Riesgo bajo

Acepte canales de un conjunto conocido de nodos con los que haya abierto previamente canales exitosos.

Riesgo medio

Solo acepte canales de nodos que hayan estado presentes en el gráfico durante un período más largo y tengan algunos canales de larga duración.

Riesgo mayor

Acepte cualquier canal entrante e implemente las mitigaciones descritas en "Consideraciones de enrutamiento".

Conclusión

En resumen, la privacidad y la seguridad son temas complejos y matizados, y aunque muchos investigadores y desarrolladores buscan mejoras en toda la red, es importante que todos los que participan en la red entiendan lo que pueden hacer para proteger su propia privacidad y aumentar la seguridad en un nivel de nodo individual.

Referencias y lecturas adicionales

En este capítulo, usamos muchas referencias de investigaciones en curso sobre seguridad Lightning. Puede encontrar estos artículos y documentos útiles enumerados por tema en las siguientes listas.

Ataques de privacidad y sondeo

- Jordi Herrera-Joancomartí et al. “Sobre la dificultad de ocultar el saldo de los canales de Lightning Network”. ***Asia CCS '19: Actas de la Conferencia ACM Asia 2019 sobre seguridad informática y de las comunicaciones*** (julio de 2019): 602–612.

- Utz Nisslmüller et al. "Hacia ataques de confidencialidad activos y pasivos en redes fuera de la cadena de criptomonedas". Versión preliminar de arXiv, ***https://arxiv.org/abs/2003.00003*** (2020).
- Serguéi Tikhomirov et al. "Sondeo de balances de canales en Lightning Network". Versión preliminar de arXiv, ***https://arxiv.org/abs/2004.00333*** (2020).
- George Kappos et al. "Un análisis empírico de la privacidad en Lightning Network". Versión preliminar de arXiv, ***https://arxiv.org/abs/2003.12470*** (2021).
- Zap código fuente con la función de sondeo.

Ataques de congestión

- Ayelet Mizrahi y Aviv Zohar. “Ataques de congestión en redes de canales de pago”. Versión preliminar de arXiv, ***https://arxiv.org/abs/2002.06564*** (2020).

Consideraciones de enrutamiento

- Marty Bent, entrevista con Joost Jager, ***Tales from the Crypt,*** audio de podcast, 2 de octubre de 2020, ***https://anchor.fm/tales-from-the-crypt/episodes/197-Joost-Jager-ekghn6.***

Capítulo 17. Conclusión

En solo unos años, Lightning Network ha pasado de ser un documento técnico a una red global de rápido crecimiento. Como la segunda capa de Bitcoin, ha cumplido la promesa de pagos rápidos, económicos y privados. Además, ha iniciado un tsunami de innovación, ya que libera a los desarrolladores de las limitaciones del consenso cerrado que existe en el desarrollo de Bitcoin.

La innovación en Lightning Network está ocurriendo en varios niveles diferentes:

- En el protocolo Core de Bitcoin, brindando uso y demanda de nuevos Códigos de operación Bitcoin Script, algoritmos de firma y optimizaciones
- A nivel del protocolo Lightning, con nuevas funciones implementadas rápidamente en toda la red
- A nivel de canal de pago, con nuevas construcciones y mejoras de canal

- Como funciones opcionales distintas implementadas de extremo a extremo por implementaciones independientes que los remitentes y destinatarios pueden usar si lo desean
- Con nuevas y emocionantes Aplicaciones Lightning (LApps) construidas sobre los clientes y protocolos

Veamos cómo estas innovaciones están cambiando Lightning ahora y en el futuro cercano.

Innovación descentralizada y asíncrona

Lightning no está sujeto a un consenso cerrado, como es el caso de Bitcoin.
Eso significa que diferentes clientes Lightning pueden implementar diferentes funciones y negociar sus interacciones (consulte “Bits de función y protocolo

Extensibilidad"). Como resultado, la innovación en Lightning Network se está produciendo a un ritmo mucho más rápido que en Bitcoin.

Lightning no solo avanza rápidamente, sino que también crea demanda de nuevas funciones en el sistema Bitcoin. Muchas innovaciones recientes y planificadas en Bitcoin están motivadas y justificadas por su uso en Lightning Network. De hecho, Lightning Network se menciona a menudo como un ejemplo de caso de uso para muchas de las nuevas funciones.

Protocolo de Bitcoin e innovación de secuencias de comandos de Bitcoin

El sistema Bitcoin es, por necesidad, un sistema conservador que tiene que preservar la compatibilidad con las reglas de consenso para evitar bifurcaciones no planificadas de la cadena de bloques o particiones de la red P2P. Como resultado, las nuevas funciones requieren mucha coordinación y prueba antes de que se implementen en mainnet, el sistema de producción en vivo.

Estas son algunas de las innovaciones actuales o propuestas en Bitcoin que están motivadas por casos de uso en Lightning Network:

neutrino

> Un protocolo de cliente ligero con características de privacidad mejoradas sobre el protocolo SPV heredado. Neutrino es utilizado principalmente por clientes Lightning para acceder a la cadena de bloques de Bitcoin.

firmas Schnorr

> Presentadas como parte de la bifurcación blanda ***Taproot ,*** las firmas de Schnorr permitirán contratos de punto fijo en el tiempo (PTLC) flexibles para la construcción de canales en Lightning. En particular, esto podría hacer uso de firmas revocables en lugar de transacciones revocables.

Raíz principal

> También como parte de la bifurcación suave de noviembre de 2021 que presenta las firmas de Schnorr, Taproot permite que los scripts complejos aparezcan como pagos de un solo pagador y un solo beneficiario, e indistinguibles de los más comunes.

tipo de pago en Bitcoin. Esto permitirá que las transacciones de cierre cooperativo (mutuo) del canal Lightning parezcan indistinguibles de los pagos simples y aumente la privacidad de los usuarios de LN.

Reenlace de entrada

También conocida con los nombres SIGHAS_NOINPUT o SIGHAS_ANYPREVOUT, esta actualización planificada del lenguaje Bitcoin Script está motivada principalmente por contratos inteligentes avanzados como el protocolo de canal de eltoo.

Pactos

Actualmente en las primeras etapas de investigación, los convenios permiten que las transacciones creen productos que restringen las transacciones futuras que los gastan. Este mecanismo podría aumentar la seguridad de los canales Lightning al hacer posible la aplicación de listas blancas de direcciones en las transacciones de compromiso.

Innovación en el protocolo Lightning

El protocolo Lightning P2P es altamente extensible y ha sufrido muchos cambios desde su creación. La regla "Está bien ser impar" utilizada en los bits de función (consulte "Bits de función y extensibilidad del protocolo") garantiza que los nodos puedan negociar las funciones que admiten, lo que permite múltiples actualizaciones independientes del protocolo.

Extensibilidad TLV

El mecanismo Tipo-Longitud-Valor (consulte “Formato Tipo-Longitud-Valor”) para extender el protocolo de mensajería es extremadamente poderoso y ya ha permitido la introducción de varias capacidades nuevas en Lightning mientras mantiene la compatibilidad hacia adelante y hacia atrás. Un ejemplo destacado, que se está desarrollando actualmente y hace uso de esto, son los pagos de cegamiento de caminos y trampolín. Esto permite que un destinatario se oculte del remitente, pero también permite que los clientes móviles envíen pagos sin

la necesidad de almacenar el gráfico de canal completo en sus dispositivos mediante el uso de un tercero al que no necesitan revelar el destinatario final.

Construcción de canales de pago

Los canales de pago son una abstracción operada por dos socios de canal. Mientras esos dos estén dispuestos a ejecutar código nuevo, pueden implementar una variedad de mecanismos de canal simultáneamente. De hecho, investigaciones recientes sugieren que los canales podrían incluso actualizarse dinámicamente a un nuevo mecanismo, sin cerrar el canal antiguo y abrir un nuevo tipo de canal.

eltoo

Un mecanismo de canal propuesto que utiliza el enlace de entrada para simplificar significativamente la operación de los canales de pago y eliminar la necesidad del mecanismo de penalización. Necesita un nuevo tipo de firma de Bitcoin antes de que pueda implementarse

Funciones opcionales de extremo a extremo

Contratos de punto de tiempo bloqueado

Un enfoque diferente a los HTLC, los PTLC pueden aumentar la privacidad, reducir la filtración de información a los nodos intermediarios y operar de manera más eficiente que los canales basados en HTLC.

Grandes canales

Los canales grandes o ***Wumbo*** se introdujeron de forma dinámica a la red sin necesidad de coordinación. Los canales que admiten grandes pagos se anuncian como parte de los mensajes de anuncio del canal y se pueden utilizar de forma opcional.

Pagos multiparte (MPP)

MPP también se introdujo de manera opcional, pero aún mejor solo requiere que el remitente y el destinatario de un pago puedan realizar MPP.

El resto de la red simplemente enruta los HTLC como si fueran pagos de una sola parte.

Enrutamiento JIT

Un método opcional que pueden usar los nodos de enrutamiento para aumentar su confiabilidad y protegerse contra el spam.

Envío de llaves

Una actualización introducida de forma independiente por las implementaciones del cliente Lightning, permite al remitente enviar dinero de forma "no solicitada" y asíncrona sin necesidad de facturar primero.

facturas HODL [1]

Pagos en los que no se cobra la HTLC final, comprometiendo al remitente con el pago, pero permitiendo al destinatario retrasar el cobro hasta que se cumpla alguna otra condición, o cancelar la factura sin cobro.
Esto también fue implementado de forma independiente por diferentes clientes Lightning y se puede usar de manera voluntaria.

Servicios de mensajes enrutados cebolla

El mecanismo de enrutamiento de cebolla y la base de datos de clave pública subyacente de los nodos se pueden usar para enviar datos que no están relacionados con los pagos, como mensajes de texto o publicaciones en foros. El uso de Lightning para habilitar la mensajería paga como una solución a las publicaciones de spam y los ataques de Sybil (spam) es otra innovación que se implementó independientemente del protocolo central.

Ofertas

Actualmente propuesto como BOLT #12 pero ya implementado por algunos nodos, este es un protocolo de comunicación para solicitar facturas (recurrentes) de nodos remotos a través de mensajes Onion.

Aplicaciones Lightning (LApps)

Si bien aún están en su infancia, ya estamos viendo el surgimiento de interesantes aplicaciones Lightning. Definidas ampliamente como una aplicación que utiliza el Protocolo Lightning o un cliente Lightning como componente, las LApps son la capa de aplicación de Lightning. Un ejemplo destacado es LNURL, que proporciona una funcionalidad similar a la que ofrece BOLT #12, pero solo sobre direcciones HTTP y Lightning. Funciona además de las ofertas para proporcionar a los usuarios una dirección de correo electrónico a la que otros pueden enviar fondos mientras el software en segundo plano solicita una factura contra el punto final LNURL del nodo. Se están creando más LAApps para juegos simples, aplicaciones de mensajería, microservicios, API de pago, dispensadores de pago (por ejemplo, bombas de combustible), sistemas de comercio de derivados y mucho más.

¡En sus marcas, listos, fuera!

El futuro se ve brillante. Lightning Network está llevando Bitcoin a nuevos mercados y aplicaciones inexplorados. Equipado con el conocimiento de este libro, puede explorar esta nueva frontera o tal vez incluso unirse como pionero y forjar un nuevo camino.

---

1 La palabra ***HODL*** proviene de una errata emocionada de la palabra “HOLD” gritada en un foro para alentar a las personas a no vender bitcoins en estado de pánico.

Apéndice A. Revisión de los fundamentos de Bitcoin

Lightning Network es capaz de ejecutarse sobre múltiples cadenas de bloques, pero está anclada principalmente en Bitcoin. Para comprender Lightning Network, necesita una comprensión fundamental de Bitcoin y sus componentes básicos.

Hay muchos buenos recursos que puede usar para obtener más información sobre Bitcoin, incluido el libro complementario ***Mastering Bitcoin,*** 2nd Edition, de Andreas M. Antonopoulos, que puede encontrar en GitHub bajo una licencia de código abierto. Sin embargo, ¡no necesita leer otro libro completo para estar listo para este!

En este capítulo, recopilamos los conceptos más importantes que necesita saber sobre Bitcoin y los explicamos en el contexto de Lightning Network. De esta manera, puede aprender exactamente lo que necesita saber para comprender Lightning Network sin distracciones.

Este capítulo cubre varios conceptos importantes de Bitcoin, que incluyen:

- Claves y firmas digitales
- Funciones hash
- Transacciones de Bitcoin y su estructura
- Encadenamiento de transacciones de Bitcoin
- Puntos de salida de transacciones
- Bitcoin Script: bloqueo y desbloqueo de scripts
- Scripts básicos de bloqueo
- Scripts de bloqueo complejos y condicionales
- Bloqueos de tiempo

Claves y Firmas Digitales

Es posible que haya escuchado que bitcoin se basa en ***la criptografía,*** que es una rama de las matemáticas que se usa ampliamente en la seguridad informática. La criptografía también se puede utilizar para probar el conocimiento de un secreto sin revelar ese secreto (firma digital), o probar la autenticidad de los datos (huella digital). Estos tipos de pruebas criptográficas son las herramientas matemáticas fundamentales para Bitcoin y se utilizan ampliamente en las aplicaciones de Bitcoin.

La propiedad de bitcoin se establece a través ***de claves digitales, direcciones de bitcoin*** y ***firmas digitales.*** Las claves digitales en realidad no se almacenan en la red, sino que los usuarios las crean y almacenan en un archivo, o base de datos simple, llamada ***billetera.*** Las claves digitales en la billetera de un usuario son completamente independientes del Protocolo Bitcoin y pueden ser generadas y administradas por el software de la billetera del usuario sin referencia a la cadena de bloques o acceso a Internet.

La mayoría de las transacciones de Bitcoin requieren que se incluya una firma digital válida en la cadena de bloques, que solo se puede generar con una clave secreta; por lo tanto, cualquiera que tenga una copia de esa clave tiene el control del bitcoin. La firma digital utilizada para gastar fondos también se conoce como ***testigo,*** un término utilizado en criptografía. Los datos de los testigos en una transacción de bitcoin dan testimonio de la verdadera propiedad de los fondos que se gastan. Las claves vienen en pares que consisten en una clave privada (secreta) y una clave pública. Piense en la clave pública como similar a un número de cuenta bancaria y la clave privada como similar al PIN secreto.

Claves privadas y públicas

Una clave privada es simplemente un número elegido al azar. En la práctica, y para facilitar la gestión de muchas claves, la mayoría de las carteras de Bitcoin generan una secuencia de claves privadas a partir de una única ***semilla*** aleatoria utilizando un algoritmo de derivación determinista. En pocas palabras, se usa un solo número aleatorio para producir una secuencia repetible de números aparentemente aleatorios que se usan como claves privadas. Esto permite a los usuarios hacer una copia de seguridad solo de la semilla y poder ***derivar*** todas las claves que necesitan de esa semilla.

Bitcoin, como muchas otras criptomonedas y cadenas de bloques, utiliza ***curvas elípticas*** por seguridad. En Bitcoin, la multiplicación de la curva elíptica en la curva elíptica ***secp256k1*** se usa como una ***función unidireccional.*** En pocas palabras, la naturaleza de las matemáticas de la curva elíptica hace que sea trivial calcular la multiplicación escalar de un punto pero imposible calcular la inversa (división o logaritmo discreto).

Cada clave privada tiene una ***clave pública correspondiente,*** que se calcula a partir de la clave privada, utilizando la multiplicación escalar en la curva elíptica. En términos simples, con una clave privada k, podemos multiplicarla con una constante ***G*** para producir una clave pública K:

$$K = k*G$$

Es imposible revertir este cálculo. Dada una clave pública K, no se puede calcular la clave privada k. La división por ***G*** no es posible en matemáticas de curvas elípticas. En cambio, uno tendría que probar todos los valores posibles de ***k*** en un proceso exhaustivo llamado ***ataque de fuerza bruta.*** Debido a que ***k*** es un número de 256 bits, agotar todos los valores posibles con cualquier computadora clásica requeriría más tiempo y energía que los disponibles en este universo.

Hachís

Otra herramienta importante utilizada ampliamente en Bitcoin y en Lightning Network son ***las funciones hash criptográficas,*** y específicamente la función hash SHA-256.

Una función hash, también conocida como ***función de resumen,*** es una función que toma datos de longitud arbitraria y los transforma en un resultado de longitud fija, llamado ***hash, resumen*** o ***huella digital*** (consulte la Figura A-1). Es importante destacar que las funciones hash son funciones ***unidireccionales ,*** lo que significa que no puede revertirlas y calcular los datos de entrada a partir de la huella digital.

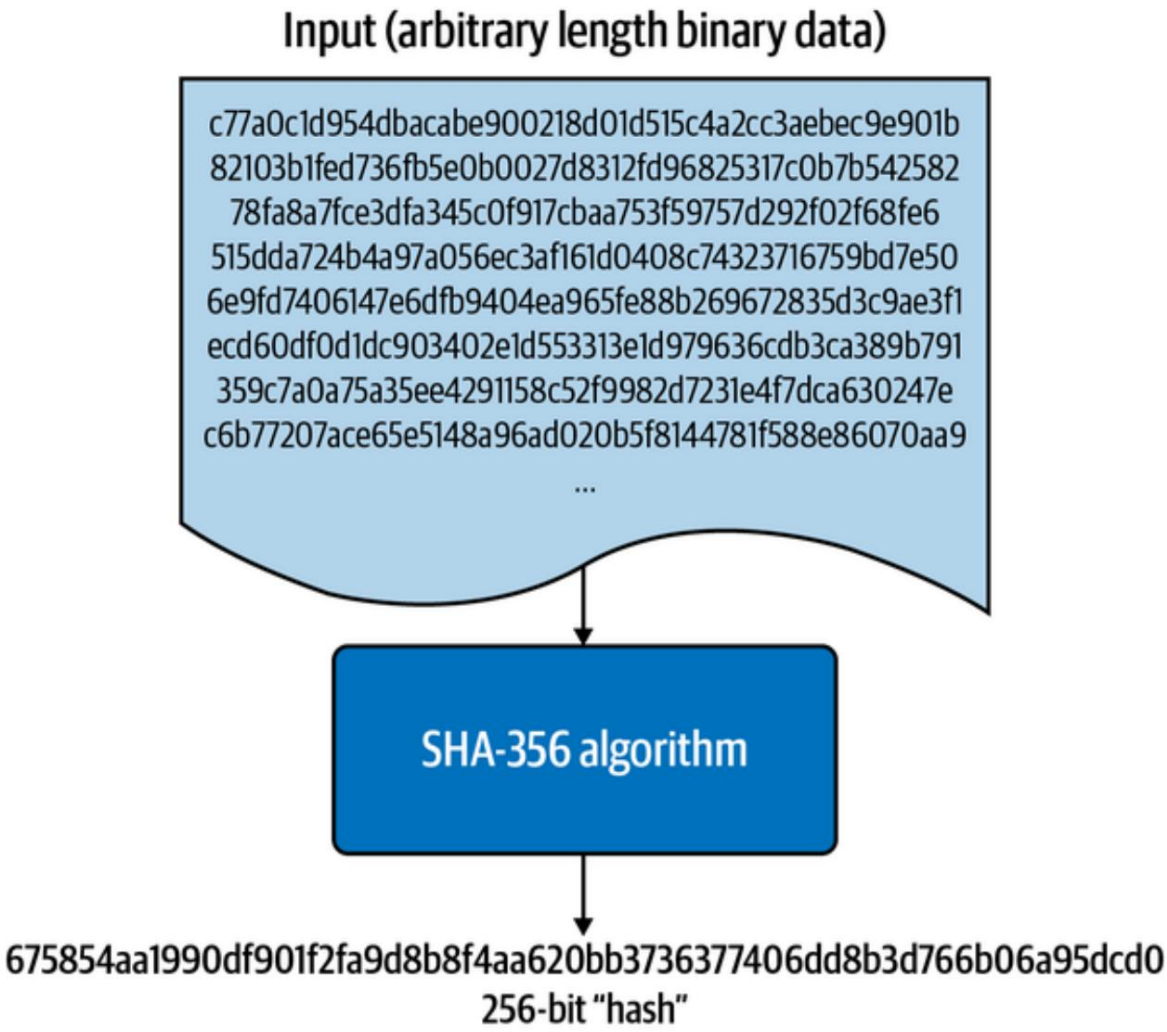

Figura A-1. El algoritmo hash criptográfico SHA-256

Por ejemplo, si usamos un terminal de línea de comandos para ingresar el texto "Dominar la red Lightning" en la función SHA-256, producirá una huella digital de la siguiente manera:

```
$ echo -n "Dominar la red Lightning" | shasum-a 256

ce86e4cd423d80d054b387aca23c02f5fc53b14be4f8d3ef14c089422b2235de
-
```

PROPINA

La entrada utilizada para calcular un hash también se denomina ***preimagen.***

La longitud de la entrada puede ser mucho mayor, por supuesto. Probemos lo mismo con el archivo PDF del documento técnico de Bitcoin de Satoshi Nakamoto:

```
$ wget http://bitcoin.org/bitcoin.pdf $ cat bitcoin.pdf | shasum -a 256
b1674191a88ec5cdd733e4240a81803105dc412d6c6708d53ab94fc248f4f553

-
```

Si bien lleva más tiempo que una sola oración, la función SHA-256 procesa el PDF de 9 páginas y lo "digiere" en una huella digital de 256 bits.

Ahora, en este punto, es posible que se pregunte cómo es posible que una función que digiere datos de tamaño ilimitado produzca una huella digital única que es un número de tamaño fijo.

En teoría, dado que hay un número infinito de posibles preimágenes (entradas) y solo un número finito de huellas dactilares, debe haber muchas preimágenes que produzcan la misma huella dactilar de 256 bits. Cuando dos preimágenes producen el mismo hash, esto se conoce como ***colisión.***

En la práctica, un número de 256 bits es tan grande que nunca encontrará una colisión a propósito. Las funciones hash criptográficas funcionan sobre la base de que la búsqueda de una colisión es un esfuerzo de fuerza bruta que requiere tanta energía y tiempo que no es posible en la práctica.

Las funciones hash criptográficas se usan ampliamente en una variedad de aplicaciones porque tienen algunas características útiles. Están:

determinista

La misma entrada siempre produce el mismo hash.

Irreversible

No es posible calcular la preimagen de un hash.

A prueba de colisiones

No es computacionalmente factible encontrar dos mensajes que tengan el mismo hash.

no correlacionado

Un pequeño cambio en la entrada produce un cambio tan grande en la salida que la salida parece no estar correlacionada con la entrada.

Uniforme/ aleatorio

Una función hash criptográfica produce hashes que se distribuyen uniformemente en todo el espacio de 256 bits de posibles salidas. La salida de un hash parece ser aleatoria, aunque no es realmente aleatoria.

Usando estas características de los hashes criptográficos, podemos construir algunas aplicaciones interesantes:

huellas dactilares

Se puede usar un hash para tomar la huella digital de un archivo o mensaje para que pueda identificarse de manera única. Los hashes se pueden utilizar como identificadores universales de cualquier conjunto de datos.

prueba de integridad

Una huella digital de un archivo o mensaje demuestra su integridad porque el archivo o mensaje no se puede alterar ni modificar de ninguna manera sin cambiar la huella digital. Esto se usa a menudo para asegurarse de que el software no haya sido manipulado antes de instalarlo en su computadora.

Compromiso/ no repudio

Puede comprometerse con una preimagen específica (por ejemplo, un número o mensaje) sin revelarla mediante la publicación de su hash. Más tarde, puede revelar el secreto y todos pueden verificar que es lo mismo con lo que se comprometió anteriormente porque produce el hash publicado.

Prueba de trabajo/ molienda de hash

Puede usar un hash para demostrar que ha realizado un trabajo computacional al mostrar un patrón no aleatorio en el hash que solo se puede producir

por conjeturas repetidas en una preimagen. Por ejemplo, el hash de un encabezado de bloque de Bitcoin comienza con muchos bits cero. La única forma de producirlo es cambiando una parte del encabezado y haciéndolo trillones de veces hasta que produzca ese patrón por casualidad.

Atomicidad

Puede hacer que una preimagen secreta sea un requisito previo para gastar fondos en varias transacciones vinculadas. Si alguna de las partes revela la preimagen para gastar una de las transacciones, todas las demás partes ahora también pueden gastar sus transacciones. Todo o nada se vuelve gastable, logrando la atomicidad en varias transacciones.

Firmas digitales

La clave privada se usa para crear firmas que se requieren para gastar bitcoin al demostrar la propiedad de los fondos utilizados en una transacción.

Una ***firma digital*** es un número que se calcula a partir de la aplicación de la clave privada a un mensaje específico.

Dado un mensaje ***m*** y una clave privada k, una función de firma ***F*** puede producir una firma S: $_{señal}$

$$S = F_{signo}(m, k)$$

Esta firma ***S*** puede ser verificada independientemente por cualquiera que tenga la clave pública ***K*** (correspondiente a la clave privada k), y el mensaje:

$$F_{verificar}(m, K, S)$$

Si ***F*** $_{verificar}$ devuelve un resultado verdadero, entonces el verificador puede confirmar que el mensaje ***m*** fue firmado por alguien que tenía acceso a la clave privada k. Es importante destacar que la firma digital prueba la posesión de la clave privada ***k*** en el momento de la firma, sin revelar k.

Las firmas digitales utilizan un algoritmo hash criptográfico. La firma se aplica a un hash del mensaje, de modo que el mensaje ***m*** se “resume” en un hash de longitud fija ***H(m)*** que sirve como huella digital.

Al aplicar la firma digital en el hash de una transacción, la firma no solo prueba la autorización, sino que también “bloquea” los datos de la transacción, asegurando su integridad. Una transacción firmada no se puede modificar porque cualquier cambio daría como resultado un hash diferente e invalidaría la firma.

Tipos de firma

Las firmas no siempre se aplican a toda la transacción. Para proporcionar flexibilidad de firma, una firma digital de Bitcoin contiene un prefijo llamado tipo de hash de firma, que especifica qué parte de los datos de la transacción se incluye en el hash. Esto permite que la firma confirme o "bloquee" todos, o solo algunos, los datos de la transacción. El tipo de hash de firma más común es SIGHASH_ALL, que bloquea todo en la transacción al incluir todos los datos de la transacción en el hash que se firma. En comparación, SIGHASH_SINGLE bloquea todas las entradas de transacciones, pero solo una salida (más información sobre entradas y salidas en la siguiente sección). Se pueden combinar diferentes tipos de hash de firma para producir seis "patrones" diferentes de datos de transacción que están bloqueados por la firma.

Se puede encontrar más información sobre los tipos de hash de firma en la sección "Tipos de hash de firma" en el Capítulo 6 de ***Mastering Bitcoin,*** Segunda Edición.

Transacciones de bitcoins

Las ***transacciones*** son estructuras de datos que codifican la transferencia de valor entre los participantes en el sistema bitcoin.

Entradas y salidas

El bloque de construcción fundamental de una transacción de bitcoin es una salida de transacción. Los ***resultados de las transacciones*** son fragmentos indivisibles de moneda bitcoin, registrados en la cadena de bloques y reconocidos como válidos por toda la red. Una transacción gasta entradas y crea salidas. Las ***entradas*** de transacciones son simplemente referencias a las salidas de transacciones previamente registradas. De esta manera,

cada transacción gasta los productos de transacciones anteriores y crea nuevos productos (ver Figura A-2).

Figura A-2. Una transacción transfiere valor de las entradas a las salidas

Los nodos completos de Bitcoin rastrean todas las salidas disponibles y gastables, conocidas como ***salidas de transacciones no*** gastadas (UTXO). La colección de todos los UTXO se conoce como el conjunto de UTXO, que actualmente cuenta con millones de UTXO. El conjunto de UTXO crece a medida que se crean nuevos UTXO y se reduce cuando se consumen UTXO. Cada transacción representa un cambio (transición de estado) en el conjunto de UTXO, al consumir uno o más UTXO como ***entradas de transacción*** y crear uno o más UTXO como ***salida de transacción.***

Por ejemplo, supongamos que una usuaria Alice tiene un UTXO de 100 000 satoshi que puede gastar. Alice puede pagarle a Bob 100 000 satoshi construyendo una transacción con una entrada (consumiendo su entrada existente de 100 000 satoshi) y una salida que “paga” a Bob 100 000 satoshi. Ahora Bob tiene un UTXO de 100ÿ000 satoshi que puede gastar, creando una nueva transacción que consume este nuevo UTXO y lo gasta en otro UTXO como pago a otro usuario, y así sucesivamente (consulte la Figura A-3).

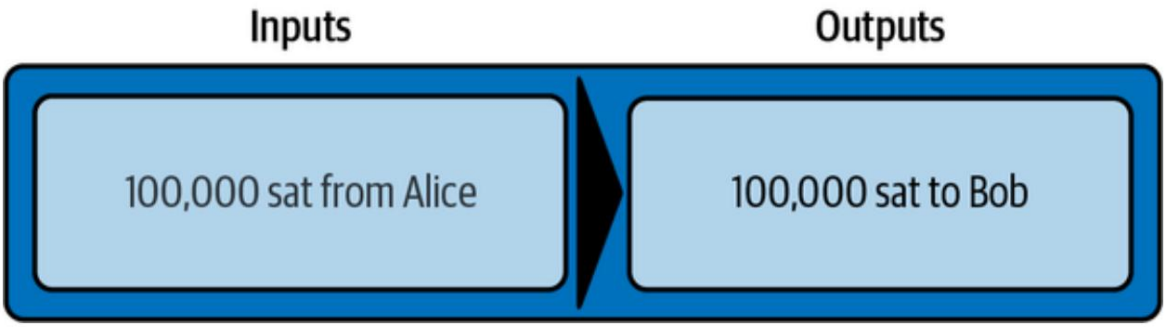

Figura A-3. Alice paga 100.000 satoshis a Bob

La salida de una transacción puede tener un valor arbitrario (entero) denominado en satoshis. Así como los dólares se pueden dividir hasta dos decimales como centavos, el bitcoin se puede dividir hasta ocho decimales como satoshis. Aunque una salida puede tener cualquier valor arbitrario, una vez creada es indivisible. Esta es una característica importante de los productos que debe enfatizarse: los productos son unidades de valor discretas e indivisibles, denominadas en satoshis enteros.
Un producto no gastado solo puede ser consumido en su totalidad por una transacción.

Entonces, ¿qué pasa si Alice quiere pagarle a Bob 50,000 satoshi, pero solo tiene un UTXO indivisible de 100,000 satoshi? Alice necesitará crear una transacción que consuma (como entrada) los 100 000 satoshi UTXO y tenga dos salidas: una que pague 50 000 satoshi a Bob y otra que pague 50 000 satoshi ***a*** Alice como “cambio” (consulte la Figura A-4).

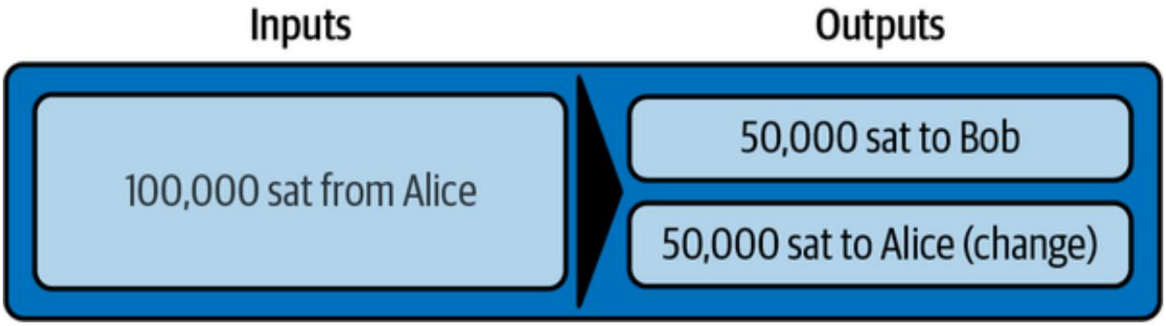

Figura A-4. Alice paga 50k sat a Bob y 50k sat a ella misma como cambio

PROPINA

No hay nada especial en una salida de cambio ni ninguna forma de distinguirla de cualquier otra salida. No tiene que ser la última salida. Podría haber más de una salida de cambio, o ninguna salida de cambio. Solo el creador de la transacción sabe qué salidas son para otros y qué salidas son para direcciones propias y, por lo tanto, "cambian".

De manera similar, si Alice quiere pagarle a Bob 85 000 satoshi pero tiene dos UTXO de 50 000 satoshi disponibles, debe crear una transacción con dos entradas (consumiendo sus 50 000 UTXO de satoshi) y dos salidas, pagando a Bob 85 000 y enviándose 15 000 satoshi de vuelta a sí misma como cambiar (ver Figura A 5).

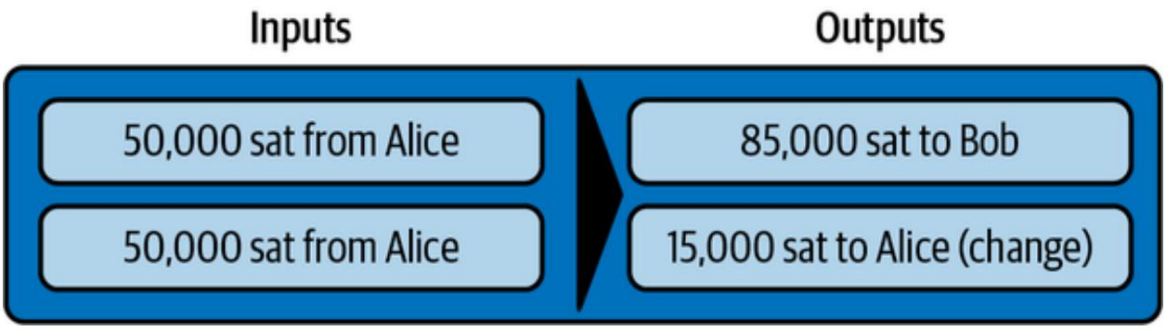

Figura A-5. Alice usa dos entradas de 50k para pagar 85k sat a Bob y 15k sat a ella misma como cambio

Las ilustraciones y ejemplos anteriores muestran cómo una transacción de Bitcoin combina (gasta) una o más entradas y crea una o más salidas. Una transacción puede tener cientos o incluso miles de entradas y salidas.

PROPINA

Si bien las transacciones creadas por Lightning Network tienen múltiples salidas, no tienen "cambio" per se, porque todo el saldo disponible de un canal se divide entre los dos socios de canal.

Cadenas de transacciones

Cada salida se puede gastar como entrada en una transacción posterior. Entonces, por ejemplo, si Bob decide gastar 10 000 satoshi en una transacción para pagarle a Chan, y Chan gasta 4000 satoshi para pagarle a Dina, se desarrollaría como se muestra en la Figura A-6.

Una salida se considera ***gastada*** si se hace referencia a ella como entrada en otra transacción registrada en la cadena de bloques. Una salida se considera ***no gastada*** (y disponible para gastar) si ninguna transacción registrada hace referencia a ella.

El único tipo de transacción que no tiene entradas es una transacción especial creada por los mineros de Bitcoin llamada ***transacción coinbase.*** La transacción de coinbase solo tiene salidas y no entradas porque crea nuevos bitcoins a partir de la minería. Cualquier otra transacción gasta una o más salidas registradas previamente como sus entradas.

Dado que las transacciones están encadenadas, si elige una transacción al azar, puede seguir cualquiera de sus entradas hacia atrás hasta la transacción anterior que la creó. Si continúa haciendo eso, eventualmente llegará a una transacción de base de monedas donde se extrajo el bitcoin por primera vez.

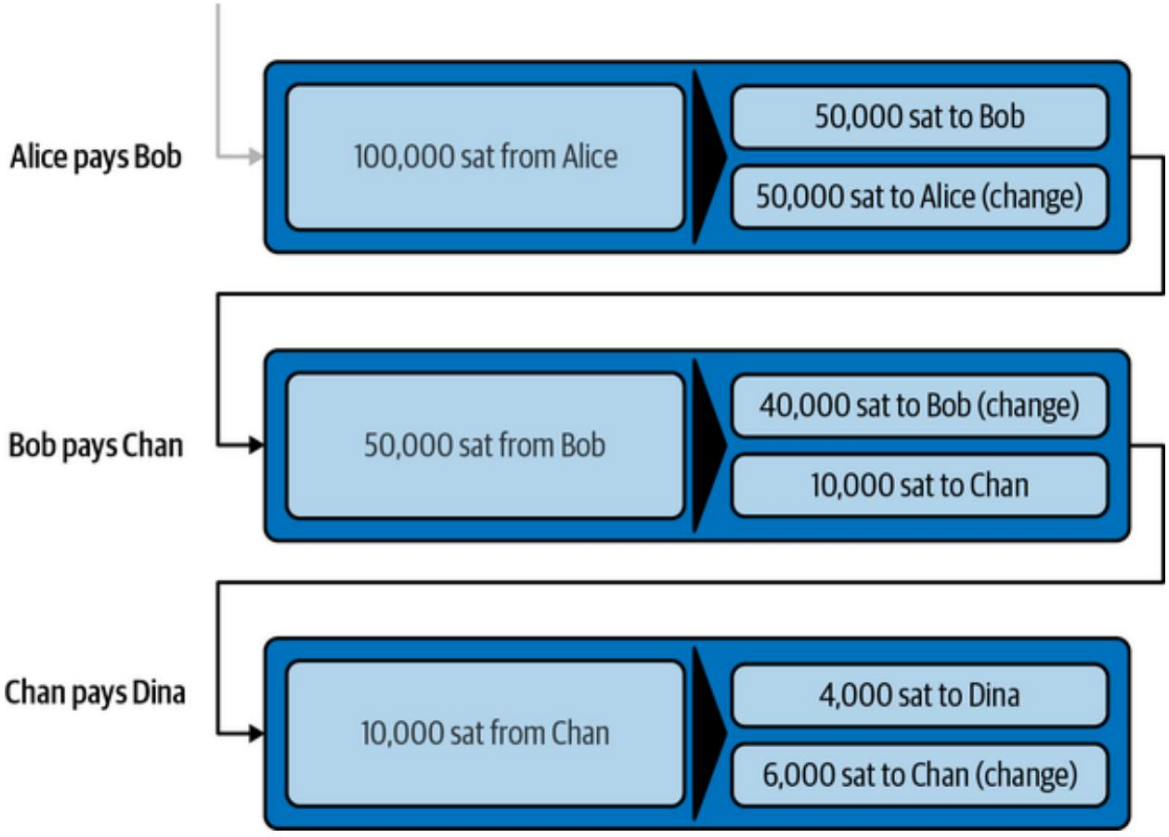

Figura A-6. Alice le paga a Bob quien le paga a Chan quien le paga a Dina

TxID: Identificadores de transacciones

Cada transacción en el sistema Bitcoin se identifica mediante un identificador único (asumiendo la existencia de BIP-0030), llamado ***ID de transacción*** o ***TxID*** para abreviar. Para producir un identificador único, usamos la función hash criptográfica SHA-256 para producir un hash de los datos de la transacción. Esta “huella digital” sirve como un identificador universal. Se puede hacer referencia a una transacción por su ID de transacción, y una vez que se registra una transacción en la cadena de bloques de Bitcoin, todos los nodos de la red de Bitcoin saben que esta transacción es válida.

Por ejemplo, un ID de transacción podría verse así:

e31e4e214c3f436937c74b8663b3ca58f7ad5b3fce7783eb84fd9a5ee5b9a54c

Esta es una transacción real (creada como ejemplo para el libro ***Mastering Bitcoin)*** que se puede encontrar en la cadena de bloques de Bitcoin. Intente encontrarlo ingresando este TxID en un explorador de bloques:

https:// blockstream.info/ tx/ e31e4e214c3f436937c74b8663b3ca58f7 ad5b3fce7783eb84fd9a5ee5b9a54c

o use el enlace corto (distingue entre mayúsculas y minúsculas):

http:// bit.ly/ AliceTx

Puntos de salida: identificadores de salida

Debido a que cada transacción tiene una identificación única, también podemos identificar la salida de una transacción dentro de esa transacción únicamente por referencia al TxID y el número de índice de salida. La primera salida de una transacción es el índice de salida 0, la segunda salida es el índice de salida 1, y así sucesivamente. Un identificador de salida se conoce comúnmente como un punto de salida .

Por convención, escribimos un punto de salida como TxID, dos puntos y el número de índice de salida:

7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18:
0

Los identificadores de salida (puntos de salida) son los mecanismos que vinculan las transacciones en una cadena. Cada entrada de transacción es una referencia a una salida específica de una transacción anterior. Esa referencia es un punto de salida: un TxID y un número de índice de salida. Entonces, una transacción "gasta" una salida específica (por número de índice) de una transacción específica (por TxID) para crear nuevas salidas que se pueden gastar por referencia al punto de salida.

La Figura A-7 presenta la cadena de transacciones de Alice a Bob a Chan a Dina, esta vez con puntos de salida en cada una de las entradas.

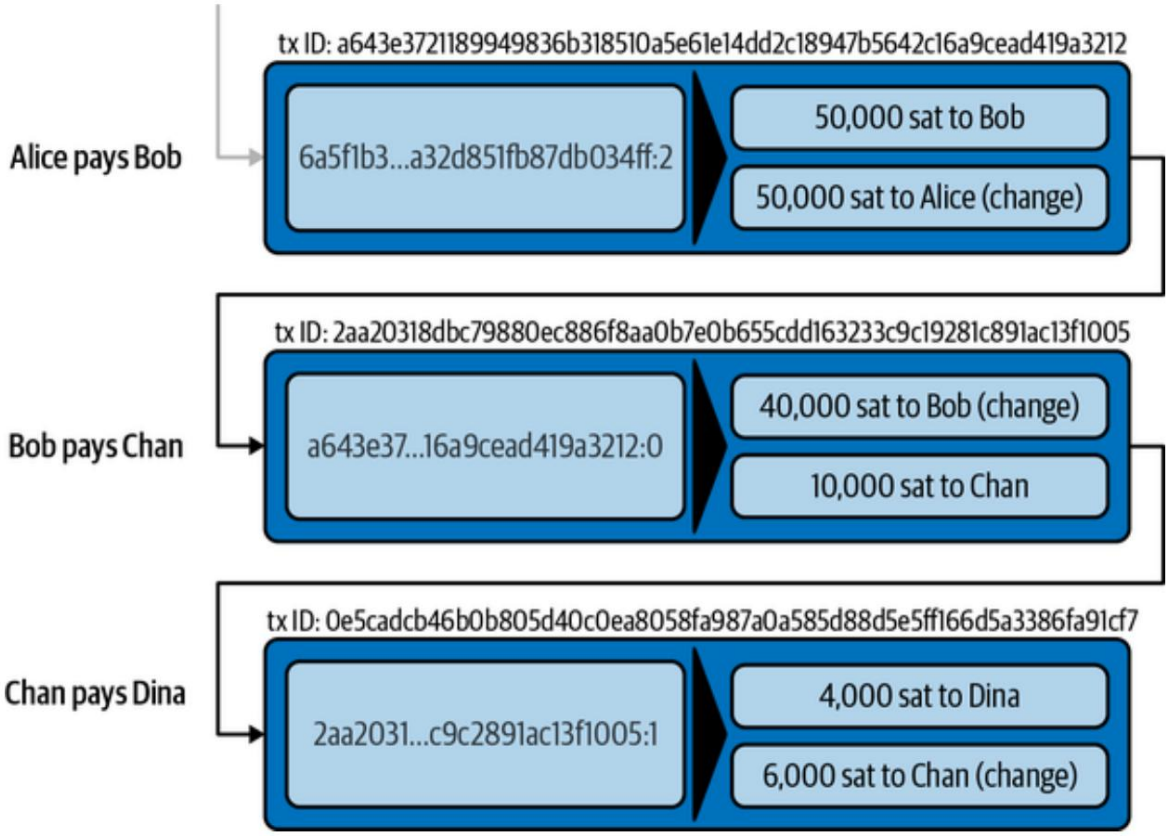

Figura A-7. Las entradas de transacción se refieren a puntos de salida que forman una cadena

La entrada en la transacción de Bob hace referencia a la transacción de Alice (por TxID) y la salida indexada 0.

La entrada en la transacción de Chan hace referencia al TxID de la transacción de Bob y la primera salida indexada, porque el pago a Chan es la salida #1. En el pago de Bob a Chan, el cambio de Bob es la salida #0.

Ahora, si observamos el pago de Alice a Bob, podemos ver que Alice está gastando un punto de salida que fue la tercera salida (índice de salida #2) en una transacción cuyo ID es 6a5f1b3[…]. No vemos esa transacción a la que se hace referencia en el diagrama, pero podemos deducir estos detalles desde el punto de partida.

Guión de Bitcoin

El elemento final de Bitcoin que se necesita para completar nuestra comprensión es el lenguaje de secuencias de comandos que controla el acceso a los puntos de salida. Hasta ahora, hemos

simplificó la descripción diciendo "Alice firma la transacción para pagarle a Bob". Detrás de escena, sin embargo, hay cierta complejidad oculta que hace posible implementar condiciones de gasto más complejas. La condición de gasto más simple y común es "presentar una firma que coincida con la siguiente clave pública". Una condición de gasto como esta se registra en cada salida como secuencia de ***comandos de bloqueo*** escrita en un lenguaje de secuencias de comandos llamado ***Bitcoin Script.***

Bitcoin Script es un lenguaje de secuencias de comandos basado en pilas extremadamente simple. No contiene bucles ni recursividad y, por lo tanto, ***Turing es incompleto*** (lo que significa que no puede expresar una complejidad arbitraria y tiene una ejecución predecible). Quienes estén familiarizados con el (ahora antiguo) lenguaje de programación FORTH reconocerán la sintaxis y el estilo.

Ejecutar secuencia de comandos de Bitcoin

En términos simples, el sistema Bitcoin evalúa Bitcoin Script ejecutando el script en una pila; si el resultado final es VERDADERO, considera cumplida la condición de gasto y válida la transacción.

Veamos un ejemplo muy simple de Bitcoin Script, que suma los números 2 y 3 y luego compara el resultado con el número 5:

```
2 3 SUMAR 5 IGUAL
```

En la Figura A-8, vemos cómo se ejecuta este script (de izquierda a derecha).

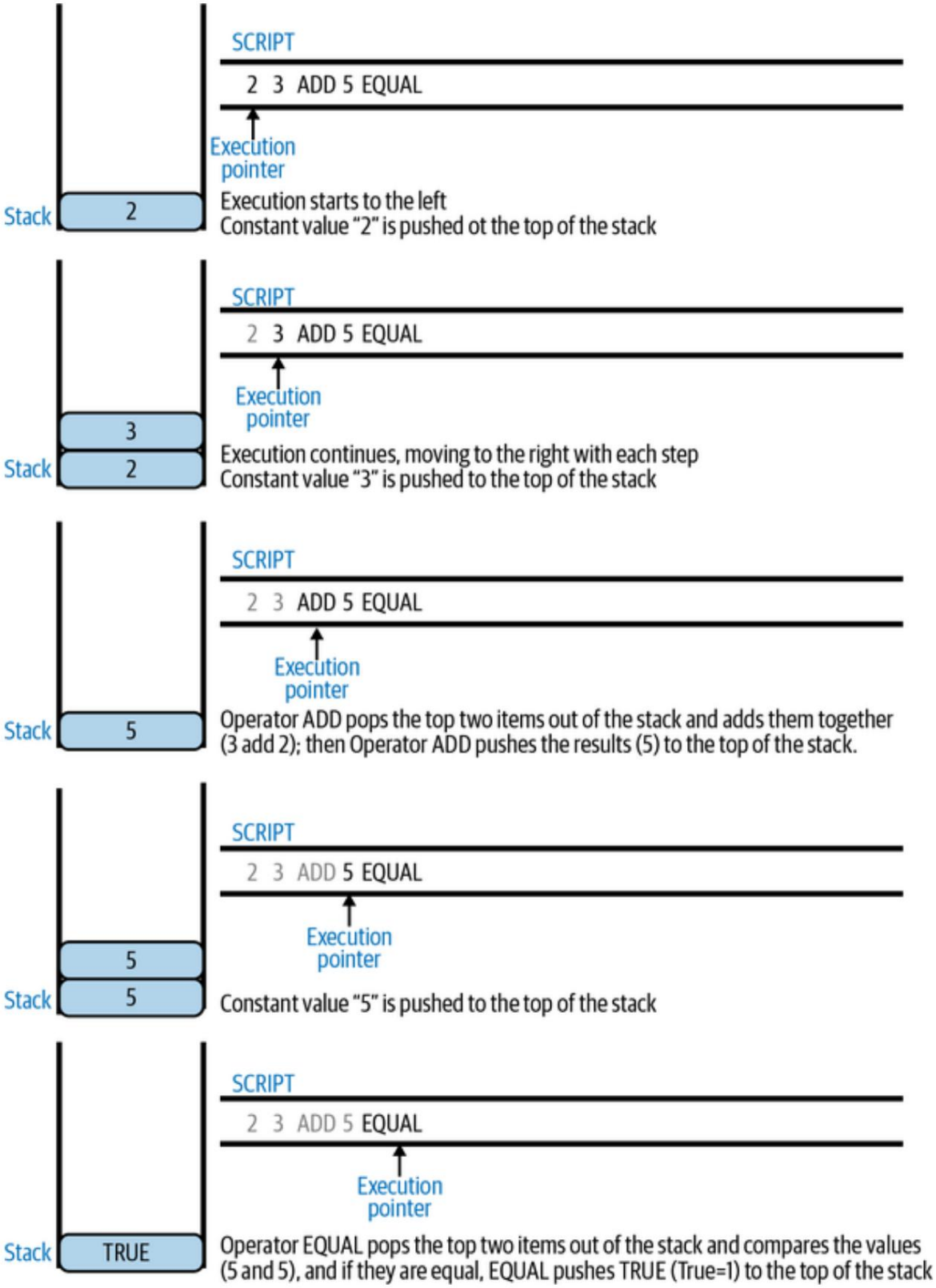

Figura A-8. Ejemplo de ejecución de Bitcoin Script

Bloqueo y desbloqueo de scripts

Bitcoin Script se compone de dos partes:

Bloqueo de secuencias de comandos

> Estos están integrados en los resultados de las transacciones, estableciendo las condiciones que deben cumplirse para gastar ese resultado. Por ejemplo, la billetera de Alice agrega un script de bloqueo a la salida que paga a Bob, que establece la condición de que se requiere la firma de Bob para gastarlo.

Desbloqueo de guiones

> Estos están integrados en las entradas de transacciones, cumpliendo las condiciones establecidas por el script de bloqueo de la salida a la que se hace referencia. Por ejemplo, Bob puede desbloquear la salida anterior proporcionando un script de desbloqueo que contenga una firma digital.

Usando un modelo simplificado, para la validación, el script de desbloqueo y el script de bloqueo se concatenan y ejecutan (P2SH y SegWit son excepciones).
Por ejemplo, si alguien bloqueó la salida de una transacción con el script de bloqueo "3 SUMA 5 IGUAL", podríamos gastarlo con el script de desbloqueo "2" en una entrada de transacción. Cualquiera que valide esa transacción concatenaría nuestro script de desbloqueo (2) y el script de bloqueo (3 ADD 5 EQUAL) y ejecutaría el resultado a través del motor de ejecución de Bitcoin Script. Se volverían VERDADEROS y podríamos gastar la salida.

Obviamente, este ejemplo simplificado sería una elección muy mala para bloquear una salida real de Bitcoin porque no hay ningún secreto, solo aritmética básica. Cualquiera podría gastar la salida proporcionando la respuesta "2".
Por lo tanto, la mayoría de los scripts de bloqueo requieren que se demuestre el conocimiento de un secreto.

Bloqueo a una clave pública (firma)

La forma más simple de una secuencia de comandos de bloqueo es la que requiere una firma.
Consideremos la transacción de Alice que le paga a Bob 50,000 satoshis. La salida

Alice crea para pagar a Bob tendrá un script de bloqueo que requerirá la firma de Bob y se vería así:

```
<Clave pública de Bob> CHECKSIG
```

El operador CHECKSIG toma dos elementos de la pila: una firma y una clave pública. Como puede ver, la clave pública de Bob está en el script de bloqueo, por lo que lo que falta es la firma correspondiente a esa clave pública. Solo Bob puede gastar este script de bloqueo, porque solo Bob tiene la clave privada correspondiente necesaria para producir una firma digital que coincida con la clave pública.

Para desbloquear este script de bloqueo, Bob proporcionaría un script de desbloqueo que solo contenga su firma digital:

```
<Firma de Bob>
```

En la Figura A-9 se puede ver el script de bloqueo en la transacción de Alice (en la salida que paga a Bob) y el script de desbloqueo (en la entrada que gasta esa salida) en la transacción de Bob.

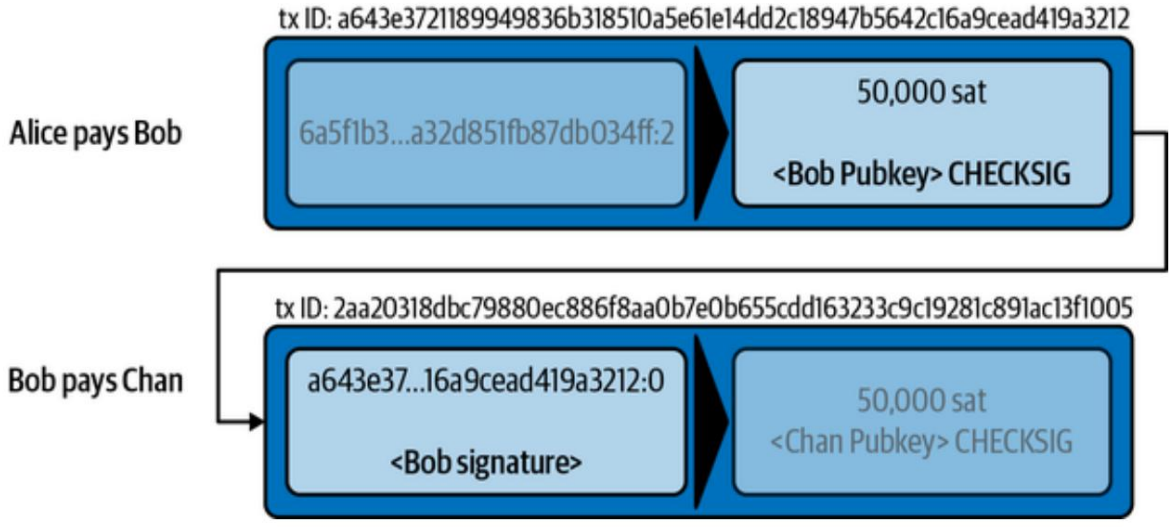

Figura A-9. Una cadena de transacciones que muestra el script de bloqueo (salida) y el script de desbloqueo (entrada)

Para validar la transacción de Bob, un nodo de Bitcoin haría lo siguiente:

1. Extraiga el script de desbloqueo de la entrada (<Firma de Bob>).

2. Busque el punto de salida que está intentando gastar (a643e37...
   3213:0). Esta es la transacción de Alice y se encontraría en la cadena de bloques.

3. Extraiga el script de bloqueo desde ese punto de salida (<Bob PubKey>
   CHECKSIG).

4. Concatene en una secuencia de comandos, colocando la secuencia de comandos de desbloqueo delante de la secuencia de comandos de bloqueo (<Firma de Bob> <Clave de publicación de Bob> CHECKSIG).

5. Ejecute este script en el motor de ejecución de Bitcoin Script para ver
   qué resultado se produce.

6. Si el resultado es VERDADERO, deduce que la transacción de Bob es válida
   porque fue capaz de cumplir la condición de gasto para gastar ese outpoint.

Bloqueo a un hash (secreto)

Otro tipo de script de bloqueo, uno que se usa en Lightning Network, es un ***hashlock.*** Para desbloquearlo, debes conocer la ***preimagen*** secreta del hash.

Para demostrar esto, hagamos que Bob genere un número aleatorio R y manténgalo secreto:

R = 1833462189

Ahora, Bob calcula el hash SHA-256 de este número:

H = SHA256(R) =>
H = SHA256 (1833462189) =>
H =
0ffd8bea4abdb0sordo6f2a8ad7941c13256a19248a7b0612407379e1460036a

Ahora, Bob le da el hash H que calculamos previamente a Alice, pero mantiene el número R en secreto. Recuerde que debido a las propiedades de los hash criptográficos, Alice no puede "revertir" el cálculo del hash y adivinar el número R.

Alice crea una salida pagando 50 000 satoshi con el script de bloqueo:

```
HASH256 H IGUAL
```

donde H es el valor hash real (0ffd8…036a) que Bob le dio a Alice.

Expliquemos este script:

El operador HASH256 extrae un valor de la pila y calcula el hash SHA-256 de ese valor. Luego empuja el resultado a la pila.

El valor H se coloca en la pila y luego el operador IGUAL verifica si los dos valores son iguales y coloca VERDADERO o FALSO en la pila según corresponda.

Por lo tanto, este script de bloqueo solo funcionará si se combina con un script de desbloqueo que contenga R, de modo que al concatenar, tenemos:

```
R HASH256 H IGUAL
```

Solo Bob conoce R, por lo que solo Bob puede producir una transacción con un script de desbloqueo que revele el valor secreto R.

Curiosamente, Bob puede dar el valor R a cualquier otra persona, quien luego puede gastar ese Bitcoin. Esto hace que el valor secreto R sea casi como un "vale" de bitcoin, ya que cualquiera que lo tenga puede gastar la salida que creó Alice. ¡Veremos cómo esta es una propiedad útil para Lightning Network!

Scripts multifirma

El lenguaje Bitcoin Script proporciona un bloque de construcción de múltiples firmas (primitivo), que se puede usar para crear servicios de depósito en garantía y configuraciones de propiedad complejas entre varias partes interesadas. Un acuerdo

que requiere múltiples firmas para gastar Bitcoin se denomina ***esquema de múltiples firmas,*** más especificado como esquema ***K-de-N ,*** donde:

- ***N*** es el número total de firmantes identificados en el esquema multifirma, y
- ***K*** es el ***quórum*** o ***umbral:*** el número mínimo de firmas para autorizar el gasto.

El script para una multifirma K-de-N es:

```
K <PubKey1> <PubKey2> ... <PubKeyN> N CHECKMULTISIG
```

donde ***N*** es el número total de claves públicas enumeradas (Clave pública 1 a Clave pública N) y ***K*** es el umbral de firmas requeridas para gastar la salida.

Lightning Network utiliza un esquema de firmas múltiples 2 de 2 para crear un canal de pago. Por ejemplo, un canal de pago entre Alice y Bob se crearía en una firma múltiple 2 de 2 como esta:

```
2 <PubKey Alice> <PubKey Bob> 2 CHECKMULTISIG
```

El script de bloqueo anterior puede satisfacerse con un script de desbloqueo que contenga un par de firmas: [2]

```
0 <Di Alicia> <Di Bob>
```

Los dos scripts juntos formarían el script de validación combinado:

```
0 <Diga Alice> <Diga Bob> 2 <PubKey Alice> <PubKey Bob> 2
COMPROBAR MULTISIG
```

Una secuencia de comandos de bloqueo de múltiples firmas puede representarse mediante una dirección de Bitcoin, codificando el hash de la secuencia de comandos de bloqueo. Por ejemplo, la transacción de financiación inicial de un canal de pago Lightning es una transacción que paga a un

dirección que codifica un script de bloqueo de un multigrado 2 de 2 de los dos socios de canal.

Guiones de bloqueo de tiempo

Otro bloque de construcción importante que existe en Bitcoin y se usa ampliamente en Lightning Network es un ***bloqueo de tiempo.*** Un bloqueo de tiempo es una restricción de gasto que requiere que haya transcurrido cierto tiempo o altura de bloque antes de que se permita el gasto. Es un poco como un cheque posfechado extraído de una cuenta bancaria que no se puede cobrar antes de la fecha del cheque.

Bitcoin tiene dos niveles de bloqueos de tiempo: bloqueos de tiempo de nivel de transacción y bloqueos de tiempo de nivel de salida.

Se registra ***un bloqueo de tiempo a nivel de transacción*** en la transacción nLockTime de la transacción y evita que se acepte toda la transacción antes de que haya pasado el bloqueo de tiempo. Los bloqueos de tiempo a nivel de transacción son el mecanismo de bloqueo de tiempo más utilizado en Bitcoin en la actualidad.

Un operador de secuencia de comandos crea un ***bloqueo de tiempo de nivel de salida .*** Hay dos tipos de bloqueos de tiempo de salida: bloqueos de tiempo absolutos y bloqueos de tiempo relativos.

Los bloqueos de ***tiempo absolutos*** de nivel de salida son implementados por el operador CHECKLOCKTIMEVERIFY, que a menudo se abrevia en una conversación como ***CLTV.*** Los bloqueos de tiempo absolutos implementan una restricción de tiempo con una marca de tiempo absoluta o altura de bloque, expresando el equivalente de "no gastable antes del bloque 800,000".

Los bloqueos de ***tiempo relativos*** de nivel de salida son implementados por el operador CHECKSEQUENCEVERIFY, a menudo abreviado en una conversación como ***CSV.***
Los bloqueos de tiempo relativos implementan una restricción de gasto relativa a la confirmación de la transacción, expresando el equivalente a "no se puede gastar hasta 1024 bloques después de la confirmación".

Scripts con Múltiples Condiciones

Una de las características más poderosas de Bitcoin Script es el control de flujo, también conocido como cláusulas condicionales. Probablemente esté familiarizado con el control de flujo

en varios lenguajes de programación que usan la construcción IF…THEN…ELSE.
Las cláusulas condicionales de Bitcoin se ven un poco diferentes, pero son esencialmente la misma construcción.

En un nivel básico, los códigos de operación condicionales de bitcoin nos permiten construir un script de bloqueo que tiene dos formas de desbloquearse, dependiendo del resultado VERDADERO/FALSO de evaluar una condición lógica. Por ejemplo, si x es VERDADERO, el script de bloqueo es A DE LO CONTRARIO, el script de bloqueo es B.

Además, las expresiones condicionales de bitcoin se pueden ***anidar*** indefinidamente, lo que significa que una cláusula condicional puede contener otra dentro de ella, que contiene otra, etc. El control de flujo de Bitcoin Script se puede usar para construir scripts muy complejos con cientos o incluso miles de posibles rutas de ejecución. No hay límite para anidar, pero las reglas de consenso imponen un límite en el tamaño máximo, en bytes, de un script.

Bitcoin implementa el control de flujo utilizando los códigos de operación IF, ELSE, ENDIF y NOTIF. Además, las expresiones condicionales pueden contener operadores booleanos como BOOLAND, BOOLOR y NOT.

A primera vista, puede encontrar confusos los scripts de control de flujo de Bitcoin. Eso es porque Bitcoin Script es un lenguaje de pila. De la misma manera que la operación aritmética 1 + 1 se ve "al revés" cuando se expresa en Bitcoin Script como 1 1 ADD, las cláusulas de control de flujo en Bitcoin también se ven "al revés".

En la mayoría de los lenguajes de programación tradicionales (de procedimiento), el control de flujo se ve así:

```
if (condición): código para
    ejecutar cuando la condición es verdadera
más:
    código para ejecutar cuando la condición es falsa
código para ejecutar en cualquier caso
```

En un lenguaje basado en pilas como Bitcoin Script, la condición lógica viene ***antes*** del IF, lo que hace que parezca "al revés", así:

```
condición
SI
```

```
    código para ejecutar cuando la condición es verdadera
MÁS
    código para ejecutar cuando la condición es falsa
TERMINARA SI
código para ejecutar en cualquier caso
```

Al leer Bitcoin Script, recuerde que la condición que se evalúa viene ***antes*** del código de operación IF.

Uso del control de flujo en secuencias de comandos

Un uso muy común para el control de flujo en Bitcoin Script es construir un script de bloqueo que ofrezca múltiples rutas de ejecución, cada una con una forma diferente de canjear el UTXO.

Veamos un ejemplo simple, donde tenemos dos firmantes, Alice y Bob, y cualquiera de ellos puede canjear. Con multisig, esto se expresaría como un script multisig 1 de 2. En aras de la demostración, haremos lo mismo con una cláusula IF:

```
SI
  <Clave pública de Alice> CHECKSIG
MÁS
  <Pubkey de Bob> CHECKSIG
TERMINARA SI
```

Al mirar este script de bloqueo, es posible que se pregunte: “¿Dónde está la condición? ¡No hay nada que preceda a la cláusula IF!

La condición no forma parte del script de bloqueo. En cambio, la condición se ***ofrecerá en el script de desbloqueo,*** lo que permitirá a Alice y Bob "elegir" qué ruta de ejecución desean.

Alice canjea esto con el script de desbloqueo:

```
<Dicho de Alicia> 1
```

El 1 al final sirve como la condición (VERDADERO) que hará que la cláusula IF ejecute la primera ruta de redención para la que Alice tiene una firma.

Para que Bob canjee esto, tendría que elegir la segunda ruta de ejecución dando un valor FALSO a la cláusula IF:

```
<Dicho de Bob> 0
```

La secuencia de comandos de desbloqueo de Bob pone un 0 en la pila, lo que hace que la cláusula IF ejecute la segunda secuencia de comandos (ELSE), que requiere la firma de Bob.

Debido a que cada una de las dos condiciones también requiere una firma, Alice no puede usar la segunda cláusula y Bob no puede usar la primera cláusula; ¡No tienen las firmas necesarias para eso!

Dado que los flujos condicionales se pueden anidar, también se pueden anidar los valores VERDADERO/FALSO en el script de desbloqueo, para navegar por una ruta compleja de condiciones.

En el Ejemplo A-1 , puede ver un ejemplo del tipo de script complejo que se usa en Lightning Network, con múltiples condiciones. Los scripts utilizados en Lightning Network están altamente optimizados y son compactos para minimizar la huella en la cadena, por lo que no son fáciles de leer y comprender. Sin embargo, vea si puede identificar algunos de los conceptos de Bitcoin Script que aprendimos en este capítulo.

Ejemplo A-1. Un script complejo utilizado en Lightning Network

```
# Al nodo remoto con clave de revocación
DUP HASH160 <RIPEMD160(SHA256(revocaciónpubkey))> IGUAL
SI
        CHECKSIG
MÁS
        <remote_htlcpubkey> INTERCAMBIAR TAMAÑO 32 IGUAL
        NOTIFICACIONES
                # Al nodo local a través de la transacción de tiempo de espera de HTLC (con tiempo de espera).
                DROP 2 SWAP <local_htlcpubkey> 2 CHECKMULTISIG ELSE

                # Al nodo remoto con preimagen.
                HASH160 <RIPEMD160(pago_hash)> EQUALVERIFY
                CHECKSIG
        TERMINARA SI
TERMINARA SI
```

1 Recuerde que el cambio no tiene que ser el último resultado de una transacción y, de hecho, es indistinguible de otros resultados.

2 El primer argumento (0) no tiene ningún significado, pero es necesario debido a un error en la implementación de firmas múltiples de Bitcoin. Este problema se describe en ***Mastering Bitcoin,*** Capítulo 7.

3 Desde el PERNO #3.

Apéndice B. Instalación y uso básicos de Docker

Este libro contiene una serie de ejemplos que se ejecutan dentro de los contenedores de Docker para la estandarización en diferentes sistemas operativos.

Esta sección lo ayudará a instalar Docker y a familiarizarse con algunos de los comandos de Docker más utilizados, para que pueda ejecutar los contenedores de ejemplo del libro.

Instalación de Docker

Antes de comenzar, debe instalar el sistema de contenedores Docker en su computadora. Docker es un sistema abierto que se distribuye de forma gratuita como ***Community Edition*** para muchos sistemas operativos diferentes, incluidos Windows, macOS y Linux. Las versiones de Windows y Macintosh se denominan ***Docker Desktop*** y consisten en una aplicación de escritorio GUI y herramientas de línea de comandos. La versión de Linux se llama ***Docker Engine*** y se compone de un demonio de servidor y herramientas de línea de comandos. Usaremos las herramientas de línea de comandos, que son idénticas en todas las plataformas.

Continúe e instale Docker para su sistema operativo siguiendo las instrucciones para "Obtener Docker" del sitio web de Docker.

Seleccione su sistema operativo de la lista y siga las instrucciones de instalación.

PROPINA

Si realiza la instalación en Linux, siga las instrucciones posteriores a la instalación para asegurarse de que puede ejecutar Docker como un usuario normal en lugar del usuario root. De lo contrario, deberá prefijar todos los comandos docker con sudo, ejecutándolos como root como: sudo docker.

Una vez que haya instalado Docker, puede probar su instalación ejecutando el contenedor de demostración hello-world de esta manera:

```
$ ventana acoplable ejecutar hola-mundo

¡Hola desde Docker!
Este mensaje muestra que su instalación parece estar funcionando correctamente.

[...]
```

Comandos básicos de Docker

En este apéndice, usamos Docker bastante extensamente. Usaremos los siguientes comandos y argumentos de Docker.

Construyendo un Contenedor

```
docker build [-t etiqueta] [directorio]
```

la etiqueta es cómo identificamos el contenedor que estamos construyendo, y el *directorio* es donde se encuentran el contexto del contenedor (carpetas y archivos) y el archivo de definición (Dockerfile).

Ejecutar un contenedor

```
docker run -it [--network netname] [ --name cname] etiqueta
```

netname es el nombre de una red Docker, *cname* es el nombre que elegimos para esta instancia de contenedor y *tag* es la etiqueta de nombre que le dimos al contenedor cuando lo construimos.

Ejecutar un comando en un contenedor

```
Comando docker exec cname
```

cname es el nombre que le dimos al contenedor en el comando de ejecución, y *command* es un ejecutable o script que queremos ejecutar dentro del contenedor.

Detener e iniciar un contenedor

En la mayoría de los casos, si estamos ejecutando un contenedor en un modo ***interactivo*** además de ***terminal ,*** es decir, con los indicadores i y t (combinados como -it) configurados, el contenedor se puede detener simplemente presionando Ctrl-C o saliendo del shell con salir o Ctrl-D. Si un contenedor no termina, puede detenerlo desde otra terminal como esta:

```
ventana acoplable detener cname
```

Para reanudar un contenedor ya existente, use el comando de inicio así:

```
nombre de inicio de la ventana acoplable
```

Eliminación de un contenedor por nombre

Si asigna un nombre a un contenedor en lugar de dejar que Docker lo asigne aleatoriamente, no podrá reutilizar ese nombre hasta que se elimine el contenedor. Docker devolverá un error como este:

```
ventana acoplable: Respuesta de error del demonio: Conflicto. El nombre del contenedor "/bitcoind"
ya está en uso...
```

Para solucionar esto, elimine la instancia existente del contenedor:

```
docker rm cname
```

cname es el nombre asignado al contenedor (bitcoind en el mensaje de error de ejemplo).

Listado de contenedores en ejecución

```
ventana acoplable pd
```

Este comando muestra los contenedores en ejecución actuales y sus nombres.

Listado de imágenes de Docker

```
imagen acoplable ls
```

Este comando muestra las imágenes de Docker que se han creado o descargado en su computadora.

Conclusión

Estos comandos básicos de Docker serán suficientes para comenzar y le permitirán ejecutar todos los ejemplos de este libro.

Apéndice C. Mensajes de protocolo de cable

Este apéndice enumera todos los tipos de mensajes definidos actualmente que se utilizan en el protocolo Lightning P2P. Además, mostramos la estructura de cada mensaje, agrupando los mensajes en agrupaciones lógicas basadas en los flujos del protocolo.

> **NOTA**
>
> Los mensajes de Lightning Protocol son extensibles y su estructura puede cambiar durante las actualizaciones de toda la red. Para obtener información autorizada, consulte la última versión de los BOLT que se encuentran en el repositorio GitHub Lightning-RFC.

Tipos de mensajes

Los tipos de mensajes definidos actualmente se enumeran en la Tabla C-1.

T
a
b
yo
y

C
-
1
.
METRO
y
s
s
a
gramo
y

t
y
pags
y
s

Escriba entero	Nombre del mensaje Categoría	
dieciséis	calor	Establecimiento de conexión
17	error	Comunicación de errores
18	silbido	Conexión vivacidad
19	apestar	Conexión vivacidad
32	canal abierto	Financiamiento del canal

33	aceptar_canal	Financiamiento del canal
34	financiación_creada	Financiamiento del canal
35	financiación_firmado	Financiamiento del canal
36	financiación_bloqueada	Canal Financiación + Canal Operación
38	cerrar	Cierre de canal
39	cierre_firmado	Cierre de canal
128	actualizar_añadir_htlc	Operación del canal
130	actualizar_cumplir_hltc	Operación del canal
131	update_fail_htlc	Operación del canal
132	cometer_sig	Operación del canal
133	revocar_y_ack	Operación del canal
134	actualizar_tarifa	Operación del canal
135	update_fail_malformed_htlc	Operación del canal
136	canal_restablecer	Operación del canal
256	canal_anuncio	Anuncio de canal
257	anuncio_nodo	Anuncio de canal
258	canal_actualizar	Anuncio de canal
259	anunciar_firmas	Anuncio de canal
261	query_short_chan_ids	Sincronización de gráfico de canal
262	answer_short_chan_ids_end	Sincronización de gráfico de canal
263	rango_canal_consulta	Sincronización de gráfico de canal

264	rango_de_canales_de_respuesta	Sincronización de gráfico de canal
265	chismes_timestamp_range	Sincronización de gráfico de canal

En la Tabla 13-1, el campo Categoría nos permite categorizar rápidamente un mensaje según su funcionalidad dentro del propio protocolo. En un nivel alto, colocamos un mensaje en uno de los ocho cubos (no exhaustivos) que incluyen:

Establecimiento de conexión

Enviado cuando se establece por primera vez una conexión punto a punto. También se utiliza para negociar el conjunto de funciones admitidas por una nueva conexión.

Comunicación de errores

Utilizado por pares para comunicar la ocurrencia de errores de nivel de protocolo entre sí.

Conexión vivacidad

Utilizado por pares para verificar que una conexión de transporte dada todavía está activa.

Financiamiento del canal

Utilizado por pares para crear un nuevo canal de pago. Este proceso también se conoce como el proceso de financiación del canal.

Operación del canal

El acto de actualizar un canal dado fuera de la cadena. Esto incluye enviar y recibir pagos, así como reenviar pagos dentro de la red.

Anuncio de canal

El proceso de anunciar un nuevo canal público a la red más amplia para que pueda usarse con fines de enrutamiento.

Sincronización de gráfico de canal

El proceso de descarga y verificación del gráfico de canales.

Observe cómo los mensajes que pertenecen a la misma categoría suelen compartir también un ***tipo de mensaje*** adyacente . Esto se hace a propósito para agrupar mensajes semánticamente similares dentro de la propia especificación.

Estructura del mensaje

Ahora detallamos cada categoría de mensaje para definir la estructura y la semántica precisas de todos los mensajes definidos dentro del protocolo LN.

Mensajes de establecimiento de conexión

Los mensajes de esta categoría son los primeros mensajes que se envían entre pares una vez que establecen una conexión de transporte. Al momento de escribir este capítulo, existe un solo mensaje dentro de esta categoría, el mensaje de inicio. El mensaje de inicio es enviado por ***ambos*** lados de la conexión una vez que se ha establecido por primera vez. No se deben enviar otros mensajes antes de que ambas partes hayan enviado el mensaje inicial.

El mensaje de inicio

La estructura del mensaje init se define de la siguiente manera:

- Tipo: 16
- Campos:
  - uint16: global_features_len
  - características_globales_len*byte: características_globales
  - uint16: características_len
  - características_len*byte: características
  - tlv_stream_tlvs

Estructuralmente, el mensaje de inicio se compone de dos segmentos de bytes de tamaño variable, cada uno de los cuales almacena un conjunto de ***bits de características.*** Como vemos en "Bits de características y extensibilidad del protocolo", los bits de características son una primitiva utilizada dentro del protocolo para anunciar el conjunto de características del protocolo que un nodo comprende (características opcionales) o exige (características requeridas).

Tenga en cuenta que las implementaciones de nodos modernos solo usarán el campo de funciones, con elementos que residen dentro del vector global_features principalmente con fines ***históricos*** (compatibilidad con versiones anteriores).

Lo que sigue después del mensaje principal es una serie de registros Tipo-Longitud-Valor (TLV) que se pueden usar para extender el mensaje de manera compatible hacia adelante y hacia atrás en el futuro. Cubriremos qué son los registros TLV y cómo se usan más adelante en este apéndice.

Luego, un par examina un mensaje de inicio para determinar si la conexión está bien definida en función del conjunto de bits de características opcionales y requeridos anunciados por ambos lados.

Una función opcional significa que un compañero conoce una función, pero no la considera fundamental para el funcionamiento de una nueva conexión. Un ejemplo de uno sería algo así como la capacidad de comprender la semántica de un campo recién agregado a un mensaje existente.

Por otro lado, las funciones requeridas indican que si el otro par no conoce la función, entonces la conexión no está bien definida. Un ejemplo de tal función sería un nuevo tipo de canal teórico dentro del protocolo: si su compañero no conoce esta función, entonces no desea mantener la conexión porque no pueden abrir su nuevo tipo de canal preferido. .

Mensajes de comunicación de errores

Los mensajes de esta categoría se utilizan para enviar errores de nivel de conexión entre dos pares. Existe otro tipo de error en el protocolo: un error de nivel de reenvío HTLC. Los errores de nivel de conexión pueden señalar cosas como el bit de función

incompatibilidad o la intención de ***forzar el cierre*** (difundir unilateralmente el último compromiso firmado).

el mensaje de error

El único mensaje en esta categoría es el mensaje de error.

- Tipo: 17
- Campos:
    - id_canal : id_canal
    - uint16: data_len
    - data_len*byte: datos

Se puede enviar un mensaje de error dentro del alcance de un canal en particular configurando channel_id al channel_id del canal que experimenta este nuevo estado de error. Alternativamente, si el error se aplica a la conexión en general, entonces el campo channel_id debe establecerse en ceros. Este channel_id todo cero también se conoce como el identificador de nivel de conexión para un error.

Dependiendo de la naturaleza del error, enviar un mensaje de error a un compañero con el que tiene un canal puede indicar que el canal no puede continuar sin una intervención manual, por lo que la única opción en ese momento es forzar el cierre del canal transmitiendo el último estado de compromiso. del canal

Conexión vivacidad

Los mensajes de esta sección se utilizan para sondear y determinar si una conexión aún está activa o no. Debido a que el protocolo LN se abstrae un poco del transporte subyacente que se utiliza para transmitir los mensajes, se define un conjunto de mensajes ping y pong a nivel de protocolo.

El mensaje de ping

El mensaje de ping se usa para verificar si la otra parte en una conexión está "en vivo". Contiene los siguientes campos:

- Tipo: 18
- Campos:
    - uint16: num_pong_bytes
    - uint16: ping_body_len
    - ping_body_len*bytes: ping_body

A continuación su compañero, el mensaje pong.

el mensaje del pong

El mensaje pong se envía en respuesta al mensaje ping y contiene los siguientes campos:

- Tipo: 19
- Campos:
    - uint16: pong_body_len
    - ping_body_len*bytes: pong_body

Cualquiera de las partes puede enviar un mensaje de ping en cualquier momento.

El mensaje ping incluye un campo num_pong_bytes que se utiliza para indicar al nodo receptor el tamaño de la carga útil que envía en su mensaje pong. El mensaje de ping también incluye un conjunto opaco de bytes ping_body que se puede ignorar de forma segura. Solo sirve para permitir que un remitente complete los mensajes de ping que envía, lo que puede ser útil para intentar frustrar ciertas técnicas de anonimización basadas en el tamaño de los paquetes en el cable.

Se debe enviar un mensaje pong en respuesta a un mensaje ping recibido. El receptor debe leer un conjunto de bytes aleatorios num_pong_bytes para enviar

atrás como el campo pong_body. El uso inteligente de estos campos/mensajes puede permitir que un nodo de enrutamiento consciente de la privacidad intente frustrar ciertas clases de intentos de anonimización de la red porque pueden crear una transcripción "falsa" que se parece a otros mensajes en función del tamaño de los paquetes enviados. Recuerde que, de manera predeterminada, Lightning Network usa un transporte ***encriptado ,*** por lo que un monitor de red pasivo no puede leer los bytes de texto sin formato y, por lo tanto, solo tiene tiempos y tamaños de paquetes para desactivar.

Financiamiento del canal

A medida que avanzamos, entramos en el territorio de los mensajes centrales que rigen la funcionalidad y la semántica del Protocolo Lightning. En esta sección, exploramos los mensajes enviados durante el proceso de creación de un nuevo canal.
Solo describiremos los campos utilizados, ya que dejamos un análisis en profundidad del proceso de financiación para el Capítulo 7.

Los mensajes que se envían durante el flujo de financiación del canal pertenecen al siguiente conjunto de cinco mensajes: canal_abierto, canal_aceptado, financiación_creada, financiación_firmada y financiación_bloqueada.

El flujo de protocolo detallado que utiliza estos mensajes se describe en el Capítulo 7.

El mensaje de canal_abierto

El mensaje open_channel inicia el proceso de financiación del canal y contiene los siguientes campos:

- Tipo: 32
- Campos:
    - cadena_hash : cadena_hash
    - 32*byte: temp_chan_id
    - uint64: financiación_satoshis
    - uint64: empujar_msat

- uint64: polvo_limit_satoshis
- uint64: max_htlc_value_in_flight_msat
- uint64 : channel_reserve_satoshis
- uint64: htlc_minimum_msat
- uint32 : feerate_per_kw
- uint16 : to_self_delay
- uint16: max_accepted_htlcs
- clave pública: financiación_clave pública
- clave pública: revocación_punto base
- clave pública: punto_base_pago
- clave pública: punto_base_pago_retrasado
- clave pública: htlc_basepoint
- clave pública: first_per_commitment_point
- byte: banderas_canal
- tlv_stream: tlvs

Este es el primer mensaje que se envía cuando un nodo desea ejecutar un nuevo flujo de financiación con otro nodo. Este mensaje contiene toda la información necesaria para que ambos pares construyan tanto la transacción de financiación como la transacción de compromiso.

Al momento de escribir este capítulo, se define un solo registro TLV dentro del conjunto de registros TLV opcionales que se pueden agregar al final de un registro definido. mensaje:

- Tipo: 0
- Datos: upfront_shutdown_script

upfront_shutdown_script es un segmento de bytes de tamaño variable que debe ser un script de clave pública válido aceptado por el algoritmo de consenso de la red Bitcoin. Al proporcionar dicha dirección, la parte que envía puede crear efectivamente un "bucle cerrado" para su canal, ya que ninguna de las partes firmará una transacción de cierre cooperativo que pague a cualquier otra dirección. En la práctica, esta dirección suele ser una derivada de una billetera de almacenamiento en frío.

El campo channel_flags es un campo de bits del cual, en el momento de escribir este artículo, solo el ***primer*** bit tiene algún tipo de significado. Si este bit está establecido, este canal se anunciará a la red pública como un canal enrutable. De lo contrario, el canal se considera no anunciado, también conocido como canal privado.

El mensaje accept_channel

El mensaje accept_channel es la respuesta al open_channel mensaje.

- Tipo: 33
- Campos:
    - 32*byte: temp_chan_id
    - uint64: polvo_limit_satoshis
    - uint64: max_htlc_value_in_flight_msat
    - uint64 : channel_reserve_satoshis
    - uint64: htlc_minimum_msat
    - uint32: profundidad_mínima
    - uint16 : to_self_delay
    - uint16: max_accepted_htlcs
    - clave pública: financiación_clave pública

    - clave pública: revocación_punto base
    - clave pública: punto_base_pago
    - clave pública: punto_base_pago_retrasado
    - clave pública: htlc_basepoint
    - clave pública: first_per_commitment_point
    - tlv_stream: tlvs

El mensaje accept_channel es el segundo mensaje enviado durante el proceso de flujo de financiación. Sirve para reconocer una intención de abrir un canal con un nuevo par remoto. El mensaje refleja principalmente el conjunto de parámetros que el respondedor desea aplicar a su versión de la transacción de compromiso. En el Capítulo 7, cuando entramos en detalle en el proceso de financiación, exploramos las implicaciones de los diversos parámetros que se pueden establecer al abrir un nuevo canal.

El mensaje financiado_creado

En respuesta, el iniciador enviará el mensaje financiado_creado.

- Tipo: 34
- Campos:
    - 32*byte: temp_chan_id
    - 32*byte: financiación_txid
    - uint16 : índice_de_salida_de_fondos
    - decir: commit_sig

Una vez que el iniciador de un canal recibe el mensaje accept_channel del respondedor, tiene todos los materiales que necesita para construir la transacción de compromiso, así como la transacción de financiación. Como los canales por defecto son financiadores únicos (solo un lado compromete fondos), solo el iniciador

necesita para construir la transacción de financiación. Como resultado, para permitir que el respondedor firme una versión de una transacción de compromiso para el iniciador, el iniciador solo necesita enviar el punto de salida de financiación del canal.

El mensaje financiado_firmado

Para concluir, el respondedor envía el mensaje financiado_firmado.

- Tipo: 34
- Campos:
    - id_canal : id_canal
    - decir: firma

Para concluir, después de que el respondedor reciba el mensaje de creación de fondos, ahora posee una firma válida de la transacción de compromiso por parte del iniciador. Con esta firma, pueden salir del canal en cualquier momento firmando su mitad de la salida de financiación multisig y transmitiendo la transacción. Esto se conoce como un cierre forzado. Por el contrario, para dar al iniciador la capacidad de cerrar el canal, el respondedor también firma la transacción de compromiso del iniciador.

Una vez que el iniciador ha recibido este mensaje, es seguro para ellos transmitir la transacción de financiación porque ahora pueden salir del acuerdo de canal unilateralmente.

El mensaje de financiación_bloqueada

Una vez que la transacción de financiación ha recibido suficientes confirmaciones, se envía el mensaje de financiación_bloqueada.

- Tipo: 36
- Campos:
    - id_canal : id_canal
    - clave pública: next_per_commitment_point

Una vez que la transacción de financiación obtiene un número mínimo de confirmaciones, ambas partes deben enviar el mensaje de financiación bloqueada. Solo después de que este mensaje haya sido recibido y enviado, el canal puede comenzar a utilizarse.

Cierre de canal

El cierre del canal es un proceso de varios pasos. Un nodo se inicia enviando el mensaje de apagado. Luego, los dos socios de canal intercambian una serie de mensajes de cierre_firmados para negociar tarifas mutuamente aceptables para la transacción de cierre. El financiador del canal envía el primer mensaje de cierre_firmado, y el otro lado puede aceptar enviando un mensaje de cierre_firmado con los mismos valores de tarifa.

El mensaje de apagado

El mensaje de apagado inicia el proceso de cierre de un canal y contiene los siguientes campos:

- Tipo: 38
- Campos:
    - id_canal : id_canal
    - u16: solo
    - len*byte: scriptpubkey

El mensaje de cierre_firmado El

mensaje de cierre_firmado lo envía cada socio de canal hasta que acuerden las tarifas. Contiene los siguientes campos:

- Tipo: 39
- Campos:
    - id_canal : id_canal

- u64: tarifa_satoshis
- firma : firma

Operación del canal

En esta sección, describimos brevemente el conjunto de mensajes utilizados para permitir que los nodos operen un canal. Por operación, nos referimos a poder enviar, recibir y reenviar pagos por un canal determinado.

Para enviar, recibir o reenviar un pago a través de un canal, primero se debe agregar un HTLC a ambas transacciones de compromiso que comprenden un enlace de canal.

El mensaje update_add_htlc

El mensaje update_add_htlc permite que cualquiera de los lados agregue un nuevo HTLC a la transacción de compromiso opuesta.

- Tipo: 128
- Campos:
  - id_canal : id_canal
  - uint64: identificación
  - uint64: cantidad_msat
  - sha256: pago_hash
  - uint32: cltv_expiry
  - 1366*byte: cebolla_enrutamiento_paquete

El envío de este mensaje permite que una de las partes inicie el envío de un nuevo pago o el reenvío de un pago existente que llegó a través de un canal entrante. El mensaje especifica la cantidad (amount_msat) junto con el hash de pago que desbloquea el pago en sí. El conjunto de instrucciones de reenvío del siguiente salto se cifran en cebolla dentro del campo cebolla_enrutamiento_paquete. En el Capítulo 10, sobre HTLC multisalto

reenvío, cubrimos en detalle el protocolo de enrutamiento de cebolla utilizado en Lightning Network.

Tenga en cuenta que cada HTLC enviado utiliza una ID que se incrementa automáticamente y que se utiliza en cualquier mensaje que modifique un HTLC (liquidar o cancelar) para hacer referencia al HTLC de una manera única en el ámbito del canal.

El mensaje update_fulfill_hltc

El mensaje update_fulfill_hltc permite el canje (recepción) de un HTLC activo.

- Tipo: 130
- Campos:
  - id_canal : id_canal
  - uint64: identificación
  - 32*byte: pago_preimagen

El receptor del HTLC envía este mensaje al proponente para canjear un HTLC activo. El mensaje hace referencia a la identificación del HTLC en cuestión y también proporciona la preimagen (que desbloquea el HLTC).

El mensaje update_fail_htlc

El mensaje update_fail_htlc se envía para eliminar un HTLC de una transacción de compromiso.

- Tipo: 131
- Campos:
  - id_canal : id_canal
  - uint64: identificación
  - uint16: solo

- len*byte: motivo

El mensaje update_fail_htlc es lo opuesto al mensaje update_fulfill_hltc en el sentido de que permite que el receptor de un HTLC elimine el mismo HTLC. Este mensaje generalmente se envía cuando un HTLC no se puede enrutar correctamente en sentido ascendente y debe devolverse al remitente para desentrañar la cadena HTLC. Como exploramos en "Mensajes de falla", el mensaje contiene un motivo de falla ***encriptado*** (motivo) que puede permitir al remitente ajustar su ruta de pago o cancelar si la falla en sí es una terminal.

El mensaje compromiso_firmado

El mensaje de compromiso_firmado se utiliza para marcar la creación de una nueva transacción de compromiso.

- Tipo: 132
- Campos:
  - id_canal : id_canal
  - decir: firma
  - uint16: num_htlcs
  - num_htlcs*sig : htlc_firma

Además de enviar una firma para la siguiente transacción de compromiso, el remitente de este mensaje también debe enviar una firma para cada HTLC presente en la transacción de compromiso.

El mensaje revocar y acusar recibo

El revoke_and_ack se envía para revocar un compromiso fechado.

- Tipo: 133
- Campos:

- id_canal : id_canal
- 32*byte: por_compromiso_secreto
- clave pública: next_per_commitment_point

Debido a que Lightning Network usa una transacción de compromiso de reemplazo por revocación, después de recibir una nueva transacción de compromiso a través del mensaje commit_sig, una parte debe revocar su compromiso anterior antes de poder recibir otro. Al revocar una transacción de compromiso, el revocador también proporciona el siguiente punto de compromiso que se requiere para permitir que la otra parte le envíe un nuevo estado de compromiso.

El mensaje update_fee

El update_fee se envía para actualizar la tarifa en las transacciones de compromiso actuales.

- Tipo: 134
- Campos:
  - id_canal : id_canal
  - uint32 : feerate_per_kw

Este mensaje solo puede ser enviado por el iniciador del canal; ellos son los que pagarán la comisión de compromiso del canal mientras esté abierto.

El mensaje update_fail_malformed_htlc

El mensaje update_fail_malformed_htlc se envía para eliminar un HTLC corrupto.

- Tipo: 135
- Campos:
  - id_canal : id_canal

        - uint64: identificación
        - sha256 : sha256_de_cebolla
        - uint16: código_fallo

Este mensaje es similar al mensaje update_fail_htlc, pero rara vez se usa en la práctica. Como se mencionó anteriormente, cada HTLC transporta un paquete de enrutamiento encriptado tipo cebolla que también cubre la integridad de partes del propio HTLC. Si una parte recibe un paquete de cebolla que de alguna manera se corrompió en el camino, entonces no podrá descifrar el paquete. Como resultado, tampoco puede reenviar correctamente el HTLC; por lo tanto, enviará este mensaje para indicar que el HTLC se ha dañado en algún lugar a lo largo de la ruta de regreso al remitente.

Anuncio de canal

Los mensajes de esta categoría se utilizan para anunciar componentes de la estructura de datos autenticados del gráfico de canales a la red más amplia. El gráfico de canal tiene una serie de propiedades únicas debido a la condición de que todos los datos agregados al gráfico de canal también deben estar anclados en la cadena de bloques base de Bitcoin. Como resultado, para agregar una nueva entrada al gráfico de canal, un agente debe tener una tarifa de transacción en cadena. Esto sirve como elemento disuasorio de spam natural para Lightning Network.

El mensaje de anuncio de canal

El mensaje channel_announcement se utiliza para anunciar un nuevo canal a la red más amplia.

- Tipo: 256
- Campos:
    - decir: node_signature_1
    - decir: node_signature_2
    - firma: bitcoin_signature_1

- firma: bitcoin_signature_2
- uint16: solo
- len*byte: características
- cadena_hash : cadena_hash
- id_canal_corto : id_canal_corto
- clave pública: node_id_1
- clave pública: node_id_2
- clave pública: bitcoin_key_1
- clave pública: bitcoin_key_2

La serie de firmas y claves públicas en el mensaje sirve para crear una ***prueba de*** que el canal realmente existe dentro de la cadena de bloques base de Bitcoin. Como detallamos en "El ID de canal corto", cada canal se identifica de forma única mediante un localizador que codifica su ***ubicación*** dentro de la cadena de bloques. Este localizador se llama short_channel_id y puede caber en un número entero de 64 bits.

El mensaje de anuncio de nodo

El mensaje node_announcement permite que un nodo anuncie/actualice su vértice dentro del gráfico de canal mayor.

- Tipo: 257
- Campos:
  - decir: firma
  - uint64 : flen
  - flen*byte: características
  - uint32: marca de tiempo
  - clave pública: node_id

- 3*bytes: rgb_color
- 32*byte: alias
- uint16: dirección
- addrlen*byte: direcciones

Tenga en cuenta que si un nodo no tiene ningún canal anunciado dentro del gráfico de canales, este mensaje se ignora para garantizar que agregar un elemento al gráfico de canales tenga un costo en la cadena. En este caso, el costo en cadena será el costo de crear el canal al que se conecta este nodo.

Además de anunciar su conjunto de características, este mensaje también permite que un nodo anuncie/actualice el conjunto de direcciones de red a las que se puede acceder.

El mensaje channel_update

El mensaje channel_update se envía para actualizar las propiedades y políticas de un borde de canal activo dentro del gráfico de canal.

- Tipo: 258
- Campos:
  - firma : firma
  - cadena_hash : cadena_hash
  - id_canal_corto : id_canal_corto
  - uint32: marca de tiempo
  - byte: banderas_mensaje
  - byte: banderas_canal
  - uint16: cltv_expiry_delta
  - uint64: htlc_minimum_msat
  - uint32 : fee_base_msat

- uint32 : fee_proportional_millionths
- uint16: htlc_maximum_msat

Además de poder habilitar/deshabilitar un canal, este mensaje permite que un nodo actualice sus tarifas de enrutamiento, así como otros campos que dan forma al tipo de pago que se permite fluir a través de este canal.

El mensaje de las firmas de anuncios

El mensaje de anuncio_firmas es intercambiado por los pares de canal para ensamblar el conjunto de firmas requeridas para producir un mensaje de anuncio de canal.

- Tipo: 259
- Campos:
  - id_canal : id_canal
  - id_canal_corto : id_canal_corto
  - decir: node_signature
  - sig: firma_bitcoin

Después de que se haya enviado el mensaje de financiación_bloqueada, si ambas partes desean anunciar su canal en la red, cada una de ellas enviará el mensaje de anuncio_firmas que permite a ambas partes colocar las cuatro firmas necesarias para generar un anuncio_firmas.
mensaje.

Sincronización de gráfico de canal

Los nodos crean una perspectiva local del gráfico del canal utilizando cinco mensajes: query_short_chan_ids, answer_short_chan_ids_end, query_channel_range, answer_channel_range y gossip_timestamp_range.

El mensaje query_short_chan_ids

El mensaje query_short_chan_ids permite que un par obtenga la información del canal relacionada con una serie de ID de canales cortos.

- Tipo: 261
- Campos:
  - cadena_hash : cadena_hash
  - u16: solo
  - len*byte: codificados_cortos_ids
  - query_short_channel_ids_tlvs: tlvs

Como aprendimos en el Capítulo 11, estos ID de canal pueden ser una serie de canales que eran nuevos para el remitente o que estaban desactualizados, lo que permite al remitente obtener el último conjunto de información para un conjunto de canales.

El mensaje answer_short_chan_ids_end

El mensaje answer_short_chan_ids_end se envía después de que un par termine de responder a un mensaje anterior query_short_chan_ids.

- Tipo: 262
- Campos:
  - cadena_hash : cadena_hash
  - byte: información_completa

Este mensaje le indica a la parte receptora que si desea enviar otro mensaje de consulta, ahora puede hacerlo.

El mensaje query_channel_range

El mensaje query_channel_range permite que un nodo consulte el conjunto de canales abiertos dentro de un rango de bloques.

- Tipo: 263
- Campos:
    - cadena_hash : cadena_hash
    - u32 : primer_número_de_bloque
    - u32 : numero_de_bloques
    - consulta_canal_rango_tlvs: tlvs

Como los canales se representan mediante un ID de canal corto que codifica la ubicación de un canal en la cadena, un nodo en la red puede usar una altura de bloque como una especie de ***cursor*** para buscar a través de la cadena y descubrir un conjunto de canales recién abiertos. .

El mensaje answer_channel_range

El mensaje answer_channel_range es la respuesta al mensaje query_channel_range e incluye el conjunto de ID de canales cortos para canales conocidos dentro de ese rango.

- Tipo: 264
- Campos:
    - cadena_hash : cadena_hash
    - u32 : primer_número_de_bloque
    - u32 : numero_de_bloques
    - byte: sincronización_completa
    - u16: solo
    - len*byte: codificados_cortos_ids
    - respuesta_canal_rango_tlvs: tlvs

Como respuesta a query_channel_range, este mensaje devuelve el conjunto de canales que se abrieron dentro de ese rango. Este proceso se puede repetir con el solicitante avanzando el cursor más abajo en la cadena para continuar sincronizando el gráfico del canal.

El mensaje gossip_timestamp_range El mensaje

gossip_timestamp_range permite que un compañero comience a recibir nuevos mensajes de chismes entrantes en la red.

- Tipo: 265
- Campos:
    - cadena_hash : cadena_hash
    - u32: primera marca de tiempo
    - u32: intervalo_marca_horaria

Una vez que un compañero haya sincronizado el gráfico del canal, puede enviar este mensaje si desea recibir actualizaciones en tiempo real sobre los cambios en el gráfico del canal. También pueden configurar los campos first_timestamp y timestamp_range si desean recibir una acumulación de actualizaciones que pueden haberse perdido mientras estaban inactivos.

Apéndice D. Fuentes y avisos de licencia

Este apéndice contiene avisos de atribución y licencia para el material incluido por permiso otorgado a través de licencias abiertas.

Fuentes

El material se obtuvo de varias fuentes públicas y con licencia abierta:

- Wiki Red Lightning de ION
- “Lightning 101: ¿Qué es una factura Lightning?” por Suredbits
- Especificaciones en progreso de Lightning Network GitHub; Atribución de Creative Commons (CC-BY 4.0)
- Página de Wikipedia, "Diffie-Hellman de curva elíptica"
- Página de Wikipedia, “Firma digital”
- Página de Wikipedia, “Función hash criptográfica”
- Página de Wikipedia, "Enrutamiento de cebolla"
- Wikimedia Commons, “Paquete de protocolos de red Lightning”
- Wikimedia Commons, “Introducción a Lightning Network Protocolo y los fundamentos de la tecnología Lightning”

Servidor BTCPay

Logotipo del servidor BTCPay , capturas de pantalla y otras imágenes usado con permiso bajo la Licencia MIT:

Industrias Lamassu AG

Imágenes del cajero automático ***Gaia*** Bitcoin que se ven en la Figura 2-3 se usan con permiso de Lamassu Industries AG. El uso de estas imágenes no constituye una promoción del producto o la empresa, sino que se proporcionan como un ejemplo visual de un cajero automático de Bitcoin.

Glosario

Este glosario rápido contiene muchos de los términos utilizados en relación con Bitcoin y Lightning Network. Estos términos se usan a lo largo del libro, así que márquelos como referencia rápida.

Dirección

Las direcciones de Bitcoin codifican de forma compacta la información necesaria para pagar a un receptor. Una dirección moderna consta de una cadena de letras y números que comienza con bc1 y se parece a bc1qw508d6qejxtdg4y5r3zarvary0c5xw7kv8f3t4. Una dirección es la abreviatura de la secuencia de comandos de bloqueo de un destinatario, que puede ser utilizada por un remitente para traspasar fondos al destinatario. La mayoría de las direcciones representan la clave pública del receptor o algún tipo de script que define condiciones de gasto más complejas. El ejemplo anterior es una dirección bech32 que codifica un programa testigo que bloquea fondos en el hash de una clave pública (consulte ***Pay-to-Witness-Public-Key-Hash).*** También hay formatos de direcciones más antiguos que comienzan con 1 o 3 que usan la codificación de direcciones Base58Check para representar hashes de clave pública o hashes de script.

sistema criptográfico asimétrico

La criptografía asimétrica, o criptografía de clave pública, es un sistema criptográfico que utiliza pares de claves: claves públicas que pueden difundirse ampliamente y claves privadas que solo conoce el propietario. La generación de tales claves depende de algoritmos criptográficos basados en problemas matemáticos para producir funciones que son fáciles de resolver en un sentido, pero muy difíciles de resolver en sentido inverso.
La seguridad efectiva solo requiere mantener privada la clave privada; la clave pública se puede distribuir abiertamente sin comprometer la seguridad.

piloto automático

Un piloto automático es un motor de recomendación para nodos Lightning que utiliza estadísticas de la topología de Lightning Network para sugerir con qué nodos deberían abrir canales. Dependiendo de la implementación del piloto automático, también se puede recomendar la capacidad del canal. Un piloto automático no es parte del Protocolo LN.

balance

El saldo de un canal de pago es la cantidad de bitcoin que pertenece a cada socio del canal. Por ejemplo, Alice podría abrir un canal con Bob por el valor de 1 BTC. El saldo del canal es entonces 1 BTC para Alice
y 0 BTC a Bob. A medida que los usuarios realicen transacciones, el saldo del canal se actualizará. Por ejemplo, si Alice envía 0,2 BTC a Bob, entonces el saldo ahora es de 0,8 BTC para Alice y 0,2 para Bob. Cuando se cierra el canal, el bitcoin en el canal se dividirá entre los dos socios del canal de acuerdo con el último saldo codificado en la transacción de compromiso.
En Lightning Network, la capacidad de enviar y recibir pagos está limitada por los saldos de los canales. Ver ***capacidad.***

bech32

bech32 se refiere a un formato genérico codificado en base32 con suma de verificación que presenta sólidas garantías de detección de errores. Si bien bech32 se desarrolló originalmente para usarse como formato de dirección para salidas SegWit nativas (BIP-173), también se usa para codificar facturas relámpago (BOLT #11). Mientras que las salidas nativas de la versión 0 de SegWit (P2WPKH y P2WSH) usan bech32, las versiones de salida nativas de SegWit superiores (por ejemplo, Pay-to-Taproot o P2TR) usan la variante mejorada bech32m (BIP-350). Las direcciones bech32m a veces se denominan direcciones "bc1", lo que refleja el prefijo de dichas direcciones. Las salidas nativas de SegWit son más eficientes en el espacio de bloques que las direcciones más antiguas y, por lo tanto, pueden reducir las tarifas de transacción para el propietario de dicha dirección.

Propuesta de mejora de Bitcoin (BIP)

Una propuesta que los miembros de la comunidad Bitcoin han presentado para mejorar Bitcoin. Por ejemplo, BIP-21 es una propuesta para mejorar el esquema del identificador uniforme de recursos (URI) de Bitcoin. Los BIP se pueden encontrar en GitHub.

bitcoins, bitcoins

Según el contexto, podría referirse al nombre de la unidad monetaria (la moneda), la red o el protocolo de habilitación subyacente. Escrito como bitcoin con una "b" minúscula generalmente se refiere a la unidad monetaria. Bitcoin con una "B" mayúscula generalmente se refiere al protocolo o sistema.

minería Bitcoin

La minería de Bitcoin es el proceso de construir un bloque a partir de transacciones recientes de Bitcoin y luego resolver un problema computacional requerido como prueba de trabajo. Es el proceso por el cual se actualiza el libro mayor compartido de bitcoin (es decir, la cadena de bloques de Bitcoin) y por el cual se incluyen nuevas transacciones en el libro mayor. También es el proceso por el cual se emiten nuevos bitcoins. Cada vez que se crea un nuevo bloque, el nodo de minería recibirá un nuevo bitcoin creado dentro de la transacción de base de monedas de ese bloque.

bloquear

Un bloque es una estructura de datos en la cadena de bloques de Bitcoin que consta de un encabezado y un cuerpo de transacciones de Bitcoin. El bloque se marca con una marca de tiempo y se compromete con un bloque predecesor (principal) específico. Cuando se aplica hash, el encabezado del bloque proporciona la prueba de trabajo que hace que la cadena de bloques sea probabilísticamente inmutable. Los bloques deben cumplir con las reglas impuestas por el consenso de la red para extender la cadena de bloques. Cuando se agrega un bloque a la cadena de bloques, se considera que las transacciones incluidas tienen su primera confirmación.

cadena de bloques

La cadena de bloques es un registro distribuido, o base de datos, de todas las transacciones de Bitcoin. Las transacciones se agrupan en actualizaciones discretas llamadas bloques,

limitada hasta 4 millones de unidades de peso. Los bloques se producen aproximadamente cada 10 minutos a través de un proceso estocástico llamado minería. Cada bloque incluye una "prueba de trabajo" computacionalmente intensiva. El requisito de prueba de trabajo se usa para regular los intervalos de bloque y proteger la cadena de bloques contra ataques para reescribir el historial: un atacante necesitaría superar la prueba de trabajo existente para reemplazar los bloques ya publicados, lo que hace que cada bloque sea probabilísticamente inmutable, ya que está enterrado debajo de los bloques posteriores. .

TORNILLO

BOLT, o Base de la tecnología Lightning, es la especificación formal de Lightning Network. A diferencia de Bitcoin, que tiene una implementación de referencia que también sirve como especificación del protocolo, las diversas implementaciones de LN siguen a BOLT para que puedan trabajar entre sí para formar la misma red. Está disponible en GitHub.

capacidad

La capacidad de un canal de pago es equivalente a la cantidad de bitcoin proporcionada por la transacción de financiación. Debido a que la transacción de financiación es visible públicamente en la cadena de bloques y el canal se anuncia a través del protocolo de chismes, la capacidad es información pública. No revela ninguna información sobre cuánto bitcoin posee cada uno de los socios del canal en el canal, es decir, el saldo. Una alta capacidad no garantiza que el canal se pueda utilizar para el enrutamiento en ambas direcciones.

c-relámpago

Implementación del Protocolo LN por parte de la empresa Blockstream con sede en Victoria. Está escrito en C. El código fuente está en GitHub.

transacción de cierre

Si ambos socios de canal acuerdan cerrar un canal, crearán una transacción de liquidación que refleje el compromiso más reciente.

transacción. Después de intercambiar firmas para una transacción de cierre, no se deben realizar más actualizaciones de canal. El cierre mutuo de un canal con la ayuda de una transacción de cierre tiene la ventaja de que se requieren menos transacciones de blockchain para reclamar todos los fondos, en comparación con forzar unilateralmente el cierre de un canal mediante la publicación de una transacción de compromiso. Además, los fondos para ambas partes se pueden gastar inmediatamente desde una transacción de cierre.

CLTV

CLTV es un acrónimo/abreviatura del operador de Bitcoin Script OP_CHECKLOCKTIMEVERIFY. Esto define una altura de bloque absoluta antes de que se pueda gastar una salida. La atomicidad del proceso de enrutamiento depende en gran medida de los valores CLTV en los HTLC. Los nodos de enrutamiento anuncian, a través del protocolo de chismes, los deltas de caducidad de CLTV esperados que desean para cualquier HTLC entrante y saliente.

base de monedas

Coinbase es un campo especial que solo se permite en la entrada única de transacciones de coinbase. La base de monedas permite hasta 100 bytes de datos arbitrarios, pero desde BIP-34, primero debe presentar la altura del bloque actual para garantizar que las transacciones de la base de monedas sean únicas. No debe confundirse con la transacción coinbase.

transacción de base de monedas

La primera transacción en un bloque que siempre crea un minero y que incluye una sola base de monedas. La transacción de coinbase puede reclamar la recompensa del bloque y asignarla a una o más salidas. La recompensa del bloque consiste en el subsidio del bloque (bitcoin recién creado) y la suma de todas las tarifas de transacción de las transacciones incluidas en el bloque. Las salidas de Coinbase solo se pueden gastar después de madurar por 100 bloques. Si el bloque incluye transacciones de SegWit, la transacción de base de monedas debe incluir un compromiso con los identificadores de transacciones testigo en un documento adicional.

producción.

almacenamiento en frio

Se refiere a mantener una cantidad de bitcoin fuera de línea. El almacenamiento en frío se logra cuando las claves privadas de Bitcoin se crean y almacenan en un entorno seguro fuera de línea. El almacenamiento en frío es importante para proteger las tenencias de bitcoin. Las computadoras en línea son vulnerables a los piratas informáticos y no deben usarse para almacenar una cantidad significativa de bitcoin.

transacción de compromiso

Una transacción de compromiso es una transacción de Bitcoin, firmada por ambos socios de canal, que codifica el último saldo de un canal. Cada vez que se realiza o reenvía un nuevo pago a través del canal, el saldo del canal se actualizará y ambas partes firmarán una nueva transacción de compromiso. Es importante destacar que, en un canal entre Alice y Bob, tanto Alice como Bob mantienen su propia versión de la transacción de compromiso, que también firma la otra parte. En cualquier momento, Alice o Bob pueden cerrar el canal si envían su transacción de compromiso a la cadena de bloques de Bitcoin. Enviar una transacción de compromiso más antigua (obsoleta) se considera ***hacer trampa*** (es decir, una violación del protocolo) en Lightning Network y puede ser penalizado por la otra parte, reclamando todos los fondos en el canal para ellos mismos, a través de una transacción de penalización.

confirmaciones

Una vez que una transacción se incluye en un bloque, tiene una confirmación. Tan pronto como se extrae ***otro*** bloque en la cadena de bloques, la transacción tiene dos confirmaciones, y así sucesivamente. Seis o más confirmaciones se consideran prueba suficiente de que una transacción no se puede revertir.

contrato

Un contrato es un conjunto de transacciones de Bitcoin que juntas dan como resultado un cierto comportamiento deseado. Algunos ejemplos son RSMC para crear un canal de pago bidireccional sin confianza, o HTLC para crear un mecanismo que permita el reenvío de pagos sin confianza a través de terceros.

Intercambio de claves Diffie-Hellman (DHKE)

En Lightning Network, se utiliza el método Elliptic Curve Diffie-Hellman (ECDH). Es un protocolo de acuerdo de clave anónimo que permite a dos partes, cada una con un par de claves públicas y privadas de curva elíptica, establecer un secreto compartido a través de un canal de comunicación inseguro. Este secreto compartido se puede utilizar directamente como clave o para derivar otra clave. La clave, o la clave derivada, se puede utilizar para cifrar las comunicaciones posteriores mediante un cifrado de clave simétrica. Un ejemplo de la clave derivada sería el secreto compartido entre la clave de sesión efímera de un remitente de una cebolla con la clave pública del nodo de un lúpulo de la cebolla, como se describe y utiliza en el formato SPHINX Mix.

firma digital

Una firma digital es un esquema matemático para verificar la autenticidad e integridad de mensajes o documentos digitales. Puede verse como un compromiso criptográfico en el que no se oculta el mensaje.

gasto doble

El doble gasto es el resultado de gastar con éxito algo de dinero más de una vez. Bitcoin protege contra el doble gasto al verificar que cada transacción agregada a la cadena de bloques se adhiera a las reglas del consenso; esto significa verificar que los insumos de la transacción no hayan sido gastados previamente.

Algoritmo de firma digital de curva elíptica (ECDSA)

El algoritmo de firma digital de curva elíptica o ECDSA es un algoritmo criptográfico utilizado por Bitcoin para garantizar que solo el titular de la clave privada correcta pueda gastar los fondos.

Destello

Implementación del Protocolo LN por parte de la empresa ACINQ con sede en París.
Está escrito en Scala. El código fuente está en GitHub.

codificación

La codificación es el proceso de convertir un mensaje en una forma diferente.
Por ejemplo, convertir un número de decimal a hexadecimal.

servidor electro

Un servidor Electrum es un nodo Bitcoin con una interfaz adicional (API).
A menudo lo requieren las billeteras de bitcoin que no ejecutan un nodo completo. Por ejemplo, estas billeteras verifican el estado de transacciones específicas o transmiten transacciones al mempool utilizando las API del servidor Electrum.
Algunas billeteras Lightning también usan servidores Electrum.

llave efímera

Las claves efímeras son claves que solo se utilizan durante un breve período de tiempo y no se conservan después de su uso. A menudo se derivan para su uso en una sesión de otra clave que se mantiene a largo plazo. Las claves efímeras se utilizan principalmente dentro del formato de mezcla SPHINX y el enrutamiento de cebolla en Lightning Network. Esto aumenta la seguridad de los mensajes o pagos transportados. Incluso si se filtra una clave efímera, solo se hace pública la información sobre una sola sesión.

bits de características

Una cadena binaria que los nodos Lightning usan para comunicarse entre sí qué características admiten. Los bits de función se incluyen en muchos mensajes Lightning, así como en BOLT #11. Se pueden decodificar usando el BOLT #9 y le dirán a los nodos qué características ha habilitado el nodo y si son compatibles con versiones anteriores. También conocidas como banderas de funciones.

Tarifa

En el contexto de Lightning Network, los nodos cobrarán tarifas de enrutamiento por reenviar los pagos de otros usuarios. Los nodos individuales pueden establecer sus

propias políticas de tarifas que se calcularán como la suma de una base_fee fija y una fee_rate que depende del monto del pago. En el contexto de Bitcoin, el remitente de una transacción paga una tarifa de transacción a los mineros por incluir la transacción en un bloque. Las tarifas de transacción de Bitcoin no incluyen una tarifa base y dependen linealmente del peso de la transacción, pero no del monto.

transacción de financiación

La transacción de financiación se utiliza para abrir un canal de pago. El valor (en bitcoins) de la transacción de financiación es exactamente la capacidad del canal de pago. El resultado de la transacción de financiación es un script de firma múltiple 2 de 2 (multisig) donde cada socio de canal controla una clave. Debido a su naturaleza multifirma, solo se puede gastar de mutuo acuerdo entre los socios de canal. Eventualmente será gastado por una de las transacciones de compromiso o por la transacción de cierre.

características globales *(campo* ***de características globales)***

Las funciones globales de un nodo Lightning son las funciones de interés para todos los demás nodos. Por lo general, están relacionados con los formatos de enrutamiento admitidos. Se anuncian en el mensaje init del protocolo peer, así como en los mensajes channel_announcement y node_announcement del protocolo gossip.

protocolo de chismes

Los nodos LN envían y reciben información sobre la topología de Lightning Network a través de mensajes de chismes que se intercambian con sus pares. El protocolo de chismes se define principalmente en BOLT #7 y define el formato de los mensajes node_announcement, channel_announcement y channel_update. Para evitar el spam, los mensajes de anuncio de nodo solo se reenviarán si el nodo ya tiene un canal, y los mensajes de anuncio de canal solo se reenviarán si la transacción de financiación del canal ha sido confirmada por la red de Bitcoin. Por lo general, los nodos Lightning

conectarse con sus socios de canal, pero está bien conectarse con cualquier otro nodo Lightning para procesar mensajes de chismes.

billetera de hardware

Una billetera de hardware es un tipo especial de billetera Bitcoin que almacena las claves privadas del usuario en un dispositivo de hardware seguro. Al momento de escribir el libro, las billeteras de hardware no están disponibles para los nodos LN porque las claves utilizadas por Lightning deben estar en línea para participar en el protocolo.

picadillo

Una huella digital de tamaño fijo de alguna entrada binaria de longitud arbitraria.
También conocido como ***resumen.***

código de autenticación de mensajes basado en hash (HMAC)

HMAC es un algoritmo para verificar la integridad y autenticidad de un mensaje basado en una función hash y una clave criptográfica. Se utiliza en el enrutamiento de cebolla para garantizar la integridad de un paquete en cada salto, así como dentro de la variante del Protocolo de ruido utilizada para el cifrado de mensajes.

función hash

Una función hash criptográfica es un algoritmo matemático que asigna datos de tamaño arbitrario a una cadena de bits de un tamaño fijo (un hash) y está diseñado para ser una función unidireccional, es decir, una función que no es factible de invertir. La única forma de recrear los datos de entrada a partir de la salida de una función hash criptográfica ideal es intentar una búsqueda de fuerza bruta de posibles entradas para ver si producen una coincidencia.

bloqueo de hash

Un hashlock es una condición de gasto de Bitcoin Script que restringe el gasto de una salida hasta que se revela un dato específico.
Los hashlocks tienen la propiedad útil de que una vez que se revela un hashlock mediante el gasto, también se pueden gastar otros hashlocks asegurados con la misma clave. Esto hace posible crear múltiples salidas que son

todos gravados por el mismo hashlock y que se pueden gastar al mismo tiempo.

hash contrato de bloqueo de tiempo (HTLC)

Un contrato hash con bloqueo de tiempo (HTLC) es una secuencia de comandos de Bitcoin que consiste en bloqueos de hash y bloqueos de tiempo para exigir que el destinatario de un pago gaste el pago antes de una fecha límite presentando la preimagen de hash o que el remitente pueda reclamar un reembolso después del bloqueo de tiempo. caduca En Lightning Network, los HTLC son salidas en la transacción de compromiso de un canal de pago y se utilizan para habilitar el enrutamiento de pagos sin confianza.

factura

El proceso de pago en Lightning Network lo inicia el destinatario (beneficiario) que emite una factura, también conocida como ***solicitud de pago.*** Las facturas incluyen el hash de pago, el monto, una descripción y el tiempo de vencimiento. Las facturas relámpago se definen en BOLT #11.
Las facturas también pueden incluir una dirección alternativa de Bitcoin a la que se puede realizar el pago en caso de que no se pueda encontrar una ruta, así como sugerencias para enrutar un pago a través de un canal privado.

enrutamiento justo a tiempo (JIT)

El enrutamiento justo a tiempo (JIT) es una alternativa al enrutamiento basado en la fuente que fue propuesto por primera vez por el coautor René Pickhardt. Con el enrutamiento JIT, los nodos intermediarios a lo largo de una ruta pueden pausar un pago en tránsito para reequilibrar sus canales antes de continuar con el pago. Esto podría permitirles reenviar con éxito pagos que de otro modo podrían haber fallado debido a la falta de capacidad de salida.

mensaje relámpago

Un mensaje Lightning es una cadena de datos encriptada que se puede enviar entre dos pares en Lightning Network. Al igual que otros protocolos de comunicación, los mensajes Lightning constan de un encabezado y un

cuerpo. El encabezado y el cuerpo tienen su propio HMAC. Los mensajes Lightning son el componente principal de la capa de mensajería.

Red Lightning, Protocolo de red Lightning, Protocolo Lightning

Lightning Network es un protocolo sobre Bitcoin (u otras criptomonedas). Crea una red de canales de pago que permite el envío sin confianza de pagos a través de la red con la ayuda de HTLC y enrutamiento cebolla. Otros componentes de Lightning Network son el protocolo de chismes, la capa de transporte y las solicitudes de pago.

Conjunto de protocolos Lightning Network

El conjunto de protocolos Lightning Network consta de cinco capas que son responsables de varias partes del protocolo. Desde abajo (la primera capa) hasta arriba (la quinta capa), estas capas se denominan capa de comunicación de red, capa de mensajería, capa de igual a igual, capa de enrutamiento y capa de pago. Varios BOLT definen partes de una o varias capas.

Nodo Lightning Network, nodo Lightning

Una computadora que participa en Lightning Network, a través del protocolo Lightning peer-to-peer. Los nodos Lightning tienen la capacidad de abrir canales con otros nodos, enviar y recibir pagos y enrutar pagos de otros usuarios. Por lo general, un usuario de un nodo Lightning también ejecutará un nodo Bitcoin.

Ind

Implementación del Protocolo LN por la empresa Lightning Labs con sede en San Francisco. Está escrito en Go. El código fuente está en GitHub.

características locales (campo: características locales)

Las características locales de un nodo LN son las características configurables de interés directo para sus pares. Se anuncian en el mensaje de inicio del

peer protocol así como en los mensajes channel_announcement y node_announcement del protocolo gossip.

tiempo de bloqueo

Locktime, o más técnicamente nLockTime, es la parte de una transacción de Bitcoin que indica el momento más temprano o el bloque más temprano cuando esa transacción se puede agregar a la cadena de bloques.

capa de mensajería

La capa de mensajería se basa en la capa de conexión de red del conjunto de protocolos Lightning Network. Es responsable de garantizar una comunicación e intercambio de información cifrados y seguros a través del protocolo de capa de conexión de red elegido. La capa de mensajería define el encuadre y el formato de Lightning Messages como se define en BOLT #1. Los bits característicos definidos en BOLT #9 también forman parte de esta capa.

milisatoshi

La unidad de cuenta más pequeña en Lightning Network. Un millisatoshi es la cien mil millonésima parte de un solo bitcoin. Un millisatoshi es una milésima parte de un satoshi. Los millisatoshis no existen ni pueden establecerse en la red Bitcoin.

pagos multiparte (MPP)

Los pagos de varias partes (MPP), a menudo también denominados pagos de rutas múltiples, son un método para dividir el monto del pago en varias partes más pequeñas y entregarlas a lo largo de una o más rutas. Dado que MPP puede enviar muchas o todas las partes a través de una sola ruta, el término pago de varias partes es más preciso que pago de múltiples rutas. En informática, los pagos de varias partes se modelan como flujos de red.

multifirma

Multifirma (multisig) se refiere a un script que requiere más de una firma para autorizar el gasto. Los canales de pago siempre se codifican como direcciones multisig que requieren una firma de cada socio del canal de pago. En el caso estándar de un canal de pago de dos partes, se utiliza una dirección multigrado 2 de 2.

nodo

Consulte ***el nodo Lightning Network.***

capacidad de la red

La capacidad de LN es la cantidad total de bitcoin bloqueado y circulado dentro de Lightning Network. Es la suma de las capacidades de cada canal público. Refleja el uso de Lightning Network hasta cierto punto porque esperamos que las personas coloquen bitcoin en los canales Lightning para gastarlo o reenviar los pagos de otros usuarios. Por lo tanto, cuanto mayor sea la cantidad de bitcoin en los canales Lightning, mayor será el uso esperado de Lightning Network. Tenga en cuenta que dado que solo se puede observar la capacidad del canal público, se desconoce la verdadera capacidad de la red. Además, dado que la capacidad de un canal puede permitir una cantidad ilimitada de pagos de ida y vuelta, la capacidad de la red no implica un límite de valor transferido en Lightning Network.

capa de conexión de red

La capa más baja del conjunto de protocolos Lightning Network. Su responsabilidad es admitir protocolos de Internet como IPv4, IPv6, TOR2 y TOR3, y usarlos para establecer un canal de comunicación criptográfico seguro como se define en el BOLT #8, o hablar DNS para el arranque de la red como se define en el BOLT #10 .

Ruido_XK

La plantilla de Noise Protocol Framework para establecer un canal de comunicación autenticado y encriptado entre dos pares de Lightning Network. X significa que no se necesita ninguna clave pública

conocido por el iniciador de la conexión. K significa que es necesario conocer la clave pública del receptor.

enrutamiento de cebolla

El enrutamiento cebolla es una técnica para la comunicación anónima a través de una red informática. En una red de cebolla, los mensajes se encapsulan en capas de cifrado, de forma análoga a las capas de una cebolla. Los datos cifrados se transmiten a través de una serie de nodos de red llamados enrutadores cebolla, cada uno de los cuales elimina una sola capa, descubriendo el próximo destino de los datos. Cuando se descifra la capa final, el mensaje llega a su destino. El remitente permanece anónimo porque cada intermediario conoce solo la ubicación de los nodos inmediatamente anteriores y posteriores.

producción

La salida de una transacción de Bitcoin, también llamada salida de transacción no gastada (UTXO). Una salida es una cantidad indivisible de bitcoin que se puede gastar, así como un script que define qué condiciones deben cumplirse para que se gaste ese bitcoin. Cada transacción de bitcoin consume algunos resultados de transacciones previamente registradas y crea nuevos resultados que pueden gastarse más tarde en transacciones posteriores. Una salida típica de bitcoin requerirá que se gaste una firma, pero las salidas pueden requerir el cumplimiento de scripts más complejos. Por ejemplo, una secuencia de comandos de múltiples firmas requiere que dos o más titulares de claves firmen antes de que se pueda gastar la salida, que es un componente fundamental de Lightning Network.

Hash de pago a clave pública (P2PKH)

P2PKH es un tipo de salida que bloquea bitcoin al hash de una clave pública. Una salida bloqueada por un script P2PKH se puede desbloquear (gastar) presentando la clave pública que coincida con el hash y una firma digital creada por la clave privada correspondiente.

Pago por hash de script (P2SH)

P2SH es un tipo de salida versátil que permite el uso de scripts de Bitcoin complejos. Con P2SH, el script complejo que detalla las condiciones para gastar la salida (script de canje) no se presenta en el script de bloqueo. En cambio, el valor está bloqueado en el hash de un script, que debe presentarse y cumplirse para gastar la salida.

dirección P2SH

Las direcciones P2SH son codificaciones Base58Check del hash de 20 bytes de un script. Las direcciones P2SH comienzan con un "3". Las direcciones P2SH ocultan toda la complejidad, para que el remitente de un pago no vea el script.

Hash de clave pública de pago a testigo (P2WPKH)

P2WPKH es el equivalente SegWit de P2PKH, utilizando un testigo segregado. La firma para gastar una salida P2WPKH se coloca en el árbol testigo en lugar del campo ScriptSig. Ver ***SegWit.***

dirección P2WPKH

El formato de dirección "native SegWit v0", las direcciones P2WPKH están codificadas en bech32 y comienzan con "bc1q".

Pay-to-Witness-Script-Hash (P2WSH)

P2WSH es el equivalente SegWit de P2SH, utilizando un testigo segregado. La firma y el script para gastar una salida P2WSH se colocan en el árbol testigo en lugar del campo ScriptSig. Ver ***SegWit.***

Dirección P2WSH

El formato de dirección de secuencia de comandos "native Segwi v0", las direcciones P2WSH están codificadas en bech32 y comienzan con "bc1q".

Pago a raíz primaria (P2TR)

Taproot, que se activará en noviembre de 2021, es un nuevo tipo de salida que bloquea bitcoin en un árbol de condiciones de gasto, o una condición de raíz única.

dirección P2TR

El formato de dirección Taproot, que representa SegWit v1, es una dirección codificada bech32m y comienza con "bc1p".

pago

Un pago Lightning ocurre cuando se transfiere bitcoin dentro de Lightning Network. Los pagos generalmente no se ven en la cadena de bloques de Bitcoin.

canal de pago

Un canal de pago es una relación financiera entre dos nodos en Lightning Network, creada mediante una transacción de bitcoin que paga una dirección de múltiples firmas. Los socios del canal pueden usar el canal para enviar bitcoins entre ellos sin comprometer todas las transacciones en la cadena de bloques de Bitcoin. En un canal de pago típico, solo se agregan a la cadena de bloques dos transacciones, la transacción de financiación y la transacción de compromiso. Los pagos enviados a través del canal no se registran en la cadena de bloques y se dice que ocurren "fuera de la cadena".

capa de pago

La capa superior y la quinta del conjunto de protocolos Lightning Network operan sobre la capa de enrutamiento. Su responsabilidad es habilitar el proceso de pago a través de las facturas BOLT #11. Si bien utiliza mucho el gráfico de canal del protocolo de chismes como se define en el BOLT #7, las estrategias reales para entregar un pago no son parte de la especificación del protocolo y se dejan a las implementaciones. Como este tema es muy importante para garantizar la confiabilidad del proceso de entrega de pagos, lo incluimos en este libro.

par

Los participantes en una red peer-to-peer. En Lightning Network, los pares se conectan entre sí a través de una comunicación encriptada y autenticada.

a través de un socket TCP, sobre IP o Tor.

capa de igual a igual

La capa de igual a igual es la tercera capa del conjunto de protocolos Lightning Network y funciona sobre la capa de mensajería. Es responsable de definir la sintaxis y la semántica de la información intercambiada entre pares a través de mensajes Lightning. Esto consiste en mensajes de control como se define en BOLT #9; mensajes de establecimiento, operación y cierre del canal como se define en el TORNILLO #2; así como chismes y enrutamiento de mensajes como se define en BOLT #7.

canal privado

Un canal no anunciado al resto de la red. Técnicamente, "privado" es un nombre inapropiado porque estos canales aún se pueden identificar a través de sugerencias de enrutamiento y transacciones de compromiso. Se describen mejor como canales "no anunciados". Con un canal no anunciado, los socios del canal pueden enviar y recibir pagos entre ellos de manera normal. Sin embargo, el resto de la red no conocerá el canal y, por lo tanto, normalmente no puede usarlo para enrutar pagos. Debido a que se desconoce el número y la capacidad de los canales no anunciados, el número total de canales públicos y la capacidad solo representan una parte del total de Lightning Network.

preimagen

En el contexto de la criptografía y específicamente en Lightning Network, la preimagen se refiere a la entrada de una función hash que produce un hash específico. No es factible calcular la preimagen a partir del hash (las funciones hash solo van en una dirección). Al seleccionar un valor aleatorio secreto como preimagen y calcular su hash, podemos comprometernos con esa preimagen y luego revelarla. Cualquiera puede confirmar que la preimagen revelada produce correctamente el hash.

Prueba de trabajo (PoW)

Datos que requieren un cálculo significativo para encontrarlos y que cualquier persona puede verificar fácilmente para probar la cantidad de trabajo que se requirió para producirlos. En Bitcoin, los mineros deben encontrar una solución numérica para el algoritmo SHA 256 que cumpla con un objetivo de toda la red, llamado objetivo de dificultad. Consulte ***Minería de Bitcoin*** para obtener más información.

Contrato de punto de tiempo fijo (PTLC)

Un contrato de bloqueo de tiempo puntual (PTLC) es un script de Bitcoin que permite un gasto condicional ya sea en la presentación de un secreto o después de que haya pasado una cierta altura de bloque, similar a un HTLC. A diferencia de los HTLC, los PTLC no dependen de una preimagen de una función hash, sino de la clave privada de un punto de curva elíptica. Por lo tanto, la suposición de seguridad se basa en el logaritmo discreto. Los PTLC aún no están implementados en Lightning Network.

bloqueo de tiempo relativo

Un bloqueo de tiempo relativo es un tipo de bloqueo de tiempo que permite que una entrada especifique el momento más temprano en que se puede agregar la entrada a un bloque. El tiempo es relativo y se basa en cuándo se registró en un bloque la salida a la que hace referencia esa entrada. Los bloqueos de tiempo relativos se establecen mediante el campo de transacción nSequence y el código de operación CHECKSEQUENCEVERIFY (CSV) Bitcoin Script, que fue introducido por BIP-68/112/113.

Contrato de Vencimiento de Secuencia Revocable (RSMC)

Este contrato se utiliza para construir un canal de pago entre dos usuarios de Bitcoin o LN que no necesitan confiar el uno en el otro. El nombre

proviene de una secuencia de estados que se codifican como transacciones de compromiso y se pueden revocar si la red de Bitcoin las publica y extrae incorrectamente.

clave de revocación

Cada RSMC contiene dos claves de revocación. Cada socio de canal conoce una clave de revocación. Conociendo ambas claves de revocación, la salida de

el RSMC se puede gastar dentro del bloqueo de tiempo predefinido. Mientras se negocia un nuevo estado de canal, las claves de revocación antiguas se comparten, por lo que se "revoca" el estado anterior. Las claves de revocación se utilizan para disuadir a los socios de canal de transmitir un estado de canal antiguo.

RIPEMD-160

RIPEMD-160 es una función hash criptográfica que produce un hash de 160 bits (20 bytes).

capa de enrutamiento

La cuarta capa del conjunto de protocolos Lightning Network opera sobre la capa peer-to-peer. Su responsabilidad es definir las primitivas criptográficas y el protocolo de comunicación necesario para permitir el transporte seguro y atómico de bitcoin desde un nodo emisor a un nodo receptor. Mientras que BOLT #4 define el formato de cebolla que se utiliza para comunicar información de transporte a pares remotos con los que no existen conexiones directas, el transporte real de las cebollas y las primitivas criptográficas se definen en BOLT #2.

topología

La topología de Lightning Network describe la forma de Lightning Network como un gráfico matemático. Los nodos del gráfico son los nodos Lightning (participantes/pares de la red). Los bordes del gráfico son los canales de pago. La topología de Lightning Network se transmite públicamente con la ayuda del protocolo de chismes, con la excepción de los canales no anunciados. Esto significa que Lightning Network puede ser significativamente más grande que la cantidad anunciada de canales y nodos. Conocer la topología es de particular interés en el proceso de enrutamiento de pagos basado en la fuente en el que el remitente descubre una ruta.

satoshi

Un satoshi es la unidad más pequeña (denominación) de bitcoin que se puede registrar en la cadena de bloques. Un satoshi es 1/100 millonésimo (0.00000001) de un bitcoin y lleva el nombre del creador de Bitcoin, Satoshi Nakamoto.

Satoshi Nakamoto

Satoshi Nakamoto es el nombre utilizado por la persona o grupo de personas que diseñaron Bitcoin y crearon su implementación de referencia original.
Como parte de la implementación, también diseñaron la primera base de datos de blockchain.
En el proceso, fueron los primeros en resolver el problema del doble gasto de la moneda digital. Su verdadera identidad sigue siendo desconocida.

firma Schnorr

Un nuevo esquema de firma digital que se activará en Bitcoin en noviembre de 2021. Permite innovaciones en Lightning Network, como PTLC eficientes (una mejora en HTLC).

secuencia de comandos, secuencia de comandos de Bitcoin

Bitcoin utiliza un sistema de secuencias de comandos para transacciones llamado Bitcoin Script. Parecido al lenguaje de programación Forth, es simple, basado en pilas y procesado de izquierda a derecha. Es deliberadamente Turing-incompleto, sin bucles ni recursividad.

ScriptPubKey (también conocido como secuencia de comandos pubkey)

ScriptPubKey o pubkey script, es un script incluido en las salidas que establece las condiciones que deben cumplirse para que se gasten dichas salidas.
Los datos para el cumplimiento de las condiciones se pueden proporcionar en un script de firma.
Véase también ***ScriptSig.***

ScriptSig (también conocido como script de firma)

ScriptSig o script de firma son los datos generados por un gastador, que casi siempre se utilizan como variables para satisfacer un script de clave pública.

clave secreta (también conocida como clave privada)

El número secreto que desbloquea bitcoin enviado a la dirección correspondiente. Una clave secreta se ve así: 5J76sF8L5j TtzE96r66Sf8cka9y44wdpJjMwCxR3tzLh3i bVPxh.

Testigo segregado (SegWit)

Segregated Witness (SegWit) es una actualización del protocolo Bitcoin introducido en 2017 que agrega un nuevo testigo para firmas y otras pruebas de autorización de transacciones. Este nuevo campo de testigo está exento del cálculo de la identificación de la transacción, lo que resuelve la mayoría de las clases de maleabilidad de transacciones de terceros. Segregated Witness se implementó como una bifurcación suave y es un cambio que técnicamente hace que las reglas del protocolo de Bitcoin sean más restrictivas.

Algoritmo hash seguro (SHA)

El algoritmo hash seguro o SHA es una familia de funciones hash criptográficas publicadas por el Instituto Nacional de Estándares y Tecnología (NIST). El protocolo Bitcoin actualmente usa SHA-256, que produce un hash de 256 bits.

ID de canal corto (scid)

Una vez que se establece un canal, el índice de la transacción de financiación en la cadena de bloques se utiliza como ID de canal corto para identificar de forma única el canal. El ID de canal corto consta de ocho bytes que se refieren a tres números. En su forma serializada, representa estos tres números como valores decimales separados por la letra "x" (p. ej., 600123x01x00). El primer número (4 bytes) es la altura del bloque. El segundo número (2 bytes) es el índice de la transacción de financiación con los bloques. El último número (2 bytes) es la salida de la transacción.

verificación de pago simplificada (SPV)

SPV o verificación de pago simplificada es un método para verificar que transacciones particulares se incluyeron en un bloque sin descargar el bloque completo. El método es utilizado por algunas billeteras ligeras de Bitcoin y Lightning.

enrutamiento basado en la fuente

En Lightning Network, el remitente de un pago decide la ruta del pago. Si bien esto reduce la tasa de éxito del proceso de enrutamiento, aumenta la privacidad de los pagos. Debido al formato mixto SPHINX utilizado por el enrutamiento de cebolla, todos los nodos de enrutamiento no conocen el autor de un pago o el destinatario final. El enrutamiento basado en la fuente es fundamentalmente diferente de cómo funciona el enrutamiento en el Protocolo de Internet.

tenedor suave

Soft fork, o cambio de bifurcación suave, es una actualización de protocolo que es compatible hacia adelante y hacia atrás, por lo que permite que tanto los nodos antiguos como los nodos nuevos continúen usando la misma cadena.

Formato de mezcla ESFINGE

Una técnica particular para el enrutamiento de cebolla utilizada en Lightning Network e inventada por George Danezis e Ian Goldberg en 2009. Con el formato SPHINX Mix, cada mensaje del paquete de cebolla se rellena con algunos datos aleatorios para que ningún salto individual pueda estimar qué tan lejos ha viajado a lo largo de la ruta. Si bien se protege la privacidad del remitente y el receptor del pago, cada nodo aún puede devolver un mensaje de error a lo largo de la ruta al remitente del mensaje.

intercambio de submarinos

Un intercambio submarino es un intercambio atómico sin confianza entre las direcciones de Bitcoin en cadena y los pagos de Lightning Network fuera de la cadena. Así como los pagos de LN usan HTLC que hacen que el reclamo final sobre los fondos esté condicionado a que el destinatario revele un secreto (preimagen hash), los intercambios de submarinos usan

el mismo mecanismo para transferir fondos a través de la barrera dentro y fuera de la cadena con un mínimo de confianza. Los intercambios submarinos inversos permiten intercambios en la dirección opuesta, desde un pago de LN fuera de la cadena hasta una dirección de Bitcoin en la cadena.

bloqueo de tiempo

Un bloqueo de tiempo es un tipo de gravamen que restringe el gasto de algunos bitcoins hasta un tiempo futuro específico o una altura de bloque. Los bloqueos de tiempo ocupan un lugar destacado en muchos contratos de Bitcoin, incluidos los canales de pago y los HTLC.

transacción

Las transacciones son estructuras de datos utilizadas por Bitcoin para transferir bitcoins de una dirección a otra. Varios miles de transacciones se agregan en un bloque, que luego se registra (se extrae) en la cadena de bloques. La primera transacción de cada bloque, denominada transacción coinbase, genera nuevos bitcoins.

maleabilidad de la transacción

La maleabilidad de la transacción es una propiedad que el hash de una transacción puede cambiar sin cambiar la semántica de la transacción. Por ejemplo, alterar la firma puede cambiar el hash de una transacción. Una transacción de compromiso necesita el hash de una transacción de financiación, y si el hash de la transacción de financiación cambia, las transacciones que dependan de ella dejarán de ser válidas. Esto hará que los usuarios no puedan reclamar los reembolsos si los hubiera. La bifurcación suave Segregated Witness soluciona este problema y, por lo tanto, fue una actualización importante para admitir Lightning Network.

capa de transporte

En las redes informáticas, la capa de transporte es una división conceptual de los métodos que utilizan las computadoras (y, en última instancia, las aplicaciones) para comunicarse entre sí. La capa de transporte proporciona servicios de comunicación.

entre computadoras, como control de flujo, verificación y multiplexación (para permitir que múltiples aplicaciones funcionen en una computadora al mismo tiempo).

salida de transacción no gastada (UTXO)

Ver ***salida.***

cartera

Una billetera es una pieza de software que contiene claves privadas de Bitcoin. Se utiliza para crear y firmar transacciones de Bitcoin. En el contexto de Lightning Network, también contiene secretos de revocación del estado del canal anterior y las últimas transacciones de compromiso prefirmadas.

torre de vigilancia

Las torres de vigilancia son un servicio de seguridad en Lightning Network que monitorea los canales de pago en busca de posibles infracciones de protocolo. Si uno de los socios del canal se desconecta o pierde su copia de seguridad, una torre de vigilancia mantiene copias de seguridad y puede restaurar la información de su canal.

Las torres de vigilancia también monitorean la cadena de bloques de Bitcoin y pueden enviar una transacción de penalización si uno de los socios intenta "hacer trampa" al transmitir un estado obsoleto. Las torres de vigilancia pueden ser administradas por los propios socios del canal o como un servicio de pago ofrecido por un tercero. Las torres de vigilancia no tienen control sobre los fondos en los propios canales.

Índice

C

D

Y

F

k

L

S

T

EN

Sobre los autores

Andreas M. Antonopoulos es un autor, orador, educador y un experto muy buscado en Bitcoin y tecnologías abiertas de cadena de bloques. Es conocido por hacer que los temas complejos sean fáciles de entender y resaltar los impactos positivos y negativos que estas tecnologías pueden tener en nuestras sociedades globales.

Andreas ha escrito dos libros técnicos más vendidos para programadores con O'Reilly Media, ***Mastering Bitcoin*** y ***Mastering Ethereum.*** También ha publicado la serie de libros ***The Internet of Money ,*** que se centran en la importancia y las implicaciones sociales, políticas y económicas de estas tecnologías. Andreas produce semanalmente contenido educativo gratuito en su canal de YouTube e imparte talleres virtuales en su sitio web. Obtenga más información en ***www.antonop.com.***

Olaoluwa Osuntokun es el cofundador y director de tecnología de Lightning Labs, y también el desarrollador principal de lnd, una de las principales implementaciones de Lightning. Recibió su licenciatura y maestría en informática de la UCSB y fue miembro de la clase Forbes 30 Under 30 de 2019. Durante sus estudios de posgrado, se centró en el campo de la criptografía aplicada, específicamente en la búsqueda cifrada. Ha sido un desarrollador activo de Bitcoin durante más de cinco años y es autor de varias propuestas de mejora de Bitcoin (BIP-157 y 158). En estos días, su enfoque principal radica en construir, diseñar y desarrollar protocolos de cadena de bloques fuera de la cadena escalables y privados, como Lightning.

René Pickhardt es un matemático capacitado y consultor de ciencia de datos que utiliza su conocimiento estadístico para investigar con NTNU sobre búsqueda de rutas, privacidad, confiabilidad de pagos y acuerdos de nivel de servicio de Lightning Network. René mantiene un canal de YouTube orientado a técnicos y desarrolladores sobre Lightning Network y ha respondido aproximadamente la mitad de las preguntas sobre Lightning Network en Bitcoin Stack Exchange, lo que lo convierte en el punto de referencia para casi todos los nuevos desarrolladores que desean unirse al espacio. René ha realizado numerosos talleres sobre Lightning Network en público y en privado, incluso enseñando a estudiantes de la

Residencia de Chaincode Labs 2019 junto con otros desarrolladores principales de Lightning.

Colofón

Los animales de la portada de ***Mastering the Lightning Network*** son hormigas de madera ***(Formica rufa).*** Comúnmente utilizado para describir un amplio grupo de hormigas, las "hormigas de madera" son aquellas que construyen nidos en áreas boscosas o infestan la madera en una casa. Sin embargo, ***Formica rufa*** se refiere específicamente a las hormigas de madera roja que construyen montículos y que se encuentran principalmente en el sur de Gran Bretaña, el norte y el centro de Europa, la cordillera de los Pirineos y Siberia.
A veces, también se encuentran en América del Norte en bosques y parques de coníferas y de hoja ancha.

También conocida como la hormiga de madera del sur, esta subespecie de hormigas de madera es agresiva, activa y grande. Las reinas de las hormigas de madera suelen tener un tamaño de 12 a 15 mm y pueden vivir hasta 15 años. Las hormigas obreras, por otro lado, son un poco más pequeñas, de 8 a 10 mm, y tienen una vida útil de entre unas pocas semanas y siete años, dependiendo de si son machos o hembras (los machos mueren poco después del apareamiento).

Capaces de producir ácido fórmico en sus abdómenes, las hormigas rojas de madera pueden expulsarlo hasta unos pocos pies de distancia cuando son amenazadas por depredadores. Sus nidos suelen ser montículos llamativos de hierba, ramitas o agujas de coníferas, a menudo construidos contra un tocón de árbol podrido en un área a la que la luz del sol puede llegar fácilmente. Las hormigas de madera viven en grandes colonias que pueden tener de 100 000 a 400 000 obreras y 100 reinas. Las hormigas rojas de madera son muy territoriales y se sabe que atacan y eliminan otras especies de hormigas del área.

Las hormigas rojas trabajadoras de la madera se alimentan hasta 50 metros de su nido para recolectar una resina natural que se encuentra goteando de los pinos. En un comportamiento exclusivo de las hormigas de madera, las hormigas individuales caminan sobre la resina para desinfectarse de bacterias y hongos. Además, también comen melaza de áfidos, pequeños insectos y arácnidos. Las hormigas rojas de la madera se usan comúnmente en la silvicultura y, a menudo, se introducen en un área como una forma de control de plagas.

Las hormigas rojas de madera son actualmente una especie protegida y están clasificadas como "casi amenazadas" por la UICN. Muchos de los animales de las portadas de O'Reilly están en peligro de extinción; todos ellos son importantes para el mundo.

La ilustración de la portada es de Karen Montgomery, basada en un grabado en blanco y negro de una placa suelta, de origen desconocido. Las fuentes de la portada son Gilroy Semibold y Guardian Sans. La fuente del texto es Adobe Minion Pro; la fuente del encabezado es Adobe Myriad Condensed; y la fuente del código es Ubuntu Mono de Dalton Maag.