

Bitcoin: Un sistema electrónico de cajas Peer a Peer

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

Resumen. Una versión puramente entre pares de efectivo electrónico permitiría que los pagos en línea fueran enviados directamente de una parte a otra sin pasar por una institución financiera. Las firmas digitales proporcionan parte de la solución, pero los principales beneficios se pierden si un tercero de confianza sigue siendo necesario para prevenir el doble gasto. Proponemos una solución al problema de doble gasto usando una red de par a par. Las transacciones de timetamps de red haciéndolas entrar en una cadena continua de pruebas de trabajo basada en hash, formando un registro que no puede ser cambiado sin volver a hacer la prueba de trabajo. La cadena más larga no sólo sirve como prueba de la secuencia de eventos presenciados, sino prueba de que vino de la mayor piscina de poder de la CPU. Mientras la mayoría de la potencia de la CPU esté controlada por nodos que no están cooperando para atacar la red, garantiza la cadena más larga y los atacantes más avanzados. La propia red requiere una estructura mínima. Los mensajes se transmiten con el mejor esfuerzo, y los nodos pueden salir y unirse a la red a voluntad, aceptando la cadena de prueba de trabajo más larga como prueba de lo que sucedió mientras se habían ido.

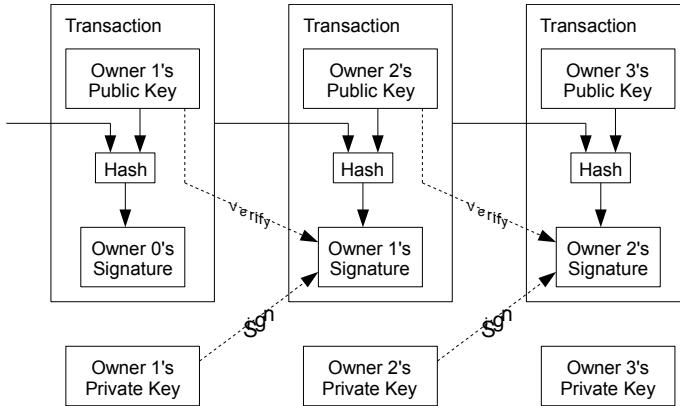
1. Introducción

El comercio en Internet ha llegado a depender casi exclusivamente de las instituciones financieras que prestan servicios como terceros de confianza para procesar pagos electrónicos. Si bien el sistema funciona lo suficientemente bien para la mayoría de las transacciones, todavía sufre de las debilidades inherentes del modelo basado en la confianza. Las transacciones completamente no reversibles no son realmente posibles, ya que las instituciones financieras no pueden evitar la mediación de controversias. El costo de la mediación aumenta los costos de transacción, limitando el tamaño mínimo de las transacciones prácticas y reduciendo la posibilidad de pequeñas transacciones casuales, y hay un costo más amplio en la pérdida de capacidad para efectuar pagos no reversibles por servicios no reversibles. Con la posibilidad de inversión, la necesidad de confianza se extiende. Los comerciantes deben ser cuidadosos con sus clientes, acosándolos para obtener más información de lo que de otro modo necesitarían. Un cierto porcentaje de fraude se acepta como inevitable. Estos costos y incertidumbres de pago se pueden evitar en persona utilizando moneda física, pero no existe ningún mecanismo para efectuar pagos a través de un canal de comunicaciones sin una parte de confianza.

Lo que se necesita es un sistema de pago electrónico basado en la prueba criptográfica en lugar de la confianza, permitiendo que cualquier dos partes dispuestas a actuar directamente entre sí sin la necesidad de un tercero confiable. Las transacciones que son computacionalmente poco prácticas para revertir protegerían a los vendedores del fraude, y los mecanismos rutinarios de garantía bloqueada podrían aplicarse fácilmente para proteger a los compradores. En este artículo proponemos una solución al problema de doble gasto utilizando un servidor de temporizador distribuido entre pares para generar pruebas computacionales del orden cronológico de las transacciones. El sistema es seguro mientras los nodos honestos controlan colectivamente más poder de CPU que cualquier grupo cooperativo de nodos atacantes.

2. Transacciones

Definimos una moneda electrónica como una cadena de firmas digitales. Cada propietario transfiere la moneda a la siguiente firmando digitalmente un hash de la transacción anterior y la clave pública del próximo propietario y agregando éstos al final de la moneda. Un beneficiario puede verificar las firmas para verificar la cadena de propiedad.

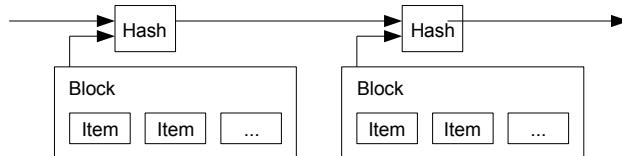


El problema, por supuesto, es que el beneficiario no puede verificar que uno de los propietarios no doblara la moneda. Una solución común es introducir una autoridad central de confianza, o menta, que comprueba cada transacción para el doble gasto. Después de cada transacción, la moneda debe ser devuelta a la menta para emitir una nueva moneda, y sólo las monedas emitidas directamente de la menta se confían en no ser doble-spent. El problema con esta solución es que el destino de todo el sistema monetario depende de la empresa que dirige la menta, con cada transacción que tiene que pasar por ellos, como un banco.

Necesitamos una manera para que el beneficiario sepa que los propietarios anteriores no firmaron ninguna transacción anterior. Para nuestros propósitos, la primera transacción es la que cuenta, por lo que no nos importan los intentos posteriores de doble discurso. La única manera de confirmar la ausencia de una transacción es ser consciente de todas las transacciones. En el modelo basado en la menta, la menta era consciente de todas las transacciones y decidió que llegaron primero. Para lograr esto sin una parte de confianza, las transacciones deben ser anunciadas públicamente [1], y necesitamos un sistema para que los participantes acuerden una sola historia del orden en que fueron recibidos. El beneficiario necesita pruebas de que en el momento de cada transacción, la mayoría de los nodos acordaron que fue el primero recibido.

3. Timestamp Server

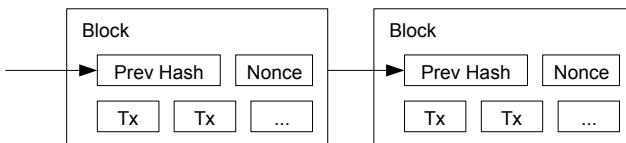
La solución que proponemos comienza con un servidor de timetamp. Un servidor de timetamp funciona tomando un hash de un bloque de artículos para ser timetamped y publicar ampliamente el hash, como en un periódico o la publicación Usenet [2-5]. El timetamp demuestra que los datos deben haber existido en ese momento, obviamente, para entrar en el hash. Cada timetamp incluye el timetamp anterior en su hash, formando una cadena, con cada timetamp adicional que refuerza los anteriores.



4. Prueba de trabajo

Para implementar un servidor de timestamp distribuido sobre una base entre pares, tendremos que usar un sistema de prueba de trabajo similar al Hashcash [6] de Adam Back, en lugar de publicaciones de periódico o Usenet. La prueba de trabajo implica el escaneo de un valor que cuando fue aplastado, como con SHA-256, el hash comienza con un número de cero bits. El trabajo promedio requerido es exponencial en el número de cero bits requeridos y se puede verificar ejecutando un solo hash.

Para nuestra red de temporizadores, implementamos la prueba de trabajo al aumentar un nonce en el bloque hasta que se encuentra un valor que da el hash del bloque los bits cero requeridos. Una vez que se ha gastado el esfuerzo de la CPU para que satisfaga la prueba de trabajo, el bloque no se puede cambiar sin rehacer el trabajo. Como bloques posteriores están encadenados después de él, el trabajo para cambiar el bloque incluiría rehacer todos los bloques después de él.



La prueba del trabajo también resuelve el problema de determinar la representación en la toma de decisiones de la mayoría. Si la mayoría se basaba en un solo voto de dirección IP, podría ser superado por cualquiera capaz de asignar muchos IPs. Proof-of-work es esencialmente un-CPU-un-vote. La decisión mayoritaria está representada por la cadena más larga, que tiene el mayor esfuerzo de prueba de trabajo invertido en ella. Si la mayoría de la potencia de la CPU está controlada por nodos honestos, la cadena honesta crecerá más rápido y superará cualquier cadena competidora. Para modificar un bloque pasado, un atacante tendría que rehacer la prueba de trabajo del bloque y todos los bloques después de él y luego ponerse al día y superar el trabajo de los nodos honestos. Demostraremos más tarde que la probabilidad de un atacante más lento que alcanza disminuye exponencialmente a medida que se agregan bloques subsiguientes.

Para compensar el aumento de la velocidad del hardware y el interés variable en la ejecución de los nodos con el tiempo, la dificultad de prueba del trabajo se determina por un promedio móvil que apunta a un número promedio de bloques por hora. Si se generan demasiado rápido, la dificultad aumenta.

5. Red

Los pasos para ejecutar la red son los siguientes:

- 1) Las nuevas transacciones se transmiten a todos los nodos.
- 2) Cada nodo recoge nuevas transacciones en un bloque.
- 3) Cada nodo trabaja en encontrar una prueba de trabajo difícil para su bloque.
- 4) Cuando un nodo encuentra una prueba de trabajo, transmite el bloque a todos los nodos.
- 5) Nodos aceptan el bloque sólo si todas las transacciones en él son válidas y no ya gastadas.
- 6) Los nodos expresan su aceptación del bloque al trabajar en la creación del siguiente bloque en la cadena, utilizando el hash del bloque aceptado como el hash anterior.

Los ganglios siempre consideran que la cadena más larga es la correcta y seguirá trabajando en ampliarla. Si dos nodos transmiten diferentes versiones del siguiente bloque simultáneamente, algunos nodos pueden recibir uno o el otro primero. En ese caso, trabajan en el primero que recibieron, pero salvan a la otra rama en caso de que se haga más tiempo. La corbata se romperá cuando se encuentre la siguiente prueba de trabajo y una rama se vuelve más larga; los nodos que estaban trabajando en la otra rama se cambiarán a la más larga.

Las nuevas transmisiones de transacciones no necesitan necesariamente llegar a todos los nodos. Mientras lleguen a muchos nodos, llegarán a un bloque antes de tiempo. Las emisiones de bloques también son tolerantes a mensajes caídos. Si un nodo no recibe un bloque, lo solicitará cuando reciba el siguiente bloque y se dé cuenta de que se perdió uno.

6. Incentivo

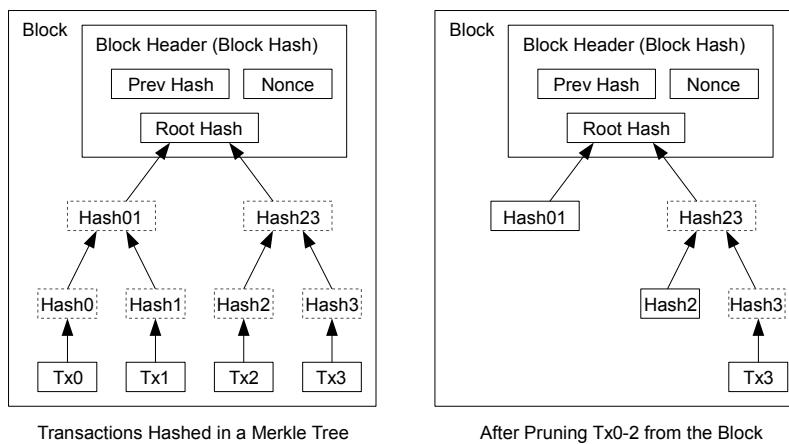
Por convención, la primera transacción en un bloque es una transacción especial que inicia una nueva moneda propiedad del creador del bloque. Esto añade un incentivo para que los nodos apoyen la red, y proporciona una forma de distribuir inicialmente monedas a la circulación, ya que no hay autoridad central para emitirlas. La adición constante de una cantidad constante de nuevas monedas es análoga a los mineros de oro que gastan recursos para añadir oro a la circulación. En nuestro caso, es tiempo de CPU y electricidad que se gasta.

El incentivo también puede financiarse con honorarios de transacción. Si el valor de salida de una transacción es inferior a su valor de entrada, la diferencia es una cuota de transacción que se añade al valor de incentivo del bloque que contiene la transacción. Una vez que un número predeterminado de monedas han entrado en circulación, el incentivo puede pasar por completo a las tasas de transacción y ser completamente libre de inflación.

El incentivo puede ayudar a alentar a los nodos a mantenerse honestos. Si un atacante codicioso es capaz de reunir más poder de CPU que todos los nodos honestos, tendría que elegir entre usarlo para defraudar a la gente robando sus pagos, o usándolo para generar nuevas monedas. Debe en contrar más provechoso jugar por las reglas, tales reglas que le favorecen con más monedas nuevas que todos los demás combinados, que socavar el sistema y la validez de su propia riqueza.

7. Reclamación del espacio de disco

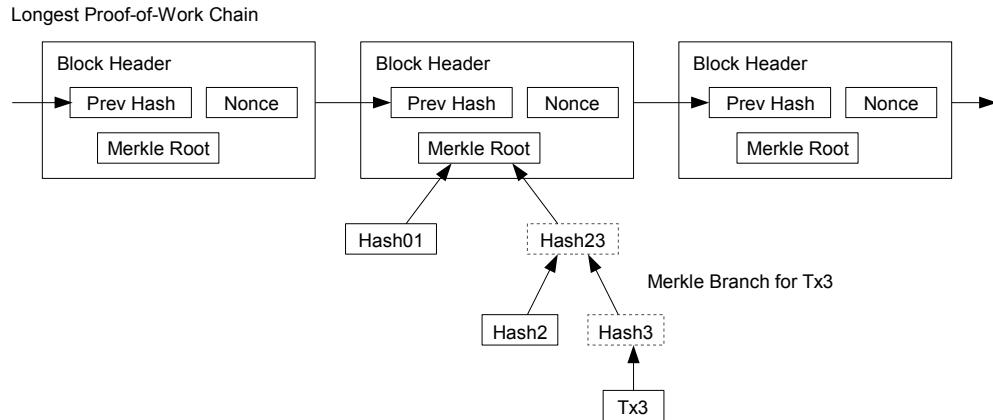
Una vez que la última transacción en una moneda es enterrada bajo suficientes bloques, las transacciones gastadas antes de que pueda ser descartada para guardar espacio de disco. Para facilitar esto sin romper el hash del bloque, las transacciones se destruyen en un árbol de Merkle [7][2][5], con sólo la raíz incluida en el hash del bloque. Los bloques antiguos pueden ser compactados por rotar ramas del árbol. Los hashes interiores no necesitan ser almacenados.



Un encabezado de bloque sin transacciones sería de unos 80 bytes. Si suponemos que los bloques se generan cada 10 minutos, $80 \text{ bytes} * 6 * 24 * 365 = 4.2\text{MB}$ al año. Con sistemas informáticos que suelen vender con 2 GB de RAM a partir de 2008, y la Ley de Moore que predice el crecimiento actual de 1.2GB al año, el almacenamiento no debe ser un problema incluso si los encabezados de bloque deben mantenerse en memoria.

8. Verificación simplificada de pagos

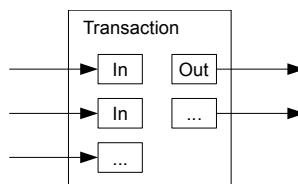
Es posible verificar los pagos sin ejecutar un nodo de red completo. Un usuario sólo necesita mantener una copia de los encabezados de bloque de la cadena de prueba de trabajo más larga, que puede conseguir por los nodos de red que se preguntan hasta que está convencido de que tiene la cadena más larga, y obtener la rama Merkle que une la transacción al bloque en el que se encuentra. No puede comprobar la transacción por sí mismo, pero al vincularla a un lugar en la cadena, puede ver que un nodo de red lo ha aceptado, y bloques añadidos después de que confirme más la red lo ha aceptado.



Como tal, la verificación es fiable mientras los nodos honestos controlan la red, pero es más vulnerable si la red es dominada por un atacante. Mientras que los nodos de red pueden verificar las transacciones por sí mismos, el método simplificado puede ser engañado por las transacciones fabricadas por un atacante mientras el atacante pueda seguir superando la red. Una estrategia para proteger contra esto sería aceptar alertas de nodos de red cuando detectan un bloque inválido, lo que incita al software del usuario a descargar el bloque completo y las operaciones alertadas para confirmar la incoherencia. Las empresas que reciben pagos frecuentes probablemente todavía querrán ejecutar sus propios nodos para una seguridad más independiente y una verificación más rápida.

9. Combinar y dividir el valor

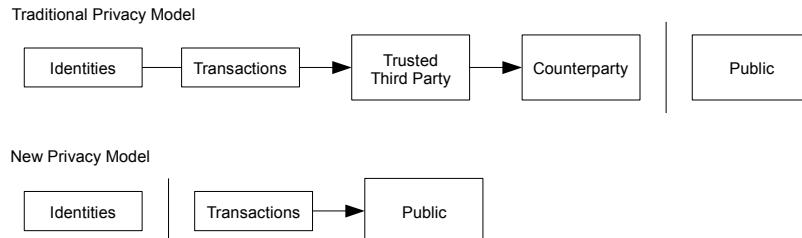
Aunque sería posible manejar monedas individualmente, no sería prudente realizar una transacción separada por cada centavo en una transferencia. Para permitir que el valor sea dividido y combinado, las transacciones contienen múltiples entradas y salidas. Normalmente habrá una sola entrada de una transacción anterior más grande o múltiples entradas que combinan cantidades más pequeñas, y en la mayoría de dos salidas: una para el pago, y otra para devolver el cambio, si hay, de regreso al remitente.



Cabe señalar que el fan-out, donde una transacción depende de varias transacciones, y esas transacciones dependen de muchas más, no es un problema aquí. Nunca es necesario extraer una copia independiente completa de la historia de una transacción.

10. Privacidad

El modelo bancario tradicional alcanza un nivel de privacidad limitando el acceso a la información a las partes involucradas y al tercero de confianza. La necesidad de anunciar públicamente todas las transacciones impide este método, pero la privacidad puede mantenerse rompiendo el flujo de información en otro lugar: manteniendo las claves públicas anónimas. El público puede ver que alguien está enviando una cantidad a otra persona, pero sin información que vincula la transacción con nadie. Esto es similar al nivel de información publicada por bolsas de valores, donde el tiempo y el tamaño de los comercios individuales, el "tape", se hace público, pero sin decir quiénes eran las partes.



Como un cortafuegos adicional, se debe utilizar un nuevo par de teclas para cada transacción para evitar que estén vinculados a un propietario común. Algunos enlaces siguen siendo inevitables con transacciones de varios insumos, que necesariamente revelan que sus insumos eran propiedad del mismo propietario. El riesgo es que si el propietario de una clave es revelado, la vinculación podría revelar otras transacciones que pertenecieron al mismo propietario.

11. Cálculos

Consideramos el escenario de un atacante tratando de generar una cadena alternativa más rápido que la cadena honesta. Incluso si esto se logra, no arroja el sistema abierto a cambios arbitrarios, como crear valor fuera del aire delgado o tomar dinero que nunca perteneció al atacante. Los nodos no van a aceptar una transacción inválida como pago, y los nodos honestos nunca aceptarán un bloque que los contenga. Un atacante sólo puede intentar cambiar una de sus propias transacciones para recuperar el dinero que ha gastado recientemente.

La carrera entre la cadena honesta y una cadena de atacantes se puede caracterizar como un proceso aleatorio binomial. El evento de éxito es la cadena honesta que se extiende por un bloque, aumentando su ventaja por +1, y el evento de fracaso es la cadena del atacante que se extiende por un bloque, reduciendo la brecha por -1.

La probabilidad de que un atacante se ponga al día de un déficit determinado es análoga al problema de la Ruina de Gambler. Supongamos que un jugador con crédito ilimitado comienza en un déficit y juega potencialmente un número infinito de ensayos para tratar de llegar a la ruptura. Podemos calcular la probabilidad de que llegue a la ruptura, o que un atacante se ponga al día con la cadena honesta, como sigue: [8]:

p = probabilidad un nodo honesto encuentra el siguiente bloque q = probabilidad el atacante encuentra el siguiente bloque q_z = probabilidad que el atacante alguna vez alcanzará desde bloques z detrás

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

Dado nuestro supuesto de que $p > q$, la probabilidad cae exponencialmente como el número de bloques que el atacante tiene que ponerse al día con aumentos. Con las probabilidades en su contra, si no hace una buena salida, sus posibilidades se vuelven descaradamente pequeñas mientras cae más atrás.

Ahora consideramos cuánto tiempo debe esperar el receptor de una nueva transacción antes de estar suficientemente seguro de que el remitente no puede cambiar la transacción. Suponemos que el remitente es un atacante que quiere hacer creer al receptor que le pagó por un tiempo, y luego cambiarlo para devolverlo a sí mismo después de que haya pasado algún tiempo. El receptor será alertado cuando eso suceda, pero el remitente espera que sea demasiado tarde.

El receptor genera un nuevo par de teclas y da la llave pública al remitente poco antes de firmar. Esto impide que el remitente prepare una cadena de bloques por delante del tiempo trabajando en ella continuamente hasta que tenga la suerte de llegar lo suficientemente lejos, y luego ejecutar la transacción en ese momento. Una vez enviado la transacción, el remitente deshonesto comienza a trabajar en secreto en una cadena paralela que contiene una versión alternativa de su transacción.

El destinatario espera hasta que la transacción se haya añadido a un bloque y z bloques han estado vinculados después de ella. No sabe la cantidad exacta de progreso que ha hecho el atacante, pero asumiendo que los bloques honestos tomaron el tiempo promedio esperado por bloque, el progreso potencial del atacante será una distribución Poisson con valor esperado:

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

Para conseguir la probabilidad de que el atacante todavía pudiera ponerse al día ahora, multiplicamos la densidad de Poisson por cada cantidad de progreso que podría haber hecho por la probabilidad que podría alcanzar desde ese punto:

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

Reordenando para evitar sumar la cola infinita de la distribución...

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

Convertir en 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++)
```

Ejecutando algunos resultados, podemos ver la probabilidad caer exponencialmente con z.

```
q=0.1 z=0 P=1.0000 z=1 P  
=0.2045873 z=2 P=0.0509  
779 z=3 P=0.0131722 z=4  
P=0.0034552 z=5 P=0.000  
9137 z=6 P=0.0002428 z=  
7 P=0.0000647 z=8 P=0.0  
000173 z=9 P=0.0000046  
z=10 P=0.0000012
```

```
q=0.3 z=0 P=1.0000 z=5 P  
=0.1773523 z=10 P=0.041  
6605 z=15 P=0.0101008 z  
=20 P=0.0024804 z=25 P  
=0.0006132 z=30 P=0.000  
1522 z=35 P=0.0000379 z  
=40 P=0.0000095 z=45 P  
=0.0000024 z=50 P=0.000  
0006
```

Solver para P menos de 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 q59
```

12. Conclusión

Hemos propuesto un sistema de transacciones electrónicas sin depender de la confianza. Comenzamos con el marco habitual de monedas hechas de firmas digitales, que proporciona un control fuerte de la propiedad, pero es incompleto sin una manera de prevenir el doble gasto. Para resolver esto, proponemos una red entre pares usando pruebas de trabajo para registrar una historia pública de transacciones que se convierte rápidamente en computacionalmente poco práctico para que un atacante cambie si los nodos honestos controlan una mayoría de poder de CPU. La red es robusta en su sencillez no estructurada. Los ganglios trabajan de inmediato con poca coordinación. No necesitan ser identificados, ya que los mensajes no se dirigen a ningún lugar en particular y sólo necesitan ser entregados en forma óptima. Los nodos pueden salir y unirse a la red a voluntad, aceptando la cadena de prueba de trabajo como prueba de lo que pasó mientras se habían ido. Votan con su poder de CPU, expresando su aceptación de bloques válidos al trabajar en extenderlos y rechazar los bloques inválidos al negarse a trabajar en ellos. Las normas e incentivos necesarios pueden aplicarse con este mecanismo de consenso.

Referencias

[1] W. Dai, "b-money", <http://www.weidai.com/bmoney.txt>, 1998. [2] H. Massias, X.S. Avila, y 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, páginas 99-111, 1991. [4] D. Bayer, S. Haber, W.S. Stornetta, "Mejorando la eficiencia y fiabilidad de la hora digital", En *Sequences II: Methods in Communication, Security and Computer Science*, páginas 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, "Protocolos para criptosistemas clave pública", En *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, páginas 122-133, abril 1980. [8] W. Feller, "Una introducción a la teoría de la probabilidad y sus aplicaciones", 1957.