Creación de contratos inteligentes en la

Red Blockchain de Ethereum con Solidity

Desde **www.ra-ma.es** podrá descargar material adicional.

Wilmar Alonso Ramírez Gil
Carlos Mario Ramírez Gil

Ra-Ma®
ediJ

Creación de contratos inteligentes en la Red Blockchain de Ethereum con Solidity

Wilmar Alonso Ramírez Gil
Carlos Mario Ramírez Gil

Ra-Ma®
edU®
Conocimiento a su alcance

Ramírez Gil, Wilmar Alonso, *et al.*

Creación de contratos inteligentes en la Red Blockchain de Ethereum con Solidity / Wilmar Alonso Ramírez Gil, Carlos Mario Ramírez Gil --. Bogotá: Ediciones de la U, 2023
438 p. ; 24 cm
ISBN 978-958-792-459-6 e-ISBN 978-958-792-460-2
1. Informática 2. Programación 3. Herramientas de desarrollo 4. Contratos inteligentes I. Tít.
621.39 ed.

Edición original publicada por © Editorial Ra-ma (España)
Edición autorizada a Ediciones de la U para Colombia

Área: Informática
Primera edición: Bogotá, Colombia, enero de 2023
ISBN. 978-958-792-459-6

Ediciones de la U es una empresa editorial que, con una visión moderna y estratégica de las tecnologías, desarrolla, promueve, distribuye y comercializa contenidos, herramientas de formación, libros técnicos y profesionales, e-books, e-learning o aprendizaje en línea, realizados por autores con amplia experiencia en las diferentes áreas profesionales e investigativas, para brindar a nuestros usuarios soluciones útiles y prácticas que contribuyan al dominio de sus campos de trabajo y a su mejor desempeño en un mundo global, cambiante y cada vez más competitivo.

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Ediciones de la U
Impresión: DGP Editores SAS
Calle 63 #70D-34, Pbx (57+1) 3203510

Impreso y hecho en Colombia
Printed and made in Colombia

A Luz Mariela y José Gerardo
cuando éramos niños

ÍNDICE

PREFACIO

Escribimos este libro para hacer que el desarrollo de contratos inteligentes sea más accesible y fácil de entender para los principiantes que buscan explorar el desarrollo en blockchain.

Puede ser desalentador cuando se lanzan términos como redes, nodos y prueba de trabajo. Este libro lo ayudará a aprender cómo crear y probar su propio contrato inteligente, crear una interfaz para que los usuarios interactúen y más. Escribimos este libro porque queremos proporcionar un recurso para las personas que quieren incursionar en el campo, pero no saben por dónde empezar. Este libro lo guiará a través de todo el proceso de creación de un contrato inteligente como lo haría en el mundo real y su implementación para ayudar a los usuarios a interactuar con él a través de una aplicación descentralizada (DApp).

AUTORES

Wilmar Alonso Ramírez Gil, Ingeniero Electricista Universidad de Antioquia, Medellín Colombia. Desarrollador en lenguajes de programación JavaScript, Solidity y el lenguaje de etiquetas Html5 para páginas Web, experiencia en el diseño de aplicaciones fundamentadas en la programación orientada a objetos en el contexto educativo; Magister Enseñanza de las Ciencias Exactas y Naturales Universidad Nacional de Colombia Seccional Medellín. Correo electrónico: williannoso@gmail.com.

Carlos Mario Ramírez Gil, Ingeniero Administrador Universidad Nacional de Colombia Seccional Medellín; Especialista en Gerencia de Sistemas Informáticos, Universidad Nacional de Colombia Seccional Medellín; Especialista en Finanzas Corporativas, Escuela de Ingeniería de Antioquia; Magister Ingeniería Administrativa Universidad Nacional de Colombia Seccional Medellín. Docente Postgrado área financiera en diversas universidades de Colombia. Amplia experiencia como ejecutivo en empresas del sector real en cargos administrativos y financieros y consultor empresarial. Desarrollador en el lenguaje de programación Python. Investigador en Blockchain aplicado a las Finanzas (DeFi – Finanzas Descentralizadas). Correo electrónico: cramirez1@hotmail.com.

1

INTRODUCCIÓN A BLOCKCHAIN, ETHEREUM Y CONTRATOS INTELIGENTES

Ethereum es una plataforma pública de código abierto basada en la tecnología blockchain. Puede verse como una computadora mundial construida sobre una red peer-to-peer (P2P). Una aplicación confiable y descentralizada se puede ejecutar sobre Ethereum sin la amenaza de una administración centralizada y problemas de punto único de fallo (SPOF por sus siglas en inglés, single-point -of- failure). Y como solo hay una máquina de este tipo en el mundo, los recursos de cálculo (como CPU, memoria, etc.) son limitados y agotados bajo la presión de la enorme escala de la base de usuarios y las aplicaciones descentralizadas (DApps por sus siglas en inglés). Por lo tanto, es comprensible que el uso de la máquina mundial Ethereum cueste dinero en forma de moneda criptográfica.

1.1 ETHEREUM

Aquí está la definición oficial de Ethereum de Ethereum.org:

Ethereum es una plataforma descentralizada que ejecuta contratos inteligentes: aplicaciones que se ejecutan exactamente según lo programado sin ninguna posibilidad de tiempo de inactividad, censura, fraude o interferencia de terceros. Estas aplicaciones se ejecutan en una cadena de bloques personalizada, una infraestructura global compartida enormemente poderosa que puede mover el valor y representar la propiedad de los activos.

Bitcoin es la primera aplicación conocida de la tecnología blockchain. Sin embargo, sigue siendo una moneda. Comparativamente, Ethereum es una plataforma distribuida y de código abierto sobre la base de la tecnología blockchain y nos brinda todo el potencial de la tecnología blockchain.

En términos generales, Ethereum es una máquina de estado basada en transacciones. La fórmula de transformación de estado es la siguiente:

$$\sigma' = \Upsilon(\sigma, T)$$

El estado inicial es σ, el estado transformado es σ', T significa transacción, Υ es función de transformación. Echemos un vistazo al siguiente ejemplo:

Figura 1.1. Máquina de estado Ethereum

- El estado inicial es que el saldo de la cuenta de Alicia es de 100 euros y el saldo de la cuenta de Pedro es 0.
- Suponga que la transacción es que Alicia le paga 10 euros a Pedro.
- La función de transformación Y es transferir 10 euros de la cuenta de Alicia a la cuenta de Pedro.
- El estado transformado es σ': el saldo de la cuenta de Alicia es de 90 euros y Pedro tiene 10 euros (Figura 1.1).

Ethereum tiene todas las características de una cadena de bloques pública: cifrado de clave pública / privada, función de hash de criptografía, árbol Merkle y bifurcación dura / blanda, etc. A continuación, se mencionan las tecnologías y términos que usaremos en los capítulos posteriores.

1.1.1 Criptografía asincrónica

La criptografía asíncrona es el mayor invento de la historia de la criptografía. El cifrado sincronizado necesita compartir la clave de antemano. Sin embargo, la criptografía asíncrona no necesita hacer eso. Como sugiere "asynchronized", la clave de cifrado y la clave de descifrado son diferentes y se denominan clave pública

y clave privada. Normalmente, la clave pública está abierta y disponible para el público, mientras que la clave privada generalmente se mantiene oculta por persona. Dado que la clave pública y la clave privada están separadas, significa que la clave pública / privada se puede utilizar en un canal inseguro. La desventaja es: todo el proceso es lento porque el cifrado / descifrado lleva tiempo; y el cifrado no es tan sólido como el cifrado sincronizado.

La seguridad del cifrado asincrónico suele estar respaldada por las matemáticas. Por el momento, existen varios enfoques principales: factorización de números primos grandes, algoritmo discreto y curva elíptica. Los algoritmos principales incluyen RSA, ElGamal y cripto-sistemas de curva elíptica (ECC). Generalmente, se utilizan en escenarios como la firma digital o el intercambio de claves, que no son aptos para el cifrado / descifrado de datos a gran escala. A partir de ahora, el algoritmo RSA se considera inseguro hasta cierto punto. Por lo tanto, se recomienda ECC.

1.1.1.1 ALGORITMO DIFFIE–HELLMAN

Aquí, vamos a presentar un algoritmo de intercambio de claves: Diffie-Hellman. Primero, presentaremos el operador mod. No es difícil. Suponiendo que tenemos un módulo A, para cualquier número de B, B mod A es el residuo de B / A.

Por ejemplo: usamos módulo A que es igual a 11

```
15 mod 11 = 4
(3 + 8) mod 11 = 0
(3^4) mod 11 = 4
```

Ahora podemos comenzar a construir un algoritmo de intercambio de claves. Pensemos en el escenario que tiene tres personas: A, B, C.

1. **A y B seleccionan su propio número privado**

   Por conveniencia, asumimos que A y B seleccionan un número pequeño: A selecciona 8 y B selecciona 9. Y en el mundo real, deberían seleccionar un número grande por motivos de seguridad.

2. **A y B seleccionan dos números públicos**

   Tanto A como B comparten dos números: uno es módulo y el otro es base. En este ejemplo, A y B comparten 11 (módulo) y 2 (base).

3. **A y B crean su propio número público-privado (PPN)**

   Ahora, A y B usarán dos números compartidos en el Paso 2 con su propio número privado para calcular su número público-privado (llamado PPN):

```
PPN = base ^ número privado mod modulo
```

   Entonces

```
A's PPN = 2^8 mod 11 = 3
B's PPN = 2^9 mod 11 = 6
```

   Como puede verse, el proceso de cálculo es irreversible debido al funcionamiento en módulo.

4. **A y B mezclan el PPN de la otra parte con su propio número privado**.

   El método de mezcla es muy similar al anterior, simplemente reemplace la base con PPN:

```
A clave compartida = B's PPN^8 mod 11 = 6^8 mod 11 = 4
B clave compartida = A's PPN^9 mod 11 = 3^9 mod 11 = 4
```

   La razón por la que obtenemos el mismo resultado es porque la operación de potencia satisface la ley conmutativa.

```
(a^b)^c = (a^c)^b = a^(bc)
```

   ¡Ahora A y B obtienen la clave compartida final (que es 4 en el ejemplo anterior) a través del proceso de mezcla! Y A y B pueden usar esta clave compartida para cifrar / descifrar. Espía: C, dado que C no puede obtener el número privado de A o B, a pesar de que conocen el PPN de A o B, C todavía no puede obtener la clave compartida final a través del proceso de mezcla.

1.1.1.2 CLAVE PRIVADA/PÚBLICA

Alicia tiene clave pública y clave privada. Puede utilizar una clave privada para generar una firma digital. Dado que la clave pública de Alicia está disponible para el público, Pedro la usa para verificar que una firma digital es de Alicia. Cuando crea una dirección de Ethereum o una dirección de Bitcoin, la cadena hexadecimal

larga (por ejemplo: 0xef.. .59) es una clave pública, mientras que la clave privada puede estar almacenada en otro lugar, podría estar en el servidor de la nube o podría estar en un dispositivo personal como un teléfono móvil o computadora personal. Si pierde la clave privada de su billetera, eso significa que pierde todos los fondos de esa billetera de forma permanente. Por lo tanto, siempre es un buen hábito hacer una copia de seguridad de su clave pública y su clave privada (Figuras 1.2 y 1.3).

Figura 1.2. Clave privada, clave pública y dirección relacionada

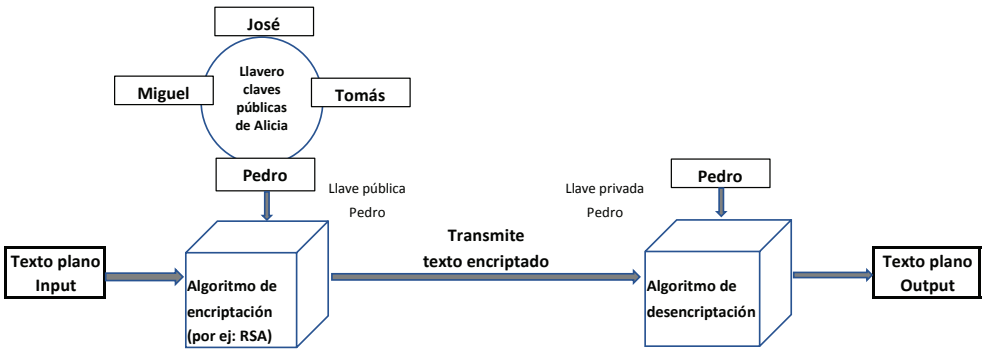

Figura 1.3. Use priv/pub para cifrar y descifrar mensajes

1.1.1.3 CIFRADO

La aplicación más importante de la clave pública / privada es el cifrado. La figura 1.3 muestra el uso:

- Alicia tiene un llavero que contiene la clave pública de Pedro, José, Miguel.
- Con la clave pública de Pedro, Alicia cifra el mensaje enviado a Pedro.
- Alicia envía texto encriptado.
- Pedro recibe texto encriptado y usa su propia clave privada para desencriptar el texto.

1.1.1.4 VERIFICANDO FIRMA

Otra aplicación importante del sistema de claves públicas / privadas es la firma digital. Y el flujo básico se muestra en la figura 1.4.

- Usando su propia clave privada, Alicia encripta el mensaje (texto sin cifrar).
- Pedro tiene la clave pública de Alicia.
- Una vez que Pedro recibe el mensaje encriptado, usando la clave pública de Alicia, puede verificar si el mensaje es de Alicia o no.

1.1.2 Función hash criptográfica

El hash criptográfico es una función hash: toma el mensaje como entrada y devuelve una cadena de tamaño fijo. La cadena de resultado se denomina valor hash, resumen del mensaje, firma digital, huella digital, resumen o suma de comprobación.

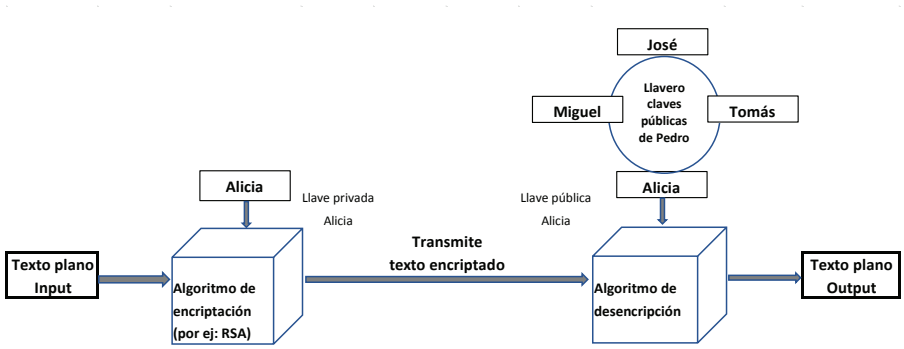

Figura 1.4. Verificar firma

La función hash debe tener tres atributos principales como se muestra a continuación:

1. **Fácil de calcular**

   Usar la función hash para calcular el valor hash para cualquier entrada debería ser fácil y rápido. Todo el proceso de cálculo no debería llevar demasiado tiempo. Es inaceptable si el cálculo del valor hash tarda 1 día o más.

2. **Irreversible**

   Si ya se conoce el resultado del hash, es muy difícil obtener el texto de origen mediante el cálculo inverso. Por ejemplo: tienes una entrada al cine, pero es muy difícil falsificarla.

3. **Resistente a colisiones**

   Un mensaje diferente debe tener un valor hash diferente. El análogo es: A y B pagan 10 euros para comprar boletos de cine y el número de asiento no debería ser el mismo.

1.1.3 Red Peer-to-Peer

Al igual que BitTorrent (protocolo descentralizado de intercambio de datos), todos los nodos de Ethereum son pares en una red distribuida. No hay un servidor centralizado. En el futuro, podría haber todo tipo de servicios cuasi centralizados para comodidad del usuario y del desarrollador. La red P2P tiene principalmente tres tipos de topología: árbol y red de tabla hash distribuida (DHT por sus siglas en inglés, Distributed Hash Table). La tecnología P2P ha cubierto casi todas las áreas de aplicación de red, por ejemplo: computación científica distribuida, intercambio de archivos, reproducción de transmisión, comunicación de voz y plataforma de juegos en línea. Algunos algoritmos P2P famosos son Kademlia, Chord, Gnutella, entre otros.

1.1.4 Blockchain

Blockchain puede verse como un libro mayor global o una base de datos simple que incluye todas las transacciones. Toda la información es pública disponible en la red y verificable. Blockchain es una tecnología de contabilidad distribuida y es la base de la moneda criptográfica Bitcoin y de Ethereum. Proporciona métodos para registrar y transmitir información de forma transparente, segura y rastreable. La tecnología blockchain hace que la organización sea transparente, democrática, distribuida, altamente eficiente, segura y confiable. La tecnología blockchain podría cambiar drásticamente las industrias existentes en los próximos años.

- **Descentralizado**

  El servicio o la aplicación se implementa en una red y ningún servidor centralizado tiene control absoluto sobre los datos y el código. Y la caída de uno o varios servidores no influirá en la aplicación o el servicio.

- **Distribuido**

  Cada servidor o nodo de la red se conecta entre sí a través del protocolo P2P.

- **Base de datos**

  La base de datos tiene varias copias en cada nodo para que el usuario pueda acceder en cualquier momento.

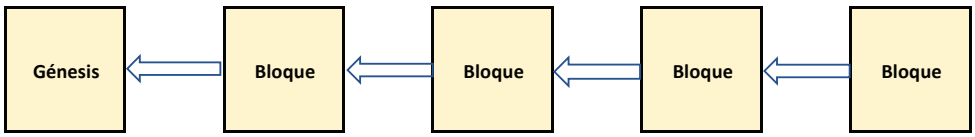

Figura 1.5. Blockchain

- **Libro mayor**

  Cada nodo tiene un sistema de contabilidad basado en el mismo libro de contabilidad público, que registra toda la información de las transacciones en la red. Y el libro mayor está sin alterar y solo se anexa (Figura 1.5).

1.1.5 Máquina virtual de Ethereum (EVM)

Además de Ethereum Virtual Machine (EVM), el usuario puede desarrollar innumerables aplicaciones. En comparación con el lenguaje de script de Bitcoin, EVM proporciona un lenguaje de programación más potente y Turing Completo. Por cada código de operación ejecutado en EVM, se ejecutará en cada nodo de la red Ethereum. La completitud de Turing significa que la computadora puede resolver cualquier fórmula matemática bajo el supuesto de que el algoritmo es correcto si tenemos suficiente tiempo y memoria.

1.1.6 Nodo

Ejecutar un nodo significa que el usuario puede acceder a los datos de la cadena a través del nodo. Un nodo completo descarga toda la cadena de bloques, mientras que un nodo ligero no descarga todos los datos y se conecta a un nodo completo para descargar los datos deseados. El nodo completo puede ser una computadora que ejecute todo el software necesario, que incluye el libro mayor distribuido completo y el software de enrutamiento P2P.

1.1.7 Minero

Un Minero ejecuta un nodo de minería para la red que procesa los bloques en blockchain. Por lo general, el minero mantiene un nodo completo que ejecuta un software de minería profesional. Pero no todos los nodos completos son nodos de minería. Si hay una actualización de código, el nodo de minería debe actualizar su software de minería con un código actualizado. Esto se puede hacer a través de una bifurcación suave, una implementación de compatibilidad con versiones anteriores. Aunque se puede hacer a través de una bifurcación dura, el código bifurcado no es compatible con el código antiguo.

1.1.8 Prueba de trabajo (PoW, proof of work)

Los mineros compiten entre sí para resolver un problema matemático. El primer minero que resuelva el problema será recompensado y tendrá derecho a empaquetar las transacciones en bloques recién generados. Cada nodo se sincronizará con un nuevo bloque automáticamente y debido a la recompensa del bloque, cada minero estará ansioso por ganar el derecho de generar el siguiente bloque nuevo. Esto conduce a un consenso en toda la red. Tenga en cuenta que Ethereum está programado para migrar al consenso de Prueba de participación (PoS).

1.1.9 App descentralizada (DApp)

En la comunidad Ethereum, las aplicaciones que utilizan contratos inteligentes se denominan APP descentralizadas (DApp). Con DApp, puede agregar interfaz de usuario (UI) y algunas funciones adicionales a su contrato inteligente, como IPFS. DApp también se puede ejecutar en un servidor centralizado si ese servidor puede interactuar con los nodos de Ethereum. DApp puede conectarse al nodo Ethereum para enviar transacciones y recuperar datos. Además de un sistema de inicio de sesión de usuario típico, el usuario tiene una dirección de billetera y los datos se guardan localmente, lo cual es diferente con las aplicaciones web actuales.

1.1.10 Solidity

Solidity es un lenguaje de programación avanzado diseñado para desarrollar contratos inteligentes que se ejecutan en Ethereum. Tiene una sintaxis similar a JavaScript. Admite tipos estáticos, integración, bibliotecas y tipo compuesto definido por el usuario. Se puede compilar en el ensamblaje EVM y, por lo tanto, se puede ejecutar en todos los nodos de Ethereum. Hay otros lenguajes de programación de contratos inteligentes: Vyper, Yul, Rust y JavaScript. Sin lugar a duda, Solidity

es el lenguaje de programación más popular para contratos inteligentes. EVM es un entorno limitado de tiempo de ejecución. Por lo tanto, todos los contratos inteligentes que descansan en Ethereum están segregados del entorno circundante. Como resultado, el contrato inteligente en EVM no puede acceder a la red, el sistema de archivos u otros procesos en Ethereum.

Solidity es un lenguaje de comprobación de tipo estático. Su compilador podría comprobar:

- Todas las funciones deben estar definidas
- El objeto no puede ser nulo
- Operador inválido

1.2 CONTRATO INTELIGENTE

El contrato inteligente es un programa de computador que se ejecuta automáticamente cuando se cumple una condición específica. El contrato inteligente es un conjunto de instrucciones desarrollado en Solidity (podría ser otro lenguaje de programación, pero en este libro solo usaremos Solidity). El lenguaje de programación Solidity se basa en IFTTT (que es la lógica IF-THIS-THEN-THAT: ejecuta el código si se cumple alguna condición). Dado que el contrato inteligente se ejecuta en EVM, no puede acceder a la red, al sistema de archivos ni a otros procesos que se ejecutan en EVM. El contrato inteligente puede acceder a datos externos a través de un Oracle (se trata de un elemento ofrecido por terceros que brinda información externa y que permite que los contratos inteligentes tomen decisiones si es necesario).

Generalmente, el contrato inteligente se puede implementar con base en dos tipos de sistema:

1. Máquina virtual (VM): Ethereum.
2. Docker: Fabric.

Todo el contenido de este libro se basa en los contratos inteligentes sobre Ethereum.

1.3 GAS

El GAS se utiliza para medir cuántos pasos necesitará una transacción en EVM. Es sencillo: si su transacción es compleja, significa que necesita más recursos informáticos (como tiempo de CPU y memoria), deberá pagar más GAS. Todos los códigos de operación en EVM costarán GAS y es poco probable que cambien en el futuro. La métrica más pequeña de GAS es wei, y 1 eth = 10^18 wei = 10^9 gwei.

1. Precio del gas: Precio unitario del gas.
2. Límite de gas: límite superior de gas que el usuario está dispuesto a pagar.

Límite de gas x Precio del gas = la tarifa máxima que el usuario está dispuesto a pagar.

En una transacción, si la tarifa máxima que especificó no se utiliza, el fondo restante se devolverá a su cuenta. Una vez que se agote el gas, la transacción se detendrá donde sea que esté y se lanzará una excepción por falta de gas. Todos los cambios de estado realizados por la transacción serán revertidos.

Pero, una vez que se usan los fondos, aunque la transacción falle, el usuario no recibirá un reembolso ya que con los fondos se recompensa a los mineros. El usuario debe pagar una tarifa por usar el recurso de cómputo y la tarifa consta de tres componentes:

1. Tasa de cálculo.
2. Tarifa de transacción (creación de contrato o llamada de mensaje).
3. Tarifa de almacenamiento (memoria, almacenamiento de datos de cuenta/contrato).

Si su contrato guarda datos en la base de datos, todos los nodos en Ethereum harán lo mismo, lo cual es muy costoso. Es por dicha razón que Ethereum cobrará por el almacenamiento y fomenta un menor uso del almacenamiento. Si la operación es borrar un elemento de almacenamiento, Ethereum lo hará de forma gratuita o incluso se le reembolsará.

Aquí hay un gráfico sobre el precio promedio de GAS: https://etherscan.io/chart/gasprice

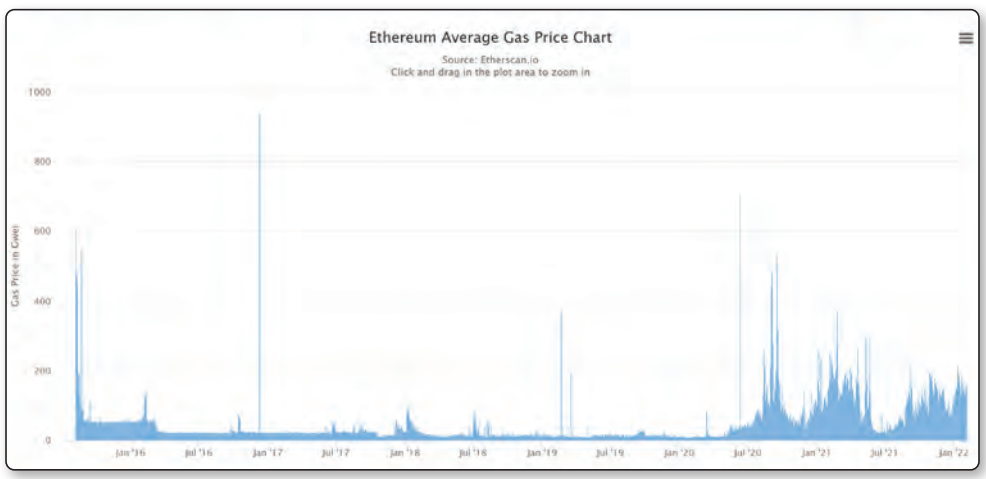

Figura 1.6. Precio promedio de Gas

1.3.1 ¿Por qué GAS?

¿Por qué necesitamos GAS? En términos generales, hay tres razones principales: finanzas, teoría y computación.

Desde el punto de vista financiero, el objetivo es incentivar a los mineros a ejecutar transacciones y contratos inteligentes utilizando su propio tiempo y recursos. Muchas operaciones complejas necesitan más recursos de cómputo; es decir, necesitan pagar más GAS. Si un usuario desea priorizar su transacción, puede enviar una transacción con un precio de GAS más alto. De esta manera, la transacción podría ser procesada antes por el minero incentivado por una tarifa de transacción más alta. Como compensación por el recurso de cómputo en el que invierte el minero, GAS se vuelve más crucial después de que el consenso migre a la Prueba de participación (POS). En la era POS, los mineros ya no son recompensados por minar bloques y empaquetar transacciones, es más importante que los mineros procesen las transacciones y se les pague por gastar recursos en la cadena de bloques.

La razón teórica es alinear los incentivos de los participantes en la red. Gran parte de la teoría de la cadena de bloques analiza cómo mitigar a los actores dañinos o maliciosos en un entorno sin confianza. GAS aborda parcialmente este problema: los mineros tienen incentivos para trabajar en la red y los usuarios no tienen incentivos para actuar mal o escribir códigos maliciosos, ya que están poniendo en riesgo su propio ether (en forma de gas).

Desde el punto de vista computacional, la razón detrás de GAS se remonta a un antiguo aspecto fundamental de la teoría informática: el problema de la interrupción. El problema de la interrupción es el problema de determinar si un programa arbitrario dejará de ejecutarse o si se ejecutará para siempre con solo mirar la descripción y los valores de entrada. En 1936, Alan Turing determinó que es imposible que cualquier máquina resuelva el problema de la interrupción. En EVM, esto significa que un minero nunca puede comenzar a procesar una transacción y saber exactamente que la transacción no continuará para siempre. Con GAS, específicamente, el límite de GAS, siempre se adjunta una cantidad finita de gas a una transacción. Incluso si un minero comenzara a procesar una transacción que estaba codificada para continuar indefinidamente, ya sea por un error o un ataque a la red, el gas eventualmente se acabaría, la transacción terminaría y el minero aún sería compensado.

1.3.2 Componentes del GAS

El GAS se divide en tres componentes: costo del GAS, precio del GAS y límite del GAS. En Ethereum, la fórmula de cálculo de la tarifa de transacción es muy simple:

Tarifa de transacción (Tx Fee) = Costo de GAS * Precio de GAS

Por ejemplo, si una transacción necesitará 50 GAS para completarse, suponiendo que el precio de GAS establecido por el usuario es 2 Gwei, entonces la tarifa total de la transacción es 50 * 2 = 100 Gwei.

1.3.2.1 COSTO DEL GAS

El costo de GAS indica que cada código de operación necesita GAS, esto se encuentra predefinido en la Guía de Ethereum (Ethereum yellow paper). Por ejemplo, un código de operación de "adición" necesita tres unidades de GAS, independientemente de la fluctuación del precio del ether. El GAS para cada código de operación no se cambiará. Es por eso que GAS se usa para estimar el costo de una transacción en lugar de ether. Si se usa ether para el costo de GAS, el precio puede variar considerablemente.

1.3.2.2 PRECIO DEL GAS

El precio de GAS indica a cuánto equivale una unidad de GAS en términos de Ether. Comúnmente, Gwei se usa como unidad de cálculo. Un Gwei equivale a mil millones de Wei, y Gwei es 10 ^ -9 Ether para ser exactos. Es decir, 1 Gwei = 0,000000001 ETH. 1 Wei es la unidad métrica más pequeña para el ether y es indivisible. Si establece el precio de GAS en 20 Gwei, eso significa que pagará 0.00000002 ether por cada paso. Evidentemente, cuanto mayor sea el precio del GAS, más pagarás. Hay varios sitios web como *ethgasstation.info* que proporcionan el precio promedio de GAS. A veces, el usuario puede estar dispuesto a pagar un precio más alto para que su transacción gane prioridad para que los mineros la elijan y la ejecuten. Esto se debe a que: El GAS especificado en la transacción se enviará al minero y el minero ordenará todas las transacciones en su grupo local por precio de GAS. La transacción con un precio de GAS más alto tendrá más posibilidades que aquellas con un precio de GAS más bajo.

1.3.2.3 LÍMITE DE GAS

El límite de GAS es el límite superior del uso de GAS para una transacción específica. Esos son los pasos máximos requeridos para ejecutar su transacción. El límite de GAS será más de lo que realmente se usa. Dado que la complejidad de la transacción varía, el GAS realmente utilizado solo se conoce después de que se haya completado la transacción. Así que antes de enviar la transacción, debe establecer un límite superior de uso de GAS. Si el límite de GAS se establece demasiado bajo, un minero intentará completar la transacción hasta que se agote el GAS. El minero será recompensado cuando se agote el GAS, ya que el minero ha dedicado tiempo y energía a la transacción de los usuarios. Y en blockchain, la transacción se configurará como falsa. El mecanismo GAS está configurado para proteger al usuario y al minero: no perderán fondos ni energía debido al código defectuoso y al ataque malicioso.

1.4 ETHER

Ether es el token emitido en Ethereum. ETH es el símbolo corto de Ether. Y ETH es una moneda criptográfica negociable. En Ethereum, Ether se usa principalmente para pagar la tarifa de transacción. La tarifa de transacción es igual al costo de GAS multiplicado por el precio de GAS y se paga en ETH. El usuario puede establecer el precio de GAS, pero tenga en cuenta que si el precio de GAS es demasiado bajo, es posible que ningún minero esté dispuesto a empaquetar la transacción.

1.5 CUENTAS

Cada cuenta tiene su propia dirección. En el espacio de direcciones, hay dos tipos de cuenta: una es una cuenta de propiedad externa (EOA por sus siglas en inglés, external owned account) controlada por clave pública/privada. Normalmente, solo las personas pueden tener dicha cuenta que se utiliza para guardar ETH; el otro es una cuenta de contrato controlada por código. Estos dos tipos tienen alguna diferencia, pero la más importante es que solo EOA puede iniciar transacciones.

Ambos tipos de cuentas pueden enviar y recibir ethers. Por lo tanto, ambos tienen un campo de saldo para realizar un seguimiento de ellos. Sin embargo, las cuentas de contrato también pueden almacenar datos. Por lo tanto, tienen un campo de almacenamiento y un campo de código que contiene instrucciones de máquina sobre cómo manipular los datos almacenados. En otras palabras, las cuentas de contrato son contratos inteligentes que viven en la cadena de bloques.

contract account = balance + code + storage
external account = balance + empty + empty

1.6 TRANSACCIÓN

Una transacción es un mensaje enviado de una cuenta a otra cuenta. Podríamos transferir ETH entre cuentas mediante el envío de transacciones a un EOA. Si la cuenta de destino es una cuenta de contrato, el envío de la transacción activará la ejecución de su código. Dado que cada transacción se ejecutará en todos los nodos de Ethereum, la cadena de bloques registrará la ejecución del código o la transacción.

En este capítulo, presentamos todos los conceptos y términos necesarios como base para comprender el lenguaje de programación Solidity y los contratos inteligentes.