ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

Ciberseguridad

Máster Universitario en Inteligencia Computacional e Internet de las Cosas

Red blockchain Ethereum

Carlos Checa Moreno

UNIVERSIDAD D CÓRDOBA

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Capítulo 1

INTRODUCCIÓN

El mundo de la tecnología se encuentra en continuo cambio y desarrollo. Concretamente, el área de la informática va tomando partida en nuestras vidas cada vez en mayor porcentaje: negocios que se digitalizan, plataformas de educación online, procesos burocráticos por internet, etc. Por esta razón, cada días es más importante la ciberseguridad implementada en los sistemas que nos dirán cuánto dinero hay en nuestra cuenta, que nos permiten realizar exámenes para nuestra universidad o que guardan un registro de todos nuestros mensajes de redes sociales o trabajo.

Unos de los problemas fundamentales de los sistemas informáticos es que, como usuarios, dependemos en múltiples ocasiones de un sistema centralizado. Por ejemplo, un banco será quien se ocupe de confirmarnos el dinero que tenemos. Sin embargo, esto tiene una serie de desventajas como podrían ser: costos y comisiones, procesos lentos y burocráticos o falta de transparencia.

Como solución a sistemas donde dependemos de un único ente tenemos las redes peer-to-peer (P2P) [1]. Estas redes consisten en organizar recursos distribuidos para cumplir una función de forma descentralizada. El P2P ofrece muchas ventajas como: mejor escalabilidad al evitar la dependencia de puntos centralizados, reducción de costos mediante la eliminación la necesidad de infraestructura costosa permitiendo la comunicación directa entre clientes y agregación de recursos, aprovechando la capacidad computacional, almacenamiento y ancho de banda de múltiples nodos. Además, los sistemas P2P promueven la autonomía y privacidad, permitiendo que los usuarios mantengan un mayor control sobre sus datos y recursos.

Una de las implementaciones de las redes P2P más sonadas de las últimas décadas es la *blockchain*. La blockchain o cadena de bloques es una estructura de datos en las que se va

encadenando la información en bloques que, a su vez, mantienen registro a bloques más antiguos. Gracias a la memoria de los bloques y técnicas criptográficas, la blockchain solo podrá ser modificada si todos los bloques son modificados, lo que lo hace un sistema extremadamente seguro.

Uno de los primeros usos que se le dio a esta estructura de datos fue Bitcoin. Mediante en el que usando la blockchain como una base de datos distribuida, se conseguiría crear una divisa que no dependería de ningún estado.

Sin embargo, en este trabajo se abordará con mayor detalle Ethereum. El cual ofrece un lenguaje de programación para trabajar sobre una blockchain. Permitiendo numerosas aplicaciones que se estudiarán en los siguientes puntos.

Capítulo 2

HISTORIA DE LA BLOCKCHAIN

Antes de comenzar a hablar de Ethereum es necesario conocer la historia que desembocó en esta innovadora tecnología.

2.1 Inicios de la blockchain

Satoshi Nakamoto, cuya identidad sigue siendo desconocida hoy en día. Fue el pseudónimo de la persona o grupo de personas que propusieron en primer lugar el bitcoin en la blockchain y la pondrían en funcionamiento en enero de 2009 [2].

Nakamoto ideó un sistema de consenso descentralizado donde las transacciones se agrupan en bloques cada diez minutos y se enlazan de forma permanente. Para garantizar la seguridad, implementó la *proof-of-work*, un mecanismo que exige a los nodos resolver problemas computacionales para validar bloques y evitar fraudes.

Primero, desde un punto de vista técnico, Bitcoin opera como un sistema de transición de estados. Su estado representa la propiedad de todos los bitcoins existentes, y cada transacción es una función de transición que modifica ese estado.

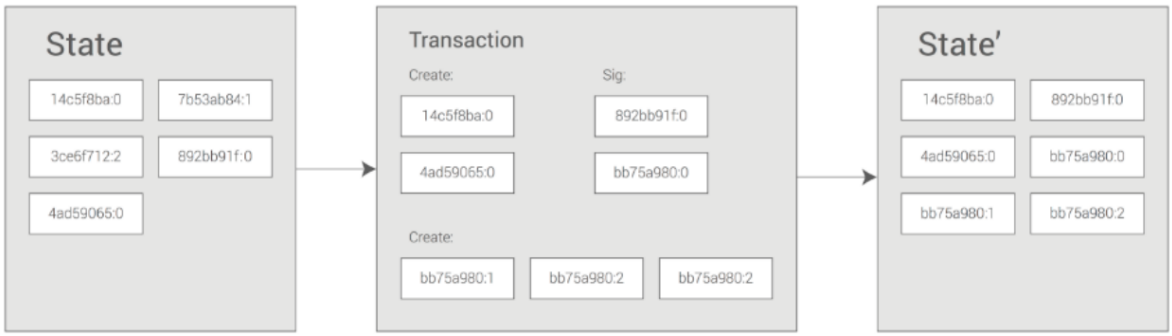

Figura 1: Bitcoin como sistema de transición de estados [2]

2.2 Minería

Para validar estas transacciones entran el juego la proof-of-work mencionada anteriormente, también llamada minería. Esta prueba se basa en generar un hash que cumpla con un requisito específico, lo que requiere una gran cantidad de intentos aleatorios hasta encontrar una solución.

El algoritmo utilizado en Bitcoin para esta prueba de trabajo es SHA-256. Como recompensa por este esfuerzo computacional, el minero que valide un bloque recibe una cantidad de Bitcoin, además de las comisiones de las transacciones incluidas en el bloque.

Sin embargo, teóricamente este no es un sistema completamente seguro. Un ataque del 51%, que ocurriría si un usuario o usuarios organizados consiguiesen más poder computacional que el resto de la red combinada, permitiría modificar transacciones pasadas. Aunque este tipo de ataque es posible, el costo en energía y hardware que sería necesario para efectuarlo hace que sea impensable en la práctica.

Así, Bitcoin se convirtió en la primera implementación exitosa de una moneda digital descentralizada, asegurando que las transacciones fueran verificadas de manera segura y sin posibilidad de manipulación por parte de una sola entidad

2.3 Creación de Ethereum

Ethereum fue presentado por primera vez en el paper de Vitalik Buterin [2]. Buterin se propuso suplir las limitaciones de Bitcoin y ampliar las aplicaciones de la tecnología blockchain

A diferencia de Bitcoin, que se centra únicamente en transacciones financieras, Ethereum introduce un lenguaje de programación completo dentro de su blockchain, lo que permite la creación de contratos inteligentes [3]. Esto permite a los desarrolladores definir sus propias reglas para la propiedad, el formato de las transacciones y las funciones de cambio de estado.

Capítulo 3

ETHEREUM

3.1 ¿Qué es Ethereum?

Ethereum es una red de ordenadores que siguen un conjunto de normas conocidas como el protocolo Ethereum [4]. Esta red actúa como base de comunicaciones, aplicaciones y activos digitales sin la necesidad de intermediarios o autoridades centrales.

Debemos tener claro que Ethereum no representa solamente una criptomoneda como Bitcoin. Aunque tenga su propia criptomoneda también llamada ETH. Si no que se refiere a la red en la cual podemos tomar parte y desarrollar contratos inteligentes con diversas reglas para conseguir muchas aplicación.

Estos contratos inteligentes se refieren a acuerdos que llegamos de forma virtual con otro usuario de la red. Por ejemplo, podemos establecer un contrato en el cual cuando un usuario A le realice una transacción de 0.1 ETH al usuario B, el B le mandará automáticamente un código con el que el primer usuario podrá acceder a una vivienda turística. De esta forma no necesitamos un intermediario que monitorice nuestros acuerdos y tome comisiones.

3.2 Características principales

Características principales de Ethereum [4]:

- **Banco para todos**: Con solo una conexión a Internet, cualquier persona puede acceder a productos de préstamo y ahorro basados en Ethereum, ofreciendo servicios financieros a quienes tradicionalmente no tienen acceso a ellos.

- **Internet abierto**: Permite a los usuarios interactuar y construir aplicaciones en la red, otorgándoles control sobre sus activos e identidad sin depender de grandes corporaciones.
- **Transacciones directas**: Facilita la coordinación y transferencia de activos digitales directamente entre personas, eliminando la necesidad de intermediarios.
- **Resistencia a la censura**: Al ser descentralizada, ningún gobierno o empresa controla Ethereum.
- **Garantías comerciales**: Mediante contratos inteligentes, se asegura que los fondos solo se transfieran si se cumplen las condiciones acordadas, proporcionando seguridad tanto a clientes como a desarrolladores.
- **Interoperabilidad de aplicaciones**: Las aplicaciones en Ethereum se construyen sobre una cadena de bloques compartida, lo que permite que interactúen y se complementen entre sí, creando mejores productos y experiencias.

3.3 Funcionamiento

La red Ethereum se basa en un sistema descentralizado de nodos, ordenadores, que interactúan entre sí para asegurar el funcionamiento de la blockchain. Cada nodo, al ejecutar el software adecuado (que veremos en el punto 3.4 Formar parte de Ethereum), se convierte en parte activa de la red, validando transacciones y asegurando la integridad del sistema.

Ethereum, emplea el algoritmo de consenso Proof of Stake (PoS) para validar las transacciones. En lugar de competir por resolver complejos problemas matemáticos como en el Proof of Work (PoW) utilizado por Bitcoin, los validadores en PoS son seleccionados de acuerdo con la cantidad de ETH que tienen bloqueado como garantía. Este modelo no solo mejora la eficiencia energética de la red, sino que también aumenta la seguridad, ya que un atacante tendría que poseer un 51% de los ETHs de la red para comprometer el sistema. Además, si un validador actúa de manera maliciosa, podría perder su participación (o "stake"), lo que lo desincentiva de intentar manipular la red.

En la siguiente figura podemos observar las diferencias y ventajas del PoS respecto al PoW. PoS mejora la escalabilidad, reduce el impacto ambiental y hace la red más accesible para los usuarios.

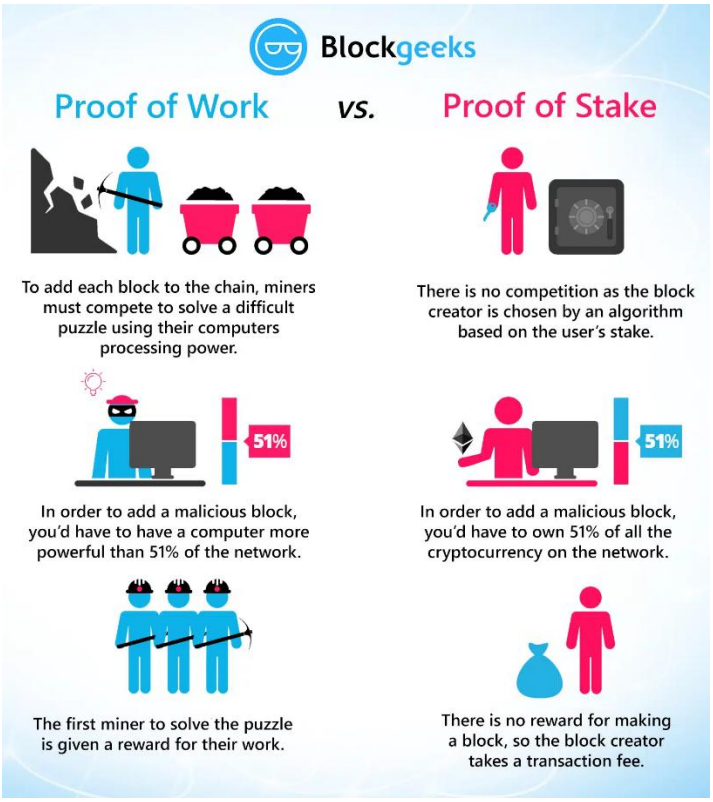

Figura 2: Proof of Work vs. Proof of Stake [6]

3.4 Seguridad

En cuanto a la seguridad criptográfica, Ethereum emplea el algoritmo Keccak-256 [8] para la generación de hashes. Keccak admite un input de cualquier tamaño y responde con una salida fija de 256 bits.

Este algoritmo emplea un *sponge construction* [9]. Este es un esquema iterativo para construir una función F que admite entradas de longitud variable y salida de longitud arbitraria, basándose en una permutación de longitud fija que opera sobre un número fijo de b bits.

En la figura 3 podemos apreciar el funcionamiento del sponge construction. Primero, la cadena de entrada se ajusta con una regla de relleno reversible y se divide en bloques de r bits. Luego, los b bits del estado se inicializan a cero y la construcción de esponja procede en dos fases:

1. **Fase de absorción**: se usa una operación XOR para combinar los bloques de entrada de r bits con los primeros r bits del estado mediante una operación XOR, mientras se aplica la función f.
2. **Fase de extracción**: los primeros r bits del estado se extraen y se devuelven como bloques de salida, acompañados de aplicaciones de la función f. El usuario tiene la libertad de elegir la cantidad de bloques de salida. Los últimos c bits del estado no se ven alterados por los bloques de entrada y no se incluyen en la salida durante esta fase.

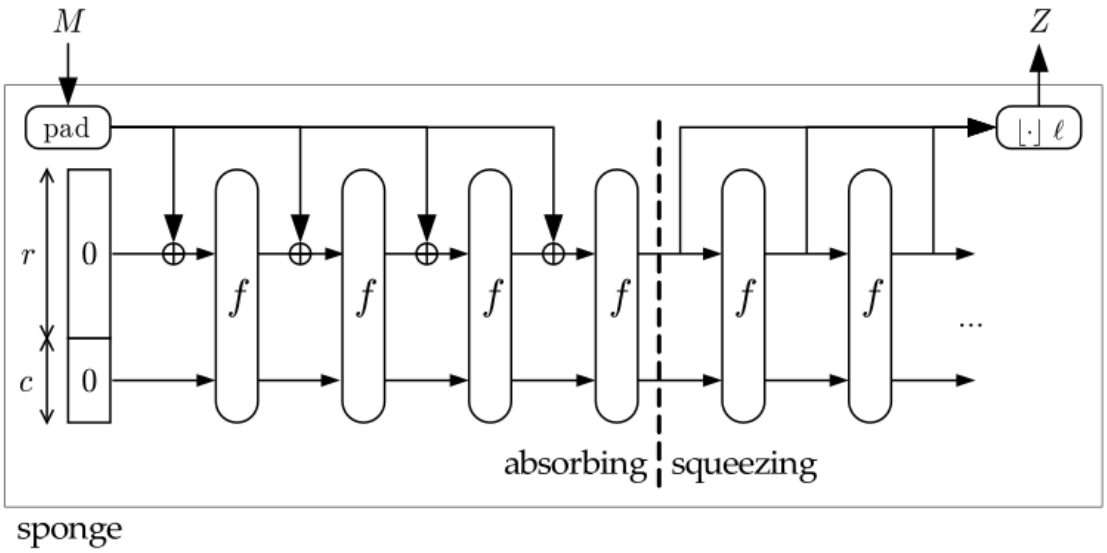

Figura 3: Sponge construction [9]

3.5 Formar parte de Ethereum

Si queremos formar parte de esta red y convertir nuestro ordenador en un nodo, de tal forma que podamos verificar bloques y datos de transacciones, deberemos ejecutar software de Ethereum.

Para funcionar correctamente, un nodo necesita ejecutar dos tipos de clientes: un cliente de ejecución y un cliente de consenso [5]:

- **Cliente de ejecución**: Se encarga de escuchar las nuevas transacciones que se transmiten en la red, ejecutarlas en la Ethereum Virtual Machine (EVM) y mantener actualizada la base de datos con el estado más reciente de Ethereum.

- **Cliente de consenso**: Implementa el algoritmo de prueba de participación (*Proof-of-Stake)* y valida la información proporcionada por el cliente de ejecución para que toda la red llegue a un acuerdo sobre el estado de la blockchain.

Además, existe un software adicional llamado validador, que se puede agregar al cliente de consenso para permitir que un nodo participe en la seguridad de la red.

Ethereum admite múltiples implementaciones de clientes en diferentes lenguajes de programación, desarrollados por distintos equipos. Esta diversidad de clientes fortalece la red, reduciendo el riesgo de que un solo fallo afecte a todo el sistema. Por ejemplo, tenemos Besu desarrollado en java, Geth y Erigon en Go y Nethermind en C#.

A pesar de estas diferencias, todos los clientes siguen las mismas reglas establecidas en las especificaciones oficiales que podemos encontrar en el Ethereum Yellow Paper [7].

Capítulo 4

PRÁCTICA. INSTALANDO UN NODO ETHEREUM

Concluyendo esta labor de investigación sobre Ethereum convertiré mi ordenador personal en un nodo de esta red. Para esta tarea usaré Geth (Go-ethereum) que es la implementación de Ethereum mediante Go.

4.1 Cliente de ejecución Geth

Primero descargaré Geth para Windows

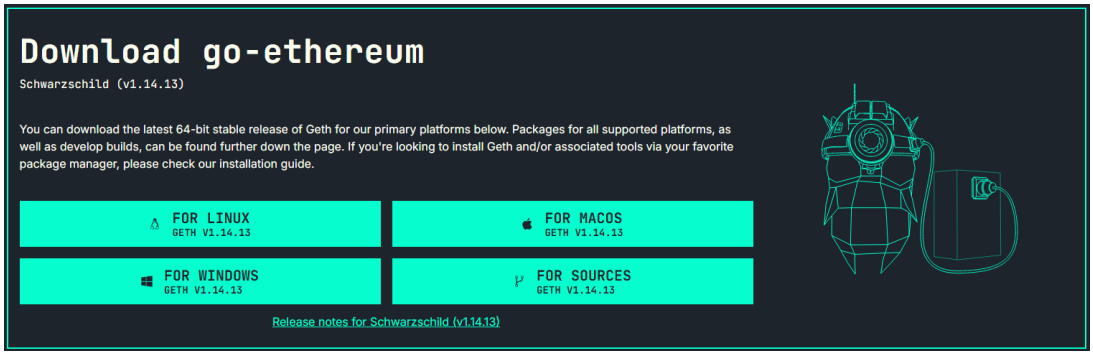

Figura 4: Descargar go-ethereum

Una vez descargado. Debemos tener en cuenta que tenemos unos requisitos de tamaño de disco muy grande. Geth nos recomienda 2TB para un nodo completo que ejecute Geth y un cliente de consenso. Geth por sí mismo nos ocupará más de 650 GB y crecerá a un ritmo de 14 GB por semana con el tamaño de caché predeterminado [10].

Simplemente ejecutando `./geth.exe` ya estaremos conectándonos a la red. En la figura 5 se muestra la ejecución de Geth en el terminal de Windows. Durante el arranque, se configuran parámetros como caché, base de datos y conexiones de red, además de mostrarse mensajes informativos y advertencias sobre la inicialización del protocolo y la validación de datos.

También se listan las principales actualizaciones (forks) de Ethereum con sus respectivos números de bloque y enlaces a documentación. Además, se confirma la transición de Ethereum de Proof of Work (PoW) a Proof of Stake (PoS) con The Merge.

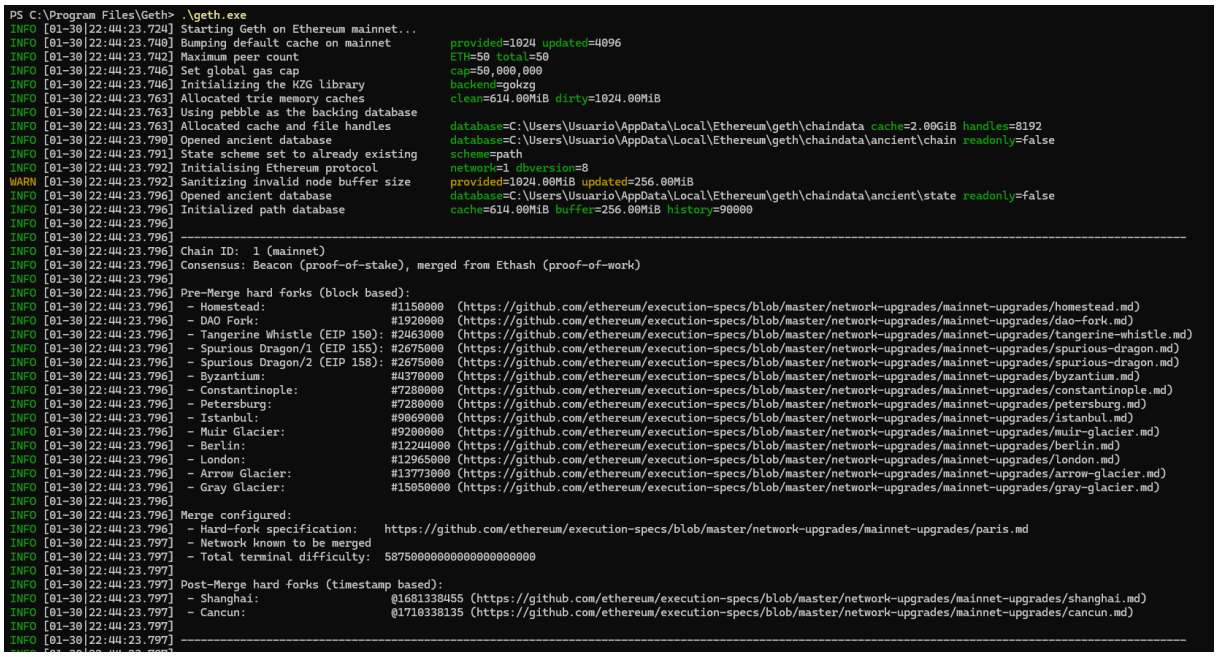

Figura 5: Inicio nodo Ethereum

En la figura 6 ya podemos ver como el está buscando pares (peers) para sincronizarse, pero muestra una advertencia indicando que está en una red post-Merge, pero no se ha detectado un cliente de consenso (necesario para Proof of Stake). Esto sugiere que el nodo necesita conectarse a un cliente de consenso para seguir la cadena.

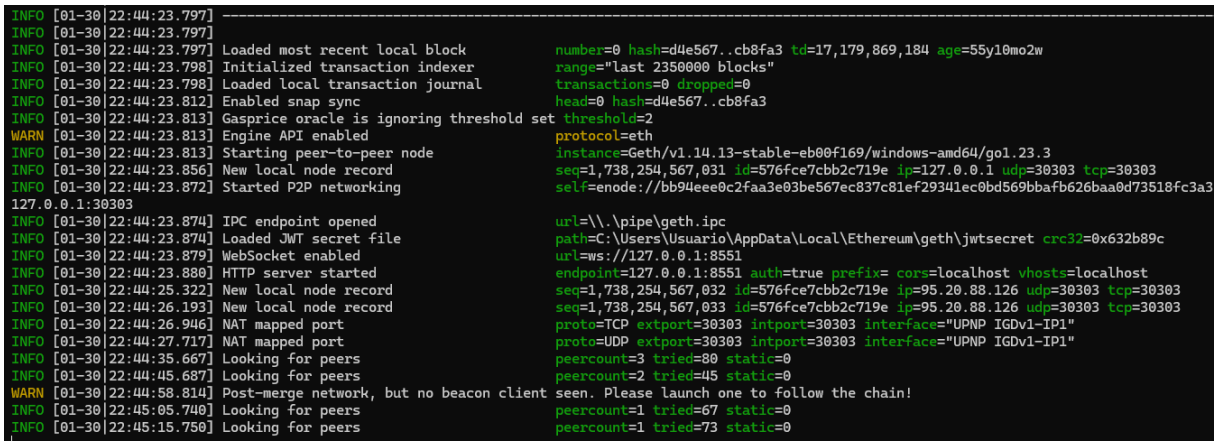

Figura 6: No beacon client

4.2 Cliente de consenso Prysm

Como cliente de consenso usaré Prysm, para descargarlo podemos usar: "`curl https://raw.githubusercontent.com/prysmaticlabs/prysm/master/prysm.bat --output prysm.bat | reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1`" [13].

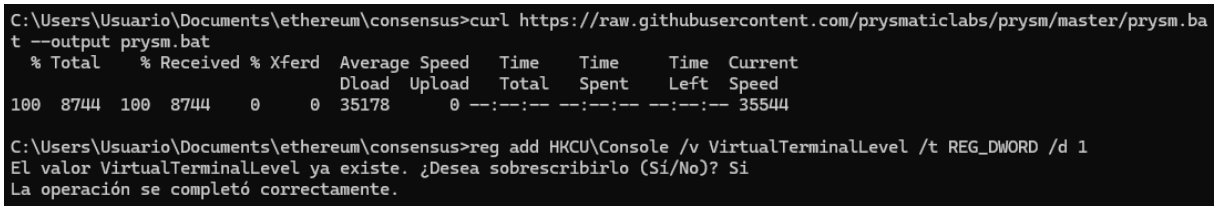

Figura 7: Descargando Prysm

Una vez descargado Prysm usaremos "`prysm.bat beacon-chain generate-auth-secret`" para crear una clave que nos servirá para sincronizar los dos clientes.

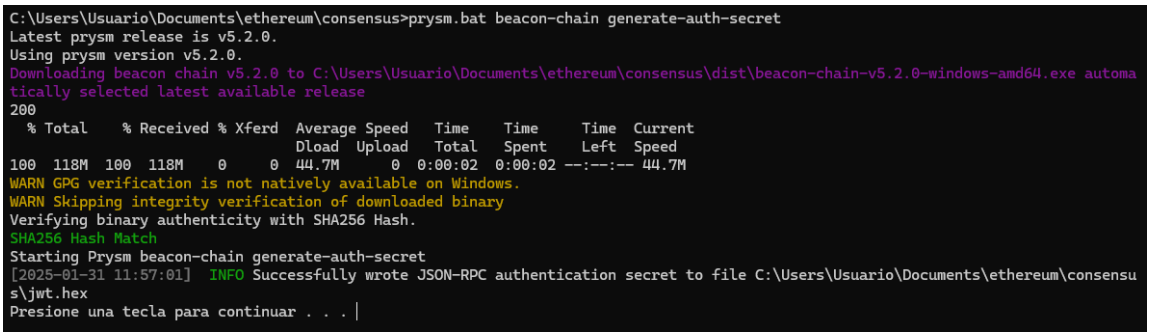

Figura 8: Obtener clave

4.3 Ejecutando nodo

Una vez tengamos listos los dos clientes y el fichero con la clave que indicaremos a Prysn y Geth para que se sincronicen ya podremos ejecutar nuestro nodo completo de Ethereum.

Primero, pondré en marcha el cliente de ejecución indicando la ruta al archivo jwt.hex que es la clave que generamos anteriormente con Prysm:

```
.\geth.exe --mainnet --http --http.api eth,net,engine,admin --authrpc.jwtsecret=C:\Users\Usuario\Documents\ethereum\jwt.hex
```

A continuación, ejecutaré el cliente de consenso Prysm indicando de igual forma la ruta a jwt.hex (la primera vez deberemos aceptar unos términos y condiciones=:

```
prysm.bat beacon-chain --execution-endpoint=http://localhost:8551 --mainnet --jwt-secret=C:\Users\Usuario\Documents\ethereum\jwt.hex --checkpoint-sync-url=https://beaconstate.info --genesis-beacon-api-url=https://beaconstate.info
```

```
C:\Users\Usuario\Documents\ethereum\consensus>prysm.bat beacon-chain --execution-endpoint=http://localhost:8551 --mainnet --jwt-se
cret=C:\Users\Usuario\Documents\ethereum\jwt.hex --checkpoint-sync-url=https://beaconstate.info --genesis-beacon-api-url=https://b
eaconstate.info
Ya existe el subdirectorio o el archivo C:\Users\Usuario\Documents\ethereum\consensus\dist.
Latest prysm release is v5.2.0.
Using prysm version v5.2.0.
Beacon chain is up to date.
WARN GPG verification is not natively available on Windows.
WARN Skipping integrity verification of downloaded binary
Verifying binary authenticity with SHA256 Hash.
SHA256 Hash Match
Starting Prysm beacon-chain --execution-endpoint=http://localhost:8551 --mainnet --jwt-secret=C:\Users\Usuario\Documents\ethereum\
jwt.hex --checkpoint-sync-url=https://beaconstate.info --genesis-beacon-api-url=https://beaconstate.info

Prysm Terms of Use

By downloading, accessing or using the Prysm implementation ("Prysm"), you (referenced herein
as "you" or the "user") certify that you have read and agreed to the terms and conditions below.

TERMS AND CONDITIONS: https://github.com/prysmaticlabs/prysm/blob/develop/TERMS_OF_SERVICE.md


Type "accept" to accept this terms and conditions [accept/decline]: (default: decline):
```

Figura 9: Términos y condiciones Prysm

En la figura 10 podemos ver el cliente de ejecución a la izquierda y a la derecha el de consenso. En el Geth vemos como, mientras arrancábamos el cliente de Prysm, nos volvió a dar el WARN de que no había beacon client. Una vez activamos el cliente de consenso, Geth empezaría a darnos los avisos de que está sincronizándose.

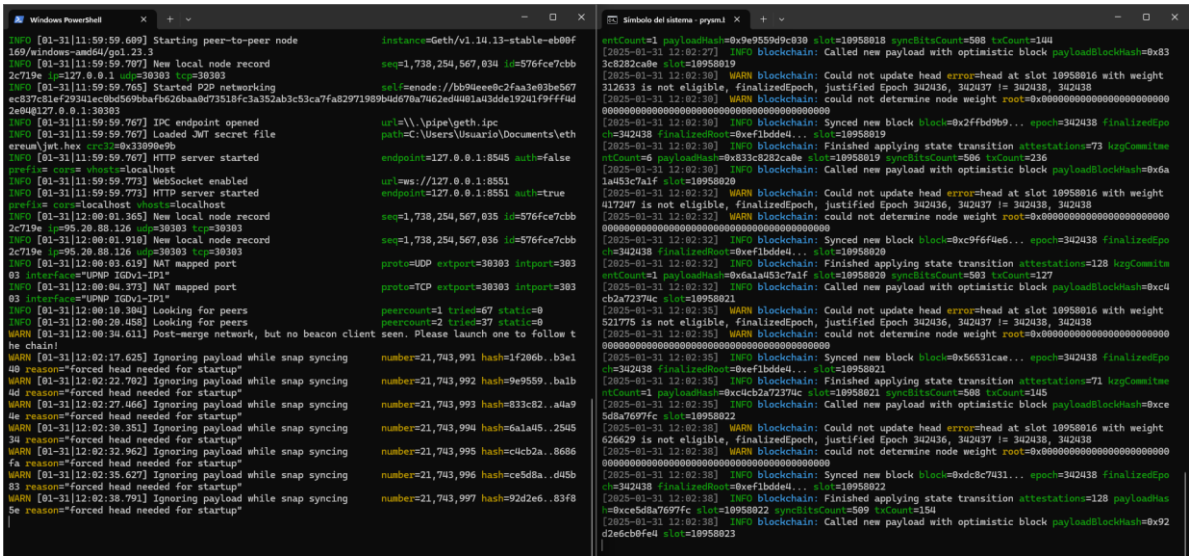

Figura 10: Geth y Prysm sincronizándose

Una vez sincronizados ambos clientes, estos softwares ya empezarían a trabajar con normalidad y estaríamos formando parte de Ethereum. En Geth observamos como se está actualizando continuamente a la head, bloques más recientes, de la cadena de bloques.

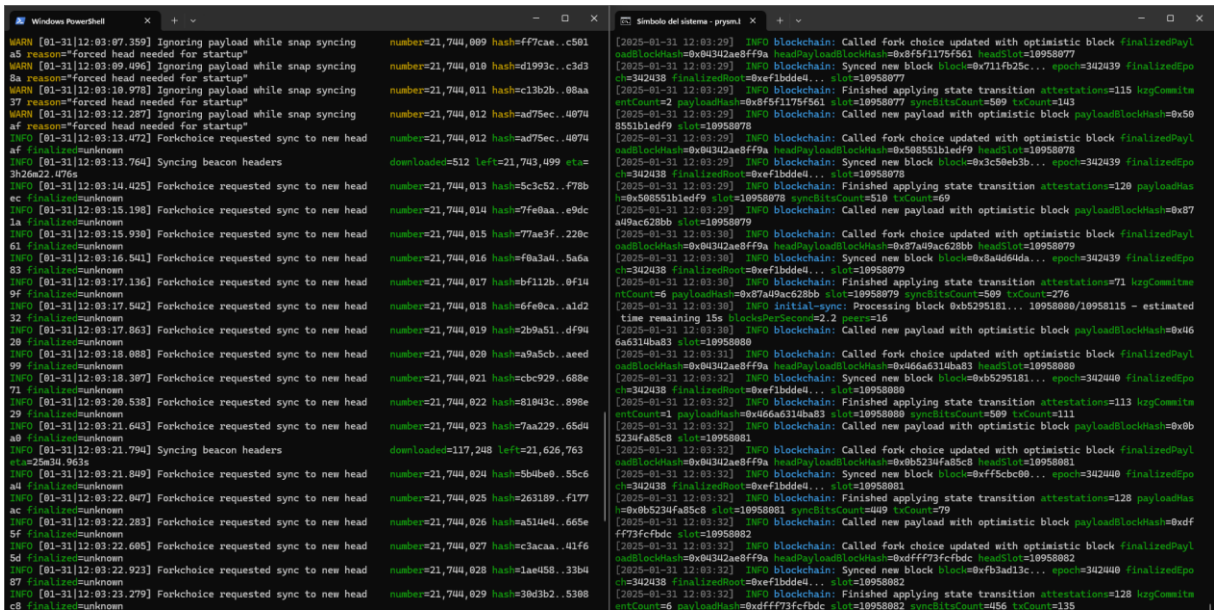

Figura 11: Nodo Ethereum ejecutándose

Capítulo 5

CONCLUSIÓN

El desarrollo y estudio de Ethereum ha permitido comprender el impacto y las capacidades de la tecnología blockchain más allá de su uso en criptomonedas. A lo largo de este trabajo, hemos abordado cómo Ethereum ha ampliado las posibilidades de la blockchain al introducir los contratos inteligente, mediante un lenguaje de programación que puede ser usado por cualquier usuario. Este enfoque ha demostrado ser una solución eficiente y segura a problemas tradicionales de centralización, ofreciendo mayores ventajas como la resistencia a la censura, la transparencia y la autonomía de los usuarios.

Además, hemos explorado cómo Ethereum, mediante el uso del algoritmo Proof of Stake (PoS), logra una mayor eficiencia energética y mejora la escalabilidad de su red, lo cual es crucial para su futuro crecimiento.

A nivel técnico. Se ha revisado el funcionamiento del algoritmo de hash usado por Ethereum, Keccak-256, que garantizan la seguridad y confiabilidad de las transacciones. También se ha instalado un nodo de Ethereum con sus respectivos clientes de ejecución, Geth, y de consenso, Prysm, lo cual ha sido de gran utilidad para afianzar los conocimientos adquiridos en la práctica.

A nivel personal, este trabajo también me ha servido para diferenciar correctamente Ethereum y Bitcoin. Un error típico entre personas que no se han interesado demasiado por Ethereum, como yo mismo antes de este trabajo, es pensar que Ethereum es simplemente una criptomoneda más como Bitcoin, cuando en realidad la criptomoneda es ETH y Ethereum es una red que nos permite realizar muchas más acciones y aplicaciones. Así mismo, instalar el nodo de Ethereum me ha supuesto una experiencia muy enriquecedora y útil para entender en profundidad esta red.

En conclusión, Ethereum ha demostrado ser una de las plataformas más innovadoras y prometedoras que da uso a blockchain. Esta red tiene un impacto significativo tanto en el ámbito financiero como en el tecnológico, abriendo el camino para aplicaciones descentralizadas en una amplia gama de sectores.

BIBLIOGRAFÍA

[1] Milojicic, Dejan & Kalogeraki, Vana & Lukose, Rajan & Nagaraja, Kiran & Pruyne, Jim & Richard, Bruno & Rollins, Sami & Xu, Zhichen. (2002). Peer-to-Peer Computing.

[2] V. Buterin, (2013) “Ethereum white paper: a next generation smart contract & decentralized application platform,”. https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[3] D. Vujičić, D. Jagodić and S. Ranđić. (2018). "Blockchain technology, bitcoin, and Ethereum: A brief overview," 17th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, pp. 1-6, doi: 10.1109/INFOTEH.2018.8345547.

[4] Ethereum. (s.f.) ¿Qué es Ethereum? Recuperado en enero de 2025 de https://ethereum.org/es/what-is-ethereum/

[5] Ethereum. (s.f.) Nodes and clients. Recuperado en enero de 2025 de https://ethereum.org/en/developers/docs/nodes-and-clients/

[6] R. Wernert. (2024). Understanding Proof of Work and Proof of Stake. https://www.summit.io/pl/blog-posts/understanding-proof-of-work-and-proof-of-stake

[7] G. Wood. (2025). Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain. https://ethereum.github.io/yellowpaper/paper.pdf

[8] Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., & Van Keer, R. (2018). Keccak Summary Specifications. STMicroelectronics, Radboud University, NXP Semiconductors. https://keccak.team/keccak_specs_summary.html

[9] Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., & Van Keer, R. (2018). The sponge and duplex constructions. STMicroelectronics, Radboud University, NXP Semiconductors https://keccak.team/sponge_duplex.html

[10] Go-ethereum. (2023). Hardware requirements. https://geth.ethereum.org/docs/getting-started/hardware-requirements

[11] J. Maldonado. Bit2Me. (2021). Cómo INSTALAR un NODO de ETHEREUM con GETH - Bit2Me Academy Pro. Youtube. https://youtu.be/JwoSzF5HLwE?si=-xv8BybfTUU6xPEp

[12] Go-ethereum. (2023). Command-line Options. https://geth.ethereum.org/docs/fundamentals/command-line-options

[13] Prysm. (2024). Quickstart: Run a node and (optionally) stake ETH using Prysm, https://docs.prylabs.network/docs/install/install-with-script