Bitcoin: un sistema de dinero en efectivo electrónico entre iguales

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

Traducido al español por CriptoNoticias

Resumen. Una versión puramente de igual-a-igual de dinero en efectivo electrónico permitiría que los pagos en línea sean enviados directamente de una parte a otra sin pasar por una institución financiera. Las firmas digitales proporcionan parte de la solución, pero los beneficios principales se pierden si aún se requiere un tercero de confianza para evitar el doble gasto. Proponemos una solución al problema del doble gasto utilizando una red de igual-a-igual. La red marca el tiempo de las transacciones al agruparlas en una cadena continua de pruebas de trabajo basadas en hash, formando un registro que no se puede cambiar sin rehacer la prueba de trabajo. La cadena más larga no solo sirve como prueba de la secuencia de eventos presenciados, sino también como prueba de que proviene del mayor grupo de potencia de CPU. Mientras la mayoría de la potencia de CPU esté controlada por nodos que no cooperan para atacar la red, ellos generarán la cadena más larga y superarán a los atacantes. La propia red requiere una estructura mínima. Los mensajes se transmiten sobre una base de mejor esfuerzo, y los nodos pueden abandonar y unirse de nuevo a la red a voluntad, aceptando la cadena de prueba de trabajo más larga como prueba de lo que sucedió mientras estaban fuera.

1. Introducción

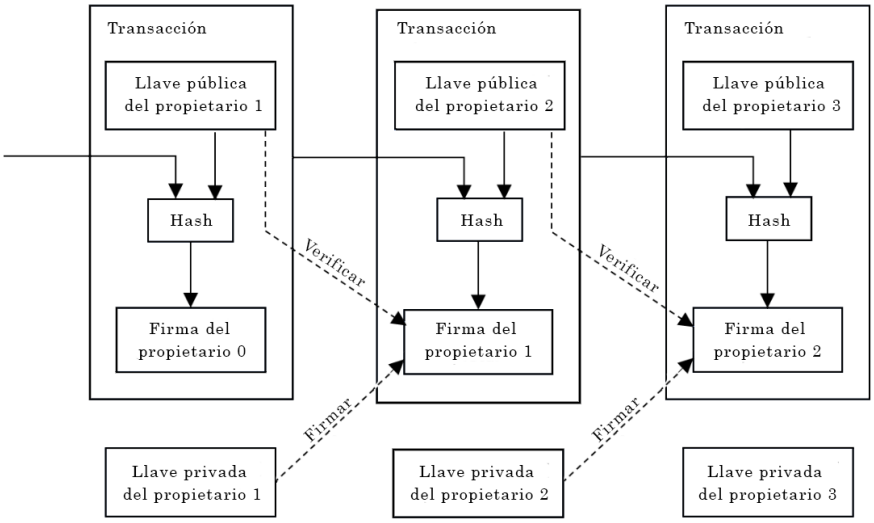
El comercio en Internet ha llegado a depender casi exclusivamente de instituciones financieras que actúan como terceros de confianza para procesar pagos electrónicos. Si bien el sistema funciona suficientemente bien para la mayoría de las transacciones, aún sufre las debilidades inherentes del modelo basado en confianza. Las transacciones completamente no reversibles no son realmente posibles, ya que las instituciones financieras no pueden evitar mediar disputas. El costo de la mediación aumenta los costos de transacción, limitando el tamaño mínimo práctico de transacción y cortando la posibilidad de pequeñas transacciones casuales, y existe un costo más amplio en la pérdida de capacidad para realizar pagos no reversibles por servicios no reversibles. Con la posibilidad de reversión, la necesidad de confianza se extiende. Los comerciantes deben ser cautelosos de sus clientes, molestándolos por más información de la que de otra manera necesitarían. Un cierto porcentaje de fraude es aceptado como inevitable. Estos costos y las incertidumbres de pago se pueden evitar en persona utilizando una moneda física, pero no existe ningún mecanismo para realizar pagos a través de un canal de comunicaciones sin un tercero de confianza.

Lo que se necesita es un sistema de pago electrónico basado en pruebas criptográficas en lugar de confianza, que permita a dos partes interesadas cualesquiera transar directamente entre ellas sin la necesidad de un tercero de confianza. Las transacciones que son computacionalmente impracticas de revertir protegerían a los vendedores del fraude, y mecanismos de fideicomiso rutinarios podrían implementarse fácilmente para proteger a los compradores. En este documento, proponemos una solución al problema del doble gasto mediante el uso de un servidor de marcas de tiempo distribuido de igual-a-igual para generar prueba computacional del orden cronológico de las transacciones. El sistema es seguro siempre que los nodos honestos controlen colectivamente más potencia de CPU que cualquier grupo colaborador de nodos atacantes.

2. Transacciones

Definimos una moneda electrónica como una cadena de firmas digitales. Cada propietario transfiere la moneda al siguiente al firmar digitalmente un hash de la transacción anterior y la llave pública del

siguiente propietario y agregarlos al final de la moneda. Un beneficiario puede verificar las firmas para verificar la cadena de propiedad.

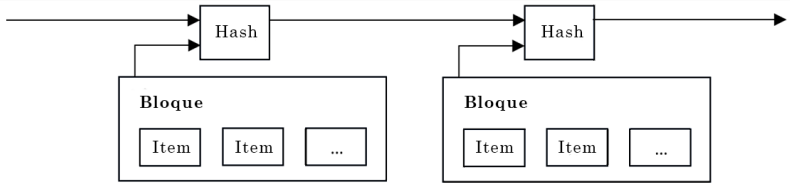


El problema, por supuesto, es que el beneficiario no puede verificar que uno de los propietarios no haya gastado dos veces la moneda. Una solución común es introducir una autoridad central de confianza, o casa de moneda, que verifica cada transacción por doble gasto. Después de cada transacción, la moneda debe devolverse a la casa de moneda para emitir una nueva moneda, y solo las monedas emitidas directamente por la casa de moneda son confiables de no haber sido doblemente gastadas. El problema con esta solución es que el destino de todo el sistema monetario depende de la compañía que gestiona la casa de moneda, con cada transacción debiendo pasar a través de ellos, justo como un banco.

Necesitamos una forma para que el beneficiario sepa que los propietarios anteriores no firmaron ninguna transacción previa. Para nuestros propósitos, la transacción más temprana es la que cuenta, por lo que no nos preocupamos por los intentos posteriores de doble gasto. La única forma de confirmar la ausencia de una transacción es estar al tanto de todas las transacciones. En el modelo basado en la casa de moneda, ésta estaba al tanto de todas las transacciones y decidía cuál llegaba primero. Para lograr esto sin una tercera parte de confianza, las transacciones deben ser anunciadas públicamente [1], y necesitamos un sistema para que los participantes acuerden un historial único del orden en que ellas se recibieron. El beneficiario necesita prueba de que al momento de cada transacción, la mayoría de los nodos acordaron que esa fue la primera recibida.

3. Servidor de Marca de Tiempo

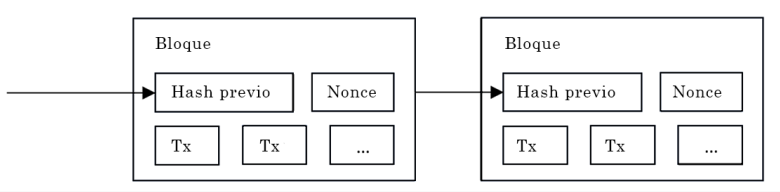
La solución que proponemos comienza con un servidor de marca de tiempo. Un servidor de marca de tiempo funciona realizando un hash de un bloque de elementos a ser marcados de tiempo y publicando ampliamente el hash, tal como en un periódico o una publicación de Usenet [2-5]. La marca de tiempo prueba que los datos deben haber existido en ese momento, obviamente, para poder entrar en el hash. Cada marca de tiempo incluye la marca de tiempo previa en su hash, formando una cadena, con cada marca de tiempo adicional reforzando las anteriores a ésta.



4. Prueba de Trabajo

Para implementar un servidor de marca de tiempo distribuido de igual-a-igual, tendremos que usar un sistema de prueba de trabajo similar al Hashcash de Adam Back [6], en lugar de publicaciones de periódico o Usenet. La prueba de trabajo implica la búsqueda de un valor que al aplicarle un hash, tal como SHA-256, el hash comienza con un número de bits cero. El trabajo promedio requerido es exponencial en el número de bits cero requeridos y se puede verificar ejecutando un simple hash.

Para nuestra red de marca de tiempo, implementamos la prueba de trabajo incrementando un nonce (o número único) en el bloque hasta que se encuentre un valor que proporcione al hash del bloque los bits cero requeridos. Una vez que el esfuerzo de la CPU se ha gastado para que cumpla con la prueba de trabajo, el bloque no se puede cambiar sin rehacer el trabajo. Como los bloques posteriores se encadenan después de éste, el trabajo para cambiar el bloque incluiría rehacer todos los bloques posteriores a éste.



La prueba de trabajo también resuelve el problema de determinar la representación en la toma de decisiones por mayoría. Si la mayoría se basara en una-dirección-IP-un-voto, podría ser subvertida por cualquier persona capaz de asignar muchas direcciones IP. La prueba de trabajo es esencialmente un-CPU-un-voto. La decisión por mayoría está representada por la cadena más larga, que cuenta con el mayor esfuerzo de prueba de trabajo invertido en ella. Si la mayoría de la potencia de CPU está controlada por nodos honestos, la cadena honesta crecerá más rápido y superará a cualquier cadena competidora. Para modificar un bloque del pasado, un atacante tendría que rehacer la prueba de trabajo del bloque y todos los bloques después de éste y luego alcanzar y superar el trabajo de los nodos honestos. Más adelante demostraremos que la probabilidad de que un atacante más lento se ponga al día disminuye exponencialmente a medida que se agregan bloques subsecuentes.

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

5. La Red

Los pasos para correr 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 para encontrar una prueba de trabajo difícil para su bloque.
4) Cuando un nodo encuentra una prueba de trabajo, difunde el bloque a todos los nodos.

5) Los nodos aceptan el bloque solo 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 previo.

Los nodos siempre consideran que la cadena más larga es la correcta y seguirán trabajando en extenderla. Si dos nodos difunden versiones diferentes del siguiente bloque simultáneamente, algunos nodos pueden recibir uno u otro primero. En ese caso, trabajan en el primero que recibieron, pero guardan la otra rama en caso de que se vuelva más larga. El empate se romperá cuando se encuentre la próxima prueba de trabajo y una rama se vuelva más larga; los nodos que estaban trabajando en la otra rama cambiarán entonces a la más larga.

Las difusiones de nuevas transacciones no necesariamente tienen que llegar a todos los nodos. Mientras lleguen a muchos nodos, entrarán en un bloque en poco tiempo. Las transmisiones de bloque también toleran los mensajes perdidos. Si un nodo no recibe un bloque, lo solicitará cuando reciba el siguiente bloque y se dé cuenta de que perdió uno.

6. Incentivo

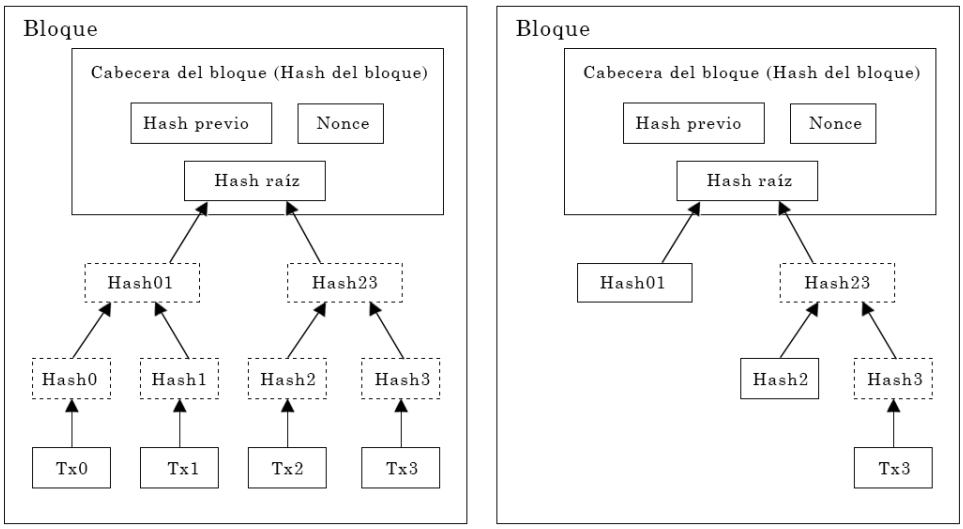
Por convención, la primera transacción en un bloque es una transacción especial que inicia una nueva moneda perteneciente al creador del bloque. Esto agrega un incentivo para que los nodos apoyen la red, y proporciona una manera de distribuir inicialmente monedas a circulación, ya que no hay una autoridad central para emitirlas. La adición estable de una cantidad constante de nuevas monedas es análoga a los mineros de oro que gastan recursos para agregar oro a circulación. En nuestro caso, son el tiempo de CPU y la electricidad que son gastados.

El incentivo también puede ser financiado con comisiones de transacción. Si el valor de salida de una transacción es menor que su valor de entrada, la diferencia es una comisión de transacción que se agrega al valor incentivo del bloque que contiene la transacción. Una vez que un número predeterminado de monedas haya entrado en circulación, el incentivo puede pasar completamente a las comisiones 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 ensamblar más poder de CPU que todos los nodos honestos, tendría que elegir entre usarlo para defraudar a las personas robando de vuelta sus pagos, o usarlo para generar nuevas monedas. Debería encontrar más rentable jugar según las reglas, reglas tales que lo favorecen con más nuevas monedas que todos los demás combinados, que socavar el sistema y la validez de su propia riqueza.

7. Recuperando Espacio en Disco

Una vez que la última transacción en una moneda está sepultada bajo suficientes bloques, las transacciones gastadas antes de ella pueden ser descartadas para ahorrar espacio en disco. Para facilitar esto sin romper el hash del bloque, las transacciones son procesadas en hashes en un árbol Merkle [7] [2] [5], con solo la raíz incluida en el hash del bloque. Los bloques viejos se pueden entonces compactar desprendiendo las ramas del árbol. Los hashes interiores no necesitan ser almacenados.



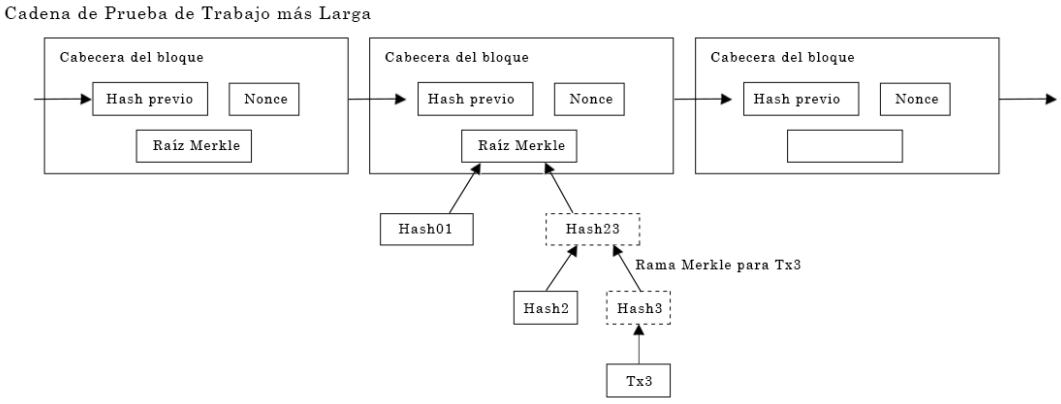
Transacciones procesadas en hashes en un árbol de Merkle

Luego de podar Tx0-2 del bloque

Una cabecera de bloque sin transacciones sería de unos 80 bytes. Si suponemos que los bloques se generan cada 10 minutos, 80 bytes * 6 * 24 * 365 = 4,2MB por año. Con sistemas informáticos siendo típicamente vendidos con 2 GB de RAM a partir de 2008, y la Ley de Moore prediciendo un crecimiento actual de 1,2 GB por año, el almacenamiento no debería ser un problema incluso si las cabeceras de bloque deben mantenerse en la memoria.

8. Verificación simplificada de pago

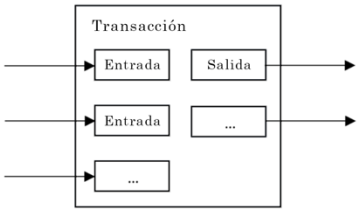
Es posible verificar los pagos sin ejecutar un nodo de red completo. Un usuario solo necesita conservar una copia de las cabeceras de bloque de la cadena de prueba de trabajo más larga, la cual puede obtener consultando los nodos de la red hasta que esté convencido de que tiene la cadena más larga, y obtener la rama Merkle que vincula la transacción con el bloque en el que fue marcada de tiempo. El usuario no puede verificar la transacción por sí mismo, pero al vincularla a un lugar en la cadena, puede ver que un nodo de red la ha aceptado, y los bloques agregados después de ésta confirman aún más que la red la ha aceptado.



Como tal, la verificación es confiable siempre que los nodos honestos controlen la red, pero es más vulnerable si la red está dominada por un atacante. Mientras que los nodos de la red pueden verificar las transacciones por sí mismos, el método simplificado puede ser engañado por las transacciones fabricadas de un atacante durante tanto tiempo como el atacante pueda seguir dominando la red. Una estrategia para protegerse contra esto sería aceptar alertas de los nodos de la red cuando detecten un bloque inválido, solicitando al software del usuario que descargue el bloque completo y las transacciones alertadas para confirmar la inconsistencia. Los negocios que reciben pagos frecuentes probablemente aún querrán ejecutar sus propios nodos para una seguridad más independiente y una verificación más rápida.

9. Combinando y Dividiendo Valor

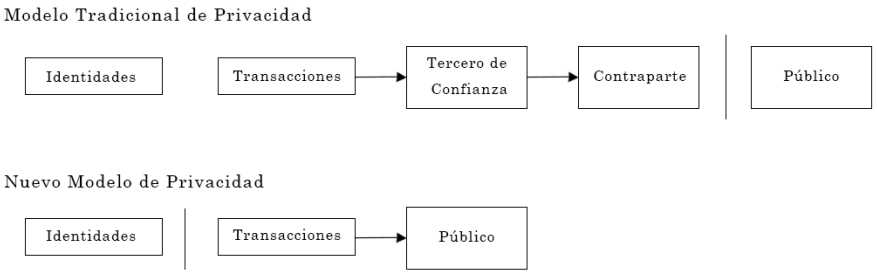
Aunque sería posible manejar las monedas individualmente, sería pesado manejar una transacción separada por cada centavo en una transferencia. Para permitir que el valor sea dividido y combinado, las transacciones contienen múltiples entradas y salidas. Normalmente habrá o una entrada única de una transacción previa más grande o entradas múltiples que combinan montos más pequeños y, como máximo, dos salidas: una para el pago y una para devolver el cambio, si corresponde, al remitente.



Debe tenerse en cuenta que la diseminación de control, 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 del historial de una transacción.

10. Privacidad

El modelo bancario tradicional logra un nivel de privacidad al limitar el acceso a la información a las partes involucradas y al tercero de confianza. La necesidad de anunciar públicamente todas las transacciones imposibilita este método, pero la privacidad todavía se puede mantener rompiendo el flujo de información en otro lugar: al mantener las llaves públicas anónimas. El público puede ver que alguien está enviando un monto a otra persona, pero sin información que vincule la transacción con alguien. Esto es similar al nivel de información liberada por las bolsas de valores, donde el tiempo y el tamaño de las operaciones individuales, la "cinta", se hace público, pero sin indicar quiénes fueron las partes.



Como un firewall adicional, un nuevo par de llaves debería usarse para cada transacción para evitar que se vinculen a un mismo propietario. Cierta vinculación sigue siendo inevitable con transacciones de múltiples entradas, que necesariamente revelan que sus entradas fueron propiedad del mismo dueño. El riesgo es que si se revela el propietario de una llave, la vinculación podría revelar otras transacciones que pertenecieron al mismo dueño.

11. Cálculos

Consideramos el escenario de un atacante que intenta generar una cadena alternativa más rápido que la cadena honesta. Incluso si esto se logra, no abre el sistema a cambios arbitrarios, como crear valor de la nada o tomar dinero que nunca perteneció al atacante. Los nodos no aceptarán una transacción inválida como pago, y los nodos honestos nunca aceptarán un bloque que las contenga. Un atacante solo puede intentar cambiar una de sus propias transacciones para recuperar el dinero que gastó recientemente.

La carrera entre la cadena honesta y una cadena del atacante puede caracterizarse como una Caminata Aleatoria Binomial. El evento de éxito es la cadena honesta siendo extendida en un bloque, aumentando su ventaja en +1, y el evento de fracaso es la cadena del atacante siendo extendida en un bloque, lo que reduce la brecha en -1.

La probabilidad de que un atacante se recupere de un déficit dado es análoga al problema de la Ruina del Jugador. Supongamos que un jugador con crédito ilimitado comienza en déficit y juega potencialmente un número infinito de intentos para tratar de alcanzar el punto de equilibrio. Podemos calcular la probabilidad de que alguna vez logre alcanzar el punto de equilibrio, o de que un atacante alcance alguna vez a la cadena honesta, de la siguiente manera [8]:

p = probabilidad de que un nodo honesto encuentre el siguiente bloque
q = probabilidad de que el atacante encuentre el siguiente bloque
q_z = probabilidad de que el atacante se recupere de estar z bloques detrás

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

Dada nuestra suposición de que p > q, la probabilidad disminuye exponencialmente a medida que aumenta el número de bloques que el atacante tiene que alcanzar. Con las probabilidades en contra de él, si no hace una estocada de suerte hacia adelante al principio, sus posibilidades se vuelven cada vez más pequeñas a medida que se queda atrás.

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

El receptor genera un nuevo par de llaves y le entrega la llave pública al remitente poco antes de firmar. Esto evita que el remitente prepare una cadena de bloques de antemano trabajando continuamente hasta que tenga la suerte de avanzar lo suficiente y luego ejecute la transacción en ese momento. Una vez que se envía la transacción, el remitente deshonesto comienza a trabajar en secreto en una cadena paralela que contiene una versión alternativa de su transacción.

El beneficiario espera hasta que la transacción se haya agregado a un bloque y z cantidad de bloques haya sido vinculada después de éste. Él 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 de Poisson con valor esperado:

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

Para obtener la probabilidad de que el atacante todavía pueda ponerse al día, multiplicamos la densidad de Poisson para cada cantidad de progreso que pudo haber realizado por la probabilidad de que pueda ponerse al día a partir de ese punto:

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

Reordenando para evitar sumar la cola infinita de la distribución…

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

Convirtiendo a código 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;
}
```

Ejecutando algunos resultados, podemos ver que la probabilidad cae exponencialmente con *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
```

Resolviendo para P menor que 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. Conclusión

Hemos propuesto un sistema para transacciones electrónicas sin depender en la confianza. Comenzamos con el marco habitual de monedas hechas de firmas digitales, lo que proporciona un fuerte control de propiedad, pero está incompleto sin una manera de evitar el doble gasto. Para resolver esto, propusimos una red de igual-a-igual utilizando prueba de trabajo para registrar un historial público de transacciones que rápidamente se vuelve computacional impráctico de cambiar para un atacante si los nodos honestos controlan la mayoría de la potencia de CPU. La red es robusta en su simpleza desestructurada. Los nodos trabajan todos a la vez con poca coordinación. Ellos no necesitan ser identificados, ya que los mensajes no se envían a ningún lugar en particular y solo deben ser entregados en base al mejor esfuerzo. Los nodos pueden abandonar y volver a unirse a la red a voluntad, aceptando la cadena de prueba de trabajo como prueba de lo que sucedió mientras estaban fuera. Ellos votan con su poder de CPU, expresando su aceptación de bloques válidos al trabajar en extenderlos y rechazando bloques inválidos al rehusarse a trabajar en ellos. Cualesquiera reglas e incentivos necesarios pueden ser impuestos 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, "Diseño de un servicio de marca de tiempo seguro con requisitos de confianza mínimos", en el *20º Simposio sobre Teoría de la Información* en Benelux, mayo de 1999.

[3] S. Haber, W.S. Stornetta, "Cómo marcar de tiempo un documento digital", en *Journal of Cryptology*, vol. 3, no 2, páginas 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Mejorando la eficiencia y la confiabilidad del estampado de tiempo digital", En *Secuencias II: Métodos en Comunicación, Seguridad y Ciencias de la Computación*, páginas 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Nombres seguros para cadenas de bits", en *Actas de la 4ª Conferencia ACM en Seguridad de Computadores y Comunicaciones*, páginas 28-35, abril de 1997.

[6] A. Back, "Hashcash - una contramedida para la denegación de servicio" http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocolos para criptosistemas de llave pública," En *Proc. 1980 Simposio de Seguridad y Privacidad*, IEEE Computer Society, páginas 122-133, abril de 1980.

[8] W. Feller, "Una introducción a la teoría de la probabilidad y sus aplicaciones", 1957.