(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 680 851**

(51) Int. CI.:

H04L 9/00 (2006.01)
G06Q 30/06 (2012.01)
G06Q 20/36 (2012.01)
G06Q 20/02 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **16.02.2017 PCT/IB2017/050865**

(87) Fecha y número de publicación internacional: **31.08.2017 WO17145019**

(96) Fecha de presentación y número de la solicitud europea: **16.02.2017 E 17708587 (5)**

(97) Fecha y número de publicación de la concesión europea: **11.04.2018 EP 3257191**

(54) Título: **Registro y método de gestión automática para contratos inteligentes ejecutados por cadena de bloques**

(30) Prioridad:

23.02.2016 GB 201603123
23.02.2016 GB 201603125
23.02.2016 GB 201603117
23.02.2016 GB 201603114
01.04.2016 GB 201605571
15.11.2016 GB 201619301

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.09.2018

(73) Titular/es:

NCHAIN HOLDINGS LIMITED (100.0%)
Fitzgerald House 44 Church Street
St. John's, AG

(72) Inventor/es:

WRIGHT, CRAIG STEVEN y
SAVANAH, STEPHANE

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Registro y método de gestión automática para contratos inteligentes ejecutados por cadena de bloques.

La presente invención se refiere, en general, a protocolos informáticos y, más concretamente, a la verificación, ejecución y/o cumplimiento de procesos controlados por condiciones como, por ejemplo, aquellos relacionados con contratos. La invención es particularmente apropiada para su uso con una red de cadena de bloques y puede usarse para sacar ventaja con un contrato inteligente.

Una cadena de bloques es un sistema informático distribuido y descentralizado compuesto de bloques no cambiables que, a su vez, están formados por transacciones. Cada bloque contiene un *hash* del bloque previo de modo que los bloques se encadenan juntos para crear un registro de todas las transacciones que se han escrito en la cadena de bloques desde el comienzo. Las transacciones contienen pequeños programas conocidos como *scripts* incorporados en sus entradas y salidas, los cuales especifican cómo y quién puede acceder a las salidas de la transacción. Cada transacción no utilizada (a la que se hace referencia como UTXO) puede utilizarse como una entrada en una nueva transacción.

La aplicación más ampliamente conocida de la tecnología de cadena de bloques es la tecnología de contabilidad Bitcoin, aunque se han propuesto y desarrollado otras implementaciones de cadena de bloques. Mientras, en la presente memoria, puede hacerse referencia a Bitcoin en aras de la conveniencia e ilustración, debe notarse que la invención no se encuentra limitada al uso con la cadena de bloques Bitcoin e implementaciones alternativas de cadena de bloques caen dentro del alcance de la invención.

La tecnología de cadena de bloques se conoce para el uso de la implementación de una criptomoneda. Sin embargo, más recientemente, los emprendedores digitales han comenzado a explorar tanto el uso del sistema de seguridad criptográfico en el que Bitcoin se basa, como los datos que pueden almacenarse en la Cadena de Bloques, para implementar nuevos sistemas. Estos incluyen, pero no se encuentran limitados a:

- Almacenamiento de metadatos
- Implementación de *tokens* digitales
- Implementación y gestión de contratos.

Uno de los principales problemas con la gestión de contratos moderna es que tiende a ser *ad-hoc*, con almacenes locales y copias de contratos que se mantienen de forma manual. Como resultado, los protocolos de ordenador conocidos como "contratos inteligentes" han comenzado a llamar la atención dado que pueden permitir la ejecución o cumplimento automático de un contrato, ya sea de forma parcial o en su totalidad. Los contratos inteligentes pueden proveer beneficios como, por ejemplo, seguridad mejorada y costes de transacción reducidos. Sin embargo, mientras existen soluciones técnicas conocidas que pretenden asegurar que dichos contratos no pueden modificarse una vez almacenados, no hay ningún registro público generalmente aceptado para verificar la validez del contrato, a saber, si aún se encuentra abierto o ha finalizado.

Por consiguiente, es deseable proveer un mecanismo implementado por ordenador que pueda controlar la visibilidad pública de la existencia de un contrato, y facilitar la capacidad de las partes relevantes de gestionar, exigir el cumplimiento y mantener procesos basados en el cumplimiento como, por ejemplo, contratos en una manera automática (a saber, por máquina antes que gestión humana). De manera importante, el presente mecanismo proveerá una capacidad técnica de especificar condiciones de control y activadores para comportamientos definidos dentro del contrato.

Debe notarse que la invención definida y descrita en la presente memoria no se encuentra limitada para su uso con contratos en el sentido legal de la palabra. El contrato puede ser un documento, archivo u otro mecanismo que define un conjunto de comportamientos que pueden activarse bajo condiciones especificadas. La condición de control puede cumplirse de forma pública. La invención no debe considerarse limitada al uso dentro de contextos legales o comercialmente orientados, y el término "contrato" no debe interpretarse en dicho sentido restrictivo. Por ejemplo, el contrato puede ser un ticket de un tren o aerolínea, o para un lugar de conciertos, y en el cual se imprime un código de acceso como, por ejemplo, un código de barras legible por máquina para proveer el desbloqueo de una barrera.

Por consiguiente, se provee una invención en la presente memoria según se define en las reivindicaciones anexas.

Según un aspecto de la presente invención, se provee un método implementado por ordenador para controlar la visibilidad y/o cumplimiento de un contrato, el método comprendiendo las etapas de:

(a) almacenar un contrato en un depósito basado en ordenador;

(b) difundir una transacción a una cadena de bloques, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador indicativo de la ubicación donde se almacena el contrato; y

c) renovar o continuar el contrato mediante:

la generación de una nueva clave mediante el uso de datos relacionados con una clave previa asociada al contrato;

la generación de un *script* que comprende la nueva clave, la ubicación del contrato y un *hash* del contrato; y

el pago de una cantidad de moneda al *script*.

Mediante la renovación o continuación del contrato mediante la generación de una nueva clave mediante el uso de datos relacionados con una clave previa asociada al contrato, la generación de un *script* que comprende la nueva clave, la ubicación del contrato y un *hash* del contrato, y el pago de una cantidad de moneda al *script*, ello provee la ventaja de que dado que la nueva clave se relaciona con la clave previa, las partes autorizadas pueden ver el contrato fuente por medio de su conexión con el contrato renovado o continuado y, de esta manera, permitir la verificación del contrato continuado, sin pérdida alguna de seguridad o privacidad. Se provee la ventaja adicional de que la memoria y capacidad de procesamiento pueden reducirse mediante el almacenamiento del contrato en un depósito fuera de la cadena basado en ordenador (a saber, no forma parte de la cadena de bloques), sin pérdida de seguridad o privacidad, dado que la clave asociada con el contrato renovado o continuado se asocia a la clave del contrato fuente.

En la situación de "continuación", la UTXO puede utilizarse mediante su envío al "nuevo" contrato continuado. Sin embargo, puede ser posible cancelar el contrato existente mediante la utilización de la salida antes del tiempo de bloqueo y, por consiguiente, cancelar todo el contrato.

La invención puede proveer un método y sistema implementados por ordenador para controlar la visibilidad y/o cumplimiento de un contrato. "Visibilidad" puede significar cómo y para quién la existencia y/o contenidos del contrato están disponibles o accesibles. El contrato puede ser un "contrato inteligente". El método puede ser un método de contrato inteligente automático. Puede ser un método para automatizar el proceso de monitoreo de la existencia, validez y/o cumplimiento del contrato. Dado que el contrato puede representarse en la forma de al menos parte de una transacción de la cadena de bloques, puede hacerse referencia a la invención como un método/sistema de tokenización. Los metadatos en la transacción pueden proveer un *token* implementado por la cadena de bloques que se usa para representar y/o acceder a un contrato.

La invención puede proveer un método/sistema que permite el almacenamiento de un contrato en un depósito (registro), donde un *hash* del contrato puede usarse como una clave de consulta para encontrar el contrato.

El método puede comprender las etapas de:

almacenar un contrato en un depósito basado en ordenador; y

difundir una transacción a una cadena de bloques, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador indicativo de la ubicación donde se almacena el contrato.

El contrato puede interpretarse como abierto o válido hasta que la UTXO se utiliza en la cadena de bloques. La cadena de bloques puede o puede no ser la cadena de bloques de Bitcoin. Esta provee el beneficio de un mecanismo innovador para representar el estado o validez de un contrato en una cadena de bloques según se representa por la UTXO.

El método puede comprender la etapa de usar un proceso, agente u otra entidad implementada por ordenador fuera de la cadena para observar el estado de la cadena de bloques y comportarse de cierta manera según si la salida se encuentra actualmente no utilizada o no. El proceso puede disponerse para interpretar la salida no utilizada como un indicador del estado del contrato. En otras palabras, mientras la salida permanece dentro de la lista UTXO en la cadena de bloques, a saber, la transacción aún no se utiliza, esta puede usarse para indicar la validez o estado "abierto" del contrato indicado o al que se hace referencia por los metadatos. El contrato puede considerarse completo (finalizado) una vez que la UTXO se ha utilizado. Dicha condición puede establecerse dentro del contrato. Sin embargo, una vez que la UTXO se ha utilizado, los metadatos pueden continuar conteniendo un indicador o referencia al contrato y un *hash* del contrato de modo que el contrato puede retener su función.

El método puede comprender la etapa de publicar la existencia del contrato. Ello puede lograrse mediante las siguientes etapas:

- El Emisor del Contrato puede crear un nuevo Documento de Contrato y publicarlo en el Depósito. La ubicación del almacén y el *hash* seguro de dicho documento pueden almacenarse para uso posterior;
- crear un *script* de rescate que cubre el documento de contrato que se asegura, en una m de n estructura multifirma donde:
  - m es al menos uno; y
  - n es m más el número de bloques de metadatos
- incluida al menos una clave pública en el *script*; esta puede ser la clave pública del Emisor del Contrato. Sin embargo, otras firmas pueden requerirse también
- pagar una cantidad de moneda p.ej., Bitcoin al *script*, preferiblemente a través de una transacción P2SH
- esperar hasta que la transacción se haya publicado en la Cadena de Bloques y extraer el ID de la transacción para la transacción publicada
- crear una nueva transacción, con un tiempo de bloqueo establecido en el tiempo de expiración del contrato, pagar la salida de la transacción otra vez al *hash* de clave pública; O

Para un contrato de dirección continuada: usar un agente informático automático para detectar la transacción en la cadena de bloques y esperar hasta el tiempo de expiración del contrato antes de activar el código para continuarlo en un nuevo contrato; O

Para un contrato basado en la finalización (donde x de y entidades acuerdan que se ha cumplido con el contrato): crear una m de n transacción multifirma y emitirla a dichas entidades para la cofirma tras la finalización.

El depósito puede ser un recurso de almacenamiento fuera del bloque. En otras palabras, el depósito puede no formar parte de la propia cadena de bloques. El depósito basado en ordenador puede ser o comprender un servidor. El depósito puede ser una base de datos u otra instalación de almacenamiento provista en un recurso basado en ordenador. El Depósito puede indexarse y permitir, de esta manera, su búsqueda. El depósito puede comprender una Tabla de *Hash* Distribuida. El contrato puede almacenarse en o en asociación con la Tabla de *Hash* Distribuida (DHT, por sus siglas en inglés).

La transacción puede además comprender un rescate determinista o dirección de *script* de bloqueo. La dirección puede ser una dirección pago al *hash* del *script* (P2SH, por sus siglas en inglés). Por consiguiente, la existencia de un contrato (o elemento definido dentro de un contrato) puede hacerse públicamente disponible mediante el uso de una transacción que se publica en la cadena de bloques mediante el uso de una dirección pago al *hash* del *script* que puede determinarse o proveerse por el emisor del contrato; y/o los metadatos del contrato.

El método puede además comprender la etapa de finalizar el contrato mediante la difusión de una transacción (adicional) a la cadena de bloques para utilizar la salida (UTXO). La transacción adicional puede comprender una entrada que es la salida (UTXO); y un *script* de desbloqueo que comprende una firma; los metadatos; y una clave pública. Esto puede proveer el beneficio de la finalización automática del contrato, mediante el uso de una transacción de cadena de bloques para utilizar la salida.

El contrato puede definir: i) al menos una condición; y ii) al menos una acción cuyo cumplimiento depende de la evaluación de la condición. La condición puede ser un ensayo que puede evaluarse para saber si es verdadera o falsa. La condición puede ser parte (p.ej., una cláusula) del contrato. La finalización o cumplimiento de la condición puede requerirse para el cumplimiento del contrato. La condición puede completarse si la evaluación determina que es verdadera.

Los metadatos pueden comprender i) una dirección o representación de una dirección de donde se almacena el contrato en el depósito basado en ordenador; y/o ii) un *hash* del contrato.

El método puede comprender la etapa de observar el estado de la cadena de bloques. Este puede comprender la búsqueda de la cadena de bloques para encontrar una transacción que contiene la UTXO. Puede comprender la etapa de verificar si el contrato ha finalizado mediante la determinación de si la transacción no utilizada UTXO se encuentra en la lista de salidas de transacciones no utilizadas para la cadena de bloques. Dicho proceso de monitoreo o verificación puede ser automático. Puede llevarse a cabo por un recurso o agente informático adecuadamente programado. Puede ser, sustancialmente, según se describe más abajo en la sección titulada "Agente Informático Ilustrativo para su uso con la invención". El agente puede llevar a cabo una acción según el

estado de utilizado o no utilizado de la UTXO. Por consiguiente, el estado de la UTXO puede controlar o influir en el comportamiento de un agente informático fuera del bloque.

El método puede comprender la etapa de difundir una transacción a la cadena de bloques que comprende una instrucción de utilizar la salida en una fecha y/u hora específicos. La instrucción puede ser una instrucción CheckLockTimeVerify.

El acceso a algunos o todos los contenidos del contrato puede restringirse a al menos una parte designada autorizada. En otras palabras, la autorización puede requerirse con el fin de acceder a ver parte de o todo el contrato. En algunas realizaciones, mecanismos de protección pueden aplicarse al propio contrato. Por ejemplo, una o más porciones del archivo pueden protegerse pero el contenido general puede ser público. Dicha protección parcial puede aplicarse tanto a la encriptación de la información dentro del contrato como al *hash* que detecta cambios en su contenido.

El contrato puede comprender un Autómata Finito Determinista (DFA, por sus siglas en inglés) para implementar el contrato. El Autómata Finito Determinista puede definirse mediante el uso de un esquema de codificación. El Autómata Finito Determinista puede implementarse mediante el uso de:

i) al menos una transacción de cadena de bloques, preferiblemente mediante el uso de un lenguaje de *scripts*;

ii) un agente informático dispuesto para monitorear el estado de la cadena de bloques (esto puede ser según se describe en la sección de más abajo titulada "Agente Informático Ilustrativo para su uso con la invención"); y/o

iii) un conjunto de instrucciones para una cartera digital.

Según otro aspecto de la presente invención, se provee un método implementado por ordenador para controlar la visibilidad y/o cumplimiento de un contrato, el método comprendiendo las etapas de:

(a) almacenar un contrato en un depósito basado en ordenador;

(b) difundir una transacción a una cadena de bloques, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador indicativo de la ubicación donde se almacena el contrato; y

(c) generar un subcontrato derivado del contrato, en donde el subcontrato se asocia a una dirección determinista y se genera mediante:

iii) el uso de una nueva clave pública derivada mediante el uso de una semilla;

iv) almacenar el subcontrato en el depósito con una referencia al contrato, y difundir una transacción a la cadena de bloques que comprende un *script* que incluye la referencia; y/o

v) añadir una referencia al subcontrato a los metadatos del contrato existente.

Mediante la generación de un subcontrato derivado del contrato, en donde el subcontrato se asocia a una dirección determinista y se genera mediante el uso de una nueva clave pública derivada mediante el uso de una semilla, el almacenamiento del subcontrato en el depósito con una referencia al contrato, y la difusión de una transacción a la cadena de bloques que comprende un *script* que incluye la referencia y/o adición de una referencia al subcontrato a los metadatos del contrato existente, se provee la ventaja de que los subcontratos pueden gestionarse, de manera independiente, sin pérdida de seguridad o privacidad, dado que se vinculan, de manera criptográfica, al contrato fuente. Además, los recursos de memoria y procesamiento pueden minimizarse mediante el almacenamiento del subcontrato en un depósito fuera del bloque.

El método puede incluir el uso de un agente basado en ordenador para monitorear la cadena de bloques y/o ejecutar acciones según el contenido del contrato. Dicho agente puede ser, sustancialmente, como se describe más abajo en la sección titulada "Agente Informático Ilustrativo para su uso con la invención".

La invención puede proveer también un sistema implementado por ordenador dispuesto para llevar a cabo cualquiera de las etapas del método mencionadas más arriba, o cualquier realización del método descrito en la presente memoria. La invención puede proveer un sistema implementado por ordenador para controlar la visibilidad y/o cumplimiento de un contrato, el sistema comprendiendo:

un depósito basado en ordenador dispuesto para almacenar un contrato; y

una cadena de bloques que comprende una transacción, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador que representa la ubicación donde se almacena el contrato.

Los metadatos también pueden almacenar un *hash* del contrato. El contrato puede ser un contrato inteligente.

El depósito puede comprender una base de datos. Este puede comprender una DHT. Puede indexarse y disponerse para la búsqueda. Puede comprender al menos un mecanismo de seguridad para controlar el acceso al contrato.

El sistema también puede comprender un agente o entidad basada en ordenador adecuadamente configurada. El agente puede disponerse para monitorear y/o buscar la cadena de bloques. Puede disponerse para llevar a cabo al menos una acción según el estado de la cadena de bloques. Puede disponerse para determinar si la UTXO se ha utilizado o no. Puede disponerse para llevar a cabo una o más acciones según si la UTXO se ha utilizado o no.

Cualquier característica descrita en la presente memoria en relación con una realización o aspecto puede también usarse en relación con cualquier otra realización o aspecto. Por ejemplo, cualquier característica descrita en relación con el método también puede usarse en relación con el sistema y viceversa.

Una lista no exhaustiva de algunos de los beneficios que pueden proveerse por la invención se provee ahora.

La invención puede proveer una disposición técnica que simplifica la gestión automática de condiciones de control estructuradas, a las cuales puede hacerse referencia en la presente memoria como "contratos". Ello, a su vez, facilita estar de acuerdo con el estado del contrato en caso de disputa. La invención también puede proveer un mecanismo para mantener un registro público seguro de contratos en una manera que permita la determinación automática de su validez por ordenador, y la liberación de sus detalles a entidades autorizadas tras la validación. Por consiguiente, la invención puede proveer un mecanismo de control de seguridad mejorada que permite o prohíbe el acceso a un recurso en una manera inteligente.

La invención también provee la capacidad de publicar un contrato a una audiencia mediante un sistema informático de modo que los detalles del contrato pueden restringirse a entidades autorizadas solamente, pero el conocimiento de la existencia del contrato se conoce públicamente. En otras palabras, puede ser de público conocimiento que existe un contrato entre A y B y esto puede verificarse públicamente, pero cualquier otro dato diferente de su existencia se restringe a partes autorizadas (que pueden, normalmente, ser A y B solamente).

También provee un mecanismo implementado por ordenador que permite a los contratos ser de tiempo limitado (a saber, expiran después de cierto tiempo o en una fecha dada); limitados a una condición (a saber, expiran una vez que el entregable especificado dentro del contrato se haya cumplido) o de finalización abierta (a saber, continúan renovándose con un período de notificación para finalizarlos).

Ello puede proveer un mecanismo para cursar una notificación para finalizar el contrato de manera pública. Por ejemplo, mediante el uso de nLockTime + CheckLockTimeVerify (CLTV) en una transacción utilizada para 'representar' la expiración.

Puede proveer un mecanismo para estructurar una jerarquía de subcontratos en una manera determinista para permitir que el control de diferentes aspectos del contrato se particione. Por ejemplo, en un proceso de desarrollo de tecnología, la fase de requisitos puede tener un conjunto diferente de activadores de control que la fase de desarrollo.

Dado que la invención puede implementarse en una plataforma de cadena de bloques, y puede extender la funcionalidad de la cadena de bloques de modo que pueda usarse en una manera técnicamente diferente, la invención puede proveer una plataforma o sistema de cadena de bloques mejorado.

La invención puede usarse para convertir cualquier transacción no utilizada (UTXO) en un contrato inteligente como, por ejemplo, para el acceso digital. Por ejemplo, es preciso considerar un escenario en donde un consumidor paga a un comerciante para acceder a un servicio durante un período. Si la dirección de pago del comerciante se implementa como un contrato inteligente, entonces la invención puede usarse para implementar un mecanismo de control de acceso para el servicio. Una verificación puede llevarse a cabo para asegurar que el dinero se ha pagado, y un proceso automático puede usarse para mover el valor al final del período a la cuenta del comerciante.

Estos y otros aspectos de la presente invención serán aparentes a partir de, y se elucidarán con referencia a, la realización descrita en la presente memoria. Ahora se describirá una realización de la presente invención, únicamente a modo de ejemplo, y con referencia a los dibujos anexos, en los cuales:

La Figura 1 muestra una visión general de cómo las transacciones de la cadena de bloques pueden usarse por una realización de la invención para implementar varias tareas relacionadas con el contrato.

La Figura 2a muestra una máquina de estado simple con dos estados: (i) el contrato está abierto y (ii) el contrato está cerrado.

La Figura 2b muestra la definición de metadatos para el escenario de la Figura 2a. Los metadatos se transportan en la salida de transacción (bitcoin) y especifican la ubicación del contrato y prueba de validez (mediante el *hash*).

La Figura 2c muestra una transacción de "emisión" relacionada con el escenario de las Figuras 2a y 2b, que inicialmente almacena el (*hash* del) contrato en la Cadena de Bloques.

La Figura 2d cancela el contrato de las Figuras 2a a 2c mediante la utilización del bitcoin.

La Figura 3a muestra metadatos ilustrativos para un escenario en donde un activo con propiedad oculta se crea y publica en la cadena de bloques.

La Figura 3b muestra una transacción ilustrativa para "financiar" el activo de la Figura 3a. Es decir, con el fin de poner algunos bitcoins en la clave pública del activo de modo que el activo puede financiar sus transacciones (como, por ejemplo, la transacción de publicación que se muestra en 3c).

La Figura 3c muestra una transacción de cadena de bloques ilustrativa para la publicación del activo de las Figuras 3a y 3b.

La Figura 3d muestra una transacción ilustrativa para el cierre del contracto relacionado con las Figuras 3a, b y c. Cuando se requiere la cancelación del contrato, se utiliza la UTXO. En el presente escenario, el requisito ha sido que tanto el Activo como el propietario oculto del activo firmen.

La Figura 4a muestra un modelo de máquina de estado ilustrativo para un escenario que implica un contrato de arrendamiento.

La Figura 4b muestra metadatos ilustrativos para el escenario de la Figura 4a.

La Figura 4c muestra una transacción ilustrativa para publicar la propiedad del activo de las Figuras 4a y 4b en la Cadena de Bloques.

La Figura 5a muestra un modelo de máquina de estado ilustrativo para un escenario en donde un contrato se renueva.

La Figura 5b muestra metadatos ilustrativos para el escenario de la Figura 5a.

La Figura 5c muestra una transacción ilustrativa que puede usarse para publicar el contrato inicial de las Figuras 5a y 5b y la renovación inicial del contrato en la Cadena de Bloques.

La Figura 5d muestra una transacción ilustrativa para la finalización del contrato de las Figuras 5a a 5d.

La Figura 6a muestra un modelo de máquina de estado ilustrativo para un escenario que implica condicionalidad del contrato.

La Figura 6b muestra metadatos ilustrativos para el escenario de la Figura 6a.

La Figura 6c muestra una transacción ilustrativa que puede usarse para crear el contrato inicial y dos subcontratos y publicarlos.

La Figura 6d muestra una transacción ilustrativa para su uso en relación con el escenario de 6a a 6c.

Las Figuras 7 a 13 muestran varios aspectos de una técnica para derivar subclaves de una clave primaria, dicha técnica siendo apropiada para su uso en relación con aspectos de la presente invención.

El cumplimiento de los contratos inteligentes construidos en la Cadena de Bloques puede exigirse a través de la lógica que se incorpora directamente en la transacción bitcoin (a saber, dentro de los *scripts* de bloqueo/desbloqueo) y/o a través de aplicaciones externas basadas en ordenador. Puede hacerse referencia a dichas aplicaciones externas basadas en ordenador como "agentes", "oráculos" o "*bots*". Además, algunas condiciones contractuales pueden exigirse a través de otros elementos de transacción bitcoin como, por ejemplo, el campo nLockTime.

Una invención se describe en la presente memoria donde el contrato se interpreta como uno que permanece en vigor siempre que exista una salida de transacción válida no utilizada UTXO en la cadena de bloques que representa el contrato. Se apreciará que dicho estado no utilizado puede influenciarse y alterarse como resultado de varios mecanismos (p.ej., un agente informático programado) cuyo comportamiento se controla por condiciones o estipulaciones en el propio contrato. Por ejemplo, el contrato estipula que expirará en cierta fecha, o que expirará cuando cierto valor alcance un umbral especificado.

El presente principio de usar salidas de transacciones no utilizadas para representar contratos puede usarse en combinación con otras características como, por ejemplo, técnicas de encriptación. Ello permite la implementación de escenarios y actividades complejas. De manera efectiva, el contexto alrededor de la salida de transacción no firmada UTXO y los metadatos asociados dentro del *script* que permite utilizarlos, permite a la transacción actuar como un indicador o referencia a un depósito fuera de la cadena que contiene los detalles formales del contrato. En la presente memoria, "fuera de la cadena" significa que no es parte de la propia cadena de bloques. Ello provee un mecanismo por medio del cual cualquiera puede usar un componente o herramienta basada en software para determinar si el contrato ha finalizado o es aún válido/abierto mediante la inspección de la cadena de bloques. Una vez que el contrato finaliza, ello se registrará en la cadena de bloques como una salida utilizada en una transacción y estará disponible para la inspección pública. La transacción de cadena de bloques se convierte en un registro permanente, inalterable y público de la existencia y estado actual del contrato.

El depósito (que puede también llamarse un "registro") puede implementarse en una variedad de maneras incluida, por ejemplo, como una tabla de *hash* distribuida (DHT). Un *hash* del contrato puede generarse y almacenarse como metadatos dentro de la transacción de cadena de bloques, y puede servir como la clave de consulta para remitirse al contrato desde la cadena de bloques. Una referencia a la ubicación del contrato también se provee dentro de los metadatos de la transacción. Por ejemplo, puede proveerse el URL para el depósito. Mientras los metadatos están abiertos a la vista pública, el propio contrato puede no estar protegido o puede estar parcialmente.

Características estándares de Bitcoin como, por ejemplo, CheckLockTimeVerify (CLTV, por sus siglas en inglés), pueden permitir que el contrato tenga una expiración formal automática en un punto en el futuro. El uso de la cadena de bloques permite que dicha fecha de expiración sea una cuestión de registro público seguro (inalterable). El presente concepto, en combinación con el uso de múltiples claves de encriptación descritas más abajo, permite al modelo CLTV continuar o renovar automáticamente el contrato a menos que se cancele de forma explícita.

El uso de subclaves deterministas, en combinación con el mecanismo de tokenización descrito en la presente memoria, permite crear subcontratos o planificaciones contra contratos.

Además, el uso de agentes informáticos (oráculos) fuera del bloque permite que la condicionalidad del contrato se incorpore en y modifique por terceros confiables. Ello significa que la acción del agente puede verse influida por condiciones (p.ej., declaraciones "SI") que se proveen dentro de la definición de contrato.

Términos Clave

Los siguientes términos pueden usarse en la presente memoria descriptiva.

- Emisor del contrato:

Esta entidad representa un actor que es responsable de la publicación del contrato en la Cadena de Bloques.

- Parte interesada:

Esta entidad representa un actor que puede necesitar determinar si un contrato particular se encuentra aún en el lugar o no, o puede necesitar determinar los detalles específicos del contrato.

- Depósito:

Esta entidad representa una ubicación que asegura / almacena una representación estructurada del contrato a la que el contrato inteligente de la Cadena de Bloques hace referencia.

- Contraparte del contrato:

Esta entidad representa la contraparte de un contrato específico. Es preciso notar que, en muchos casos, esta entidad no estará presente.

- Contrato:

Este es el documento o archivo estructurado almacenado dentro del depósito y al que se hace referencia desde la Cadena de Bloques. El contrato puede ser cualquier tipo de contrato o acuerdo. Ello puede incluir, por ejemplo, contratos financieros, títulos de propiedad, contratos de servicios y más. Un contrato puede ser público o privado en términos de su contenido. El contrato se formaliza en que se expresa en una manera estructurada mediante el uso de un esquema de codificación.

Modelo de Contrato

Los elementos básicos del modelo de contrato son los siguientes:

- Un esquema de codificación que permite una descripción completa de cualquier tipo de contrato. El esquema puede ser una nueva construcción o puede usar una instalación existente como, por ejemplo, XBRL, XML, JSON (etc.);

- Un DFA (Autómata Finito Determinista) para implementar el Contrato que puede definirse totalmente dentro del esquema de codificación. Este está formado por:

  ◦ Un conjunto de parámetros, y dónde originar dichos parámetros;

  ◦ Un conjunto de definiciones de estado

  ◦ Un conjunto de transiciones entre los estados, incluido el activador para la transición y las normas seguidas durante la transición

  ◦ Tabla de definición de normas.

- Definiciones de los parámetros específicos para la presente instancia del Contrato;

- Mecanismos para asegurar y proteger el Contrato;

- Un "navegador" para permitir que el contrato sea legible por humanos en lenguaje legal formal; y

- Un "cumplidor" para convertir el esquema de codificación en código de oráculo y/o *script* como, por ejemplo, un *script* Bitcoin.

Implementación del contrato

Cuando el Contrato se registra en un depósito, la dirección asociada, p.ej., URL y el *hash* pueden usarse como metadatos dentro de una transacción de la Cadena de Bloques para asociar la transacción en la cadena al propio contrato de control. Ello puede implementarse en una variedad de formas, pero un esquema de codificación apropiado se provee más abajo para completar esto en la sección titulada "Esquema de codificación".

Existe un número de métodos diferentes sobre cómo el DFA contenido dentro de la definición de contrato puede implementarse:

- Como una transacción de la Cadena de Bloques o secuencia de transacción. Varias formas de DFA pueden implementarse directamente dentro del lenguaje de *script* Bitcoin; la persona con experiencia en la técnica comprenderá esto y la presente invención no se encuentra limitada con respecto a la manera en la cual el DFA se implementa mediante transacciones de la cadena de bloques;

- Como un proceso o secuencia de procesos basada en agente (p.ej., oráculo). La sección de más abajo titulada "Agente Informático Ilustrativo para su uso con la invención" describe el proceso básico para definir y ejecutar un agente apropiado para monitorear la Cadena de Bloques y, posiblemente, otras fuentes externas.

- Como un conjunto de instrucciones para una Cartera digital. En el presente contenido, una cartera inteligente es simplemente, de manera eficaz, un proceso de oráculo local que puede manejar ciertas condiciones contractuales como, por ejemplo, una asignación de entradas de transacción a una transacción de Cadena de Bloques.

Es preciso notar que una definición de contrato dada puede implementarse como una mezcla de los tres mecanismos de más arriba, donde cada transición de estado de contrato es, de manera eficaz, una implementación separada. Existe un número de métodos para crear la implementación a partir de una definición de contrato, incluida la elaboración artesanal de las transacciones / código relevantes.

Publicación de la Existencia del Contrato

Con el fin de publicar la existencia de un contrato (o un elemento definido dentro de un contrato), una transacción Tx se publica en la Cadena de Bloques mediante el uso de una dirección de pago al *hash* del *script* (P2SH). Una transacción P2SH es una en la cual el receptor debe proveer un *script* que concuerde con el *hash* del *script*, y también datos que hagan que la evaluación del *script* sea verdadera, con el fin de que la transacción se utilice. En relación con las realizaciones de la presente invención, el pago al *hash* del *script* (P2SH) puede determinarse inmediatamente a partir de:

- El emisor del contrato; y

- Los metadatos del contrato.

Según algunas realizaciones de la invención, la transacción no utilizada puede interpretarse como un indicador del estado del contrato. Un proceso fuera de la cadena puede disponerse para monitorear la cadena de bloques y comportarse de cierta manera según si la salida se utiliza o no. En otras palabras, mientras dicha salida permanece dentro de la lista UTXO en la cadena de bloques (a saber, la transacción aún no se utiliza), ello indica la validez del contrato indicado o al que se hace referencia por los metadatos. El contrato se considera completo una vez que dicha salida se ha utilizado. Dicha condición (de que el contrato permanece válido/abierto solamente siempre que exista una UTXO para este) puede ser una condición del propio contrato. Sin embargo, no es una estipulación necesaria del protocolo dado que en otras realizaciones una condición de finalización alternativa pueda estar en su lugar. Es preciso notar que incluso después de que la transacción se haya utilizado (y, por lo tanto, ya no existe en la lista UTXO) aún reside permanentemente en la Cadena de Bloques y aún retiene un indicador o referencia al contrato y un *hash* del contrato de modo que puede retener su función incluso después de haberse utilizado.

Subcontratos/Condiciones

Un subcontrato es un contrato que se relaciona directamente con un contrato existente. Una condición es una cláusula dentro de un contrato existente que debe satisfacerse para cumplir con los términos de dicho contrato.

Según una realización de la invención, los subcontratos y condiciones pueden implementarse en la misma manera, a saber, como un contrato que se implementa como una UTXO con una dirección de *script* de rescate determinista. En ambos casos, la entidad puede interpretarse como completa cuando la UTXO se utiliza (en el caso de una condición, ello indica que la condición se ha cumplido). Según se establece más arriba, los metadatos aún contendrán un indicador o referencia a la ubicación de la entidad dentro del depósito, y también un *hash* de esta. Por lo tanto, en otras realizaciones, el subcontrato o condición pueden permanecer en existencia y retener funcionalidad incluso después de que la salida se haya utilizado, según las condiciones contractualmente especificadas.

Existe una cantidad de mecanismos que pueden usarse para crear la dirección determinista para una condición o subcontrato:

- Derivar una nueva clave pública mediante el uso de información de la semilla;

- Crear y publicar el subcontrato, con una referencia al contrato marco, dentro del depósito y mediante el uso de este como la referencia de metadatos; y

- Añadir la referencia de la condición / subcontrato a los metadatos del contrato existente.

Protección del Contrato

La representación formal del contrato (a saber, el documento o archivo que especifica el contenido del contrato) puede asegurarse de varias maneras según las necesidades formales de dicho contrato específico, aunque en todos los casos un registro público de la existencia del contrato se publicará en la Cadena de Bloques contenida dentro del registro de metadatos (es preciso ver la sección titulada "Esquema de codificación" para detalles de una estructura de metadatos específica).

Desde dicho registro de cadena de bloques, las entidades autorizadas podrán aprender la ubicación de la representación formal, junto con el *hash* para determinar que la representación formal no se ha modificado desde que la transacción se ha publicado.

Sin embargo, es posible asegurar además la propia representación formal a través de un número de métodos:

- El propio depósito del documento puede presentar mecanismos de control de acceso; y

- El propio Contrato puede protegerse a través de técnicas de encriptación estándares que limitan el acceso a aquellas entidades con acceso a las claves de desencriptación relevantes.

En muchos casos, el propio Contrato tendrá protección parcial. Por ejemplo, algunas secciones dentro del archivo pueden protegerse mientras el contenido general es público, p.ej., los detalles de cómo implementar un préstamo a tasa fija se publican pero el conocimiento de quién ha solicitado el préstamo, por qué monto y a qué tasa solo lo conocen las partes contratantes.

Dicha protección parcial se aplica tanto a la encriptación de la información dentro del contrato como al *hash* que detecta cambios en su contenido.

Para un número de contratos, los detalles del contrato pueden modificarse durante su duración y esto no debe requerir la reemisión del propio contrato. Ello puede lograrse mediante la determinación del alcance del *hash* en un subconjunto del contrato. Un ejemplo donde esto puede ser útil es en la implementación de un fondo de inversión. El contrato que constituye la base del fondo de inversión no puede cambiar, pero el beneficiario del fondo puede

modificarse a través de la venta del contrato. En una realización, el registro de los cambios puede lograrse mediante el uso de subcontratos.

Finalización del Contrato

Dado que la Cadena de Bloques provee un registro permanente e inalterable de transacciones, un contrato no puede finalizarse simplemente eliminando el documento de Contrato asociado. Ello significa que el depósito seguro del contrato debe tener las mismas normas de almacenamiento y retención que la propia Cadena de Bloques que se admite a través de un número de mecanismos estándares. Ello significa que la solución debe presentar un mecanismo para detectar la expiración de un contrato a través del registro de la Cadena de Bloques directamente.

El método de finalización se define como una condición en el contrato y puede llevarse a cabo en una variedad de maneras, todas las cuales se cubren conceptualmente por la presente invención. En una realización preferida de la invención, la finalización se gestiona a través de la utilización de la UTXO que representa el contrato.

Para un número de tipos de contrato, la expiración del contrato puede publicarse simultáneamente con la publicación del propio Contrato. De manera eficaz, se crean dos transacciones, una para publicar el contrato y obtener la salida de transacción que representa el contrato y una segunda para utilizar dicha salida. Dicha segunda transacción tiene un CheckLockTimeVerify establecido en ella para utilizar la salida en una fecha futura dada (que representa el fin del contrato). Según el comentario previo, esta es nuestra manera estándar pero no la única manera.

Dicha autoutilización puede extenderse para admitir la continuación de un contrato (por ejemplo, contratos que se extienden de forma automática por un período adicional de doce meses si no se cancelan). En la presente situación, la UTXO se utiliza mediante su envío al "nuevo" contrato continuado. Sin embargo, es posible cancelar el contrato antiguo mediante la utilización de la salida antes del tiempo de bloqueo y, por consiguiente, cancelar todo el contrato.

Modelo de Caso de Uso

La Figura 1 muestra una visión general de un modelo de caso de uso según una realización de la invención. El presente modelo de caso de uso ilustrativo demuestra cómo las transacciones Bitcoin estándares pueden usarse para implementar elementos del DFA directamente dentro de los *scripts* del Bitcoin. Ejemplos de casos de uso clave se proveen ahora en aras de la ilustración.

Creación del Contrato

El Emisor del Contrato (que es el actor primario en el presente ejemplo) desea publicar un contracto en la Cadena de Bloques para visibilidad pública. Dicho proceso se ilustra en la Tabla 1:

Etapa	Detalles
100.10	El Emisor del Contrato crea un nuevo Documento de Contrato y lo publica en un Depósito, almacena la ubicación del almacén y el *hash* seguro de dicho documento para uso posterior. Es preciso notar que el presente Depósito puede ser público, privado o semiprivado según la naturaleza del propio Documento de Contrato. El Depósito se indexa y, de esta manera, permite su búsqueda por una variedad de atributos.
100.20	El Emisor del Contrato crea un *script* de rescate que cubre el documento de contrato que se asegura, en una m de n estructura multifirma donde: - m es al menos uno; y - n es m más el número de bloques de metadatos (que serán al menos dos). La única clave pública que debe suministrarse siempre a dicho *script* es la del Emisor del Contrato. Sin embargo, según los términos del contrato, también pueden requerirse otras firmas.
100.30	El Emisor del Contrato paga una cantidad de moneda nominal, p.ej., Bitcoin, al *script* de rescate calculada en la etapa 100.20 a través de una transacción P2SH estándar.
100.40	El Emisor del Contrato espera hasta que la transacción se haya publicado en la Cadena de Bloques y extrae el ID de la transacción para la transacción publicada.

Etapa	Detalles
100.50	Para un contrato de duración fija, el Emisor del Contrato entonces crea una nueva transacción, con un tiempo de bloqueo establecido en el tiempo de expiración del contrato, y paga la salida desde la etapa 100.40 otra vez al *hash* de clave pública del Emisor del Contrato. Para un contrato de duración continuada, un agente basado en ordenador puede recoger la transacción y esperar hasta el tiempo de expiración del contrato antes de activar el caso de uso de "continuación" de la tabla 3 de más abajo para continuarlo en el contrato. Para un contrato basado en la finalización (donde x de y entidades acuerdan que se ha cumplido con el contrato), una m de n transacciones multifirma se crea y emite a dichas entidades para la cofirma tras la finalización).

Existen dos realizaciones o variaciones clave del presente escenario que se explican en detalle más abajo:

- Creación de un subcontrato a partir de un contrato existente
- Continuación de un contrato existente en uno nuevo (renovación)

Creación de un Subcontrato

En la presente situación, el Emisor del Contrato desea crear un subcontrato a partir de un contracto existente. Dicho proceso se ilustra en la Tabla 2:

Etapa	Detalles
150.10	El Emisor del Contrato crea una nueva subclave a partir de su clave pública usada para crear el contrato primario mediante el uso de un valor de semilla en la derivación de la información de subclave del contrato primario. Esta puede ser cualquier derivación que el Emisor del Contrato desea (y con la que se ha comprometido), pero ejemplos de semillas apropiadas pueden incluir: - ID / índice de transacción de la UTXO del contrato creado en la etapa 100.40; o - *Hash* del *script* de rescate creado en la etapa 100.20. Debe notarse que el presente ejemplo supone que la clave pública a la que se hace referencia más arriba será la clave pública del Emisor del Contrato; sin embargo, la persona con experiencia en la técnica apreciará que no hay nada para prevenir que esta sea la subclave derivada (a saber, un subcontrato de un subcontrato).
150.20	Según la naturaleza del subcontrato que se está creando, el Emisor del Contrato: - Usa la ubicación y el *hash* del documento de contrato marco; o - Crea un nuevo Documento de Contrato con un enlace al contrato marco incorporado en este, almacena la ubicación del documento y protege el *hash* de dicho documento para uso posterior; o - Crea un nuevo Documento de Contrato con un enlace al contrato marco incorporado en este, más una lista de los campos del Documento de Contrato original que se cubre. De manera eficaz, este es un documento que especifica que dicho subcontrato cubre secciones específicas de otro documento antes que duplicar la información original. Es preciso notar que el presente Depósito puede ser público, privado o semiprivado según la naturaleza del propio Documento de Contrato.

Etapa	Detalles
150.30	El Emisor del Contrato crea un *script* de rescate que cubre el documento de contrato que se está asegurando, en una m de n estructura multifirma donde: - m es al menos uno; y - n es m más el número de bloques de metadatos (que serán al menos dos). La única clave pública que debe suministrarse siempre a dicho *script* es la del Emisor del Contrato. Sin embargo, según los términos del contrato, también pueden requerirse otras firmas.
150.40	El Emisor del Contrato paga una cantidad de moneda nominal, p.ej., Bitcoin, al *script* de rescate calculada en la etapa 150.30 a través de una transacción P2SH (pago al *hash* del *script*) estándar.
150.50	El Emisor del Contrato espera hasta que la transacción se haya publicado en la Cadena de Bloques y extrae el ID de la transacción para la transacción publicada.
150.60	Para un subcontrato de duración fija, el Emisor del Contrato entonces crea una nueva transacción, con un tiempo de bloqueo establecido en el tiempo de expiración del contrato, y paga la salida de la etapa 150.50 otra vez al *hash* de clave pública del Emisor del Contrato.

Según una o más realizaciones, el subcontrato puede monitorearse de manera independiente. Por ejemplo, es preciso considerar un contrato de construcción de propiedad donde se requiere la aprobación de un topógrafo y el contrato establece "sujeto a la aprobación de <x>". Con el fin de implementar esto, se crea la etapa 150.60 y se circula a <x> para la firma. El *script* de repago no es de tiempo bloqueado sino que se crea como un m de n elemento multifirma donde el signatario requerido es <x>. En algunas realizaciones, la transacción tendrá dos salidas: la tarifa a <x> más el pago de la UTXO generado en la etapa 150.50.

Caso de uso a modo de ejemplo: Renovación de contrato existente

En el presente caso de uso, el Emisor del Contrato desea continuar un contrato existente en uno nuevo. Un proceso ilustrativo se provee en la tabla 3:

Etapa	Detalles
175.10	El Emisor del Contrato comprobará la Cadena de Bloques para determinar si el contrato se ha cancelado o no mediante la validación si la UTXO previa se ha utilizado o no. Si se ha utilizado, el proceso finaliza.
175.20	El Emisor del Contrato crea una nueva subclave a partir de su clave pública usada para crear el contrato primario mediante el uso de esta como un valor de semilla en la derivación de la información de subclave de la secuencia de contrato primario. Esta puede ser cualquier derivación determinista que el Emisor del Contrato desee (y con la que se ha comprometido), pero puede ser: - Número de secuencia (p.ej., instancia "1" continuada); o - Rango de fechas para el contrato continuado Lo establecido más arriba supone que la clave pública mencionada más arriba será la clave pública del Emisor del Contrato, pero, en la práctica, no hay nada para prevenir que esta sea una subclave derivada (a saber, un subcontrato de un subcontrato). Es preciso ver la sección titulada "Método de generación de subclave" para un ejemplo de cómo puede crearse la subclave.
175.30	El Emisor del Contrato toma la ubicación y el *hash* del documento de contrato existente. Es preciso notar que el presente Depósito puede ser público, privado o semiprivado según la naturaleza del propio Documento de Contrato.

Etapa	Detalles
175.40	El Emisor del Contrato crea un *script* de rescate que cubre el documento de contrato que se está asegurando, en una m de n estructura multifirma donde: - m es al menos uno; y - n es m más el número de bloques de metadatos (que serán al menos dos). Las dos claves públicas que deben suministrarse siempre a dicho *script* son la del Emisor del Contrato y la del Cliente. Sin embargo, según los términos del contrato, también pueden requerirse otras firmas.
175.50	El Emisor del Contrato paga una cantidad nominal de Bitcoin al *script* de rescate calculada en la etapa 175.40 a través de una transacción P2SH estándar.
175.60	El Emisor del Contrato espera hasta que la transacción se haya publicado en la Cadena de Bloques y extrae el ID de la transacción para la transacción publicada.
175.70	Un proceso (como, por ejemplo, una implementación basada en oráculo o *bot*) recogerá la transacción y esperará hasta el tiempo de expiración del contrato antes de reactivar al proceso de "continuación" de la tabla 3 para continuarlo nuevamente si este no se ha cancelado.

Ejemplo: Verificación del Contrato

En el presente caso de uso, una Parte Interesada desea confirmar que hay un contrato en existencia para cubrir la actividad sobre la que está solicitando información. Dicho proceso se muestra en la tabla 4:

Etapa	Detalles
200.10	La Parte Interesada comprobará la Cadena de Bloques para confirmar si la UTXO relacionada con el contrato en el que está interesada se ha utilizado o no. Donde la UTXO aún no se ha utilizado, entonces el contrato permanece válido. Donde la UTXO aún no se ha utilizado, pero existe una transacción de tiempo de bloqueo pendiente, entonces ello determinará el tiempo de expiración para el contrato. Donde la UTXO se ha utilizado, entonces el contrato se ha completado en algún aspecto.

La variable principal de más arriba supone que la Parte Interesada conoce la transacción que gobierna el contrato a través de alguna otra ruta (en general, aquella es que son el Emisor del Contrato o la Contraparte del Contrato). Sin embargo, cualquier entidad que tenga acceso al Documento de Contrato y conocimiento del Emisor del Contrato podrá llevar a cabo la verificación mediante:

- La derivación del *script* de rescate para la transacción UTXO; y

- La exploración de la Cadena de Bloques para encontrar una UTXO con dicho *hash* de *script* de rescate coincidente.

Ejemplo: Cierre del Contrato

En el presente caso de uso, un Emisor del Contrato o Contraparte del Contrato desea cerrar un contrato existente. Dicho proceso se ilustra en la tabla 5:

Etapa	Detalles
300.10	El Iniciador del cierre comprobará la cadena de bloques para determinar si el contrato se ha cancelado o no mediante la validación de si la UTXO previa se ha utilizado o no. Si se ha utilizado, el proceso finaliza dado que el contrato ya se ha cerrado.
300.20	Si existe una transacción de cierre existente, entonces el iniciador simplemente firmará dicha transacción y

Etapa	Detalles
	la presentará a la Cadena de Bloques.
300.30	Si no hay una transacción de cierre existente, entonces el iniciador creará la transacción con la entrada de transacción siendo la UTXO del último contrato, y el *script* de desbloqueo siendo su firma, los metadatos asociados al contrato y a su clave pública.
300.40	En el punto en el que la transacción se acepta en la Cadena de Bloques, entonces será de público conocimiento que el contrato se ha cerrado (aunque solo los participantes conocerán el motivo específico).

Condiciones Contractuales

El mismo mecanismo descrito más arriba puede usarse para monitorear las condiciones dentro de un contrato dado como, por ejemplo, puntos de comprobación. Por ejemplo, si se determina que un contrato vale 100 BTC, con 20 BTC que se pagarán en el punto de comprobación 1 a 5, entonces el modelo de subcontrato descrito más arriba puede usarse para derivar un contrato marco más cinco subcontratos. Cada uno de dichos subcontratos puede marcarse como completo mediante el uso de los mismos, o diferentes, signatarios (como, por ejemplo, notarios o similares, por ejemplo). De esta manera, un registro público puede mantenerse y mostrar que las condiciones adosadas al contrato se han cumplido. Es posible entonces combinar este concepto con un proceso o aplicación ("*bot*") que puede usarse para activar los pagos de 20 BTC una vez que el contrato se haya marcado como completo.

En aras de la ilustración, algunos escenarios a modo de ejemplo se proveen más abajo, los cuales muestran algunas de las aplicaciones para las cuales la invención puede usarse. En todos dichos escenarios, el contenido del propio contrato se considera irrelevante y no restrictivo.

Escenario a modo de ejemplo 1: Registro Público de un activo

En el presente escenario, Bob decide publicar su propiedad de un activo (p.ej., su casa) en la Cadena de Bloques. Nada más se lleva a cabo en la presente etapa; es simplemente un activo que puede luego usarse en transacciones posteriores. En la presente situación, no hay ninguna fecha de finalización del contrato. La Figura 2a muestra una máquina de estado simple con dos estados: (i) el contrato está abierto y (ii) el contrato está cerrado. La Figura 2b muestra la definición de metadatos transportada en la salida de Transacción de bitcoin y que especifica la ubicación del contrato y prueba de validez mediante el *hash*. La Figura 2c muestra una transacción de "emisión" que inicialmente almacena el contrato en la Cadena de Bloques (aunque, en realidad, solo almacena el *hash*, no el contrato real). La Figura 2d cancela el contrato mediante la utilización del bitcoin.

Escenario a modo de ejemplo 2: Creación y Registro de un Activo con Propiedad Oculta

Esta es una versión ligeramente mejorada del escenario 1 donde Bob desea publicar el activo en la Cadena de Bloques, pero no desea revelar directamente su titularidad.

En esta situación, Bob primero crea una subclave a partir de su clave pública para representar el activo. Dicha subclave se publica entonces como parte de los detalles del activo en la Cadena de Bloques. Nuevamente, en la presente situación, no hay ninguna fecha de finalización para el activo. (Un ejemplo detallado se provee más abajo para una manera en la cual la subclave puede generarse. Es preciso ver la sección de más abajo titulada "Método de generación de subclave").

La máquina de estado para el presente escenario es igual a aquella para el escenario 1, según se muestra en la Figura 2a. La Figura 3a muestra la definición de metadatos para el presente escenario. Los metadatos se transportan en la salida de Transacción de bitcoin y especifican la ubicación del contrato y prueba de validez (mediante el *hash*). La Figura 3b muestra la transacción para "financiar" el activo. Es decir, poner algunos bitcoins en la clave pública del activo de modo que el activo puede financiar sus transacciones (como, por ejemplo, la transacción de publicación en la Figura 3c). La Figura 3b no muestra la creación de Bob de la subclave del activo dado que no es una transacción Bitcoin.

La Figura 3c muestra la transacción de cadena de bloques para la publicación del activo. La Figura 3d muestra la transacción para el cierre del contrato. Cuando se requiere la cancelación del contrato, la UTXO se utiliza. En la presente situación, el requisito ha sido que tanto el Activo como el propietario oculto del activo firmen.

Escenario a modo de ejemplo 3: Contrato de arrendamiento

En la presente situación ilustrativa, Bob celebra un contrato de arrendamiento con Eve por un término fijo de tres años. Los términos del contrato especificarán un número de pagos. Los detalles del pago no son relevantes con respecto a la presente invención. Sin embargo, el contrato tiene un término fijo sin cláusulas de interrupción.

Este tiene un modelo de máquina de estado simple según se muestra en la Figura 4a. La Figura 4b muestra los metadatos para el presente escenario. La Figura 4c muestra la transacción para publicar la titularidad del activo en la Cadena de Bloques. En primer lugar, Bob provee cierta financiación para el activo, luego el activo se publica a sí mismo.

Escenario a modo de ejemplo 4: Renovación de Contrato

En la presente situación ilustrativa, Bob decide arrendar una casa a Eve de forma anual continuada, donde él necesita cursar notificación con dos meses de antelación para cancelar el arrendamiento en la fecha de renovación, de lo contrario, este se renovará automáticamente. Este tiene un modelo de máquina de estado simple según se muestra en la Figura 5a. La Figura 5b muestra los metadatos para el presente escenario. La Figura 5c muestra la transacción para publicar el contrato inicial y la renovación inicial del contrato en la Cadena de Bloques.

Después del primer año, Bob continúa con el arrendamiento y no la finaliza. Inmediatamente después de la publicación de EVE-S3-T2, ello se recoge entonces por un agente informático automático y se renueva por otro año. Debe notarse que también es posible que ello pueda realizarse por EVE mediante el uso de una lógica interna propia de ella.

Después del segundo año, Bob decide finalizar el arrendamiento y presenta una transacción mediante el uso de la misma entrada que EVE-S3-T3. Sin embargo, dado que dicha transacción no se ha presentado todavía, la entrada no se utiliza y si la transacción de Bob se publica en la Cadena de Bloques primero, esta invalidará EVE-S3-T3. Mientras las cantidades implicadas son triviales, el *bot* no refrendará la transacción a menos que la salida se dirija al *hash* de clave pública de Eve (o lo que establezca, en realidad, el contrato). La transacción para la finalización del contrato de Bob se muestra en la Figura 5d.

Escenario a modo de ejemplo 5: Confidencialidad del Contrato

En la presente situación ilustrativa, Bob celebra un contrato con un grupo de constructores para entregar una nueva propiedad, y especifica una cantidad de condiciones dentro del contrato que requieren la aprobación independiente (la primera siendo la aprobación de los planos por parte de la autoridad de planificación local). Este tiene un modelo de máquina de estado simple según se muestra en la Figura 6a. La Figura 6b muestra los metadatos para el presente escenario. La Figura 6c muestra la transacción en donde Bob crea el contrato inicial y los dos subcontratos (después de derivar la subclave relevante, posiblemente mediante el uso de la técnica de generación de subclave descrita más abajo) y los publica. La Figura 6d muestra la transacción para cuando el permiso de planificación se ha aprobado.

Esquema de Codificación

Los metadatos que se usan para hacer referencia al contrato pueden formatearse en una variedad de formas. Sin embargo, un esquema de codificación apropiado se describe aquí.

Un contrato es transferible si los derechos que define se confieren al tenedor o propietario del contrato. Un ejemplo de un contrato no transferible es uno en el cual los participantes se nombran - es decir, donde los derechos se confieren a una entidad específica nombrada antes que al tenedor del contrato. Solo los contratos transferibles se describen en el presente esquema de codificación.

Un *token* representa un contrato específico que detalla o define derechos conferidos por un contrato. Según la presente invención, el *token* es una representación del contrato en la forma de una transacción bitcoin.

El presente método de codificación usa metadatos que comprenden tres parámetros o artículos de datos. Dichos datos pueden ser indicativos de:

i) una cantidad de acciones disponibles en virtud del contrato (puede hacerse referencia a ello en la presente memoria como "NumShares");

ii) una cantidad de unidades de transferencia que se transferirán de un emisor a al menos un receptor (puede hacerse referencia a ello en la presente memoria como "ShareVal"); y

iii) un factor para calcular un valor para la cantidad de unidades de transferencia (puede hacerse referencia a ello en la presente memoria como una "tasa de asignación/PeggingRate").

Una ventaja del presente esquema de codificación es que puede usarse para encapsular o representar contratos como *tokens* en una cadena de bloques mediante el uso de solamente los tres parámetros descritos más arriba. De

hecho, el contrato puede especificarse mediante el uso de un mínimo de dichos tres artículos de datos. Dado que el presente esquema de codificación puede usarse para cualquier tipo de contrato transferible, algoritmos comunes pueden concebirse y aplicarse. Detalles adicionales de dichos artículos de metadatos se proveen de la siguiente manera.

Un *token* divisible es uno en el cual el valor en una salida de transacción puede subdividirse en cantidades más pequeñas asignadas a lo largo de múltiples *tokens* (a saber, asignadas a lo largo de múltiples transacciones). El arquetipo es moneda fiduciaria tokenizada. Los contratos divisibles se definen como aquellos que especifican una PeggingRate diferente a cero. Para contratos divisibles, el valor tokenizado transferido en la salida de transacción se relaciona con el valor del bitcoin (BTC) subyacente mediante PeggingRate. Es decir, el contrato especifica los derechos del tenedor en términos de una tasa de asignación. Para *tokens* no divisibles, no hay PeggingRate y el contrato especifica los derechos del tenedor en términos de un valor fijo (p.ej., como un bono al portador: "el presente contrato es rescatable por exactamente $1000" o un vale "el presente contrato es rescatable por un corte de pelo"). Para contratos no divisibles, el valor BTC de la transacción subyacente es irrelevante para el valor del contrato.

La frase "valor BTC subyacente" se refiere a la cantidad de bitcoins (BTC) adosada a la salida de transacción. En el protocolo Bitcoin, toda salida de transacción debe tener una cantidad BTC diferente de cero para considerarse válida. De hecho, la cantidad BTC debe ser mayor que un mínimo establecido (conocido como "restos") que, al tiempo de la escritura, se establece actualmente en 546 *satoshis*. 1 bitcoin se define como igual a 100 millones de *satoshis*. Dado que las transacciones bitcoin se usan aquí solamente como un medio para facilitar un intercambio de titularidad, la cantidad BTC subyacente real es arbitraria: el valor verdadero reside en las especificaciones del contrato. En teoría, cada *token* puede transportarse por los restos.

Según el presente esquema de codificación, específicamente para *tokens* divisibles, el valor BTC subyacente no tiene un significado: soporta una relación con el valor del contrato mediante PeggingRate. PeggingRate es arbitraria y se elige para mantener pequeña la cantidad BTC subyacente. La razón para usar PeggingRate antes que simplemente toda transacción de *token* subyacente con restos es que el protocolo de la presente invención facilita la divisibilidad: cuando un *token* se divide en varias salidas de transacción de cantidades más pequeñas no es necesario ajustar el contrato original. Más bien, el valor de contrato de cada *token* subdividido se calcula simplemente según PeggingRate y la cantidad subdividida de valor BTC subyacente.

Un *token* limitado es uno en el cual un valor de emisión total es fijo (o "limitado") por un número de acciones fijo diferente a cero según se define por una cantidad llamada NumShares. Por lo tanto, no pueden emitirse más acciones en un contrato limitado. Por ejemplo, un contrato para la propiedad de parte de un caballo de carrera se limita al 100% del caballo de carrera (p.ej., 100 acciones al 1% cada una o 10 acciones al 10% cada una, etc.). Un contrato ilimitado implica que el emisor puede suscribir emisiones adicionales de acciones, por ejemplo mediante la adición de la cantidad requerida de moneda fiduciaria a su Cuenta de Reserva. NumShares debe establecerse explícitamente en todos los contratos. Los contratos limitados deben tener NumShares > 0; los contratos ilimitados se denotan mediante el establecimiento de NumShares = 0.

El ejemplo arquetípico es una reserva de moneda (análoga a una reserva de oro) de modo que el valor total mantenido en la cuenta bancaria de reserva concuerda con el valor total en pagarés en existencia (a saber, *tokens* no rescatados). Este concepto se extiende más allá de las reservas de moneda para incluir inventario de existencias. Por ejemplo, un emisor de *tokens* de camisetas impresas con licencia puede comenzar con un inventario de 10.000 camisetas disponibles y puede emitir un *token* divisible para representar dichas 10.000 camisetas (donde, digamos, cada acción = 1 camiseta). El *token* original puede subdividirse y cada *token* subdividido será rescatable para un número de camisetas según el valor BTC subyacente de la salida de transacción según se define por PeggingRate. Si la demanda aumenta, sin embargo, el emisor puede decidir emitir acciones adicionales (a saber, aumentar el número de acciones en circulación en (digamos) otras 10.000). En dicho caso, le corresponde al emisor depositar 10.000 camisetas adicionales en su cuenta de reserva (a saber, almacén de existencias) con el fin de suscribir la emisión adicional. Por consiguiente, el número total de camisetas disponibles (donde las existencias actúan como "cuenta de reserva") en cualquier momento = el número de total de acciones no rescatadas.

PeggingRates solo se aplica a contratos divisibles, en donde el valor de una acción (representada por una cantidad llamada ShareVal) se asigna a la cantidad BTC subyacente. Por ejemplo, el contrato puede especificar que el emisor promete rescatar el *token* a una tasa de $10.000 para cada 1 BTC subyacente. Ello significará (por ejemplo) que una transacción con un valor de salida subyacente tokenizado de 15.400 *satoshis* será rescatable por $1,54. Un valor de 0 para PeggingRate indica que el contrato no es divisible (a saber, solo puede transferirse en su totalidad, como un bono al portador). Cuando PeggingRate se establece en 0 (lo cual significa un *token* no divisible) el valor BTC subyacente no es relevante para el valor del contrato y puede establecerse en cualquier cantidad. Normalmente, en el presente caso, es deseable mantener la cantidad BTC subyacente tan pequeña como sea posible (a saber, establecida en restos) para minimizar los costes operativos.

NumShares es el número total (fijo) de acciones disponibles en el contrato (Limitado). Para contratos limitados NumShares debe ser un número total mayor que cero. Para contratos ilimitados NumShares no es fijo dado que más acciones pueden emitirse en cualquier momento (siempre que se suscriban), lo cual se denota mediante el establecimiento del valor en 0.

Una acción se define como la unidad de transferencia y ShareVal es el valor de dicha unidad. Por ejemplo, para la moneda fiduciaria, la unidad de transferencia puede establecerse en 1 céntimo. O, por ejemplo, puede establecerse en 50 céntimos, en cuyo caso las transferencias solo pueden ejecutarse en "lotes" de 50 céntimos. ShareVal también puede expresarse como un porcentaje: por ejemplo, si un criador quiere vender un caballo de carrera en 10 acciones iguales, entonces ShareVal = 10%. ShareVal debe ser > 0 y debe definirse en el contrato.

TotalIssuance representa el valor total de acciones emitidas. Este valor solo se refiere a contratos limitados dado que para los contratos ilimitados la emisión no es fija y más acciones pueden emitirse. Si las acciones se expresan como un porcentaje, entonces TotalIssuance = 100% por definición.

Para contratos limitados, NumShares, ShareVal y TotalIssuance se relacionan de la siguiente manera:

$$\text{NumShares x ShareVal} = \text{TotalIssuance.}$$

Un valor de 0 para TotalIssuance implica que es un contrato ilimitado. Un ejemplo de un contrato ilimitado es una moneda fiduciaria (entonces TotalIssuance se establece en 0); ejemplos de contratos limitados son: (i) monedas conmemorativas de edición limitada (1000 acuñadas, donde 1 acción = 1 moneda): TotalIssuance = 1000 x 1 = 1000 monedas; y (ii) asientos en un lugar con tickets, donde TotalIssuance = número total de asientos disponibles.

La circulación se define como el valor total de *tokens* no utilizados (a saber, según se determina por las transacciones en UTXO - salida de transacción no utilizada). El conjunto total de todas las transacciones no utilizadas se mantiene en una lista disponible para todos los nodos bitcoin. Por ejemplo, si un emisor inicialmente emite $10.000 como *tokens* tipo moneda fiduciaria y con el tiempo el valor de $5500 de *tokens* se rescata, entonces la circulación = $4500 (siendo el valor de *tokens* no rescatados). Este valor debe conciliar con el saldo en la cuenta de reserva asociada.

Método de Generación de Subclave

Más arriba, la Tabla 3 y los escenarios a modo de ejemplo se refieren a situaciones donde es ventajoso generar una subclave a partir de una clave (maestra) original. Un método para lograr esto se provee ahora para la ilustración de una manera en la cual ello puede llevarse a cabo.

La Figura 7 ilustra un sistema 1 que incluye un primer nodo 3 que está en comunicación con un segundo nodo 7 en una red de comunicaciones 5. El primer nodo 3 tiene un primer dispositivo de procesamiento 23 asociado y el segundo nodo 5 tiene un segundo dispositivo de procesamiento 27 asociado. El primer y segundo nodos 3, 7 pueden incluir un dispositivo electrónico como, por ejemplo, un ordenador, teléfono, tableta, dispositivo de comunicación móvil, servidor de ordenador, etc. En un ejemplo, el primer nodo 3 puede ser un dispositivo de cliente (usuario) y el segundo nodo 7 puede ser un servidor. El servidor puede ser un servidor de proveedor de carteras digitales.

El primer nodo 3 se asocia a un primer par de criptografía asimétrica que tiene una clave privada maestra de primer nodo (V_{1C}) y una clave pública maestra de primer nodo (P_{1C}). El segundo nodo (7) se asocia a un segundo par de criptografía asimétrica que tiene una clave privada maestra de segundo nodo (V_{1S}) y una clave pública maestra de segundo nodo (P_{1S}). En otras palabras, el primer y segundo nodos están, cada uno, en posesión de los respectivos pares de claves públicas-privadas.

El primer y segundo pares de criptografía asimétrica para los respectivos primer y segundo nodos 3, 7 pueden generarse durante un proceso de registro como, por ejemplo, el registro de una cartera. La clave pública para cada nodo puede compartirse públicamente como, por ejemplo, en una red de comunicaciones 5.

Con el fin de determinar un secreto común (SC) tanto en el primer nodo 3 como en el segundo nodo 7, los nodos 3, 7 llevan a cabo etapas de métodos 300, 400 respectivos sin comunicar claves privadas en la red de comunicaciones 5.

El método 300 llevado a cabo por el primer nodo 3 incluye determinar 330 una segunda clave privada de primer nodo (V_{2C}) según al menos la clave privada maestra de primer nodo (V_{1C}) y un Valor de Generador (VG). El Valor de Generador puede basarse en un mensaje (M) que se comparte entre el primer y segundo nodos, que puede incluir compartir el mensaje en la red de comunicaciones 5 según se describe en mayor detalle más abajo. El método 300 también incluye determinar 370 una segunda clave pública de segundo nodo (P_{2S}) según al menos la clave pública maestra de segundo nodo (P_{1S}) y el Valor de Generador (VG). El método 300 incluye determinar 380 el secreto común (SC) según la segunda clave privada de primer nodo (V_{2C}) y la segunda clave pública de segundo nodo (P_{2S}).

De manera importante, el mismo secreto común (SC) también puede determinarse en el segundo nodo 7 por el método 400. El método 400 incluye determinar 430 una segunda clave pública de primer nodo (P_{2C}) según la clave pública maestra de primer nodo (P_{1C}) y el Valor de Generador (VG). El método 400 además incluye determinar 470 una segunda clave privada de segundo nodo (V_{2S}) según la clave privada maestra de segundo nodo (V_{1S}) y el Valor de Generador (VG). El método 400 incluye determinar 480 el secreto común (SC) según la segunda clave privada de segundo nodo (V_{2S}) y la segunda clave pública de primer nodo (P_{2C}).

La red de comunicaciones 5 puede incluir una red de área local, una red de área amplia, redes móviles, red de comunicaciones radioeléctricas, Internet, etc. Dichas redes, donde los datos pueden transmitirse mediante un medio de comunicaciones como, por ejemplo, cable eléctrico, fibra óptica, o de manera inalámbrica, pueden ser susceptibles a escuchas clandestinas como, por ejemplo, por un dispositivo de escucha secreta 11. El método 300, 400 puede permitir al primer nodo 3 y segundo nodo 7 determinar, de forma independiente, un secreto común sin transmitir el secreto común en la red de comunicaciones 5.

Por consiguiente, una ventaja es que el secreto común (SC) puede determinarse de forma segura e independiente por cada nodo sin tener que transmitir una clave privada en una red de comunicaciones 5 potencialmente insegura. A su vez, el secreto común puede usarse como una clave secreta (o como la base de una clave secreta).

Los métodos 300, 400 pueden incluir etapas adicionales. Es preciso ver la Figura 11. El método 300 puede incluir, en el primer nodo 3, generar un mensaje firmado (MF1) según el mensaje (M) y la segunda clave privada de primer nodo (V_{2C}). El método 300 además incluye enviar 360 el primer mensaje firmado (MF1), en la red de comunicaciones, al segundo nodo 7. A su vez, el segundo nodo 7 puede llevar a cabo la etapa de recibir 440 el primer mensaje firmado (MF1). El método 400 también incluye la etapa de validar 450 el primer mensaje firmado (MF2) con la segunda clave pública de primer nodo (P_{2C}) y autenticar 460 el primer nodo 3 según el resultado de la validación del primer mensaje firmado (MF1). De manera ventajosa, ello permite al segundo nodo 7 autenticar que el supuesto primer nodo (donde el primer mensaje firmado se ha generado) es el primer nodo 3. Ello se basa en la suposición de que solo el primer nodo 3 tiene acceso a la clave privada maestra de primer nodo (V_{1C}) y, por lo tanto, solo el primer nodo 3 puede determinar la segunda clave privada de primer nodo (V_{2C}) para generar el primer mensaje firmado (MF1). Se apreciará que, de forma similar, un segundo mensaje firmado (MF2) puede generarse en el segundo nodo 7 y enviarse al primer nodo 3 de modo que el primer nodo 3 puede autenticar el segundo nodo 7 como, por ejemplo, en un escenario entre pares.

El compartir el mensaje (M) entre el primer y segundo nodos puede lograrse en una variedad de formas. En un ejemplo, el mensaje puede generarse en el primer nodo 3 que luego se envía, en la red de comunicaciones 5, al segundo nodo 7. De manera alternativa, el mensaje puede generarse en el segundo nodo 7 y luego enviarse, en la red de comunicaciones 5, al segundo nodo 7. En algunos ejemplos, el mensaje (M) puede ser público y, por lo tanto, puede transmitirse en una red 5 no segura. Uno o más mensajes (M) pueden almacenarse en un almacén de datos 13, 17, 19. La persona con experiencia en la técnica se dará cuenta de que compartir el mensaje puede lograrse en una variedad de formas.

De manera ventajosa, un registro para permitir la recreación del secreto común (SC) puede mantenerse sin que el registro tenga, por sí solo, que almacenarse de forma privada o transmitirse de manera segura.

Método de registro 100, 200

Un ejemplo de un método de registro 100, 200 se describirá con referencia a la Figura 9, donde el método 100 se lleva a cabo por el primer nodo 3 y el método 200 se lleva a cabo por el segundo nodo 7. Ello incluye establecer el primer y segundo pares de criptografía asimétrica para los respectivos primer y segundo nodos 3, 7.

Los pares de criptografía asimétrica incluyen claves privadas y públicas asociadas como, por ejemplo, aquellas usadas en la encriptación de clave pública. En el presente ejemplo, los pares de criptografía asimétrica se generan mediante el uso de la Criptografía de Curva Elíptica (ECC, por sus siglas en inglés) y propiedades de funciones de curva elíptica.

En el método 100, 200, ello incluye que el primer y segundo acuerdan 110, 210 sobre un sistema ECC común y usan un punto base (G). (Nota: puede hacerse referencia al punto base como un Generador Común, pero el término "punto base" se usa para evitar confusión con el Valor de Generador VG). En un ejemplo, el sistema ECC común puede basarse en secp256K1 que es un sistema ECC usado por Bitcoin. El punto base (G) puede seleccionarse, generarse de manera aleatoria, o asignarse.

Volviendo, ahora, al primer nodo 3, el método 100 incluye establecerse 110 en el sistema ECC común y punto base (G). Ello puede incluir recibir el sistema ECC común y punto base del segundo nodo 7, o un tercer nodo 9. De manera alternativa, una interfaz de usuario 15 puede asociarse al primer nodo 3, por medio de lo cual un usuario puede proveer, de forma selectiva, el sistema ECC común y/o punto base (G). En incluso otra alternativa, uno o ambos del sistema ECC común y/o punto base (G) pueden seleccionarse, de manera aleatoria, por el primer nodo 3. El primer nodo 3 puede enviar, en la red de comunicaciones 5, una notificación indicativa del uso del sistema ECC

común con un punto base (G) al segundo nodo 7. A su vez, el segundo nodo 7 puede establecerse 210 mediante el envío de una notificación indicativa de un reconocimiento del uso del sistema ECC común y punto base (G).

El método 100 también incluye el primer nodo 3 que genera 120 un primer par de criptografía asimétrica que incluye la clave privada maestra de primer nodo (V_{1C}) y la clave pública maestra de primer nodo (P_{1C}). Ello incluye generar la clave privada maestra de primer nodo (V_{1C}) según, al menos en parte, un entero aleatorio en un rango permisible especificado en el sistema ECC común. Ello también incluye determinar la clave pública maestra de primer nodo (P_{1C}) según una multiplicación de punto de curva elíptica de la clave privada maestra de primer nodo (P_{1C}) y el punto base (G) según la fórmula:

$$P_{1C} = V_{1C} \times G \qquad \text{(Ecuación 1)}$$

Por consiguiente, el primer par de criptografía asimétrica incluye:

V_{1C} : La clave privada maestra de primer nodo que se mantiene secreta por el primer nodo.

P_{1C}: La clave pública maestra de primer nodo que se conoce públicamente.

El primer nodo 3 puede almacenar la clave privada maestra de primer nodo (V_{1C}) y la clave pública maestra de primer nodo (P_{1C}) en un primer almacén de datos 13 asociado al primer nodo 3. En aras de la seguridad, la clave privada maestra de primer nodo (V_{1C}) puede almacenarse en una porción segura del primer almacén de datos 13 para asegurar que la clave permanece privada.

El método 100 además incluye enviar 130 la clave pública maestra de primer nodo (P_{1C}), en la red de comunicaciones 5, al segundo nodo 7, como se muestra en la Figura 9. El segundo nodo 7 puede, al recibir 220 la clave pública maestra de primer nodo (P_{1C}), almacenar 230 la clave pública maestra de primer nodo (P_{1C}) en un segundo almacén de datos 17 asociado al segundo nodo 7.

De manera similar al primer nodo 3, el método 200 del segundo nodo 7 incluye generar 240 un segundo par de criptografía asimétrica que incluye la clave privada maestra de segundo nodo (V_{1S}) y la clave pública maestra de segundo nodo (P_{1S}). La clave privada maestra de segundo nodo (V_{1S}) también es un entero aleatorio dentro del rango permisible. A su vez, la clave pública maestra de segundo nodo (P_{1S}) se determina por la siguiente fórmula:

$$P_{1S} = V_{1S} \times G \qquad \text{(Ecuación 2)}$$

Por consiguiente, el segundo par de criptografía asimétrica incluye:

V_{1S} : La clave privada maestra de segundo nodo que se mantiene secreta por el segundo nodo.

P_{1S}: La clave pública maestra de segundo nodo que se conoce públicamente.

El segundo nodo 7 puede almacenar el segundo par de criptografía asimétrica en el segundo almacén de datos 17. El método 200 además incluye enviar 250 la clave pública maestra de segundo nodo (P_{1S}) al primer nodo 3. A su vez, el primer nodo 3 puede recibir 140 y almacenar 150 la clave pública maestra de segundo nodo (P_{1S}).

Se apreciará que, en algunas alternativas, las respectivas claves maestras públicas pueden recibirse y almacenarse en un tercer almacén de datos 19 asociado al tercer nodo 9 (como, por ejemplo, un tercero confiable). Ello puede incluir un tercero que actúa como un directorio público como, por ejemplo, una autoridad de certificación. Por consiguiente, en algunos ejemplos, la clave pública maestra de primer nodo (P_{1C}) puede solicitarse y recibirse por el segundo nodo 7 solo cuando se requiere la determinación del secreto común (SC) (y viceversa).

Las etapas de registro pueden necesitar solamente ocurrir una vez como un establecimiento inicial.

Inicio de sesión y determinación del secreto común por el primer nodo 3

Un ejemplo de determinación de un secreto común (SC) se describirá ahora con referencia a la Figura 10. El secreto común (SC) puede usarse para una sesión, tiempo, transacción, u otro propósito particular entre el primer nodo 3 y el segundo nodo 7 y puede no ser deseable, o seguro, usar el mismo secreto común (SC). Por consiguiente, el secreto común (SC) puede cambiarse entre diferentes sesiones, tiempos, transacciones, etc.

Lo siguiente se provee para la ilustración de la técnica de transmisión segura que se ha descrito más arriba.

Generación de un mensaje (M) 310

En el presente ejemplo, el método 300 llevado a cabo por el primer nodo 3 incluye generar 310 un mensaje (M). El mensaje (M) puede ser aleatorio, pseudoaleatorio, o definido por el usuario. En un ejemplo, el mensaje (M) se basa en el tiempo Unix y en un *nonce* (un valor arbitrario). Por ejemplo, el mensaje (M) puede proveerse como:

$$\text{Mensaje } (M) = \text{UnixTime} + \text{nonce} \qquad \text{(Ecuación 3)}$$

En algunos ejemplos, el mensaje (M) es arbitrario. Sin embargo, se apreciará que el mensaje (M) puede tener valores selectivos (como, por ejemplo, Tiempo Unix, etc.) que pueden ser útiles en algunas aplicaciones.

El método 300 incluye enviar 315 el mensaje (M), en la red de comunicaciones 3, al segundo nodo 7. El mensaje (M) puede enviarse en una red no segura dado que el mensaje (M) no incluye información sobre las claves privadas.

Determinación de un Valor de Generador (VG) 320

El método 300 además incluye la etapa de determinar 320 un Valor de Generador (VG) según el mensaje (M). En el presente ejemplo, ello incluye determinar un *hash* criptográfico del mensaje. Un ejemplo de un algoritmo de *hash* criptográfico incluye SHA-256 para crear un Valor de Generador (VG) de 256 bits. Es decir:

$$VG = \text{SHA-256}(M) \qquad \text{(Ecuación 4)}$$

Se apreciará que otros algoritmos de *hash* pueden usarse. Ello puede incluir otros algoritmos de *hash* en la familia de Algoritmos de *Hash* Seguros (SHA, por sus siglas en inglés). Algunos ejemplos particulares incluyen instancias en el subconjunto SHA-3, incluidos SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256. Otros algoritmos *hash* pueden incluir aquellos en la familia *Digest* de Mensajes de Evaluación de Primitivas de Integridad RACE (RIPEMD, por sus siglas en inglés). Un ejemplo particular puede incluir RIPEMD-160. Otras funciones *hash* pueden incluir familias basadas en la función *hash* Zémor-Tillich y funciones *hash* basadas en *knapsack*.

Determinación de una segunda clave privada de primer nodo 330

El método 300 luego incluye la etapa 330 de determinar 330 la segunda clave privada de primer nodo (V_{2C}) según la clave privada maestra de segundo nodo (V_{1C}) y el Valor de Generador (VG). Ello puede basarse en una suma escalar de la clave privada maestra de primer nodo (V_{1C}) y el Valor de Generador (VG) según la siguiente fórmula:

$$V_{2C} = V_{1C} + VG \qquad \text{(Ecuación 5)}$$

Por consiguiente, la segunda clave privada de primer nodo (V_{2C}) no es un valor aleatorio sino que, en cambio, se deriva, de manera determinista, de la clave privada maestra de primer nodo. La clave pública correspondiente en el par criptográfico, a saber, la segunda clave pública de primer nodo (P_{2C}), tiene la siguiente relación:

$$P_{2C} = V_{2C} \times G \qquad \text{(Ecuación 6)}$$

La sustitución de V_{2C} de la Ecuación 5 en la Ecuación 6 provee:

$$P_{2C} = (V_{1C} + VG) \times G \qquad \text{(Ecuación 7)}$$

donde el operador "+" se refiere a la suma escalar y el operador "x" se refiere a la multiplicación de punto de curva elíptica. Teniendo en cuenta que el álgebra de criptografía de curva elíptica es distributiva, la Ecuación 7 puede expresarse como:

$$P_{2C} = V_{1C} \times G + VG \times G \qquad \text{(Ecuación 8)}$$

Finalmente, la Ecuación 1 puede sustituirse en la Ecuación 7 para proveer:

$$P_{2C} = P_{1C} + VG \times G \qquad \text{(Ecuación 9.1)}$$

$$P_{2C} = P_{1C} + \text{SHA-256}(M) \times G \qquad \text{(Ecuación 9.2)}$$

En las ecuaciones 8 a 9.2, el operador "+" se refiere a la suma de punto de curva elíptica. Por consiguiente, la segunda clave pública de primer nodo (P_{2C}) correspondiente puede ser derivable dado el conocimiento de la clave pública maestra de primer nodo (P_{1C}) y el mensaje (M). El segundo nodo 7 puede tener dicho conocimiento para determinar, de forma independiente, la segunda clave pública de primer nodo (P_{2C}) como se describirá en mayor detalle más abajo con respecto al método 400.

Generación de un primer mensaje firmado (MF1) según el mensaje y la segunda clave privada de primer nodo 350

El método 300 además incluye generar 350 un primer mensaje firmado (MF1) según el mensaje (M) y la segunda clave privada de primer nodo (V_{2C}) determinada. La generación de un mensaje firmado incluye aplicar un algoritmo de firma digital para firmar digitalmente el mensaje (M). En un ejemplo, esto incluye aplicar la segunda clave privada de primer nodo (V_{2C}) al mensaje en un Algoritmo de Firma Digital de Curva Elíptica (ECDSA, por sus siglas en inglés) para obtener el primer mensaje firmado (MF1). Ejemplos de ECDSA incluyen aquellos basados en sistemas ECC con secp256k1, secp256r1, secp384r1, se3cp521r1.

El primer mensaje firmado (MF1) puede verificarse con la segunda clave pública de primer nodo (P_{2C}) correspondiente en el segundo nodo 7. La presente verificación del primer mensaje firmado (MF1) puede usarse por el segundo nodo 7 para autenticar el primer nodo 3, lo cual se describirá en el método 400 más abajo.

Determinación de una segunda clave pública de segundo nodo 370'

El primer nodo 3 puede luego determinar 370 una segunda clave pública de segundo nodo (P_{2S}). Según se describe más arriba, la segunda clave pública de segundo nodo (P_{2S}) puede basarse al menos en la clave pública maestra de segundo nodo (P_{1S}) y en el Valor de Generador (VG). En el presente ejemplo, dado que la clave pública se determina 370' como la clave privada con multiplicación de punto de curva elíptica con el punto base (G), la segunda clave pública de segundo nodo (P_{2S}) puede expresarse, de manera similar a la Ecuación 6, como:

$$P_{2S} = V_{2S} \times G \qquad \text{(Ecuación 10.1)}$$

$$P_{2S} = P_{1S} + VG \times G \qquad \text{(Ecuación 10.2)}$$

La prueba matemática para la Ecuación 10.2 es igual a la descrita más arriba para derivar la Ecuación 9.1 para la segunda clave pública de primer nodo (P_{2C}). Se apreciará que el primer nodo 3 puede determinar 370 la segunda clave pública de segundo nodo de manera independiente del segundo nodo 7.

Determinación del secreto común 380 en el primer nodo 3

El primer nodo 3 puede entonces determinar 380 el secreto común (SC) según la segunda clave privada de primer nodo (V_{2C}) determinada y la segunda clave pública de segundo nodo (P_{2S}) determinada. El secreto común (SC) puede determinarse por el primer nodo 3 mediante la siguiente fórmula:

$$S = V_{2C} \times P_{2S} \qquad \text{(Ecuación 11)}$$

Método 400 llevado a cabo en el segundo nodo 7

Ahora se describirá el método 400 correspondiente llevado a cabo en el segundo nodo 7. Se apreciará que algunas de estas etapas son similares a aquellas descritas más arriba que se han llevado a cabo por el primer nodo 3.

El método 400 incluye recibir 410 el mensaje (M), en la red de comunicaciones 5, del primer nodo 3. Ello puede incluir el mensaje (M) enviado por el primer nodo 3 en la etapa 315. El segundo nodo 7 entonces determina 420 un Valor de Generador (VG) según el mensaje (M). La etapa de determinar 420 el Valor de Generador (VG) por el segundo nodo 7 es similar a la etapa 320 llevada a cabo por el primer nodo descrita más arriba. En el presente ejemplo, el segundo nodo 7 lleva a cabo la presente etapa de determinación 420 de manera independiente del primer nodo 3.

La siguiente etapa incluye determinar 430 una segunda clave pública de primer nodo (P_{2C}) según la clave pública maestra de primer nodo (P_{1C}) y el Valor de Generador (VG). En el presente ejemplo, dado que la clave pública se determina 430' como la clave privada con multiplicación de punto de curva elíptica con el punto base (G), la segunda clave pública de primer nodo (P_{2C}) puede expresarse, de manera similar a la Ecuación 9, como:

$$P_{2C} = V_{2C} \times G \qquad \text{(Ecuación 12.1)}$$

$$P_{2C} = P_{1C} + VG \times G \qquad \text{(Ecuación 12.2)}$$

La prueba matemática para las Ecuaciones 12.1 y 12.2 es la misma que aquella descrita más arriba para las Ecuaciones 10.1 y 10.2.

El segundo nodo 7 autentica el primer nodo 3

El método 400 puede incluir etapas llevadas a cabo por el segundo nodo 7 para autenticar que el supuesto primer nodo 3 es el primer nodo 3. Según se describe previamente, ello incluye recibir 440 el primer mensaje firmado (MF1) del primer nodo 3. El segundo nodo 7 puede entonces validar 450 la firma en el primer mensaje firmado (MF1) con la segunda clave pública de primer nodo (P_{2C}) que se ha determinado en la etapa 430.

La verificación de la firma digital puede llevarse a cabo según un Algoritmo de Firma Digital de Curva Elíptica (ECDSA) según se describe más arriba. De manera importante, el primer mensaje firmado (MF1) que se ha firmado con la segunda clave privada de primer nodo (V_{2C}) solo debe verificarse correctamente con la segunda clave pública de primer nodo (P_{2C}) correspondiente, dado que V_{2C} y P_{2C} forman un par criptográfico. Dado que dichas claves son deterministas en la clave privada maestra de primer nodo (V_{1C}) y la clave pública maestra de primer nodo (P_{1C}) que se han generado en el registro del primer nodo 3, la verificación del primer mensaje firmado (MF1) puede usarse como una base de autenticación de que un supuesto primer nodo que envía al primer mensaje firmado (MF1) es el mismo primer nodo 3 durante el registro. Por consiguiente, el segundo nodo 7 puede además llevar a cabo la etapa de autenticar (460) el primer nodo 3 según el resultado de la validación (450) del primer mensaje firmado.

El segundo nodo 7 determina el secreto común

El método 400 puede además incluir el segundo nodo 7 que determina 470 una segunda clave privada de segundo nodo (V_{2S}) según la clave privada maestra de segundo nodo (V_{1S}) y el Valor de Generador (VG). De manera similar a la etapa 330 llevada a cabo por el primer nodo 3, la segunda clave privada del segundo nodo (V_{2S}) puede basarse en una suma escalar de la clave privada maestra de segundo nodo (V_{1S}) y el Valor Generador (VG) según las siguientes fórmulas:

$$V_{2S} = V_{1S} + VG \qquad \text{(Ecuación 13.1)}$$

$$V_{2S} = V_{1S} + \text{SHA-256}(M) \qquad \text{(Ecuación 13.2)}$$

El segundo nodo 7 puede entonces, de manera independiente del primer nodo 3, determinar 480 el secreto común (SC) según la segunda clave privada de segundo nodo (V_{2S}) y la segunda clave pública de primer nodo (P_{2C}) según la siguiente fórmula:

$$S = V_{2S} \times P_{2C} \qquad \text{(Ecuación 14)}$$

Prueba del secreto común (SC) determinado por el primer nodo 3 y segundo nodo 7

El secreto común (SC) determinado por el primer nodo 3 es igual al secreto común (SC) determinado en el segundo nodo 7. Ahora se describirá la prueba matemática de que la Ecuación 11 y Ecuación 14 proveen el mismo secreto común (SC).

Volviendo al secreto común (SC) determinado por el primer nodo 3, la Ecuación 10.1 puede sustituirse en la Ecuación 11 de la siguiente manera:

$$S = V_{2C} \times P_{2S} \qquad \text{(Ecuación 11)}$$

$$S = V_{2C} \times (V_{2S} \times G)$$

$$S = (V_{2C} \times V_{2S}) \times G \qquad \text{(Ecuación 15)}$$

Volviendo al secreto común (SC) determinado por el segundo nodo 7, la Ecuación 12.1 puede sustituirse en la Ecuación 14 de la siguiente manera:

$$S = V_{2S} \times P_{2C} \qquad \text{(Ecuación 14)}$$

$$S = V_{2S} \times (V_{2C} \times G)$$

$$S = (V_{2S} \times V_{2C}) \times G \qquad \text{(Ecuación 16)}$$

Dado que el álgebra ECC es conmutativa, la Ecuación 15 y Ecuación 16 son equivalentes, ya que:

$$S = (V_{2C} \times V_{2S}) \times G = (V_{2S} \times V_{2C}) \times G \qquad \text{(Ecuación 17)}$$

El secreto común (SC) y la clave secreta

El secreto común (SC) puede ahora usarse como una clave secreta, o como la base de una clave secreta en un algoritmo de clave simétrica para proteger la comunicación entre el primer nodo 3 y segundo nodo 7.

El secreto común (SC) puede ser en la forma de un punto de curva elíptica (x_s, y_s). Este puede convertirse en un formato de clave estándar mediante el uso de funciones estándares públicamente conocidas acordadas por los nodos 3, 7. Por ejemplo, el valor x_s puede ser un entero de 256 bits que puede usarse como una clave para la encriptación AES_{256}. También puede convertirse en un entero de 160 bits mediante el uso de RIPEMD160 para aplicaciones que requieren dicha clave de longitud.

El secreto común (SC) puede determinarse según se requiera. De manera importante, el primer nodo 3 no necesita almacenar el secreto común (SC) dado que este puede volver a determinarse según el mensaje (M). En algunos ejemplos, los mensajes (M) usados pueden almacenarse en un almacén de datos 13, 17, 19 (u otro almacén de datos) sin el mismo nivel de seguridad que el requerido para las claves privadas maestras. En algunos ejemplos, el mensaje (M) puede estar públicamente disponible. Sin embargo, según alguna aplicación, el secreto común (SC) puede almacenarse en el primer almacén de datos (X) asociado al primer nodo siempre que el secreto común (SC) se mantenga tan seguro como la clave privada maestra de primer nodo (V_{1C}).

De manera ventajosa, esta técnica puede usarse para determinar múltiples secretos comunes que pueden corresponder a múltiples claves de secreto seguras según un par de criptografía de una sola clave maestra.

Jerarquía de Valores de Generador (claves)

Por ejemplo, puede determinarse una serie de Valores de Generador (VGs) sucesivos, donde cada VG sucesivo puede determinarse según el Valor de Generador (VG) precedente. Por ejemplo, en lugar de repetir las etapas 310 a 370 y 410 a 470 para generar claves de un solo propósito sucesivas, mediante acuerdo previo entre los nodos, el Valor de Generador (VG) previamente usado puede volver a usarse como *hash* de manera repetida por ambas partes para establecer una jerarquía de Valores de Generador. De hecho, el Valor de Generador, según el *hash* de un mensaje (M), puede ser un mensaje de próxima generación (M') para la próxima generación de Valor de Generador (VG'). Ello permite a las sucesivas generaciones de secretos compartidos calcularse sin la necesidad transmisiones adicionales de establecimiento de protocolo, en particular, la transmisión de múltiples mensajes para cada generación de secretos comunes. El secreto común de próxima generación (SC') puede calcularse de la siguiente manera.

En primer lugar, tanto el primer nodo 3 como el segundo nodo 7 determinan, de forma independiente, la próxima generación del Valor de Generador (VG'). Esto es similar a las etapas 320 y 420, pero adaptado con las siguientes fórmulas:

$$M' = \text{SHA-256}(M) \qquad \text{(Ecuación 18)}$$

$$VG' = \text{SHA-256}(M') \qquad \text{(Ecuación 19.1)}$$

$$VG' = \text{SHA-256}(\text{SHA-256}(M)) \qquad \text{(Ecuación 19.2)}$$

El primer nodo 3 puede entonces determinar la próxima generación de la segunda clave pública de segundo nodo (P_{2S}') y la segunda clave privada de primer nodo (V_{2C}') similar a las etapas 370 y 330 descritas más arriba, pero adaptadas con las siguientes fórmulas:

$$P_{2S}' = P_{1S} + VG' \times G \quad \text{(Ecuación 20.1)}$$

$$V_{2C}' = V_{1C} + VG' \quad \text{(Ecuación 20.2)}$$

El segundo nodo 7 puede entonces determinar la próxima generación de la segunda clave pública de primer nodo (P_{2C}') y la segunda clave privada de segundo nodo (V_{2S}') similar a las etapas 430 y 470 descritas más arriba, pero adaptadas con las siguientes fórmulas:

$$P_{2C}' = P_{1C} + VG' \times G \quad \text{(Ecuación 21.1)}$$

$$V_{2S}' = V_{1S} + VG' \quad \text{(Ecuación 21.2)}$$

El primer nodo 3 y el segundo nodo 7 pueden entonces determinar, cada uno, el secreto común de próxima generación (SC'). En particular, el primer nodo 3 determina el secreto común de próxima generación (SC') con la fórmula:

$$SC' = V_{2C}' \times P_{2S}' \quad \text{(Ecuación 22)}$$

El segundo nodo 7 determina el secreto común de próxima generación (SC') con la fórmula:

$$SC' = V_{2S}' \times P_{2C}' \quad \text{(Ecuación 23)}$$

Generaciones adicionales (SC", SC"', etc.) pueden calcularse de la misma manera para crear una jerarquía de cadena. La presente técnica requiere que tanto el primer nodo 3 como el segundo nodo 7 realicen un seguimiento del Mensaje (M) original o del Valor de Generador (VG) originalmente calculado, y con cuyo nodo se relaciona. Dado que esto es información públicamente conocida, no hay cuestiones de seguridad con respecto a la retención de la presente información. Por consiguiente, esta información puede mantenerse en "tablas de *hash*" (que vinculan valores *hash* a claves públicas) y distribuirse libremente a lo largo de la red 5 (por ejemplo, mediante el uso de Torrent). Además, si un secreto común (SC) individual en la jerarquía se ve comprometido alguna vez, ello no afecta a la seguridad de otros secretos comunes en la jerarquía siempre que las claves privadas V_{1C}, V_{1S} permanezcan seguras.

Estructura de árbol de las claves

Así como una jerarquía de cadena (lineal) según se describe más arriba, puede crearse una jerarquía en la forma de una estructura de árbol. Con una estructura de árbol, una variedad de claves para diferentes propósitos como, por ejemplo, claves de autenticación, claves de encriptación, claves de firma, claves de pago, etc. pueden determinarse, por medio de lo cual dichas claves se vinculan, todas, a una sola clave maestra mantenida de forma segura. Ello se ilustra mejor en la Figura 12 que muestra una estructura de árbol 901 con una variedad de claves diferentes. Cada una de estas puede usarse para crear un secreto compartido con otra parte. La ramificación de árbol puede lograrse de varias maneras, tres de las cuales se describen más abajo.

(i) Generación de clave maestra

En la jerarquía de cadena, cada "enlace" nuevo (par de claves Públicas/Privadas) se crea mediante la adición de un Mensaje con *hash* repetido de multiplicación a la clave maestra original. Por ejemplo, (solo se muestra la clave privada del primer nodo 3 en aras de la claridad):

$$V_{2C} = V_{1C} + \text{SHA-256}(M) \quad \text{(Ecuación 24)}$$

$$V_{2C}' = V_{1C} + \text{SHA-256}(\text{SHA-256}(M)) \quad \text{(Ecuación 25)}$$

$$V_{2C}'' = V_{1C} + \text{SHA-256}(\text{SHA-256}(\text{SHA-256}(M))) \qquad \text{(Ecuación 26)}$$

... y así sucesivamente.

Con el fin de crear una rama, cualquier clave puede usarse como una clave submaestra. Por ejemplo, V_{2C}' puede usarse como una clave submaestra (V_{3C}) mediante la adición del *hash* a esta como se hace para la clave maestra regular:

$$V_{3C} = V_{2C}' + \text{SHA-256}(M) \qquad \text{(Ecuación 27)}$$

La clave submaestra (V_{3C}) puede tener una clave de próxima generación (V_{3C}'), por ejemplo:

$$V_{3C}' = V_{2C}' + \text{SHA-256}(\text{SHA-256}(M)) \qquad \text{(Ecuación 28)}$$

Ello provee una estructura de árbol 903 mediante el uso del método de generación de clave maestra según se muestra en la Figura 13.

(ii) Asociación Lógica

En el presente método, todos los nodos en el árbol (pares de claves públicas/privadas) se generan como una cadena (o en cualquier otra forma) y las relaciones lógicas entre los nodos en el árbol se mantienen por una tabla en la cual cada nodo en el árbol se asocia simplemente a su nodo padre en el árbol mediante el uso de un indicador. Por consiguiente, el indicador puede usarse para determinar los pares de claves públicas/privadas relevantes para determinar la clave secreta (SC) común para la sesión.

(iii) Multiplicidad de Mensajes

Nuevos pares de claves privadas/públicas pueden generarse mediante la introducción de un nuevo mensaje en cualquier punto en la cadena o árbol. El propio mensaje puede ser arbitrario o puede llevar algún significado o función (p.ej., puede relacionarse con un número de cuenta bancaria "real", etc.). Puede ser deseable que dichos nuevos mensajes para formar los nuevos pares de claves privadas/públicas se retengan de forma segura.

Agente Informático Ilustrativo para su uso con la invención

La presente invención puede utilizar un agente o recurso informático para llevar a cabo aspectos automáticos del proceso de contrato. Un ejemplo de un agente apropiado se provee más abajo, aunque otras implementaciones pueden usarse.

El agente puede funcionar en conjunto con la cadena de bloques, mediante el uso de este como la cinta no borrable en la implementación de una máquina de Turing. Este agente se ejecuta en paralelo con la red de cadena de bloques, y monitorea y maneja la ejecución de un proceso (en bucle). El proceso en bucle se diseña para llevar a cabo una tarea dada como, por ejemplo, la automatización de un proceso o control de un dispositivo o sistema. El presente recurso paralelo monitorea el estado de la cadena de bloques y puede hacer que las transacciones se escriban en la cadena de bloques. En un sentido, este utiliza la Cadena de Bloques como una cinta no borrable de la Máquina de Turing, con las siguientes definiciones y características:

1. la Cadena de Bloques actúa como la cinta de la Máquina de Turing. Cada transacción en la Cadena de Bloques representa una celda en la cinta. Dicha celda puede contener símbolos de un alfabeto finito.

2. La cabeza de la cinta puede leer información de los bloques que ya se han escrito en la Cadena de Bloques.

3. La cabeza de la cinta puede escribir nuevos bloques, que contienen muchas transacciones, hasta el final de la Cadena de Bloques. Sin embargo, no puede escribir en bloques que ya existen. Como tal, la cinta de la Cadena de Bloques es no borrable.

4. Los metadatos para cada transacción pueden almacenarse como parte de una transacción de pago al *hash* del *script* (P2SH) multifirma.

Una función importante del agente es actuar como una entidad automática que monitorea el estado actual de la Cadena de Bloques. También puede recibir una señal o entrada de cualquier fuente fuera del bloque. Según el estado y/o una entrada recibida de la Cadena de Bloques, el agente puede llevar a cabo ciertas acciones. El agente decide qué acción(es) se llevará(n) a cabo. Estas pueden o pueden no implicar acciones en el "mundo real" (a saber, fuera del bloque) y/o acciones en la Cadena de Bloques (como, por ejemplo, crear y difundir nuevas transacciones). La acción que el agente realiza puede activarse por el estado de la Cadena de Bloques. El agente también puede

decidir el próximo conjunto de transacciones que se difundirán a la red Bitcoin y que, posteriormente, se escribirán en la Cadena de Bloques.

Las acciones del agente se ejecutan en paralelo y simultáneamente a la red de la Cadena de Bloques (p.ej., Bitcoin). En un sentido, esto extiende la función del *script* de la cadena de bloques (p.ej., Bitcoin). El presente monitoreo continuo implementa las construcciones de flujo de control "en bucle" haciendo del sistema combinado de agente combinado y sistema de Cadena de Bloques un sistema Turing Completo.

La Máquina de Turing incluye dos pilas:

- Pila de datos: Esta se representa por la Cadena de Bloques según se describe más arriba.

- Pila de control: Esta se representa por la función de agente. Esta almacena información relacionada con la función de flujo de control de repetición.

La separación de la pila de control de la pila de datos provee la ventaja de prevenir que bucles infinitos ocurran dentro del núcleo Bitcoin, y así mitigar ataques por denegación de servicio.

El agente gestiona y ejecuta subrutinas que pueden repetirse en bucle mediante cualquier tipo de construcción en bucle (p.ej., PARA-PRÓXIMO; REPETIR HASTA; etc.). Una realización ilustrativa descrita en la presente memoria incluye un proceso que usa un ejemplo de la construcción "repetir". El usuario puede especificar el índice (*i*) y el límite (*J*). Estos representan el número de iteración actual (normalmente, comienza desde 0) y el número total de iteraciones del bucle de repetición respectivamente.

Para cada iteración:

1. El Índice aumenta en 1. Para la condición de salida, las iteraciones se detendrán cuando el índice alcance el límite.

2. Un bloque de código que contiene una declaración "si la *condición* entonces la *acción*" (ICTA, por sus siglas en inglés) se ejecuta; la acción puede ser cualquier acción en o fuera de la cadena de bloques.

3. Un *hash* criptográfico de la presente subrutina se calcula. Este puede almacenarse en la Cadena de Bloques como parte de una transacción. Dado que el *hash* es único para cada código, ello permitirá la verificación de qué código se ha usado.

La estructura del bucle incluye un bloque de código. Cada bloque de código contiene una declaración "Si la *condición* entonces la *acción*" (ICTA). Esta monitorea el estado actual de la Cadena de Bloques para transacciones que concuerdan con la:

- Condición de inicio o activación (p.ej. cuando se alcanza una fecha particular).

- Condición de repetición (a saber, metadatos o *hash* asociados a la iteración previa).

- Condición de detención (a saber, última iteración del bucle).

La declaración ICTA permite al agente decidir la próxima transacción que se realizará, según el estado actual de la cadena de bloques. Llevar a cabo la próxima transacción implica difundir la transacción en la red Bitcoin y escribir la nueva transacción en la Cadena de Bloques. Ello actúa como un registro de que la presente iteración se ha ejecutado. Una vez que la transacción se ha escrito en la Cadena de Bloques, el Administrador descubrirá, posteriormente, que la iteración previa se ha ejecutado y escrito en la Cadena de Bloques, y ejecutará la próxima iteración. Esto último continúa hasta que el bucle de repetición abandona cuando el índice (*i*) alcanza el límite (*J*) especificado en el bloque de código.

Cada transacción se guarda en la cadena de bloques en una manera que puede volver a usarse. En una implementación de Bitcoin, cada firma en una transacción se anexa con una bandera SIGHASH. Dicha bandera puede asumir diferentes valores, cada uno de los cuales indica si otras partes de la transacción pueden modificarse sin participación del propietario de dicha firma. Una transacción reutilizable tiene la bandera SIGHASH 'SigHash_AnyoneCanPay' en una de las entradas de transacción. Ello permite que cualquiera contribuya a las entradas de la transacción. Este parámetro permite que la función ICTA del agente se ejecute y repita múltiples veces y con diferentes entradas. El uso de la función puede limitarse a partes autorizadas - por ejemplo, mediante derechos de autor de la transacción reutilizable.

La sección "Si *condición*" del bloque de código ICTA puede monitorear cualquier tipo de condición. Ello es similar a otros lenguajes de programación (p.ej., C, C++, Java) y no se encuentra limitado a la información almacenada en la Cadena de Bloques. Por ejemplo, puede monitorear la fecha y hora (a saber, cuándo se alcanza cierta fecha y hora)

o monitorear el clima (a saber, cuándo la temperatura se encuentra por debajo de los 10 °C y cuándo está lloviendo), monitorear las condiciones de un contrato o un fideicomiso (a saber, cuándo la compañía A compra la compañía B).

La sección "Entonces la *acción*" del bloque de código ICTA puede ejecutar un número de acciones. La invención no se encuentra limitada con respecto al número o tipo de acciones que pueden tomarse. La acción no se encuentra limitada a una transacción en la Cadena de Bloques, aunque una transacción que contiene metadatos relacionados con la acción puede escribirse en la Cadena de Bloques.

Los metadatos pueden ser de cualquier forma. Sin embargo, en una realización, los metadatos pueden almacenar un hiperenlace a un archivo que contiene más datos o instrucciones relacionadas con la acción. Los metadatos pueden almacenar un hiperenlace a una tabla de *hash* que contiene más datos o instrucciones relacionadas con la acción junto con un *hash* de la acción que actúa como la clave de consulta para la tabla de *hash*.

La pila de control del agente puede implementarse en un número de maneras que son específicas a las necesidades de cada usuario. Por ejemplo, el bucle de repetición de la pila de control puede basarse en cualquier lenguaje Completo de Turing. Una elección posible de lenguaje es el lenguaje basado en pila estilo Forth. Una ventaja del uso de dicho lenguaje es que mantiene la pila de control coherente en el estilo de programación con los *scripts* Bitcoin que ya se conocen y son de amplio uso.

Uso de la Pila Alterna del *Script* Bitcoin como un Espacio de Almacenamiento de Datos

El *script* Bitcoin contiene comandos, también llamados códigos op, que permiten a los usuarios mover datos en una pila alternativa, conocida como la "pila alt".

Los códigos op son:

- OP_TOALTSTACK - que mueve datos de la parte superior de la pila principal a la parte superior de la pila alt.
- OP_FROMALTSTACK - que mueve datos de la parte superior de la pila alt a la parte superior de la pila principal.

Ello permite que datos de etapas intermedias de cálculos se almacenen en la pila alt, similar a la función de "memoria" que permite que los datos se almacenen en la calculadora. En una realización, la pila alt se usa para configurar *scripts* bitcoin para resolver pequeñas tareas de cálculo y devolver los resultados en el cálculo.

Uso de un Registro de Código para Administrar al Agente

El agente también administra un registro de todos los códigos que posee y ejecuta. Dicho registro se estructura como una tabla de consulta o diccionario que mapea una clave específica hacia un valor específico. El par de clave y valor se representa por el *hash* del bloque de código (H_1) y la dirección IPv6 de donde se almacena el código respectivamente. Con el fin de recuperar el bloque de código mediante el uso de la clave H_1, la tabla de consulta se usa para recuperar el valor asociado (este es la ubicación donde se almacena el código) y, por consiguiente, recupera el código fuente. La implementación del registro de código puede variar.

Metadatos de transacción del código de agente, y regeneración del bucle

La información requerida para regenerar el bucle del agente en una iteración particular se almacena como metadatos en la transacción registrada en la Cadena de Bloques.

De esta manera, una transacción en la cadena de bloques almacena o provee acceso a la información sobre una iteración dada del bucle que se está ejecutando en el agente. Dicha información puede incluir los valores de variables asociadas al bucle como, por ejemplo, índice *i*, y cualquier otra información necesaria como, por ejemplo, valores para parámetros usados en el bloque de código o datos relacionados con la ubicación que especifican dónde puede accederse a información adicional requerida.

Los propios metadatos se almacenan como parte de un *script* de pago al *hash* del *script* (P2SH) multifirma en la transacción. Los metadatos registrados con la transacción también dan la capacidad de registrar un registro de auditoría de cómo el código se ha ejecutado en el pasado.

Existen varias formas en las cuales el agente puede regenerar el bloque de código de bucle de repetición en cada iteración. El bloque de código puede preprogramarse en el propio agente, o puede almacenarse en un archivo privado o públicamente disponible, o almacenarse como una entrada en un archivo de tabla de *hash* privado o público, o una combinación de lo establecido más arriba. El bloque de código puede ser estático con variables preprogramadas o puede ser estático pero contener parámetros que pueden rellenarse. Los parámetros pueden ser valores únicos de cualquier formato de datos, o pueden ser pequeños fragmentos de código, o combinaciones de lo establecido más arriba. Los parámetros pueden rellenarse mediante la recuperación de ellos directamente de los metadatos en una transacción (p.ej., transacción bitcoin) o de una fuente externa como, por ejemplo, una base de

datos interna o un archivo privado/público o tabla de *hash* o cualquier combinación de lo establecido más arriba. Los indicadores de la fuente externa de valores de parámetro pueden almacenarse en los metadatos en una transacción.

Las siguientes etapas proveen un ejemplo de cómo el agente puede regenerar un bloque de código de bucle de repetición en la *i*ésima iteración. En el presente ejemplo, el registro de código es una tabla de *hash* por medio de la cual los valores de *hash* actúan como claves de consulta para la tabla y se almacenan en metadatos en las transacciones.

1. El agente monitorea la Cadena de Bloques para transacciones que contienen *hashes* del bloque de código que concuerdan con entradas en el registro de código.

2. El agente encuentra una transacción que contiene el *hash* (H_1) correspondiente.

3. El agente lee 'Metadata-CodeHash', obtiene el campo CodeHash para obtener H_1 y lo usa para recuperar el código (C_1). Si RIPEMD-160(SHA256(C_1)) es igual a H_1, el código no se ha cambiado y es seguro proceder a la próxima etapa.

4. El agente lee 'Metadata-CodeHash' que almacena el índice *I*, y regenera el código en la $i^{ésima}$ iteración. En otras palabras, el bucle se "recarga" en la iteración apropiada.

5. La firma del Usuario se incluye en el comando P2SH para verificar el origen de los metadatos.

6. El agente lee 'Metadata-OutputHash' y 'Metadata-OutputPointer' para recuperar la salida de las etapas previas, si dichos datos se requieren para dicha iteración del bucle.

Debe notarse que las realizaciones mencionadas más arriba ilustran, antes que limitan, la invención, y que aquellos con experiencia en la técnica podrán diseñar muchas realizaciones alternativas sin apartarse del alcance de la invención según se define por las reivindicaciones anexas. En las reivindicaciones, cualquier signo de referencia colocado entre paréntesis no se interpretará como uno que limita las reivindicaciones. La expresión "que comprende" y "que comprenden", y similares, no excluyen la presencia de elementos o etapas diferentes de aquellas enumeradas en una reivindicación o en la memoria descriptiva en su conjunto. En la presente memoria, "comprende(n)" significa "incluye(n) o consiste(n) en" y "que comprende(n)" significa "que incluye(n) o consiste(n) en". La referencia singular de un elemento no excluye la referencia plural de dichos elementos y viceversa. La invención puede implementarse por medio de hardware que comprende varios elementos distintos, y por medio de un ordenador adecuadamente programado. En una reivindicación de dispositivo que enumera varios medios, varios de dichos medios pueden realizarse por el único artículo de hardware. El mero hecho de que ciertas medidas se incluyan en reivindicaciones dependientes mutuamente diferentes no indica que una combinación de dichas medidas no se pueda usar con el fin de obtener una ventaja.

REIVINDICACIONES

1. Un método implementado por ordenador para controlar la visibilidad y/o cumplimiento de un contrato, el método comprendiendo las etapas de:

(a) almacenar un contrato en un depósito basado en ordenador;

(b) difundir una transacción a una cadena de bloques, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador indicativo de la ubicación donde se almacena el contrato;

(c) interpretar el contrato como abierto o válido hasta que la salida no utilizada (UTXO) se utiliza en la cadena de bloques;

y

d) renovar o continuar el contrato mediante:

la generación de una nueva clave mediante el uso de datos relacionados con una clave previa asociada al contrato;

la generación de un *script* que comprende la nueva clave, la ubicación del contrato y un *hash* del contrato; y

el pago de una cantidad de moneda al *script*.

2. Un método según la reivindicación 1 en donde la transacción además comprende una dirección de *script* de rescate determinista, preferiblemente en donde la dirección de *script* de rescate es una dirección de pago al *hash* del *script* (P2SH).

3. Un método según la reivindicación 2 y que además comprende la etapa de finalizar el contrato mediante la difusión de una transacción adicional a la cadena de bloques para utilizar la salida (UTXO).

4. Un método según la reivindicación 3 en donde la transacción adicional comprende:

una entrada que es la salida (UTXO); y

un *script* de desbloqueo que comprende una firma; los metadatos; y una clave pública.

5. Un método según cualquier reivindicación precedente en donde el contrato define:

i) al menos una condición; y

ii) al menos una acción cuyo cumplimento depende de la evaluación de la condición; y/o

en donde los metadatos comprenden:

i) una dirección o representación de una dirección de donde se almacena el contrato en el depósito basado en ordenador; y/o

ii) un *hash* del contrato.

6. Un método según cualquier reivindicación precedente y que comprende la etapa de:

verificar si el contrato ha finalizado mediante la determinación de si la transacción no utilizada UTXO se encuentra en la lista de salidas de transacciones no utilizadas para la cadena de bloques.

7. Un método según cualquier reivindicación precedente, en donde

i) el contrato se almacena en una Tabla de *Hash* Distribuida (DHT); y/o

ii) el método comprende la etapa de:

difundir una transacción a la cadena de bloques que comprende una instrucción de utilizar la salida en una fecha y/u hora especificadas, preferiblemente en donde la instrucción es una instrucción CheckLockTimeVerify.

8. Un método según cualquier reivindicación precedente en donde:

i) el acceso a algunos o todos los contenidos del contrato se restringe a al menos una parte designada autorizada; y/o

ii) el contrato comprende un Autómata Finito Determinista (DFA) para implementar el contrato;

preferiblemente en donde el Autómata Finito Determinista se define mediante el uso de un esquema de codificación.

9. Un método según la reivindicación 8 en donde el Autómata Finito Determinista se implementa mediante el uso de:

i) al menos una transacción de la cadena de bloques, preferiblemente mediante el uso de un lenguaje de *scripts*;

ii) un agente informático dispuesto para monitorear el estado de la cadena de bloques; y/o

iii) un conjunto de instrucciones para una cartera digital.

10. Un método implementado por ordenador para controlar la visibilidad y/o cumplimiento de un contrato, el método comprendiendo las etapas de:

(a) almacenar un contrato en un depósito basado en ordenador;

(b) difundir una transacción a una cadena de bloques, la transacción comprendiendo:

i) al menos una salida no utilizada (UTXO); y

ii) metadatos que comprenden un identificador indicativo de la ubicación donde se almacena el contrato;

(c) interpretar el contrato como abierto o válido hasta que la salida no utilizada (UTXO) se utiliza en la cadena de bloques;

y

(d) generar un subcontrato derivado del contrato, en donde el subcontrato se asocia a una dirección determinista y se genera mediante:

iii) el uso de una nueva clave pública derivada mediante el uso de una semilla;

iv) el almacenamiento del subcontrato en el depósito con una referencia al contrato, y la difusión de una transacción a la cadena de bloques que comprende un *script* que incluye la referencia; y/o

v) la adición de una referencia al subcontrato a los metadatos del contrato existente.

11. Un método según la reivindicación 10 en donde la transacción además comprende una dirección de *script* de rescate determinista, preferiblemente en donde la dirección de *script* de rescate es una dirección de pago al *hash* del *script* (P2SH).

12. Un método según la reivindicación 11 y que además comprende la etapa de completar el contrato mediante la difusión de una transacción adicional a la cadena de bloques para utilizar la salida (UTXO);

preferiblemente en donde la transacción adicional comprende:

una entrada que es la salida (UTXO); y

un *script* de desbloqueo que comprende una firma; los metadatos; y una clave pública.

13. Un método según cualquiera de las reivindicaciones 10 a 12, en donde:

i) el contrato define:

a) al menos una condición; y

b) al menos una acción cuyo cumplimento depende de la evaluación de la condición;

y/o

ii) los metadatos comprenden:

a) una dirección o representación de una dirección de donde se almacena el contrato en el depósito basado en ordenador; y/o

b) un *hash* del contrato.

14. Un método según cualquiera de las reivindicaciones 10 a 13 y que comprende la etapa de:

verificar si el contrato ha finalizado mediante la determinación de si la transacción no utilizada UTXO se encuentra en la lista de salidas de transacciones no utilizadas para la cadena de bloques.

15. Un método según cualquiera de las reivindicaciones 10 a 14, en donde el contrato se almacena en una Tabla de *Hash* Distribuida (DHT).

16. Un método según cualquiera de las reivindicaciones 10 a 15 y que comprende la etapa de:

difundir una transacción a la cadena de bloques que comprende una instrucción de utilizar la salida en una fecha y/u hora especificadas, preferiblemente en donde la instrucción es una instrucción CheckLockTimeVerify.

17. Un método según cualquiera de las reivindicaciones 10 a 16, en donde:

i) el acceso a algunos o todos los contenidos del contrato se restringe a al menos una parte autorizada designada; y/o

ii) el contrato comprende un Autómata Finito Determinista (DFA) para implementar el contrato;

preferiblemente en donde:

el Autómata Finito Determinista se define mediante el uso de un esquema de codificación; y/o

el Autómata Finito Determinista se implementa mediante el uso de:

i) al menos una transacción de cadena de bloques, preferiblemente mediante el uso de un lenguaje de *scripts*;

ii) un agente informático dispuesto para monitorear el estado de la cadena de bloques; y/o

iii) un conjunto de instrucciones para una cartera digital.

18. Un sistema dispuesto para llevar a cabo el método de cualquier reivindicación precedente.

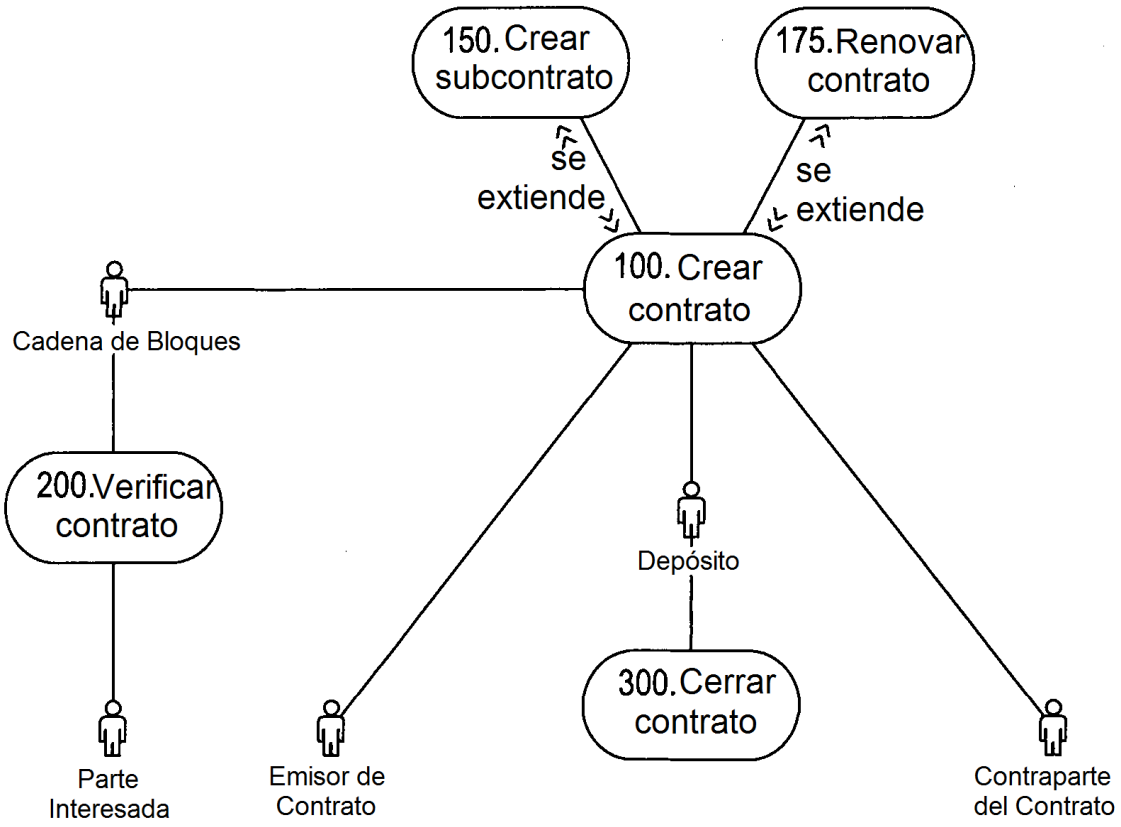

150. Crear subcontrato
175. Renovar contrato
se extiende
se extiende
100. Crear contrato
Cadena de Bloques
200. Verificar contrato
Depósito
300. Cerrar contrato
Parte Interesada
Emisor de Contrato
Contraparte del Contrato

Fig. 1

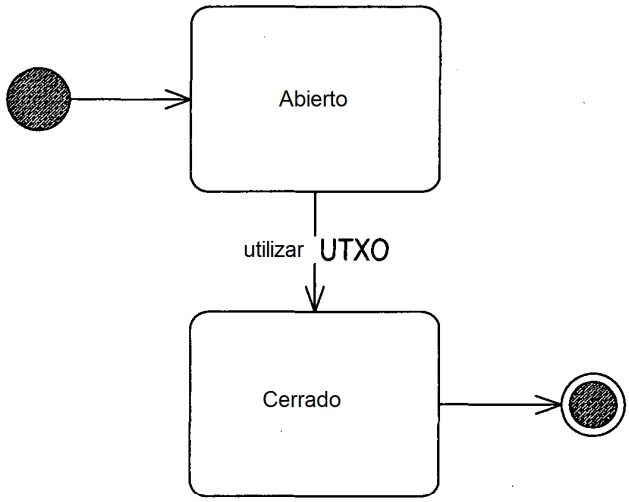


Fig. 2a

Definición de Escenario

Bob: Deseo registrar mi casa en la Cadena de Bloques

Metadatos Base de la Casa

Campo	Subcampo	Bytes	Valor	Comentarios
Metadatos de Activo A	Tipo de Contrato	4	0x0000FF04	Indica unidad
	Indicador de Contrato	16	xxxx.xxxx.xxxx.xxxx (...) .xxxx	Dirección del archivo *Definición de Activo*
	Relleno	12		De reserva
Metadatos de Activo B	*Hash* del Contrato	20	################...	*Hash* del archivo *Definición de Activo* (¡no la tokenización!)
	Jurisdicción	2	EN	Especifica que el activo se cubre por la ley de Inglaterra
	Opciones	2	0x0000	Sin opciones especificadas
	Relleno	8		De reserva

Fig. 2b

Esta es la transacción para publicar la propiedad del activo en la Cadena de Bloques; esta es una transacción bastante directa			
Publicación de Bob			
BOB-S1-T1	ID de Transacción		
Número de versión	Número de Versión		
1	Número de entradas		
<Salida de BTC previo no utilizado de Bob - supongamos 500.000 *satoshi*>	Salida Trans Prev		
IDX-00	Índide de Salidas Trans Prev		
Longitud del *script*	Longitud del *script*		
Sig-Bob PubK-Bob	ScriptSig		
Número de secuencia	Número de secuencia		
1	Número de salidas		
2.000	Valor de salida		
Longitud de *script* de salida	Longitud de *script* de salida		*Script* de rescate; permitir que Bob cancele el contrato en cualquier momento
OP_HASH160 < *hash* de *script* de rescate > OP_EQUAL	*Script* de salida	<<<<	OP_1AssetMetaDataA AssetMetadataB PubK-Bob OP_3 OP_CHECKMULTISIG
498.000	Valor de salida		
Longitud de *script* de salida	Longitud de *script* de salida		
OP_DUP OP_HASH160 <PubK-Bob Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida		
Tiempo de Bloqueo	Tiempo de Bloqueo		

Fig. 2c

Cuando Bob se deshace del activo, o cuando ya no quiere que sea de conocimiento público (o semipúblico), entonces simplemente utiliza la salida de transacción	
Cancelación del Contrato de Bob	
BOB-S1-T2	ID de Transacción
Número de versión	Número de versión
1	Número de entradas
BOB-S1-T1	Salida Trans Prev
IDX-00	Índice de Salidas Trans Prev
Longitud de *script*	Longitud de *Script*
Sig-Bob OP_1AssetMetadataA AssetMetadataB PubK-Bob OP_3 OP_CHECKMULTISIG	ScriptSig
Número de secuencia	Número de secuencia
1	Número de salidas
1.000	Valor de salida
Longitud de *script* de salida	Longitud de *script* de salida
OP_DUP OP_HASH160<PubK-Bob Hash>OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
Tiempo de Bloqueo	Tiempo de Bloqueo

Fig. 2d

Definición de Escenario

Bob: Deseo crear un activo con titularidad oculta y publicarlo en la Cadena de Bloques

Metadatos Base de la Casa

Campo	Subcampo	Bytes	Valor	Comentarios
Metadatos de Activo A	Tipo de Contrato	4	0x0000FF04	Indica unidad
	Indicador de Contrato	16	xxxx.xxxx.xxxx.xxxx (...) .xxxx	Dirección del archivo *Definición de Activo*
	Relleno	12		De reserva
Metadatos de Activo B	*Hash* del Contrato	20	################...	*Hash* del archivo *Definición de Activo* (¡no la tokenización!)
	Jurisdicción	2	EN	Especifica que el activo está cubierto por la ley de Inglaterra
	Opciones	2	0x0000	Sin opciones especificadas
	Relleno	8		De reserva

Fig. 3a

Financiación de Bob del Activo	
BOB-S2-T1	
1	
<Salida previa BTC no utilizada de Bob - supongamos 500.000 *satoshi*>	
IDX-00	
Sig-Bob PubK-Bob	
2	
4.000	Valor de salida
OP_DUP OP_HASH160<PubK-Asset Hash>OP_EQUALVERIFY OP_CHECKSIG	
496.000	
Longitud de *script* de salida	Longitud de *script* de salida
OP_DUP OP_HASH160 <PubK-Bob Hash> OP_EQUALVERIFY OP_CHECKSIG	

Fig. 3b

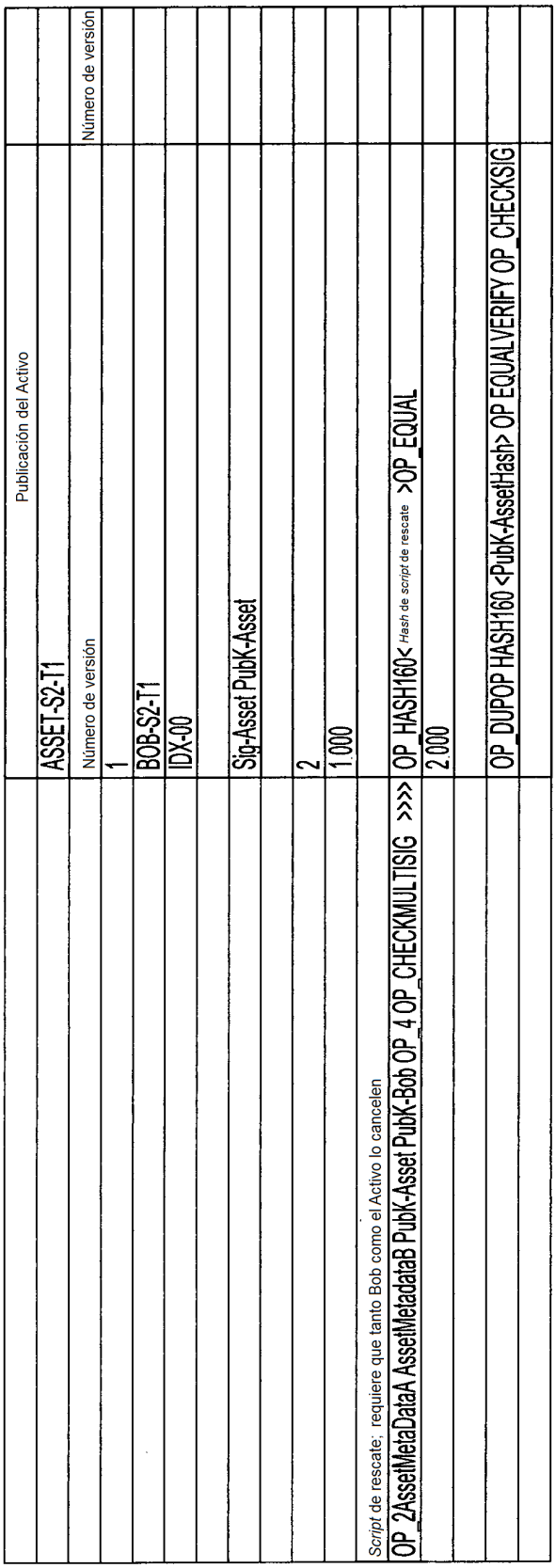

Publicación del Activo
ASSET-S2-T1
Número de versión
Número de versión
1
BOB-S2-T1
IDX-00
Sig-Asset PubK-Asset
2
1.000
Script de rescate; requiere que tanto Bob como el Activo lo cancelen
OP_2AssetMetaDataA AssetMetadataB PubK-Asset PubK-Bob OP_4 OP_CHECKMULTISIG >>>>
OP_HASH160< Hash de script de rescate >OP_EQUAL
2.000
OP_DUPOP HASH160 <PubK-AssetHash> OP EQUALVERIFY OP_CHECKSIG

Fig. 3c

Cierre del Contrato	
ASSET-S2-T2	ID de Transacción
Número de versión	Número de versión
1	Número de entradas
ASSET-S2-T1	Salida Trans Prev
IDX-00	Índice de Salidas Trans Prev
Longitud de *script*	Longitud de *script*
Sig-Asset Sig-Bob OP_2 AssetMetaDataA AssetMetadataB PubK-Asset PubK-Bob OP_4 OP_CHECKMULTISIG	ScriptSig
Número de secuencia	Número de secuencia
1	Número de salidas
1.000	Valor de salida
Longitud de *script* de salida	Longitud de *script* de salida
OP_DUP OP_HASH160<PubK-Bob Hash>OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
Tiempo de Bloqueo	Tiempo de Bloqueo

Fig. 3d

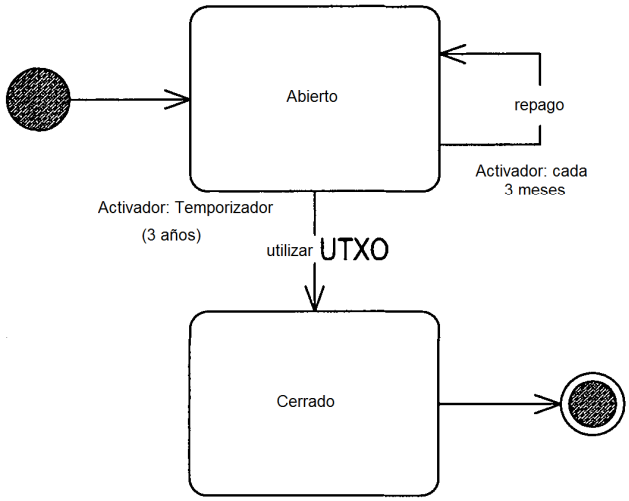


Fig. 4a

Definición de Escenario

Bob: Estoy rentando un coche de Eve por un período de tres años

Metadatos Base del Arrendamiento

Campo	Subcampo	Bytes	Valor	Comentarios
Metadatos de Activo A	Tipo de Contrato	4	0x0000FF04	Indica unidad
	Indicador de Contrato	16	xxxx.xxxx.xxxx.xxxx (...) .xxxx	Dirección del archivo *Definición de Activo*
	Relleno	12		De reserva
Metadatos de Activo B	*Hash* del Contrato	20	################...	*Hash* del archivo *Definición de Activo* (¡no la tokenización!)
	Jurisdicción	2	EN	Especifica que el activo está cubierto por la ley de Inglaterra
	Opciones	2	0x0000	Sin opciones especificadas
	Relleno	8		De reserva

Fig. 4b

		Creación de Eve del Contrato de Arrendamiento	
		EVE-S3-T1	ID de Transacción
		Número de versión	Número de versión
		1	Número de entradas
		<Salida previa BTC no utilizada de Eve - supongamos 500.000 *satoshi*>	Salida Trans Prev
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-Eve PubK-Eve	ScriptSig
		Número de secuencia	Número de secuencia
		2	Número de salidas
		2.000	Valor de salida
Script de Rescate; requiere simplemente que Eve lo cierre (es preciso notar que una solución que requiere que Bob & Eve lo cierren también es posible)		Longitud de *script* de salida	Longitud de *script* de salida
OP_1AssetMetaDataA AssetMetadataB PubK-Eve OP_3 OP_CHECKMULTISIG	>>>>	OP_HASH160< *Hash* de *script* de rescate >OP_EQUAL	*Script* de salida
		498.000	Valor de salida
		Longitud de *script* de salida	Longitud de *script* de salida
		OP_DUP OP_HASH160 <PubK-Eve Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
		Tiempo de Bloqueo	Tiempo de Bloqueo
		Finalización del Contrato con Bloqueo de Tiempo de Eve	
		EVE-3-T2	ID de Transacción
		Número de versión	Número de versión
		1	Número de entradas
		EVE-3-T1	Salida Trans Prev
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-Eve OP_1AssetMetaDataA AssetMetadataB PubK-Eve OP_3 OP_ CHECKMULTISIG	ScriptSig
		Número de secuencia	Número de secuencia
		1	Número de salidas
		2.000	Valor de salida
		Longitud de *script* de salida	Longitud de *script* de salida
		OP_DUP OP_HASH160<PubK-Eve Hash>OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
NOTA: La Transacción se presenta con antelación antes que al final del período de tres años	>>>>	Fecha en la que EVE-S3-T1 se ha transmitido más 3 años	Tiempo de Bloqueo

Fig. 4c

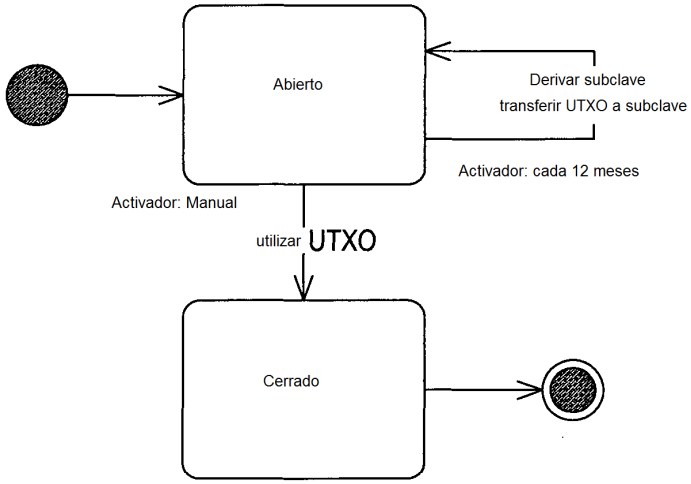


Fig. 5a

Definición de Escenario

Bob: Estoy alquilando una casa de Eve de manera anual, pero puede cancelar dentro de los 2 meses del aniversario

Metadatos de Base de Arrendamiento

Campo	Subcampo	Bytes	Valor	Comentarios
Metadatos de Activo A	Tipo de Contrato	4	0x0000FF04	Indica unidad
	Indicador de Contrato	16	xxxx.xxxx.xxxx.xxxx (...) .xxxx	Dirección del archivo *Definición de Activo*
	Relleno	12		De reserva
Metadatos de Activo B	*Hash* del Contrato	20	################...	*Hash* del archivo *Definición de Activo* (¡no la tokenización!)
	Jurisdicción	2	EN	Especifica que el activo está cubierto por la ley de Inglaterra
	Opciones	2	0x0000	Sin opciones especificadas
	Relleno	8		De reserva

Fig. 5b

		Creación del Contrato de Arrendamiento de Eve	
		EVE-S4-T1	ID de Transacción
		Número de versión	Número de versión
		1	Número de entradas
		<Salida previa BTC no utilizada de Eve - supongamos 500.000 *satoshi*>	Salida Trans Prev
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-Eve PubK-Eve	ScriptSig
		Número de secuencia	Número de secuencia
		2	Número de salidas
		1.000	Valor de salida
Script de Rescate; requiere que 2 de Bob, Eve y el agente informático independiente lleven a cabo el desbloqueo		Longitud de *script* de salida	Longitud de *script* de salida
OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-Eve PubK-Oracle OP_5 OP_CHECKMULTISIG	>>>>	OP_HASH160< *Hash* de *script* de rescate >OP_EQUAL	*Script* de salida
		499.000	Valor de salida
		Longitud de *script* de salida	Longitud de *script* de salida
		OP_DUP OP_HASH160 <PubK-Eve Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
		Tiempo de Bloqueo	Tiempo de Bloqueo
		Renovación con Bloqueo de Tiempo del Contrato de Eve	
		EVE-S4-T2	ID de Transacción
		Número de versión	Número de versión
		2	Número de entradas
		<El pago de minería de Eve de 1000 *satoshi* (es preciso notar que no puede darse ningún cambio a partir de esta transacción debido al efecto del bloqueo de tiempo) significa que ella generará, probablemente, una transacción previa para obtener una entrada del valor exacto>	Salida Trans Prev
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-Eve Sig-Bob OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-Eve OP_5 OP_CHECKMULTISIG	ScriptSig
		Número de secuencia	Número de secuencia
		1	Número de salidas
		1.000	Valor de salida
Script de Rescate; requiere que 2 de Bob, la subclave de renovación del contrato de Eve y el Aente independiente (oráculo) realicen		Longitud de *script* de salida	Longitud de *script* de salida
OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-Eve SK1 PubK-Oracle OP_5 OP_CHECKMULTISIG		OP_HASH160 <*Hash* de *script* de rescate> OP_EQUAL	*Script* de salida
NOTA: La Transacción se presenta con anticipación	>>>>	Fecha en la que EVE-S4-T1 se ha transmitido más 1 año	Tiempo de Bloqueo

Fig. 5c

Después del primer año, Bob continúa con el arrendamiento y no finaliza. Inmediatamente después de la publicación de EVE-S3-T2, esta se recoge por el Agente (oráculo) y se renueva por otro año (es preciso notar que también es posible que esto pueda realizarlo Eve mediante el uso de su propia lógica interna)

		Renovación del Contrato con Bloqueo de Tiempo de Eve	
		EVE-S4-T3	ID de Transacción
		Número de versión	Número de versión
		2	Número de entradas
		<El pago de minería de Eve de 1000 *satoshi* (es preciso notar que ningún cambio puede darse a partir de la presente transacción debido al efecto del bloqueo de tiempo) significa que ella generará, probablemente, una transacción previa para obtener una entrada del valor exacto>	Salida Trans Prev
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-Eve PubK-Eve	ScriptSig
		EVE-S4-T2	Número de secuencia
		IDX-00	Índice de Salidas Trans Prev
		Longitud de *script*	Longitud de *script*
		Sig-EveSK1 Sig-Oracle OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-EveSK1 PubK-Eve OP_5 OP_CHECKMULTISIG	ScriptSig
		Número de secuencia	Número de secuencia
		1	Número de salidas
		1.000	Valor de salida
Script de Rescate; requiere que 2 de Bob, la segunda subclave de renovación de contrato de Eve y el Agente independiente lleven a cabo el desbloqueo		Longitud de *script* de salida	Longitud de *script* de salida
OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-EveSK2 PubK-Oracle OP_5 OP_CHECKMULTISIG		OP_HASH160 <*hash* de *script* de rescate> OP_EQUAL	*Script* de salida
NOTA: La Transacción se presenta con anticipación	>>>>	Fecha en la que EVE-S4-T2 se ha transmitido más 1 año	Tiempo de Bloqueo

Fig. 5c (continuada)

Finalización del Contrato de Bob	
BOB-S4-T1	ID de Transacción
Número de versión	Número de versión
2	Número de entradas
<El pago de minería de Bob de 1000 *satoshi* (es preciso notar que ningún cambio puede darse a partir de la presente transacción debido al efecto del bloqueo de tiempo) significa que generará, probablemente, una transacción preiva para obtener una entrada del valor exacto>	Salida Trans Prev
IDX-00	Índice de Salidas Trans Prev
Longitud de *script*	Longitud de *script*
Sig-Bob PubK-Bob	ScriptSig
EVE-S4-T2	Número de secuencia
IDX-00	Índice de Salidas Trans Prev
Longitud de *script*	Longitud de *script*
Sig-Bob Sig-Oracle OP_2AssetMetaDataA AssetMetadataB PubK-Bob PubK-EveSK1 PubK-Eve OP_3 OP_CHECKMULTISIG	ScriptSig
Número de secuencia	Número de secuencia
1	Número de salidas
1.000	Valor de salida
Longitud de *script* de salida	Longitud de *script* de salida
OP_DUP OP_HASH160<PubK-Eve Hash>OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida
Tiempo de Bloqueo	Tiempo de Bloqueo

Fig. 5d

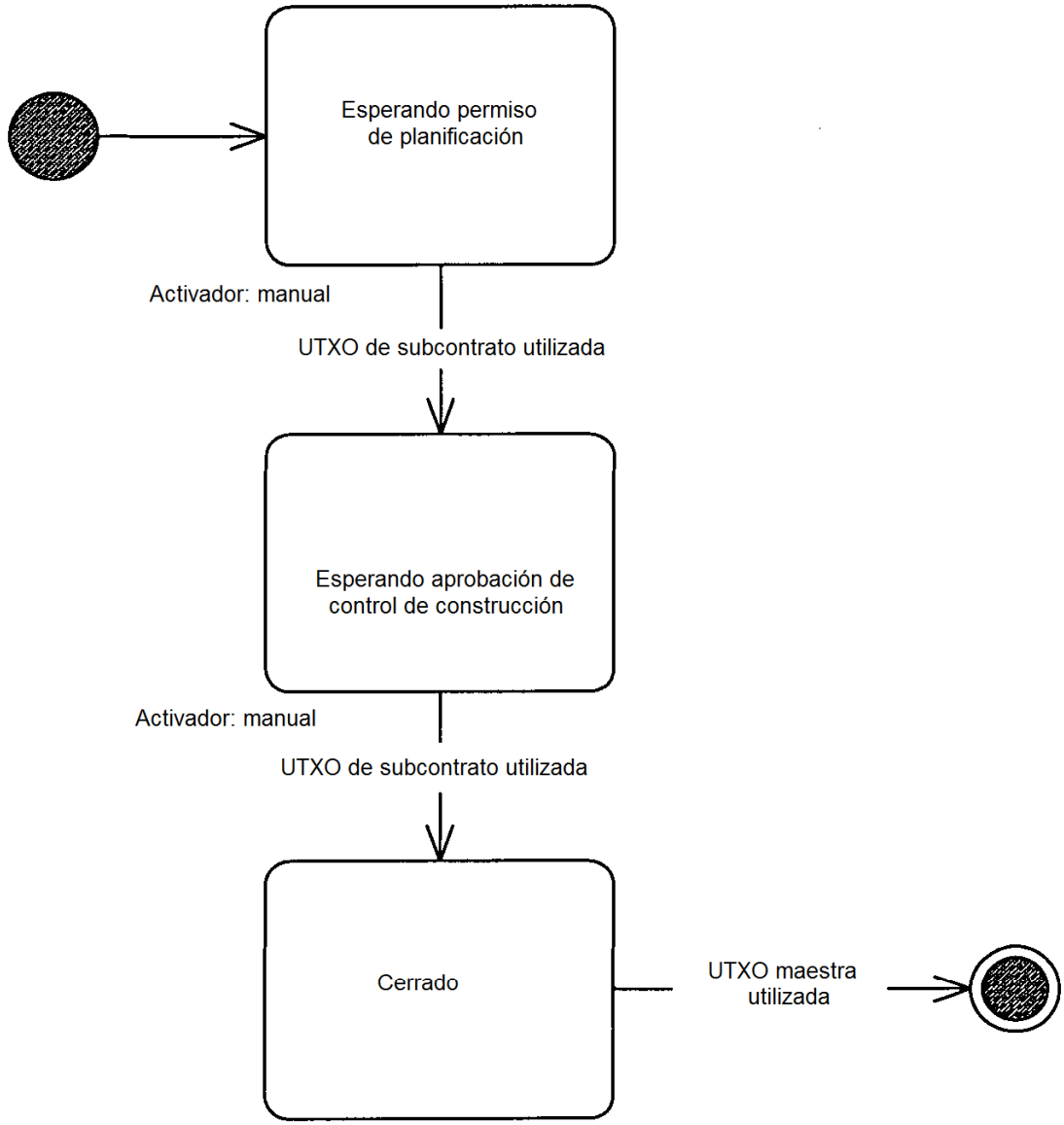

Esperando permiso de planificación
Activador: manual
UTXO de subcontrato utilizada
Esperando aprobación de control de construcción
Activador: manual
UTXO de subcontrato utilizada
Cerrado
UTXO maestra utilizada

Fig. 6a

Definición de Escenario

Bob: Estoy construyendo una propiedad y necesito dos evaluaciones independientes en diferentes momentos en el proceso (permiso de planificación y acreditación de construcción) antes de cumplir con el contrato

Metadatos Base de la Casa

Campo	Subcampo	Bytes	Valor	Comentarios
Metadatos de Activo A	Tipo de Contrato	4	0x0000FF04	Indica unidad
	Indicador de Contrato	16	xxxx.xxxx.xxxx.xxxx (...) .xxxx	Dirección del archivo *Definición de Activo*
	Relleno	12		De reserva
Metadatos de Activo B	*Hash* de Contrato	20	################...	*Hash* del archivo *Definición de Activo* (¡no la tokenización!)
	Jurisdicción	2	EN	Especifica que el activo está cubierto por la ley de Inglaterra
	Opciones	2	0x0000	No se especifican opciones
	Relleno	8		De reserva

Fig. 6b

Creación del Contrato de Construcción de Propiedad de Bob		
BOB-S5-T1	ID de Transacción	
Número de versión	Número de versión	
1	Número de entradas	
<Salida previa BTC no utilizada de Bob - supongamos 500.000 *satoshi*>	Salida Trans Previa	
IDX-00	Índice de Salidas Trans Prev	
Script length	Longitud de script	
Sig-Bob PubK Bob	ScriptSig	
Número de secuencia	Número de secuencia	
2	Número de salidas	
2.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	
		Script de Rescate; requiere que 2 de Bob y Oráculo concluyan
OP_HASH160 <*hash* de *script* de rescate> OP_EQUAL	*Script* de salida	>>>> OP_1AssetMetaDataA AssetMetadataB PubK-Bob PubK-Oracle OP_4 OP_CHECKMULTISIG
497.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	
OP_DUP OP_HASH160 <PubK-Bob Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida	
Tiempo de Bloqueo	Tiempo de Bloqueo	

Fig. 6c

Creación del subcontrato de Bob mediante el uso de su clave derivada para confirmar la aprobación de planificación		
BOB-S5-T2	ID de Transacción	
Número de versión	Número de versión	
1	Número de entradas	
BOB-S5-T1	Salida Trans Prev	
IDX-01	Índice de Salidas Trans Prev	
Longitud de *script*	Longitud de *script*	
Sig-Bob PubK Bob	ScriptSig	
Número de secuencia	Número de secuencia	
2	Número de salidas	
2.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	*Script* de Rescate; requiere aprobación de Autoridad de Planifiación & aprobación de Oráculo, y que Bob sustituya a cualquiera
OP_HASH160 <*hash* de *script* de rescate> OP_EQUAL	*Script* de salida	>>>> OP_2AssetMetaDataA AssetMetadataB PubK-BobSK1 PubK-PlanningAuthority PubK-Oracle OP_5
494.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	
OP_DUP OP_HASH160 <PubK-Bob Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida	
Tiempo de Bloqueo	Tiempo de Bloqueo	

Fig. 6c (continuada)

Creación del subcontrato de Bob mediante el uso de su clave derivada para confirmar aprobación estándar de construcción		
BOB-S5-T3	ID de Transacción	
Número de versión	Número de versión	
1	Número de entradas	
BOB-S5-T2	Salida Trans Prev	
IDX-01	Índice de Salidas Trans Prev	
Longitud de *script*	Longitud de *script*	
Sig-Bob PubK Bob	ScriptSig	
Número de secuencia	Número de secuencia	
2	Número de salidas	
2.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	*Script* de Rescate; requiere aprobación del Estándar de Construcción & aprobación de Oráculo pero Bob puede reemplazar a cualquiera
OP_HASH160 <*hash* de *script* de rescate> OP_EQUAL	*Script* de salida	>>>> OP_2AssetMetaDataA AssetMetadataB PubK-BobSK2 PubK-BuildingStandard PubK-Oracle OP_5
491.000	Valor de salida	
Longitud de *script* de salida	Longitud de *script* de salida	
OP_DUP OP_HASH160 <PubK-Bob Hash> OP_EQUALVERIFY OP_CHECKSIG	*Script* de salida	
Tiempo de Bloqueo	Tiempo de Bloqueo	

Fig. 6c (continuada)

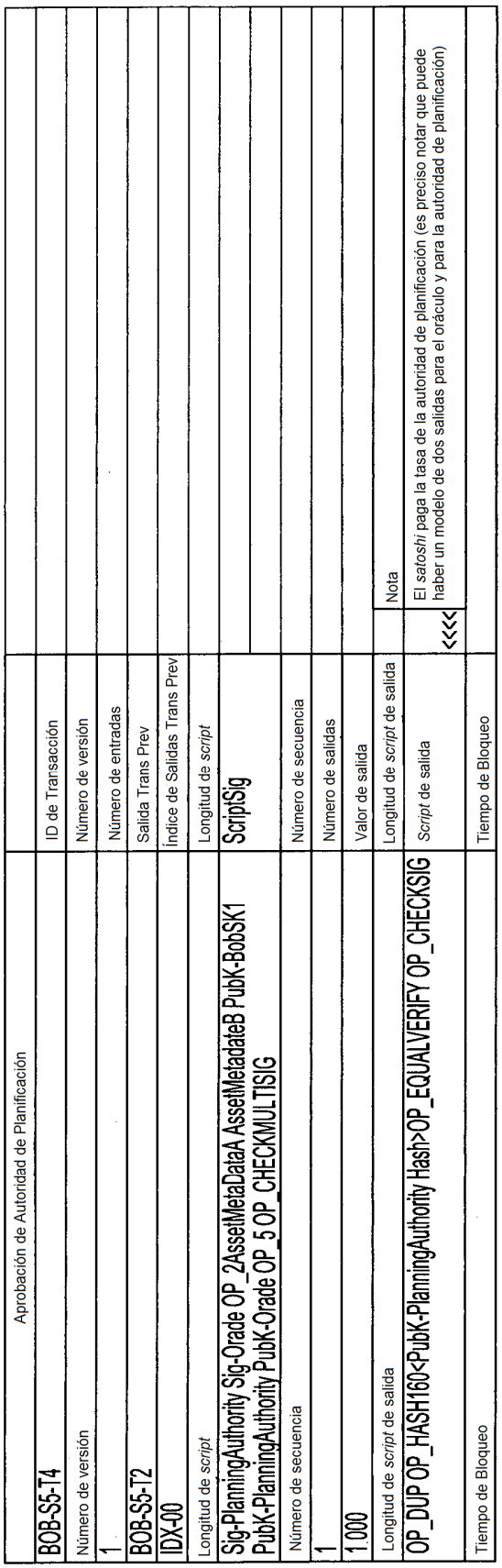

Aprobación de Autoridad de Planificación
BOB-S5-T4
Número de versión
1
BOB-S5-T2
IDX-00
Longitud de script
Sig-PlanningAuthority Sig-Orade OP_2AssetMetaDataA AssetMetadateB PubK-BobSK1 PubK-PlanningAuthority PubK-Orade OP_5 OP_CHECKMULTISIG
Número de secuencia
1
1.000
Longitud de script de salida
OP_DUP OP_HASH160<PubK-PlanningAuthority Hash>OP_EQUALVERIFY OP_CHECKSIG
Tiempo de Bloqueo
ID de Trarsacción
Número de versión
Número de entradas
Salida Trans Prev
Índice de Salidas Trans Prev
Longitud de script
ScriptSig
Número de secuencia
Número de salidas
Valor de salida
Longitud de script de salida
Script de salida
Tiempo de Bloqueo
Nota
El satoshi paga la tasa de la autoridad de planificación (es preciso notar que puede haber un modelo de dos salidas para el oráculo y para la autoridad de planificación)
<<<<

Fig. 6d

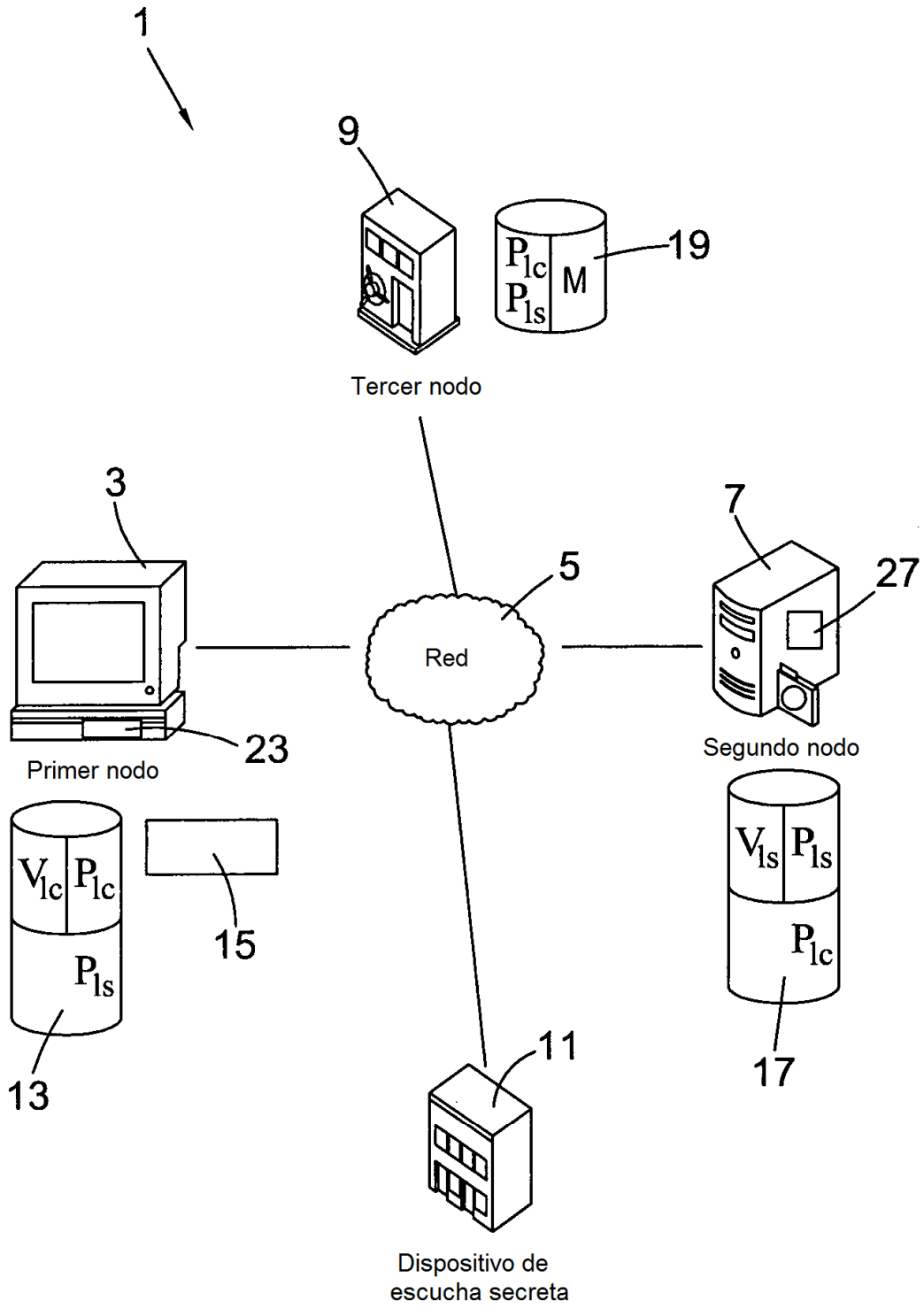
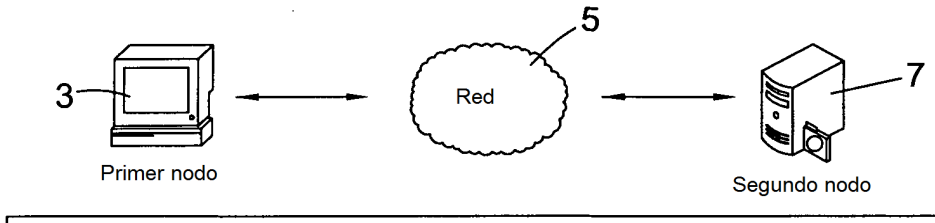

1
9
19
P_{lc}
P_{ls}
M
Tercer nodo
3
7
27
5
Red
23
Primer nodo
Segundo nodo
V_{lc}
P_{lc}
P_{ls}
15
V_{ls}
P_{ls}
P_{lc}
13
17
11
Dispositivo de
escucha secreta

Fig. 7



Compartir Mensaje (M) entre el primer y segundo nodos

300

330

Determinar una segunda clave privada de primer nodo (V_{2C}) según al menos la clave privada maestra de primer nodo (V_{1C}) y un Valor de Generador (VG)

Generar un primer mensaje firmado (MF1) según el mensaje (M) y la segunda clave privada de primer nodo (V_{2C})

350

Enviar el primer mensaje firmado (MF1) al segundo nodo

360

400

Determinar una segunda clave pública de primer nodo (P_{2C}) según la clave pública maestra de primer nodo (P_{1C}) y el Valor de Generador (VG)

430

440

Recibir el primer mensaje firmado (MF1) del primer nodo

450

Validar firma en el primer mensaje firmado (MF1) con la segunda clave pública de primer nodo (P_{2C}) determinada

460

Autenticar el primer nodo según el resultado de la validación del primer mensaje firmado (MF1)

370

Determinar una segunda clave pública de segundo nodo (P_{2S}) según la clave pública maestra de segundo nodo (P_{1S}) y el Valor de Generador (VG)

470

Determinar una segunda clave privada de segundo nodo (V_{2S}) según la clave privada maestra de segundo nodo (V_{1S}) y el Valor de Generador (VG)

480

Determinar el secreto común (SC) según la segunda clave privada de primer nodo (V_{2C}) y la segunda clave pública de segundo nodo (P_{2S})

380

Determinar el secreto común (SC) según la segunda clave privada de segundo nodo (V_{2S}) y la segunda clave pública de primer nodo (P_{2C})

Fig. 8

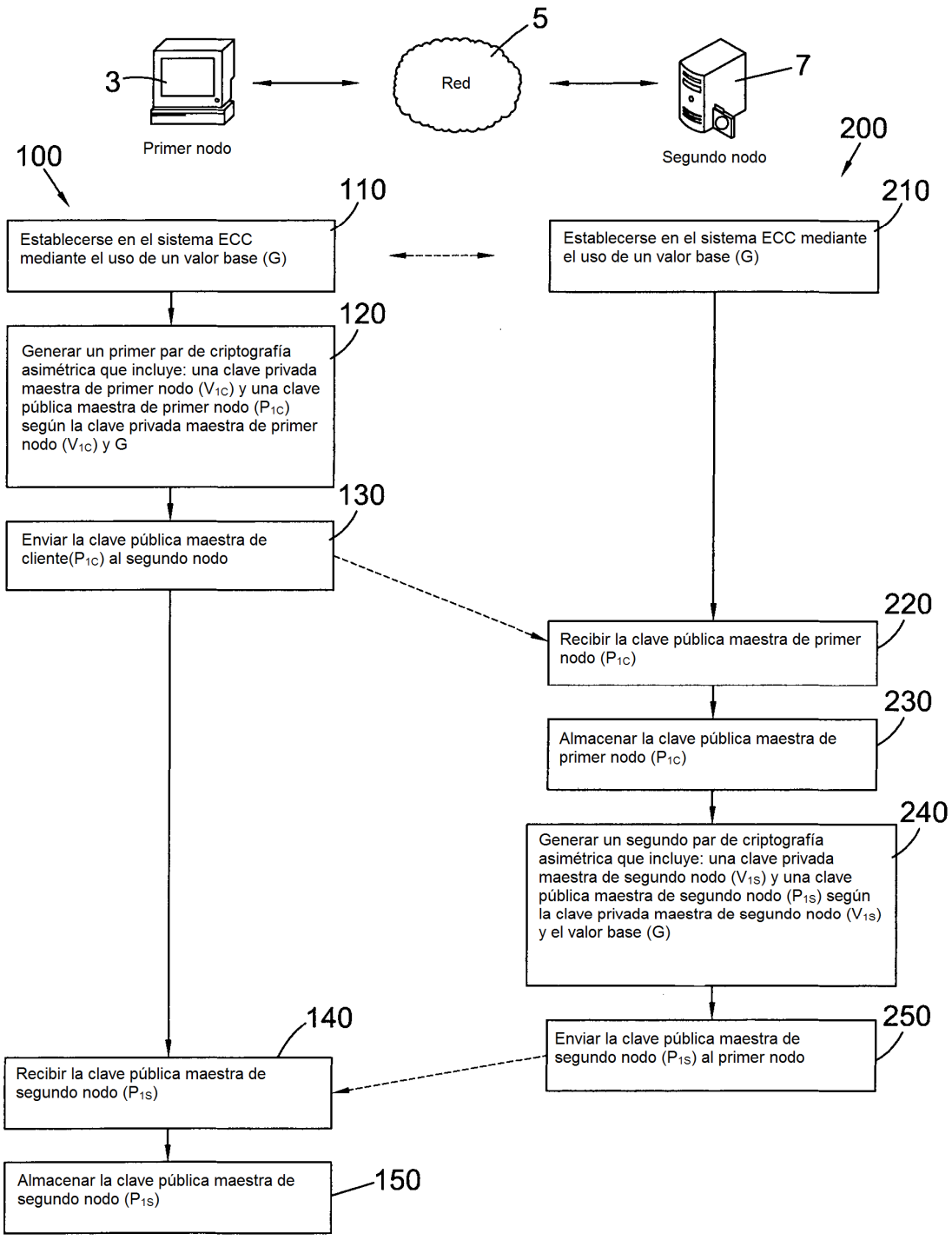

3
Primer nodo
5
Red
7
Segundo nodo
100
200
110
Establecerse en el sistema ECC mediante el uso de un valor base (G)
210
Establecerse en el sistema ECC mediante el uso de un valor base (G)
120
Generar un primer par de criptografía asimétrica que incluye: una clave privada maestra de primer nodo (V_{1C}) y una clave pública maestra de primer nodo (P_{1C}) según la clave privada maestra de primer nodo (V_{1C}) y G
130
Enviar la clave pública maestra de cliente(P_{1C}) al segundo nodo
220
Recibir la clave pública maestra de primer nodo (P_{1C})
230
Almacenar la clave pública maestra de primer nodo (P_{1C})
240
Generar un segundo par de criptografía asimétrica que incluye: una clave privada maestra de segundo nodo (V_{1S}) y una clave pública maestra de segundo nodo (P_{1S}) según la clave privada maestra de segundo nodo (V_{1S}) y el valor base (G)
140
Recibir la clave pública maestra de segundo nodo (P_{1S})
250
Enviar la clave pública maestra de segundo nodo (P_{1S}) al primer nodo
Almacenar la clave pública maestra de segundo nodo (P_{1S})
150

Fig. 9

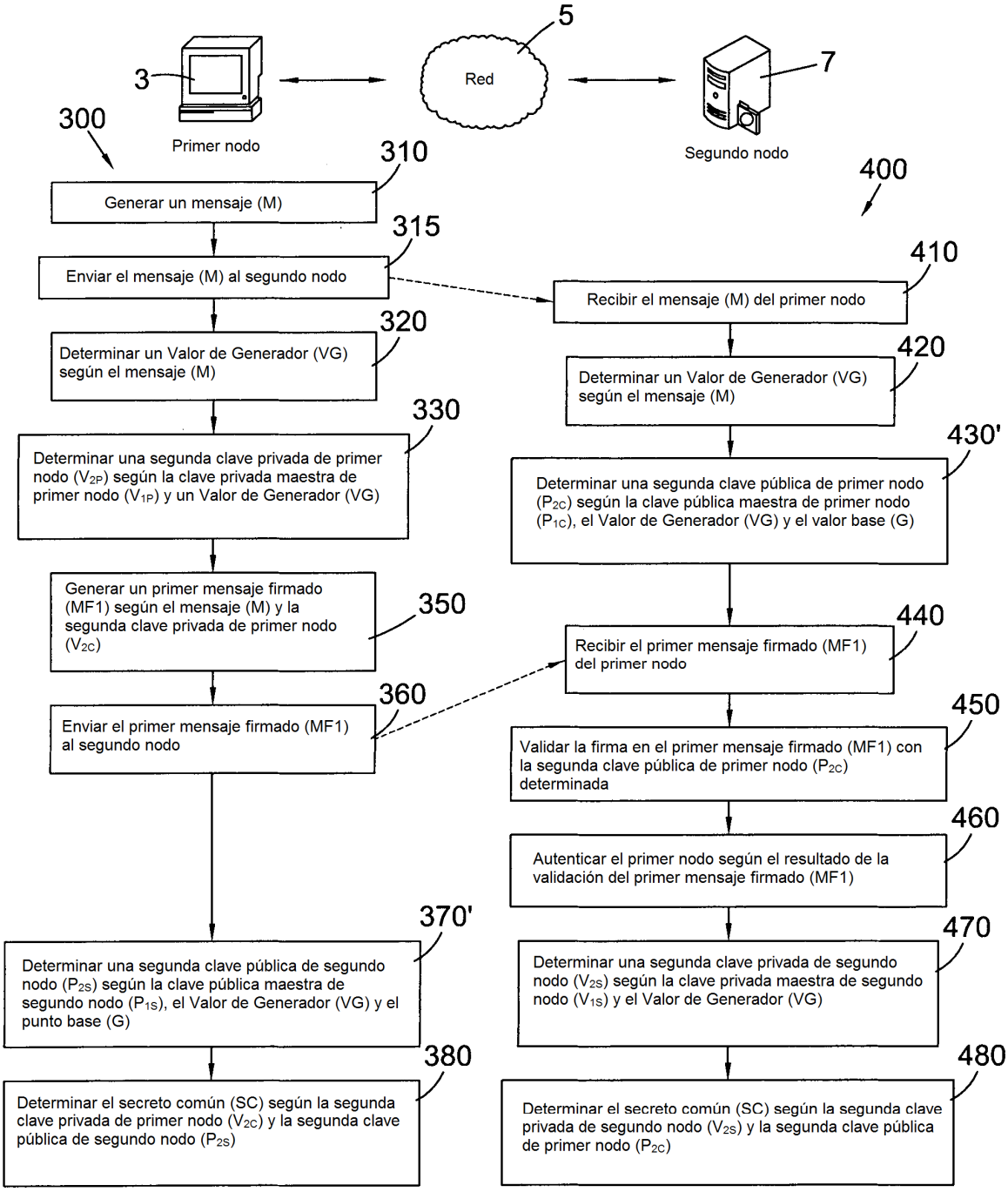

5
Red
3
7
300
Primer nodo
Segundo nodo
310
Generar un mensaje (M)
400
315
Enviar el mensaje (M) al segundo nodo
410
Recibir el mensaje (M) del primer nodo
320
Determinar un Valor de Generador (VG) según el mensaje (M)
420
Determinar un Valor de Generador (VG) según el mensaje (M)
330
Determinar una segunda clave privada de primer nodo (V_{2P}) según la clave privada maestra de primer nodo (V_{1P}) y un Valor de Generador (VG)
430'
Determinar una segunda clave pública de primer nodo (P_{2C}) según la clave pública maestra de primer nodo (P_{1C}), el Valor de Generador (VG) y el valor base (G)
Generar un primer mensaje firmado (MF1) según el mensaje (M) y la segunda clave privada de primer nodo (V_{2C})
350
440
Recibir el primer mensaje firmado (MF1) del primer nodo
360
Enviar el primer mensaje firmado (MF1) al segundo nodo
450
Validar la firma en el primer mensaje firmado (MF1) con la segunda clave pública de primer nodo (P_{2C}) determinada
460
Autenticar el primer nodo según el resultado de la validación del primer mensaje firmado (MF1)
370'
Determinar una segunda clave pública de segundo nodo (P_{2S}) según la clave pública maestra de segundo nodo (P_{1S}), el Valor de Generador (VG) y el punto base (G)
470
Determinar una segunda clave privada de segundo nodo (V_{2S}) según la clave privada maestra de segundo nodo (V_{1S}) y el Valor de Generador (VG)
380
Determinar el secreto común (SC) según la segunda clave privada de primer nodo (V_{2C}) y la segunda clave pública de segundo nodo (P_{2S})
480
Determinar el secreto común (SC) según la segunda clave privada de segundo nodo (V_{2S}) y la segunda clave pública de primer nodo (P_{2C})

Fig. 10

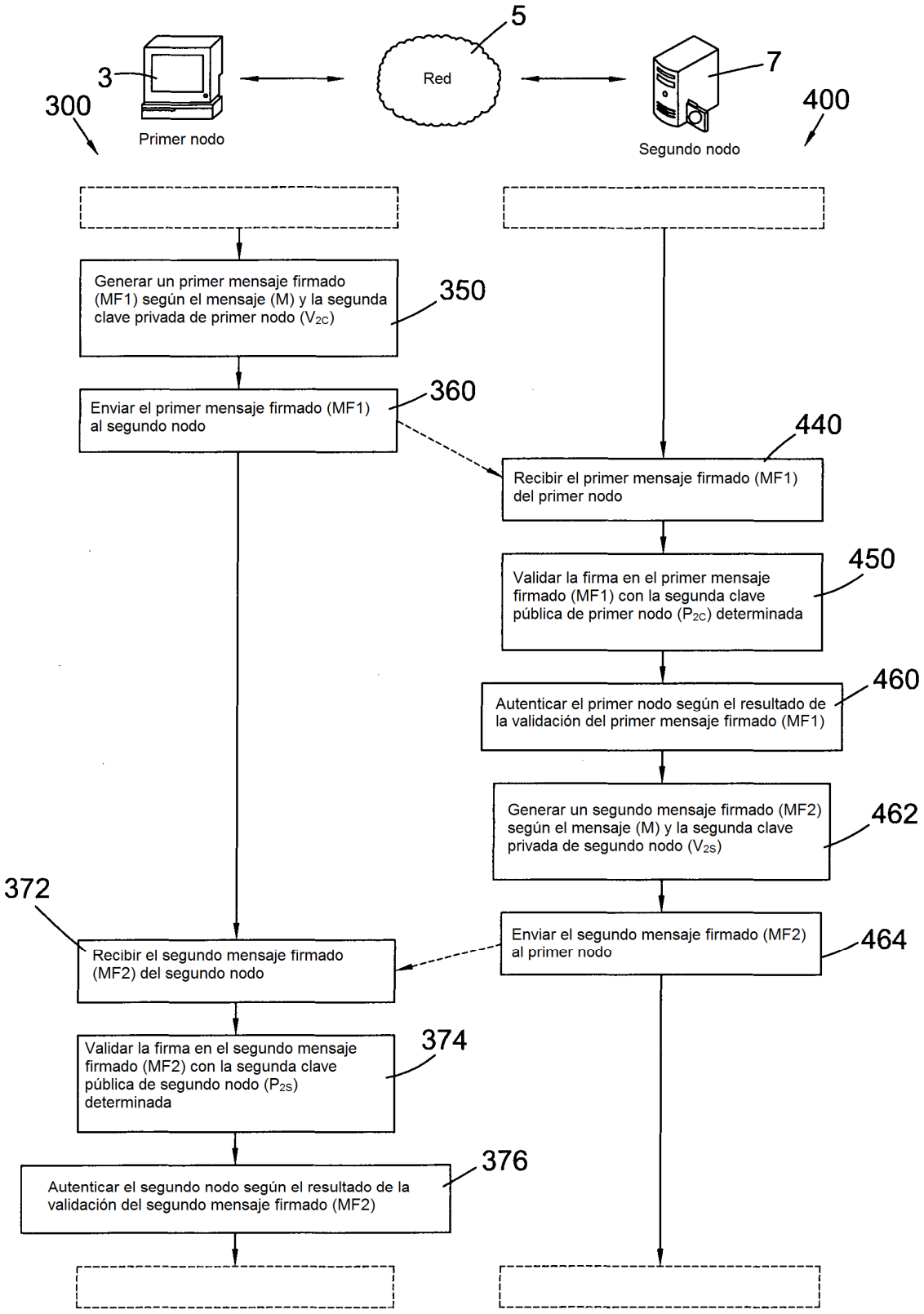

5
Red
3
300
Primer nodo
7
400
Segundo nodo
Generar un primer mensaje firmado (MF1) según el mensaje (M) y la segunda clave privada de primer nodo (V_{2C})
350
Enviar el primer mensaje firmado (MF1) al segundo nodo
360
440
Recibir el primer mensaje firmado (MF1) del primer nodo
Validar la firma en el primer mensaje firmado (MF1) con la segunda clave pública de primer nodo (P_{2C}) determinada
450
Autenticar el primer nodo según el resultado de la validación del primer mensaje firmado (MF1)
460
Generar un segundo mensaje firmado (MF2) según el mensaje (M) y la segunda clave privada de segundo nodo (V_{2S})
462
372
Recibir el segundo mensaje firmado (MF2) del segundo nodo
Enviar el segundo mensaje firmado (MF2) al primer nodo
464
Validar la firma en el segundo mensaje firmado (MF2) con la segunda clave pública de segundo nodo (P_{2S}) determinada
374
Autenticar el segundo nodo según el resultado de la validación del segundo mensaje firmado (MF2)
376

Fig. 11

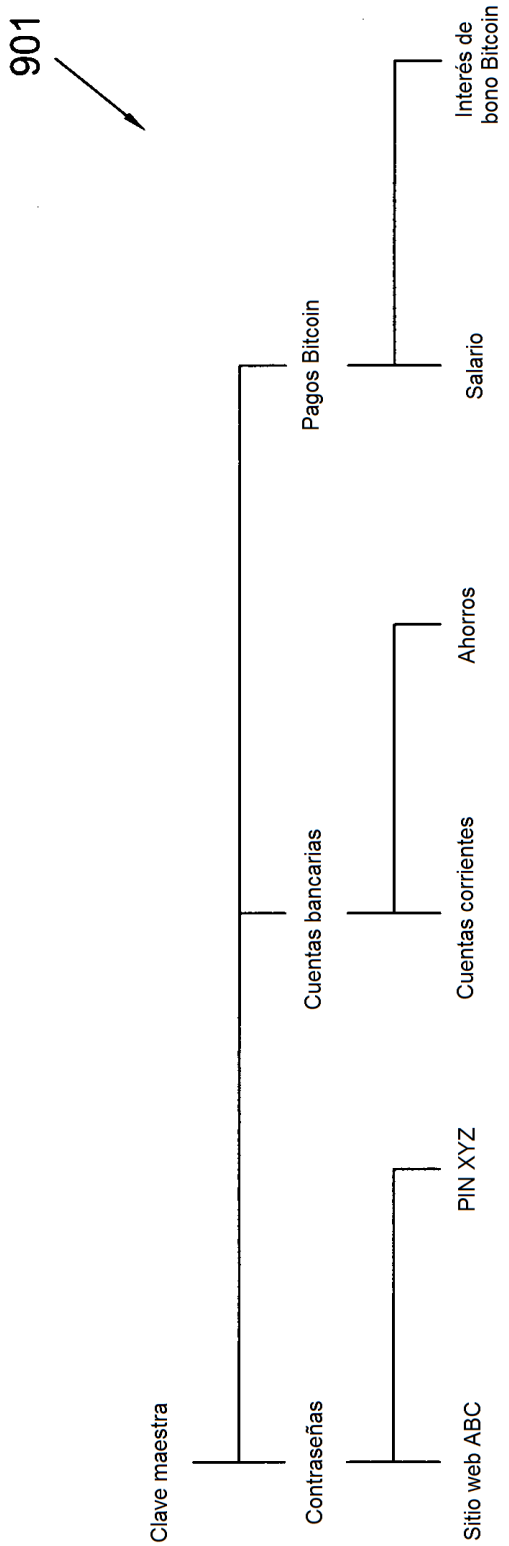

901
Clave maestra
Contraseñas
Sitio web ABC
PIN XYZ
Cuentas bancarias
Cuentas corrientes
Ahorros
Pagos Bitcoin
Salario
Interés de bono Bitcoin

Fig. 12

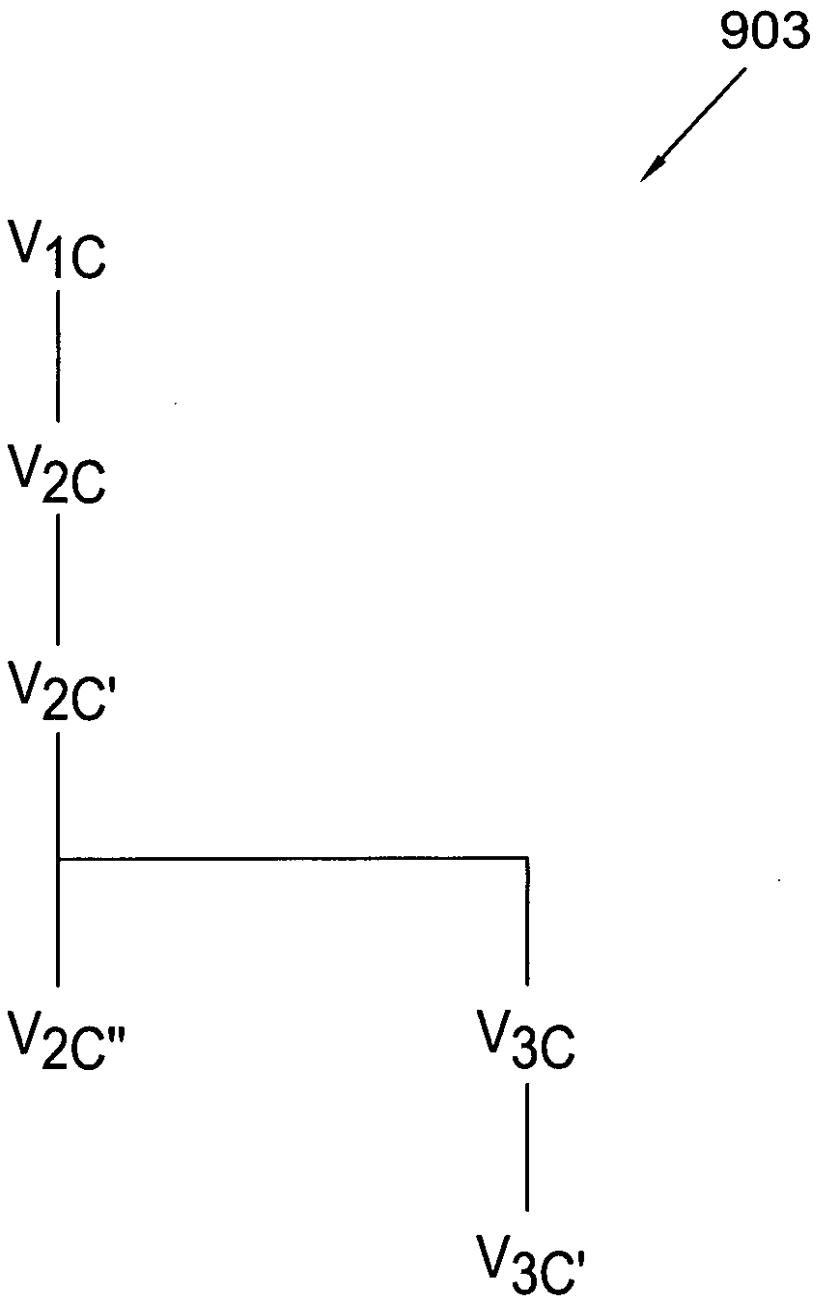


Fig. 13