

APLICACIONES CRIPTOGRAFICAS PARA UN SISTEMA DE CADENA DE BLOQUES

Antecedentes de la invención

El sistema de bitcoins se desarrolló para permitir que el dinero electrónico se transfiera directamente de una parte a otra sin pasar a través de una institución financiera, tal como se describe en el artículo titulado "Bitcoin: A Peer-To-Peer Electronic Cash System" de Satoshi Nakamoto. Un bitc  n (por ejemplo, una moneda electr  nica) est   representado por una cadena de transacciones que transfiere la propiedad de una parte a otra. Para transferir la propiedad de un bitc  n, se genera una nueva transacci  n y se agrega a una pila de transacciones en un bloque. La nueva transacci  n, que incluye la clave p  blica del nuevo propietario, est   firmada digitalmente por el propietario con la clave privada del propietario para transferir la propiedad al nuevo propietario tal como se representa por la nueva clave p  blica del propietario. Una vez que el bloque est   lleno, el bloque se "limita" con un encabezado de bloque que es un resumen de *hash* de todos los identificadores de transacci  n dentro del bloque. El encabezado de bloque se registra como la primera transacci  n en el siguiente bloque de la cadena, creando una jerarqu  a matem  tica denominada "cadena de bloques". Para verificar el propietario actual, puede seguirse la cadena de bloques de transacciones para verificar cada transacci  n desde la primera transacci  n hasta la   ltima transacci  n. El nuevo propietario solo necesita tener la clave privada que coincida con la clave p  blica de la transacci  n que transfiri   el bitc  n. La cadena de bloques crea una prueba matem  tica de propiedad en una entidad representada por una identidad de seguridad (por ejemplo, una clave p  blica), que en el caso del sistema de bitcoins es seudoan  nima.

Para garantizar que un propietario anterior de un bitc  n no gastara dos veces el bitc  n (es decir, transferir la propiedad del mismo bitc  n a dos partes), el sistema de bitcoins mantiene un libro mayor de transacciones distribuido. Con el libro mayor distribuido, un libro mayor de todas las transacciones para un bitc  n se almacena de forma redundante en varios nodos (es decir, computadoras) de una red de cadena de bloques. El libro mayor en cada nodo se almacena como una cadena de bloques. En una cadena de bloques, las transacciones se almacenan en el orden en el cual las transacciones son recibidas por los nodos. Cada nodo en la red de cadena de bloques tiene una r  plica completa de la cadena de bloques completa. El sistema de bitcoins tambi  n implementa las t  cnicas para garantizar que cada nodo almacene la cadena de bloques id  ntica, aunque los nodos pueden recibir transacciones en diferentes   rdenes. Para verificar que las transacciones en un libro mayor almacenadas en un nodo sean correctas, puede accederse a los bloques en la cadena de bloques del m  s antiguo al m  s reciente, generando un nuevo *hash* del bloque y comparando el nuevo *hash* con el *hash* generado cuando se cre   el bloque. Si los *hash* son iguales, entonces se verifican las transacciones en el bloque. El sistema de bitcoins tambi  n implementa las t  cnicas para garantizar que no ser  a factible cambiar una transacci  n y regenerar la cadena de bloques empleando una t  cnica inform  ticamente costosa para generar un c  digo generado aleatoriamente que se agrega al bloque cuando se crea.

Aunque el sistema de bitcoins ha sido muy exitoso, est   limitado a transacciones en bitcoins u otras criptomonedas. Actualmente se est  n realizando esfuerzos para utilizar la cadena de bloques para

respaldar las transacciones de cualquier tipo, como las relacionadas con la venta de vehículos, venta de derivados financieros, venta de acciones, pagos de contratos, etcétera. Tales transacciones utilizan testigos de identidad, que también se conocen como bonos de portador digital, para identificar de manera única algo que puede estar poseído o que puede poseer otras cosas. Un testigo de identidad para un activo físico o digital se genera mediante un *hash* criptográfico de información unidireccional que identifica de forma única el activo. Los testigos también tienen un propietario que utiliza un par adicional de claves públicas/privadas. La clave pública del propietario se establece como la identidad del propietario del testigo y al realizar acciones contra los testigos, la prueba de propiedad se establece al proporcionar una firma generada por la clave privada del propietario y se valida contra la clave pública que figura como propietaria del testigo. Una persona puede ser identificada de manera única, por ejemplo, utilizando una combinación de nombre de usuario, número de seguro social y biométrica (por ejemplo, huella digital). Un producto (por ejemplo, un refrigerador) puede identificarse de forma única, por ejemplo, utilizando el nombre de su fabricante y su número de serie. Los testigos de identidad para cada uno serían un *hash* unidireccional criptográfico de tales combinaciones. El testigo de identidad para una entidad (por ejemplo, persona o empresa) puede ser la clave pública de un par de claves públicas/privadas, donde la entidad tiene la clave privada. Los testigos de identidad pueden utilizarse para identificar personas, instituciones, productos, contratos, códigos de computadora, acciones, derivados, bonos, seguros, préstamos, documentos, etcétera. Los testigos de identidad también pueden utilizarse para identificar conjuntos de activos. Un testigo de identidad para un conjunto puede ser un *hash* unidireccional criptográfico de los testigos digitales de los activos en el conjunto. La creación de un testigo de identidad para un activo en una cadena de bloques establece la procedencia del activo, y el testigo de identidad puede utilizarse en transacciones (por ejemplo, compra, venta, aseguramiento) del activo almacenado en una cadena de bloques, creando un registro de auditoría completo de las transacciones.

Para registrar una transacción simple en una cadena de bloques, cada parte y activo involucrado con la transacción necesita una cuenta que esté identificada por un testigo digital. Por ejemplo, cuando una persona desea transferir un automóvil a otra persona, el propietario actual y el siguiente propietario crean cuentas, y el propietario actual también crea una cuenta que está identificada de manera única por su número de identificación del vehículo. La cuenta para el automóvil identifica al dueño actual. El propietario actual crea una transacción con cargo a la cuenta del automóvil que indica que la transacción es una transferencia de la transferencia de propiedad, indica las claves públicas (es decir, testigos de identidad) del propietario actual y del siguiente propietario, indica el testigo de identidad del automóvil, y la transacción está firmada por la clave privada del propietario actual. La transacción es evidencia de que el siguiente propietario ahora es el propietario actual.

Para permitir transacciones más complejas, algunos sistemas utilizan “contratos inteligentes”. Un contrato inteligente es un código informático que implementa transacciones de un contrato. El código informático puede ejecutarse en una plataforma segura (por ejemplo, una plataforma Ethereum) que sea compatible con el registro de transacciones en cadenas de bloques. Además, el propio contrato inteligente se registra como una transacción en la cadena de bloques utilizando un testigo de identidad

que es un *hash* (es decir, un testigo de identidad) del código informático para que el código informático que se ejecuta pueda autenticarse. Cuando se implementa, un constructor del contrato inteligente ejecuta la inicialización del contrato inteligente y su estado. El estado de un contrato inteligente se almacena de forma persistente en la cadena de bloques (por ejemplo, a través de un árbol de Merkle). Cuando se registra una transacción con cargo a un contrato inteligente, se envía un mensaje al contrato inteligente y se ejecuta el código informático del contrato inteligente para implementar la transacción (por ejemplo, debitar una cierta cantidad del saldo de una cuenta). El código informático garantiza que se cumplan todos los términos del contrato antes de que la transacción se registre en la cadena de bloques. Por ejemplo, un contrato inteligente puede apoyar la venta de un activo. La información en un contrato inteligente para vender un automóvil pueden ser los testigos de identidad del vendedor, comprador y automóvil y el precio de venta en dólares estadounidenses. El código informático garantiza que el vendedor es el propietario actual del automóvil y que el comprador tiene fondos suficientes en su cuenta. El código informático entonces registra una transacción que transfiere la propiedad del automóvil al comprador y una transacción que transfiere el precio de venta de la cuenta del comprador a la cuenta del vendedor. Si la cuenta del vendedor está en dólares estadounidenses y la cuenta del comprador está en dólares canadienses, el código informático puede recuperar la tasa de cambio de una moneda, determinar cuántos dólares canadienses debe cargarse en la cuenta del vendedor y registrar la tasa de cambio. Si cualquiera de las transacciones no tiene éxito, ninguna transacción se registra.

Cuando se envía un mensaje a un contrato inteligente para registrar una transacción, el mensaje se envía a cada nodo que mantiene una réplica de la cadena de bloques. Cada nodo ejecuta el código informático del contrato inteligente para implementar la transacción. Por ejemplo, si 100 nodos mantienen una réplica de una cadena de bloques, el código informático se ejecuta en cada uno de los 100 nodos. Cuando un nodo completa la ejecución del código informático, el resultado de la transacción se registra en la cadena de bloques. Los nodos emplean un algoritmo de consenso para decidir qué transacciones mantener y qué transacciones descartar. Aunque la ejecución del código informático en cada nodo ayuda a asegurar la autenticidad de la cadena de bloques, requiere grandes cantidades de recursos informáticos para mantener tal ejecución redundante del código informático.

Los contratos inteligentes, como el de la venta de un activo, a menudo necesitan recuperar información con base en eventos o en el tiempo, como tipo de cambio, precios de acciones, tasas de interés, etcétera. Para sustentar la necesidad de tal información, se han desarrollado “oráculos” para registrar las transacciones con la información necesaria. Por ejemplo, un oráculo puede registrar una transacción en la cadena de bloques que enumera los tipos de cambio a una determinada hora cada día. El código informático para un contrato inteligente puede buscar a través de la cadena de bloques para encontrar el tipo de cambio necesario. Desafortunadamente, las plataformas de cadena de bloques actuales no proporcionan una manera de garantizar que la información agregada por un oráculo sea confiable.

Breve descripción de la invención

Se proporciona un método para delegar el comportamiento de un contrato inteligente asociado

con una cadena de bloques al código que no forma parte de la cadena de bloques. Un sistema ordena la ejecución por una máquina virtual del contrato inteligente. Durante la ejecución del contrato inteligente, el contrato inteligente envía a un servicio de contenedores de *cryptlets*, a través de un *cryptodelegate*, una solicitud para delegar un comportamiento a un *cryptlet* que se ejecuta en una computadora anfitriona certificada. Durante la ejecución del servicio de contenedores de *cryptlets*, el servicio de contenedores de *cryptlets* identifica una computadora anfitriona para ejecutar el código del *cryptlet* dentro de un contenedor de *cryptlets*. El servicio de contenedores de *cryptlets* ordena a la computadora anfitriona identificada que ejecute el código del *cryptlet* para realizar el comportamiento delegado. Una vez realizado el comportamiento delegado, el servicio de contenedores de *cryptlets* recibe del *cryptlet* una respuesta al comportamiento solicitado. El servicio de contenedores de *cryptlets* envía la respuesta al contrato inteligente como un mensaje que es validado por el *cryptodelegate*. El *cryptodelegate* y el servicio de contenedores de *cryptlets* proporcionan comunicaciones seguras entre el contrato inteligente y el *cryptlet*.

Breve descripción de los dibujos

La figura 1 es un diagrama de bloques que ilustra un ejemplo en el cual un contrato inteligente solicita los servicios de un *cryptlet*.

La figura 2 es un diagrama de bloques que ilustra una conexión entre un *cryptodelegate* y un *cryptlet* en algunas modalidades.

La figura 3 es un diagrama de flujo que ilustra una visión general de una inicialización de un contrato inteligente que emplea un *cryptlet* en algunas modalidades.

Las figuras 4A, 4B y 4C son diagramas de bloques que ilustran una arquitectura de la CMP que es compatible con los casos de uso de eventos o *cryptlets* de utilidad en algunas modalidades.

Las figuras 5A, 5B, 5C y 5D son diagramas de bloques que ilustran una arquitectura de la CMP que es compatible con los casos de uso de control, *cryptlets* de contrato y uso de enclaves en algunas modalidades.

Descripción detallada de la invención

La combinación de un contrato inteligente y un libro mayor distribuido (es decir, una cadena de bloques) brinda la oportunidad para que un consorcio de entidades (por ejemplo, bancos), conocido como miembros, registre las transacciones entre los miembros y otras partes. Un libro mayor distribuido tiene varias ventajas para un consorcio, como ser criptográficamente auténtico, compartido, distribuido y un libro mayor. Un libro mayor distribuido es criptográficamente auténtico en el sentido de que el uso de claves públicas y privadas garantiza que las transacciones sean invulnerables al fraude y establezcan una coherencia compartida. La distribución de un libro mayor distribuido significa que hay muchas réplicas del libro mayor. El valor de un libro mayor distribuido aumenta a medida que aumenta el número de miembros en un consorcio debido a que pueden registrarse más transacciones que involucren a miembros en el libro mayor distribuido y los miembros pueden compartir los costos asociados con el

mantenimiento del libro mayor distribuido. Dado que el libro mayor distribuido es un “libro mayor”, sirve como un registro inalterable de las transacciones y puede utilizarse para establecer el cumplimiento de los requisitos reglamentarios. Un libro mayor distribuido para tal consorcio es privado en el sentido de que una entidad necesita el permiso de los miembros existentes para convertirse en miembro.

5 Debido a que los contratos inteligentes pueden utilizarse para representar prácticamente todos los tipos de contratos, muchos consorcios diferentes pueden utilizar los contratos inteligentes. Los miembros de un consorcio pueden aceptar un conjunto estándar de contratos inteligentes que se utilizarán para registrar las transacciones. Por ejemplo, un consorcio de bancos puede convenir un contrato inteligente relacionado con hipotecas con una tasa ajustable. Un contrato inteligente de este tipo
10 puede programarse para hacer cumplir los términos de una hipoteca, como debitar una cuenta de un acreedor hipotecario y acreditar una cuenta de un deudor hipotecario mediante la determinación de la tasa ajustable adecuada y calcular un pago variable con base en los términos de la hipoteca. Los contratos inteligentes brindan una oportunidad para que un consorcio reemplace un modelo de negocio existente (por ejemplo, contratos en papel) con contratos inteligentes y tenga las ventajas de un libro
15 mayor distribuido tal como se abordó anteriormente.

Sin embargo, para que un consorcio de este tipo sea realmente efectivo, el consorcio necesita más servicios que aquellos a los cuales se les puede proporcionar solo contratos inteligentes y un libro mayor distribuido. Por ejemplo, los miembros de un consorcio pueden necesitar sus sistemas existentes (por ejemplo, un sistema de contabilidad o un sistema de informes) para acceder a los datos del libro
20 mayor distribuido, sistemas para ayudar con la administración y operación del libro mayor distribuido, sistemas para ayudar a garantizar la privacidad de los datos en el libro mayor distribuido (por ejemplo, los términos de los contratos), sistemas que son compatibles con la administración de identidades y claves, sistemas para respaldar los análisis (por ejemplo, inteligencia empresarial y aprendizaje automático), etcétera. Además, las partes de un contrato pueden desear evitar los costos asociados con la ejecución
25 de la lógica que implementa un contrato de forma redundante en cada nodo.

Para satisfacer las necesidades de un consorcio, se proporciona una plataforma de *middleware* criptográfico (CMP, por sus siglas en inglés) que permite que los contratos inteligentes interactúen con los programas informáticos para proporcionar los servicios adicionales necesarios de un consorcio y para
30 ayudar a evitar la ejecución redundante de la lógica que implementa un contrato. Los programas informáticos están “fuera de la cadena de bloques” en el sentido de que, a diferencia de los contratos inteligentes, no se registran en el libro mayor distribuido. Estos programas son subprogramas criptográficos, a los cuales en el presente documento se les denomina “*cryptlets*”.

En algunas modalidades, la CMP delega el comportamiento asociado con un contrato instalado en una cadena de bloques a una aplicación distinta a la cadena de bloques. La CMP proporciona un
35 delegado criptográfico (“*cryptodelegate*”) y un servicio de contenedores de *cryptlets*. El *cryptodelegate* interactúa con una máquina virtual que ejecuta el código de un contrato para proporcionar acceso a un *cryptlet* que se ejecuta fuera de la máquina virtual. El *cryptodelegate* recibe de la máquina virtual un testigo de identidad del *cryptlet* y una indicación de un comportamiento solicitado que debe realizar el

cryptlet tal como lo establece el código del contrato. Por ejemplo, el comportamiento solicitado puede estar suministrando el precio actual de una determinada mercancía. El *cryptodelegate* interactúa con el servicio de contenedores de *cryptlets* para proporcionar un canal de comunicación seguro desde la máquina virtual a la computadora anfitriona (por ejemplo, una computadora certificada y un contenedor de *cryptlets*) donde se ejecutará el *cryptlet*. El servicio de contenedores de *cryptlets* almacena información relacionada con el registro de *cryptlets* con el que es compatible. Por ejemplo, la información puede incluir una asignación del testigo de identidad del *cryptlet* a la computadora anfitriona donde se ejecuta el *cryptlet* y varias políticas (por ejemplo, aislamiento del proceso) con relación a la ejecución del *cryptlet*. El servicio de contenedores de *cryptlets* ordena al *cryptlet* que proporcione el servicio solicitado y proporcione la respuesta al *cryptodelegate* y cadena de bloques.

La figura 1 es un diagrama de bloques que ilustra un ejemplo en el cual un contrato inteligente solicita los servicios de un *cryptlet*. La figura ilustra un banco 101, un fondo de cobertura 102 y una compañía de seguros 103 que son miembros de un consorcio cuyo libro mayor distribuido está almacenado en los nodos 110 y cuyos contratos inteligentes son ejecutados por una máquina virtual 120 (por ejemplo, una máquina virtual Ethereum). Los miembros son parte de un contrato representado por un contrato inteligente en el cual un cálculo que utiliza un determinado tipo de cambio debe realizarse todos los días a las 16:00 GMT. Como parte del contrato, las partes acuerdan que el contrato inteligente utilizará el tipo de cambio proporcionado por un *cryptlet* 131 que se ejecuta en una computadora anfitriona certificada 130. La computadora anfitriona se certifica en el sentido de que se garantiza que se ejecute el *cryptlet* en un entorno confiable. Se genera una clave de identidad (por ejemplo, una clave pública) para el código informático del *cryptlet* para su inclusión en el contrato inteligente y para su uso en la autenticación del *cryptlet* en tiempo de ejecución. El contrato inteligente incluye una referencia a la computadora anfitriona certificada y al *cryptlet*. Cuando el contrato inteligente se instala por primera vez en la cadena de bloques, su constructor se ejecuta y se registra para recibir un evento de “activación” todos los días a las 16:00 GMT y qué valores desea recibir al despertar. El evento de activación puede generarse por un *cryptlet* de uso especial que envía mensajes de activación a contratos inteligentes. Cuando se activa, el contrato inteligente es ejecutado por la máquina virtual con cualquier valor certificado proporcionado por el *cryptlet*. El canal seguro es validado por el *cryptodelegate*, que es un adaptador que interactúa con la máquina virtual y asegura que solo el *cryptlet* al cual se hace referencia puede proporcionar este evento y los datos para la ejecución.

La figura 2 es un diagrama de bloques que ilustra una conexión entre un *cryptodelegate* y un *cryptlet* en algunas modalidades. Una máquina virtual 210 incluye un *cryptodelegate* 211 y ejecuta un contrato inteligente 212. El *cryptodelegate* interactúa con un *CryptletContainerService* 220. El *CryptletContainerService* (o servicio de contenedores de *cryptlets*) proporciona servicios para buscar un *cryptlet*, hacer cumplir las políticas del *cryptlet*, verificar firmas, etcétera. El *cryptodelegate* utiliza el servicio de contenedores de *cryptlets* para establecer una conexión segura 240 con un *cryptlet* 231 dentro de un contenedor de *cryptlets* 230 que está alojado en una computadora certificada. Un contenedor de *cryptlets* incluye tanto el *cryptlet* como el entorno de ejecución (por ejemplo, el sistema operativo) para el

cryptlet. El contenedor de *cryptlets* puede cargarse y ejecutarse por una máquina virtual de la computadora anfitriona certificada.

La figura 3 es un diagrama de flujo que ilustra una visión general de una inicialización de un contrato inteligente que emplea un *cryptlet* en algunas modalidades. Una máquina virtual de contrato inteligente 310 ejecuta una función de inicialización 320 de un contrato inteligente y de un *cryptodelegate* 330. Una computadora anfitriona de servicio de contenedores de *cryptlets* (CCS, por sus siglas en inglés) 340 ejecuta el contenedor de *cryptlets* 350. Un contenedor de *cryptlets* 360 ejecuta un *cryptlet* 370. La función de inicialización del contrato inteligente registra 321 el *cryptlet* enviando, por ejemplo, el código de *cryptlet* y la información de configuración al *cryptodelegate*. El *cryptodelegate* establece 331 una conexión segura con el servicio de contenedores de *cryptlets* y envía 332 la solicitud de registro al servicio de contenedores de *cryptlets*. El servicio de contenedor *cryptlets* genera y verifica 351 un *hash* del código de *cryptlet*. El servicio de contenedores de *cryptlets* entonces registra 352 el *cryptlet* en una cadena de bloques de registro de *cryptlet*. La función de inicialización del contrato inteligente puede entonces suscribirse 322 para recibir las notificaciones para el *cryptlet* (por ejemplo, cambios en el precio de un producto) mediante el envío de una solicitud al *cryptlet* a través del *cryptodelegate*. La solicitud puede incluir un identificador de una función de retrollamada del contrato inteligente. El *cryptodelegate* reenvía 333 la solicitud de suscripción al servicio de contenedores de *cryptlets*. El servicio de contenedores de *cryptlets* reenvía 353 la solicitud al *cryptlet*. El servicio de contenedores de *cryptlets* también puede ordenar la creación de instancias del contenedor de *cryptlets* que incluye el *cryptlet*. Al recibir la solicitud, el *cryptlet* registra 371 la solicitud. Eventualmente, el *cryptlet* puede decidir publicar en el contrato inteligente un evento (por ejemplo, un cambio en el precio del producto) con base en la suscripción. En tal caso, el *cryptlet* envía el evento a través del servicio de contenedores de *cryptlets* a la cadena de bloques que valida el *cryptodelegate*, y hace que la función de retrollamada del contrato inteligente sea invocada para manejar el evento.

En algunas modalidades, la CMP puede implementarse como un servicio basado en la nube. Varios consorcios pueden interactuar con el servicio basado en la nube para proporcionar los contratos inteligentes y *cryptlets* que están disponibles solo para los miembros de cada consorcio. El servicio basado en la nube puede ser compatible con el registro de un consorcio y sus miembros. El servicio basado en la nube puede proporcionar un servicio de autenticación, servicios de administración de cadena de bloques, máquinas virtuales para el alojamiento de contratos inteligentes de los miembros, servicios de *cryptodelegate*, servicios de *cryptlet*, servicios de seguridad, servicios de nube estándar, etcétera. Siempre que se ejecute un contrato inteligente para un consorcio, el servicio basado en la nube puede crear una instancia de una máquina virtual para ejecutar de forma segura el contrato inteligente y comunicarse con los *cryptlets* a través de un *cryptodelegate* y un servicio de contenedores de *cryptlets*.

En modalidades, se puede llamar a un *cryptodelegate* de manera similar a EIP - 7 DELEGATECALL (<https://github.com/ethereum/EIPs/blob/master/EIPS/eip-7.md>) de la plataforma Ethereum. Alternativamente, puede definirse una instrucción especial de la máquina virtual para ser compatible con la llamada de *cryptlets*. Sin embargo, en lugar de funcionar como una llamada regular (por

ejemplo, en términos de GAS) a otro contrato inteligente, es una llamada sin costo al código informático fuera de la cadena de bloques de un *cryptlet*. Un *cryptlet* puede estar en cualquier idioma y se ejecuta dentro de un contenedor seguro, aislado y confiable, y se comunica con el uso de comunicaciones seguras. Los *cryptlets* pueden utilizarse en sistemas de contrato inteligente y en sistemas de contrato no inteligente, como los sistemas de salida de transacción no gastada (UTXO, por sus siglas en inglés) cuando se necesita información o funcionalidad adicional.

Un *cryptodelegate* es una función “gancho” dentro de una máquina virtual de contrato inteligente que llama al *cryptlet* para el contrato inteligente que extiende la envoltura segura y auténtica para transacciones a la ejecución del *cryptlet*. Un *cryptlet* llamado a través de un *cryptodelegate* es de confianza ya sea firmado con la misma firma que el contrato inteligente o una firma digital de confianza y validación cuando se escribió o en el momento del diseño. Los *cryptlets* tienen sus propias identidades compuestas, que pueden incluir su *hash* binario, versión y certificado de editor para firmar transacciones. Los *cryptlets* globales o compartidos pueden tener identificadores certificados que también incluirán la firma de la computadora anfitriona certificada que se puede asignar a direcciones locales para el flujo de identidad y seguridad adicional. Cada llamada a través de un *cryptodelegate* verifica la validez de las firmas en tiempo de ejecución y las registra junto con la transacción. Para los sistemas de contrato que no son inteligentes, los *cryptlets* se pueden llamar con adaptadores que realizan las funciones que el *cryptodelegate* haría en una máquina virtual de contrato inteligente.

La CMP puede ser compatible con dos tipos de *cryptlets*: utilidad y contrato. Los *cryptlets* de utilidad pueden crearse antes de que se redacten los contratos inteligentes y se realicen funciones comunes como el cifrado, publicación de eventos con base en el tiempo, acceso a datos externos, etcétera. Los desarrolladores de contratos inteligentes pueden utilizar los *cryptlets* de utilidad que necesitan acceso a datos externos o funcionalidad de una manera confiable. Los *cryptlets* de utilidad pueden agruparse en bibliotecas, algunas proporcionadas por el solicitante y otras por terceros como proveedores de *software* independientes. Las capacidades proporcionadas por la CMP pueden implementarse utilizando los *cryptlets* de utilidad.

Se crea un *cryptlet* de contrato y se vincula a una instancia de contrato inteligente específica. Un *cryptlet* de contrato puede alojar una lógica sofisticada que no es del todo adecuada para ejecutarse en un entorno de máquina virtual de cadena de bloques con fines de rendimiento y escala. Con los *cryptlets* de contrato, un contrato inteligente simplemente contiene el estado y las variables clave para el contrato inteligente, mientras el *cryptlet* contiene la lógica de programación, incluida la lógica para acceder a fuentes de datos externas y crear y manejar eventos del contrato inteligente. Un *cryptlet* de contrato es el sustituto del tiempo de ejecución para un contrato inteligente en particular. Los *cryptlets* de contrato utilizados con contratos inteligentes permiten situaciones más avanzadas donde se desean rendimiento y escalabilidad. Por ejemplo, un *cryptlet* de contrato puede realizar la operación completa de un contrato inteligente en paralelo, mientras otros contratos se ejecutan en la máquina virtual de cadena de bloques sin vincularlo. El uso de *cryptlets* de contrato significa que el código de contrato inteligente que se ejecuta en todos los nodos de la cadena de bloques es mucho más liviano debido a que solo se escribe en la

base de datos de la cadena de cadenas y difiere del manejo de la mayor parte de la informática, integración con datos externos, etcétera, en la computadora anfitriona designada para ejecutar el *cryptlet* de contrato delegado del contrato inteligente.

Los *cryptlets* de contrato pueden crearse cuando se implementa un contrato inteligente en la cadena de bloques u otros métodos de implementación. Cuando se implementa el contrato inteligente, el constructor del contrato inteligente registra las propiedades y el código (por ejemplo, C#, F#, Java y C++) para el *cryptlet*. El contrato inteligente proporciona la información de configuración que puede incluir una definición de interfaz para el *cryptlet* y varias propiedades, como tipos personalizados, la clave pública, etcétera, para el *cryptlet*. La definición de interfaz para un *cryptlet* se ilustra a continuación:

```
//Utility cryptlet
contract PropertyPrivacyCryptlet{
    address cryptlet = 0x39e2842....;
    function();
    event();
}

//Contract cryptlet base type
contract ContractCryptlet{
    uint minimumInstances;
    uint maximumInstances;
    address cryptlet;
    function();
    event();
    // check that only the cryptlet surrogate
    // is able to call the smart contract
    modifier onlyBy(address _cryptlet)
    {
        if (msg.sender != _cryptlet)
            throw;
    }
}
```

Las propiedades de un *cryptlet* pueden proporcionarse utilizando un esquema de *cryptlet* con base en la notación de objetos de JavaScript (JSON, por sus siglas en inglés), tal como se ilustra a continuación:

5

10

15

20

25

```
{
  "title": "cryptlet Schema",
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "publicKey": {
      "type": "string"
    },
    "config": {
      "description": "describe what cryptlet does",
      "isolation": "boolean",
      "...":
    }
  },
  "required": ["name", "publicKey"]
}
```

30

35

Cuando un contrato inteligente registra un *cryptlet* con el servicio de contenedores de *cryptlets*, el servicio de contenedores de *cryptlets* genera un *hash* del código de *cryptlet* y asegura que el *hash* coincida con la clave pública proporcionada por el contrato inteligente. El servicio de contenedores de *cryptlets* entonces registra la información de configuración y el código de *cryptlet* en una cadena de bloques de registro de *cryptlet* que puede vincularse a los metadatos del almacenamiento de datos distribuidos e información de políticas para el *cryptlet*. La cadena de bloques de registro de *cryptlet* es una base de datos de cadena de bloques mantenida por el servicio de contenedores de *cryptlets*. Los *cryptlets* de utilidad se registran independientemente de los contratos inteligentes y son consultados por ellos.

Una definición de interfaz de *cryptlet* puede importarse en un contrato inteligente utilizando las siguientes instrucciones de importación:

```
import "github.com/cryptlets/encryption/propertyPrivacy.sol" as propertyPrivacy;
import "github.com/cryptlets/stocktools/equityTools" as equityTools;
```

Los contratos inteligentes que utilizan un *cryptlet* pueden derivarse de una clase base de *cryptletProxy* que proporciona la mayor parte de la información de configuración predeterminada. Un contrato inteligente que utiliza un *cryptlet* de utilidad puede representarse de la siguiente manera:

```
contract cryptletProxy{
    function cryptletProxy() {owner = msg.sender;}
    address owner;
    string name;
    address publicKey;
    string attestedHost;
    string interface;
}

contract PropertyPrivacy is cryptletProxy{
    name = propertyPrivacryptlet;
    publicKey = `0xd5f9d8d94886e70b06e474c3fb14fd43e2f23970`;
    attestedHost = `19 3E EA B7 C5 54 60 1D 81 BD...`;
    interface = propertyPrivacy; //import statement
};

contract StockClient is cryptletProxy{
    name = stockClientcryptlet;
    publicKey = `0xe7f9d8d94886e70b06e474c3fb14fd43e2f2381`;
    attestedHost = `19 3E EA B7 C5 54 60 1D 81 BD...`;
    interface = equityTools; //import statement
};
```

Un contrato inteligente puede, durante la ejecución de su constructor, registrarse para recibir eventos de un *cryptlet* de utilidad de la siguiente manera:

```

var stockClient;
function init(){
    stockClient = StockClient();
    [event(stockclient.PriceUpdate.Subscribe('au', CalculatePrice))];
}
//callback method that is run when the cryptlet PriceUpdate event fires
function CalculatePrice(uint price){...}

```

El constructor `stockClient()` para el *cryptlet* se invoca para registrar el *cryptlet* `StockClient` con el servicio de contenedores de *cryptlets*. El método de evento se invoca para enviar un evento para suscribirse y recibir eventos de `PriceUpdate` para oro. El evento especifica el método de retrollamada del contrato inteligente, `CalculatePrice`, que se invocará cuando el *cryptlet* envíe un evento de actualización de precio al contrato inteligente. El contrato inteligente también puede invocar métodos del *cryptlet* de la siguiente manera:

```
var msft = StockClient.getCurrentPrice('msft');
```

La CMP permite que los comportamientos de *cryptlet*, como el aislamiento del proceso para ejecutar un contenedor de *cryptlet* y un *cryptlet* en un enclave aislado, se especifiquen durante la ejecución del contrato inteligente. Alternativamente, la CMP puede permitir que los comportamientos se establezcan mediante una política del servicio de contenedores de *cryptlets*, que puede anular la configuración del código, de la siguiente manera:

```
PropertyPrivacy.processIsolation = true;
```

Además, el código de un contrato inteligente puede especificar un atributo para indicar que la ejecución de la lógica en el punto de atribución se delega a través del *cryptodelegate*. Por ejemplo, si una propiedad en un contrato inteligente debe estar cifrada y ser visible solo para las partes del contrato, un atributo se puede aplicar de la siguiente manera:

```
[encryptField='ContractSignersOnly']
uint public trade_amount = 0;
```

En este ejemplo, el *cryptodelegate* se asegurará de que la cantidad comercial se cifre y solo se pueda descifrar cuando se proporcione a un firmante del contrato.

El siguiente es otro ejemplo de un *cryptlet*:

```

contract CreditDefaultSwap is Contractcryptlet{
    uint price;
    uint rate;
    //constructor definition of contract cryptlet
    //that will act as an execution surrogate
    function CreditDefaultSwap(){
        minimumInstances = 1;
        maximumInstances = 5;
        //code could be brought in with an include statement
        string code = "c#, java, C++, etc... code ...";
    }
}

```

Los *cryptlets* de contrato permiten la ejecución paralela que no requiere que todos los nodos de la red ejecuten el código de contrato inteligente. Por lo tanto, muchos *cryptlets* de contrato pueden ejecutar diferentes códigos de contrato inteligentes mediante subrogación al mismo tiempo en diferentes máquinas y registrar sus transacciones firmadas de forma auténtica en la base de datos de cadena de bloques. Los *cryptlets* de contrato permiten que las cadenas de bloques de contrato inteligentes se puedan escalar a volúmenes mucho más grandes y calcular la potencia, a la vez que se mantiene la integridad de la cadena de bloques.

En las modalidades, para implementar una llamada a un *cryptlet*, la CMP puede utilizar una máquina virtual que ejecute los contratos inteligentes y que se modifique para agregar una nueva instrucción para acceder al *cryptodelegate*. Alternativamente, un compilador para una máquina virtual Ethereum puede marcar los descriptores de accesos de variables y métodos estáticos de un *cryptlet* con la dirección del *cryptodelegate* de la siguiente manera:

```
value=0x5b Mnemonic=JUMPDEST δ=0 α=0
```

La máquina virtual modificada puede utilizar DELEGATECALL en la intercepción durante el tiempo de ejecución para asignarse a JUMPDEST del *cryptodelegate* de la siguiente manera:

```
value=0xf4 Mnemonic=DELEGATECALL δ=7 α=1
```

De esta manera, el *cryptodelegate* puede procesar de forma sincrónica en otro hilo mientras el hilo de llamada espera en proceso. Una vez que DELEGATECALL se asigna a JUMPDEST del

cryptodelegate que se aprueba en la clave pública del *cryptlet*, así como a varios atributos, el *cryptodelegate* recopila la ejecución.

El *cryptodelegate* puede incluirse en github.com/ethereum/core/cryptodelegate como un paquete incluido en el núcleo de la siguiente manera:

5

10

15

20

25

30

35

```
package backends
```

```
5 import (
```

```
    "math/big"
```

```
    "github.com/ethereum/go-ethereum/accounts/abi/bind"
```

```
    "github.com/ethereum/go-ethereum/common"
```

```
10    "github.com/ethereum/go-ethereum/core"
```

```
    "github.com/ethereum/go-ethereum/core/state"
```

```
    "github.com/ethereum/go-ethereum/core/types"
```

```
    "github.com/ethereum/go-ethereum/core/vm"
```

```
15    "github.com/ethereum/go-ethereum/ethdb"
```

```
    "github.com/ethereum/go-ethereum/event"
```

```
    "github.com/ethereum/go-ethereum/params"
```

```
)
```

```
20 // Default chain configuration which sets homestead phase at block 0
```

```
// (i.e. no frontier)
```

```
var chainConfig = &core.ChainConfig{HomesteadBlock:
```

```
25 params.MainNetHomesteadBlock}
```

```
// This nil assignment ensures compile time that SimulatedBackend
```

```
//implements bind.ContractBackend.
```

```
30 var _ bind.ContractBackend = (*SimulatedBackend)(nil)
```

```
// SimulatedBackend implements bind.ContractBackend,
```

```
// simulating a blockchain in the background.
```

```
// Its main purpose is to allow easily testing contract bindings.
```

```
35 type SimulatedBackend struct {
```

```
    database ethdb.Database // In memory database to store our testing data
```

```
    blockchain *core.BlockChain // Ethereum blockchain to handle the consensus
```

```
    cryptodelegate *core.CryptoDelegate // Ethereum CryptoDelegate module
```

```
    handling secure envelope for cryptlets
```

El *cryptodelegate* puede incluirse alternativamente dentro de la propia máquina virtual.

En el inicio del nodo, una opción de línea de comandos especifica dónde el *cryptodelegate* busca el registro y política de *cryptlet* de la siguiente manera:

5

```
geth --cryptoDelegatePath 'https://azure.com/myconsortium' --  
cryptoDelegateSignature '0x231dw...'
```

Una vez que el *cryptodelegate* recibe la llamada omitida, pasa la clave pública de *cryptlet* al contenedor de *cryptlet* junto con la carga útil solicitada y opciones como el aislamiento. La carga útil
10 puede ser un método estático llamado en el *cryptlet* con parámetros, la suscripción a un evento de *cryptlet* publicado con detalles de retrollamada, o creación de un *cryptlet* de contrato.

En algunas modalidades, el contenedor de *cryptlets* se encapsulará dentro de un enclave protegido, un área segura, a prueba de manipulación indebida, segura para la ejecución del código y los datos. El contenedor de *cryptlets* y los *cryptlets* que aloja se ejecutarán dentro del enclave seguro para
15 proporcionar una ejecución segura y los datos con las garantías certificadas de que los resultados son auténticos y a prueba de manipulaciones. El contenedor de *cryptlets* que utiliza el enclave proporciona un entorno seguro y privado para que se ejecuten los *cryptlets*. Los *cryptlets* de contrato y aquellos que realizan servicios de encriptación son ejemplos de los que se ejecutarán dentro de un enclave. Los contenedores de *cryptlets* que se ejecutan dentro de un enclave pueden comunicarse de manera segura
20 con otros *cryptlets* en otros enclaves de manera segura, así como comunicarse con otros dispositivos de seguridad como un HSM (módulos de seguridad de *hardware*). Las protecciones de enclavamiento pueden proporcionarse por *hardware* o *software*.

En las modalidades, en el lado del *cryptlet*, el servicio de contenedores de *cryptlets* escucha un puerto para las solicitudes firmadas de las *cryptodelegate* registrados (mediante la firma en los
25 parámetros geth) para instanciar las claves públicas de *cryptlet*. El servicio de contenedores de *cryptlets* comprueba la cadena de bloques de registro de *cryptlet* para el *cryptlet* y verifica la política para la validación de firmas, el aislamiento del proceso, el método para llamar a las dependencias, las posibles funciones de conversión de tipos, etcétera.

En algunas modalidades, una llamada de *cryptlet* puede encaminarse a una instancia sin estado
30 estática precargada del *cryptlet* en un contenedor de *cryptlet*, encaminarse a una instancia de un contenedor aislado de proceso nuevo y opcional (utilizando un enclave), o encaminarse a un contenedor de *cryptlet* que tiene afinidad inteligente de contrato para *cryptlets* de contrato. Los *cryptlets* pueden precargarse en un contenedor de *cryptlets* para el rendimiento.

Una vez que el servicio de contenedores de *cryptlets* crea y/o encamina la llamada al contenedor
35 de *cryptlets*, el contenedor de *cryptlets* crea una instancia del *cryptlet* e invoca la llamada apropiada. Si la llamada es un registro de evento, los detalles de la retrollamada se registran y el *cryptlet* puede o no permanecer activo. Por ejemplo, si la llamada es un registro de evento para una retrollamada en una fecha y hora específicas, el *cryptlet* debería permanecer en ejecución. Para los *cryptlets* de larga

ejecución, la configuración para el número máximo de instancias en la cadena de bloques puede proporcionar un nivel de tolerancia a fallas sin tener una relación de 1-1 entre *cryptlet* y nodo. Las funciones de retrollamada pueden ser llamadas de mensaje desde el *cryptlet* a la función de retrollamada del contrato inteligente de suscripción.

En algunas modalidades, la envoltura de seguridad se extiende desde el contrato inteligente en la cadena de bloques a través del *cryptodelegate* que instancia o localiza el *cryptlet* que se comunica con el certificado apropiado (HTTPS/SSL) en la computadora anfitriona certificada y que verifica la firma del *cryptlet* y el contenedor ejecutándolo en cada llamada. El *cryptodelegate* también puede registrar las llamadas a los *cryptlets* y los datos que proporcionan en el contrato inteligente junto con la firma del *cryptlet*, la firma de contenedor de *cryptlets* opcional y la dirección de computadora anfitriona certificada, de modo que está disponible un libro mayor auténtico para la auditoría.

En modalidades, los *cryptlets* y sus contenedores de *cryptlets* pueden estar firmados por o incluir firmas digitales de identidades que permitirían que un *cryptlet* funcione “en nombre de” una o más identidades. Por ejemplo, un usuario podría crear un *cryptlet* y firmar con su firma digital, y cuando se invoca el *cryptlet*, realiza las acciones como un agente para el usuario en un proceso empresarial.

De este modo, los *cryptlets* proporcionan un modelo de seguridad formal para interactuar con el mundo real, proporcionando la interoperabilidad con los sistemas existentes de una manera confiable para consorcios empresariales y proporcionando un nivel adicional de reutilización para los desarrolladores que escriben los *cryptlets*. Los *cryptlets* también proporcionan un nivel completo de *middleware* para la cadena de bloques de cualquier tipo y proporcionan un entorno de ejecución flexible, permitiendo una delegación parcial de la lógica y una delegación completa para una mayor escala.

Las figuras 4A, 4B y 4C son diagramas de bloques que ilustran una arquitectura de la CMP que es compatible con los *cryptlets* de utilidad y casos de uso de eventos en algunas modalidades. Una arquitectura de *cryptlet* 400 de utilidad incluye un nodo de cadena de bloques 410 y un sistema de *cryptlet* 420. El nodo de cadena de bloques 410 incluye una máquina virtual 411, un componente de algoritmo de consenso 414, una base de datos 415 (por ejemplo, libro mayor distribuido y árboles de Merkle) y un componente de red 416. La máquina virtual incluye un código de cifrado 412 y ejecuta el código de un contrato inteligente 413. El sistema de *cryptlet* incluye un servicio de contenedores de *cryptlets* 430 y un contenedor de *cryptlets* 440 que contiene un *cryptlet* 441. El servicio de contenedores de *cryptlets* incluye un almacenamiento de metadatos de *cryptlet* 431, una cadena de bloques de registro de *cryptlets* 432 y un almacenamiento de políticas 433. El almacenamiento de cadena de bloques de registro de *cryptlets* almacena la información sobre los *cryptlets* registrados. El almacenamiento de metadatos de *cryptlets* almacena metadatos para el *cryptlet* registrado y el almacenamiento de políticas almacena la información de política para el servicio de contenedores de *cryptlets*. El servicio de contenedores de *cryptlets* también incluye una interfaz de registro 434 para registrar los *cryptlets*, una interfaz de búsqueda 435 para buscar información sobre los *cryptlets* y una interfaz de canal seguro 436. Cuando la máquina virtual ejecuta las instrucciones 417 del código del contrato inteligente, un mensaje se envía al *cryptlet* stockClient a través del canal seguro al *cryptlet*. Durante la inicialización del contrato

inteligente, se invocó una función de inicialización (por ejemplo, el constructor) del contrato inteligente para ubicar y establecer una conexión segura con el *cryptlet* stockClient.

Las figuras 5A, 5B, 5C y 5D son diagramas de bloques que ilustran una arquitectura de la CMP que es compatible con los *cryptlets* de contrato, casos de uso de control y uso de enclaves en algunas modalidades. Las figuras 5A y 5B son similares a las figuras 4A y 4B, excepto que el contrato incluye una indicación del código del *cryptlet* de contrato, que se envía al servicio de contenedores de *cryptlets* como un paquete de *cryptlets* de contrato de creación 517. El servicio de contenedor de *cryptlets* 530 también incluye un productor de *cryptlets* para crear instancias de *cryptlets* de contrato. El código del contrato inteligente solicita el comportamiento del *cryptlet* del contrato, y las respuestas se envían como mensajes al contrato inteligente. La figura 5D ilustra el uso de enclaves protegidos mediante *hardware* o *software* para crear operaciones privadas, seguras y a prueba de manipulaciones para la ejecución y los datos. Los *cryptlets* que se ejecutan en un enclave están alojados en el contenedor de *cryptlets* apropiado para el lenguaje en el cual está escrito el *cryptlet*.

Los sistemas informáticos en los cuales puede implementarse la CMP pueden incluir una unidad central de procesamiento, dispositivos de entrada, dispositivos de salida (por ejemplo, dispositivos de visualización y altavoces), dispositivos de almacenamiento (por ejemplo, unidades de memoria y de disco), interfaces de red, unidades de procesamiento de gráficos, acelerómetros, interfaces de enlace de radio celular, dispositivos de sistema de posicionamiento global, etcétera. Los dispositivos de entrada pueden incluir teclados, dispositivos señaladores, pantallas táctiles, dispositivos de reconocimiento de gestos (por ejemplo, para gestos al aire), dispositivos de seguimiento de cabeza y ojos, micrófonos para el reconocimiento de voz, etcétera. Los sistemas informáticos de los clientes pueden incluir computadoras de escritorio, computadoras portátiles, tabletas, lectores electrónicos, asistentes digitales personales, teléfonos inteligentes, dispositivos de juegos, servidores, etcétera. Los sistemas informáticos de los servidores pueden incluir servidores de un centro de datos, sistemas paralelos de gran capacidad, etcétera. Los sistemas informáticos pueden acceder a medios legibles por computadora que incluyen medios de almacenamiento legibles por computadora y medios de transmisión de datos. Los medios de almacenamiento legibles por computadora son medios de almacenamiento tangibles que no incluyen una señal de propagación transitoria. Los ejemplos de medios de almacenamiento legibles por computadora incluyen la memoria como la memoria primaria, la memoria caché y la memoria secundaria (por ejemplo, DVD) y otro almacenamiento. El medio de almacenamiento legible por computadora puede tener grabado en él, o puede estar codificado con, las instrucciones ejecutables por computadora o lógica que implementa la CMP. Los medios de transmisión de datos se utilizan para transmitir datos a través de señales transitorias, propagantes u ondas portadoras (por ejemplo, electromagnetismo) a través de una conexión alámbrica o inalámbrica. Los sistemas informáticos pueden incluir un procesador de cifrado seguro como parte de una unidad central de procesamiento para generar y almacenar de manera segura las claves y para cifrar y descifrar los datos utilizando las claves.

La CMP puede describirse en el contexto general de las instrucciones ejecutables por computadora, como módulos de programa y componentes, ejecutados por una o más computadoras,

procesadores u otros dispositivos. En general, los módulos o componentes de programas incluyen rutinas, programas, objetos, estructuras de datos, etcétera, que realizan tareas particulares o implementan tipos de datos particulares. Normalmente, la funcionalidad de los módulos de programa puede combinarse o distribuirse según se desee en varios ejemplos. Los aspectos de la CMP pueden implementarse en *hardware* utilizando, por ejemplo, un circuito integrado de aplicación específica (ASIC, por sus siglas en inglés) o matriz de puertas programable *in-situ* (FPGA).

Los siguientes párrafos describen diversas modalidades de aspectos de la CMP. Una implementación de la CMP puede emplear cualquier combinación de las modalidades. El procesamiento descrito a continuación puede realizarse por un dispositivo informático con un procesador que ejecuta las instrucciones ejecutables por computadora almacenadas en un medio de almacenamiento legible por computadora que implementa la CMP.

En algunas modalidades, se proporciona un sistema informático para delegar el comportamiento asociado con un contrato instalado en una cadena de bloques a una aplicación distinta a la cadena de bloques. El sistema informático comprende uno o más medios de almacenamiento legibles por computadora que almacenan las instrucciones ejecutables por computadora de un *cryptodelegate* y un servicio de contenedores de *cryptlets* y uno o más procesadores para ejecutar el almacenamiento de instrucciones ejecutables por computadora en uno o más medios de almacenamiento legibles por computadora. El *cryptodelegate* incluye las instrucciones que reciben del código de parte del contrato que ejecuta en una máquina virtual una identidad de un *cryptlet* y un comportamiento solicitado que debe realizar el *cryptlet*, proporciona al servicio de contenedores de *cryptlets* la identidad y el comportamiento solicitado, recibe de parte del servicio de contenedores de *cryptlets* una respuesta generada por el *cryptlet* que realiza el comportamiento solicitado y envía la respuesta al código del contrato. El servicio de contenedores de *cryptlets* incluye las instrucciones que almacenan la información relacionada con los *cryptlets* registrados, que incluye las referencias a los *cryptlets*, recibe del *cryptodelegate* la identidad y el comportamiento solicitado, identifica una computadora anfitriona para ejecutar el *cryptlet* y el *cryptlet*, autentifica el *cryptlet*, proporciona el comportamiento solicitado al *cryptlet*, recibe la respuesta generada por el *cryptlet*, y la envía a la cadena de bloques y verifica la respuesta mediante el *cryptodelegate*. En algunas modalidades, la clave virtual es ejecutada por la máquina virtual. En algunas modalidades, el comportamiento solicitado es enviar eventos al contrato. En algunas modalidades, el *cryptodelegate* incluye además instrucciones para recibir el código del *cryptlet* del contrato y proporcionar el código al servicio de contenedores de *cryptlets* y donde el servicio de contenedores de *cryptlets* verifica el código generando un *hash* del código y verifica que coincida con un *hash* proporcionado por el contrato, registra el código en una cadena de bloques de registro de *cryptlet* y ordena a una computadora anfitriona certificada para que ejecute el *cryptlet* para realizar el comportamiento solicitado. En las modalidades, el *cryptodelegate* registra en una cadena de bloques asociada con el contrato una indicación de cada solicitud y respuesta proporcionada al contrato. En algunas modalidades, el contrato proporciona al *cryptodelegate* una indicación de una computadora anfitriona que debe ejecutar el *cryptlet*. En algunas modalidades, el *cryptlet* se ejecuta dentro de un entorno proporcionado por un contenedor de *cryptlets*.

En algunas modalidades, el contenedor de *cryptlets* se ejecuta dentro de un entorno de enclave protegido para una operación segura y a prueba de manipulaciones. En algunas modalidades, el contrato proporciona al *cryptodelegate* a la información de configuración para el *cryptlet*. En algunas modalidades, la información de configuración incluye una definición de interfaz para el *cryptlet* y la especificación de los atributos del *cryptlet*.

En algunas modalidades, se proporciona un método realizado por un sistema informático. El método recibe de un contrato inteligente que está siendo ejecutado por una máquina virtual una solicitud para registrar un *cryptlet* de contrato para realizar el comportamiento en nombre del contrato inteligente, el código del *cryptlet* de contrato y una indicación de una computadora anfitriona certificada para ejecutar el código del *cryptlet* de contrato. El método verifica el código del *cryptlet* de contrato. El método almacena el código del *cryptlet* de contrato en una cadena de bloques de registro de *cryptlets* y en el almacenamiento de datos. El método recibe del contrato inteligente una solicitud para que el *cryptlet* de contrato realice el comportamiento. El método ordena que el *cryptlet* de contrato se ejecute en la computadora anfitriona certificada. El método envía al *cryptlet* de contrato que se ejecuta en la computadora anfitriona certificada la solicitud para realizar el comportamiento. En algunas modalidades, las comunicaciones con el contrato inteligente y el *cryptlet* de contrato se realizan a través de canales seguros. En algunas modalidades, la verificación del código incluye generar un *hash* del código y comparar el *hash* con una clave pública asociada con el código. En las modalidades, la solicitud que se recibe del contrato inteligente se recibe a través de un *cryptodelegate* que se ejecuta por la máquina virtual. En las modalidades, el *cryptodelegate* registra en una cadena de bloques del contrato inteligente una indicación de comunicación con el contrato inteligente y el *cryptodelegate*. En algunas modalidades, la máquina virtual se ejecuta en un nodo de una cadena de bloques y la computadora anfitriona certificada es una computadora que es externa a los nodos de la cadena de bloques. En algunas modalidades, la cadena de bloques de registro de *cryptlets* está separada de una cadena de bloques del contrato inteligente. En las modalidades, los miembros de un consorcio aprueban el código del contrato inteligente y el *cryptlet*.

En algunas modalidades, se proporciona un método realizado por uno o más sistemas informáticos. El método ordena la ejecución mediante una máquina virtual de un contrato inteligente asociado con una cadena de bloques. Durante la ejecución del contrato inteligente, el método envía a través de un *cryptodelegate* a un servicio de contenedores de *cryptlets* una solicitud para delegar un comportamiento a un *cryptlet* que se ejecuta en una computadora anfitriona certificada. Durante la ejecución del servicio de contenedores de *cryptlets*, el método identifica una computadora anfitriona para ejecutar el código del *cryptlet*, ordena a la computadora anfitriona identificada que ejecute el código del *cryptlet* para realizar el comportamiento delegado, recibe del *cryptlet* una respuesta al comportamiento solicitado y envía la respuesta al contrato inteligente en la cadena de bloques validada por el *cryptodelegate*. En las modalidades, el *cryptodelegate* establece un canal de comunicaciones seguro para el servicio de contenedores de *cryptlets* y el servicio de contenedores de *cryptlets* establece un canal de comunicaciones seguro con la computadora anfitriona. En algunas modalidades, el *cryptlet* es

un *cryptlet* de contrato, el código del *cryptlet* se proporciona o identifica por el contrato inteligente, y la computadora anfitriona se identificada por el contrato inteligente. En algunas modalidades, el *cryptlet* es un *cryptlet* de utilidad que está disponible para los contratos inteligentes de diferentes consorcios.

Aunque la materia objeto se ha descrito en un lenguaje específico para las características estructurales y/o acciones metodológicas, debe entenderse que la materia objeto definida en las reivindicaciones adjuntas no está necesariamente limitada a las características o acciones específicas descritas anteriormente. Más bien, las características y acciones específicas descritas anteriormente se describen como formas ejemplares de la implementación de las reivindicaciones.

10

15

20

25

30

35