur que el remitent no pot canviar la transacció. Suposem que el remitent és un atacant que vol fer creure al destinatari que durant un temps l'ha pagat, per posteriorment canviar el pagament i pagar-se a si mateix l'import de la transacció. El receptor serà avisat quan això passi, però el remitent espera que ja sigui massa tard.

El receptor genera un nou parell de claus i dóna la clau pública al remitent poc abans de signar. Això impedeix que el remitent prepari una cadena de blocs abans d'hora treballant-hi contínuament fins que tingui la sort d'avançar-se prou i després executi la transacció en aquell moment. Un cop enviada la transacció, el remitent deshonest comença a treballar en secret en una cadena paral·lela que conté una versió alternativa de la seva transacció.

El destinatari espera fins que la transacció s'hagi afegit a un bloc i que un nombre de z blocs s'hagin enllaçat després d'aquesta. Ell no sap la quantitat exacta de progrés que ha realitzat l'atacant, però suposant que els blocs honestos han tardat el temps mitjà esperat per bloc, el progrés potencial de l'atacant serà una distribució de Poisson amb el valor esperat:

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

Per obtenir la probabilitat que l'atacant pugui atrapar la cadena honesta, multipliquem la densitat de Poisson per cada quantitat de progrés que hagués pogut fer, per la probabilitat que pogués atrapar la cadena a partir d'aquest moment:

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

Reorganització per evitar sumar la cua infinita de la distribució...

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

Convertint-ho a codi C...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Executant alguns resultats, podem veure que la probabilitat disminueix exponencialment amb 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
```

Resolució de P inferior a 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. Conclusió

Hem proposat un sistema per realitzar transaccions electròniques sense dependre de la confiança. Havíem començat amb el marc habitual de monedes fetes de signatures digitals, que proporciona un fort control de la propietat, però és incomplet sense una manera d'evitar la doble despesa. Per solucionar-ho, hem proposat una xarxa d'igual a igual que utilitza proves de treball per enregistrar un historial públic de transaccions, que ràpidament es fa computacionalment poc pràctica perquè un atacant la modifiqui si els nodes honestos controlen la majoria de la potència de CPU. La xarxa és robusta en la seva senzillesa no estructurada. Els nodes treballen tots a la vegada amb poca coordinació. No necessiten ser identificats, ja que els missatges no s'encaminen a cap lloc concret i només necessiten ser lliurats en base al màxim esforç. Els nodes poden marxar i tornar-se a unir a la xarxa a voluntat, acceptant la cadena de prova de treball com a prova del que ha passat mentre no hi eren. Els nodes voten amb la seva potència de CPU, expressant la seva acceptació de blocs vàlids treballant per ampliar-los i rebutjant blocs invàlids en negar-se a treballar-hi. Qualssevol regles i incentius necessaris es poden aplicar amb aquest mecanisme de consens.

Referències

[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.