Análisis de la privacidad en Ethereum mediante la aplicación de protocolos de prueba de conocimiento nulo

Antonina Serebriakova

Máster Universitario de Ciberseguridad y Privacidad

"Sistemas de blockchain"

Tutor de TF
Alberto Ballesteros Rodríguez

Profesor responsable de la asignatura
Víctor García Font

Junio, 2023

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis de la privacidad en Ethereum mediante la aplicación de protocolos de prueba de conocimiento nulo*
Nombre del autor:	*Antonina Serebriakova*
Nombre del consultor/a:	*Alberto Ballesteros Rodríguez*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*Junio, 2023*
Titulación o programa:	*Máster Universitario de Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Sistemas de blockchain*
Idioma del trabajo:	*Castellano*
Palabras clave	*Privacidad, Ethereum, Protocolos de conocimiento nulo*

Resumen del Trabajo

Este trabajo tiene por objetivo realizar un análisis de la privacidad en Ethereum, una blockchain ampliamente utilizada en el ámbito de las criptomonedas. Se analizan en detalle los métodos utilizados para garantizar la privacidad, poniendo el foco en los protocolos zk-SNARK y zk-STARK. Estos protocolos, conocidos como pruebas de conocimiento nulo, desempeñan un papel clave a la hora de garantizar la seguridad y privacidad de las transacciones en Ethereum.

Se discuten los mecanismos y características subyacentes de los protocolos zk-SNARK y zk-STARK, resaltando tanto sus ventajas como desventajas. Mientras que zk-SNARK requiere una fase inicial de configuración confiable y es vulnerable a los ataques cuánticos, zk-STARK ofrece resistencia a dichos ataques y no requiere dicha configuración, aunque a costa de una mayor complejidad y tamaño de las pruebas.

Además, se exploran varias aplicaciones de estos protocolos en proyectos reales basados en la blockchain Ethereum, proporcionando una visión de su implementación práctica y del impacto en la privacidad de los usuarios de Ethereum.

Por último, se reflexiona sobre las aplicaciones reales de estos métodos y su potencial para mejorar significativamente la privacidad en Ethereum.

Abstract

This study aims to examine privacy within Ethereum, a blockchain widely used in the cryptocurrency domain. The methods used to protect privacy are thoroughly examined, with a particular focus on the zk-SNARK and zk-STARK protocols. These protocols, also known as zero-knowledge proofs, play a crucial role in maintaining security and privacy in Ethereum transactions.

A discussion is provided on the basic mechanisms and features of the zk-SNARK and zk-STARK protocols, highlighting both their advantages and disadvantages. While zk-SNARK requires an initial trusted setup phase and is vulnerable to quantum attacks, zk-STARK offers resistance to these attacks and does not require such a setup, although with an increase in complexity and proof size.

In addition, an exploration is conducted into the various applications of these protocols in real-world projects based on the Ethereum blockchain, providing insights into their practical implementation and the impact on the privacy of Ethereum users.

To conclude, thoughts are put forward regarding the real-world applications of these methodologies and their potential to significantly improve privacy within Ethereum.

Índice

1. Introducción

En los últimos años, el uso de criptomonedas y la realización de transacciones financieras en blockchain han experimentado un rápido crecimiento. La blockchain Ethereum se ha destacado como una de las más populares y completas, permitiendo la creación de tokens y criptomonedas personalizadas en una sola blockchain.

La transparencia de los datos es una de las principales características de Ethereum, lo que impide que los usuarios falsifiquen transacciones. Sin embargo, esta misma naturaleza de Ethereum ha hecho que la blockchain sea vulnerable a amenazas de seguridad y ataques, debido al hecho de que la transparencia implica un compromiso con la privacidad.

Aunque las transacciones en Ethereum pueden parecer anónimas a simple vista, la blockchain es intrínsecamente no privada. Cada transacción se transmite a todos los nodos de la red, lo que permite que se puedan realizar análisis de transacciones y desanonimización de usuarios. La falta de anonimato de las transacciones en la blockchain de Ethereum es un problema importante que puede afectar a grandes empresas y propietarios de tokens, ya que la divulgación de información confidencial puede influir en el estado del mercado.

En este contexto, la adopción de soluciones de privacidad es fundamental para garantizar la protección de la información de los usuarios y mejorar la adopción de la tecnología blockchain en entornos regulados y no regulados.

A lo largo de este trabajo, se investigarán las fortalezas y debilidades de las soluciones existentes de privacidad y se analizará su aplicabilidad en diferentes casos de uso en el ecosistema Ethereum. Además, se discutirá el potencial, los desafíos y oportunidades de los protocolos zk-SNARK y zk-STARK para garantizar la privacidad en Ethereum, proporcionando una visión integral sobre cómo estas tecnologías pueden contribuir a mejorar la privacidad en la blockchain.

1.1. Problema a resolver

En la actual era digital, la privacidad es un tema crucial, especialmente cuando se trata de tecnologías emergentes como blockchain y criptomonedas. Ethereum es una de las principales plataformas de blockchain, y su popularidad ha aumentado drásticamente en los últimos años gracias al desarrollo de aplicaciones descentralizadas (DApps) en diversos sectores.

Sin embargo, la privacidad sigue siendo un problema importante en la plataforma Ethereum. A pesar de que la tecnología blockchain es inherentemente segura y transparente, la información almacenada en ella es completamente pública y rastreable. Esto significa que, aunque los usuarios de Ethereum pueden mantener sus identidades privadas, su actividad en la plataforma no lo es.

Para abordar este problema, se han desarrollado diversas soluciones tecnológicas, como las tecnologías de zk-SNARK y zk-STARK. Estos protocolos son conocidos como sistemas de prueba de conocimiento nulo, que permiten a los usuarios demostrar que poseen cierta información sin revelar la información en sí misma. Esto significa que los usuarios pueden realizar transacciones en la plataforma Ethereum de forma completamente privada, sin que terceros puedan acceder a los datos de las mismas.

Aunque los protocolos de zk-SNARK y zk-STARK son prometedores, también presentan desafíos significativos. Una de las principales limitaciones es la complejidad técnica de estas soluciones, lo que dificulta su implementación en aplicaciones descentralizadas. Además, estas soluciones también pueden afectar la escalabilidad y el rendimiento de la plataforma Ethereum.

Más allá de los desafíos técnicos, también es importante considerar el contexto sociopolítico más amplio en el que se inserta la privacidad en Ethereum. La tesis de Gilles Deleuze [3] sobre la sociedad de control es especialmente relevante en este contexto. Deleuze sostiene que la sociedad contemporánea se ha transformado de una sociedad disciplinaria a una sociedad de control, en la que los mecanismos de vigilancia y control están más dispersos y se manifiestan de maneras más insidiosas.

En esta sociedad de control, la privacidad y la libertad individual están siendo erosionadas por el control tecnológico y la vigilancia constante. Por lo tanto, la implementación de tecnologías de privacidad como zk-SNARK y zk-STARK en Ethereum es esencial para proteger la privacidad y la libertad individual en este contexto sociopolítico más amplio.

En resumen, este trabajo se enfoca en la mejora de la privacidad en Ethereum mediante el análisis de tecnologías de prueba de conocimiento nulo como zk-SNARK y zk-STARK. Para ello se explorarán los desafíos técnicos de estas soluciones, así como su relevancia en el contexto de la sociedad de control de Deleuze. En última instancia, este trabajo pretende argumentar que la privacidad en Ethereum es fundamental para proteger la libertad individual y reducir las desigualdades en la era digital actual.

1.2. Estado del arte

Cualquiera puede unirse a la red principal de Ethereum. Además, todos los datos de las transacciones y los contratos inteligentes son públicos, lo que significa que todas las transacciones entre y desde una dirección pueden ser vistas por todos los usuarios de la red. No es posible ocultar estas transacciones ni las direcciones que las realizan, por lo que un usuario de Ethereum no puede ser realmente anónimo. Si se encontrara una forma de vincular la dirección a una identidad real, ya sea en la actualidad o en el futuro, se conocería la identidad de la transacción.

El 15 de septiembre de 2022, Ethereum cambió su sistema de consenso de prueba de trabajo a prueba de participación, proceso conocido como “Merge”. Este cambio no solo disminuyó el consumo energético en un 99%, sino que

también sentó las bases para Ethereum más seguro y escalable. A pesar de las ventajas de la Merge, se produjo un declive en la privacidad, lo que genera preocupación entre los usuarios de Ethereum.

Con la implementación del PoS, los validadores tienen que mantener una cantidad mínima de Ether en sus cuentas para participar en la creación de bloques y la validación de transacciones. Esto significa que los validadores tienen un incentivo económico para mantener su identidad y reputación en la red, lo que puede llevar a una mayor centralización y disminución del anonimato [22,23,24]. Además, dado que los validadores tienen más responsabilidad y poder en la red, existe un riesgo potencial de que se conviertan en objetivos de ataques de correlación, donde un atacante puede vincular transacciones y direcciones a individuos específicos.

En general, la privacidad no se prioriza tanto como otros aspectos clave de la blockchain, como la descentralización y la escalabilidad. La búsqueda de transparencia en las redes de blockchain a menudo compromete la privacidad de las personas y las empresas.

Muchos profesionales han considerado desde hace tiempo las pruebas de conocimiento nulo como un método adecuado para mejorar la privacidad. Las pruebas de conocimiento nulo son métodos para demostrar el conocimiento de algún dato revelando una cantidad muy pequeña de información al respecto. Los zk-SNARKs y zk-STARKs son dos de los métodos más populares de aplicar pruebas de conocimiento nulo en criptomonedas y sistemas distribuidos, pero no son las únicas. Existen otras técnicas y protocolos basados en pruebas de conocimiento nulo, como zk-Boo [38], zk-SHARK [39], Bulletproofs [40] y Sonic [41], que también se utilizan en diferentes contextos para garantizar la privacidad y la integridad de los datos.

Sonic es un protocolo zk-SNARK que destaca por su conjunto de confianza que admite una cadena de referencia estructurada universal y continuamente actualizable que se escala linealmente en tamaño.

Zk-Boo es un esquema de prueba de conocimiento nulo eficiente que utiliza evaluaciones de circuitos booleanos para reducir el tamaño y el tiempo de las pruebas. Zk-SHARK es un protocolo que combina las ventajas de los zk-SNARK y los zk-STARK, ofreciendo una solución escalable para lograr la verificación rápida, pruebas cortas y configuración no confiable.

Por último, Bulletproofs es un esquema de prueba de rango no interactivo que no requiere un conjunto de confianza y es especialmente útil para reducir el tamaño de las transacciones.

Durante la conferencia técnica ETH Seúl 2022, los desarrolladores de Ethereum se unieron para tratar cómo mejorar la privacidad y la escalabilidad en las aplicaciones descentralizadas. Vitalik Buterin, uno de los cofundadores de Ethereum, abrió el evento ETH Seúl destacando el papel de las pruebas de conocimiento nulo (ZK) en el mejoramiento de la privacidad en Ethereum. Según Buterin [15]: "Con las pruebas ZK, es posible comprobar que uno es humano sin

revelar dicha información. Además, se pueden implementar sistemas de reputación que permiten demostrar si se ha realizado o no alguna acción".

En el caso de zk-SNARKs, recientes investigaciones han contribuido a mejorar la eficiencia y la escalabilidad de las pruebas de conocimiento nulo. Por un lado, Plonk [11] es un sistema de pruebas zk-SNARK que permite una configuración de confianza universal y actualizable, por el otro, se tiene Halo 2 [14], una implementación de zk-SNARK de alto rendimiento que elimina la necesidad de una configuración de confianza y sienta las bases para la escalabilidad en Zcash. Aplicaciones como Zether y Tornado Cash demuestran el potencial de zk-SNARKs para garantizar la privacidad en las transacciones de Ethereum.

Zk-STARK es un protocolo desarrollado por investigadores como Eli Ben-Sasson, Michael Riabzev y Madars Virza. A diferencia de zk-SNARKs, zk-STARKs no requieren una "configuración confiable" y son resistentes a ataques cuánticos. Aunque menos maduras en términos de adopción práctica, están ganando terreno gracias a su resistencia cuántica y la falta de configuración confiable. Investigaciones como las pruebas basadas en FIR (Fast Reed-Solomon Interactive Oracle Proofs), un protocolo para la verificación eficiente de ecuaciones polinomiales a través de interacciones con oráculos, y aplicaciones como StarkWare y DiversiFi, son ejemplos del creciente interés en zk-STARKs.

Si los desarrolladores pueden implementar con éxito este método, zk-STARKs podría convertirse en una solución para garantizar la confidencialidad, ofreciendo uno de los mejores niveles de anonimato disponibles.

En resumen, tanto zk-SNARKs como zk-STARKs están dando pasos importantes para abordar los desafíos de privacidad en Ethereum. Es probable que estas tecnologías continúen evolucionando y encontrando nuevas aplicaciones en el ecosistema blockchain a medida que la investigación y el desarrollo avancen en los próximos años.

1.3. Objetivos del trabajo

- Analizar el funcionamiento y la seguridad de los protocolos zk-SNARK y zk-STARK para la preservación de la privacidad en Ethereum.
- Investigar y comparar las ventajas y desventajas de ambos protocolos en términos de privacidad y seguridad.
- Evaluar el impacto de los protocolos zk-SNARK y zk-STARK en la escalabilidad de la red de Ethereum.
- Identificar y analizar los posibles riesgos de privacidad en la utilización de Ethereum y cómo los protocolos zk-SNARK y zk-STARK pueden mitigarlos.
- Determinar la eficacia de los protocolos zk-SNARK y zk-STARK en la preservación de la privacidad en transacciones financieras y de otro tipo en Ethereum.
- Analizar cómo los protocolos zk-SNARK y zk-STARK pueden mejorar la privacidad en aplicaciones descentralizadas (dApps) y contratos inteligentes en Ethereum.

- Investigar las realizaciones prácticas de la implementación de los protocolos zk-SNARK y zk-STARK en aplicaciones y contratos inteligentes de Ethereum.
- Analizar los protocolos zk-SNARK y zk-STARK con otros métodos y tecnologías utilizadas en la preservación de la privacidad en Ethereum.

1.4. Impacto en sostenibilidad, ético-social y de diversidad

Privacidad y derechos humanos
Entre las razones a realizar este trabajo, es importante destacar la preocupación por la protección de la privacidad y la libertad individual en un contexto sociopolítico en el que estos valores están siendo erosionados por el control tecnológico y la vigilancia constante. Al explorar cómo las tecnologías de prueba de conocimiento nulo pueden mejorar la privacidad y la seguridad en Ethereum, este trabajo puede contribuir a la protección de los derechos humanos y a reducir las desigualdades en el entorno digital.

Consumo de energía
La implementación de zk-SNARKs y zk-STARKs puede reducir el consumo de energía en Ethereum al disminuir la cantidad de datos necesarios para validar transacciones y contratos inteligentes. Al optimizar la eficiencia y la escalabilidad de la plataforma, estos protocolos pueden contribuir a una mayor sostenibilidad ambiental.

Inclusión financiera
La mejora de la privacidad en Ethereum mediante zk-SNARKs y zk-STARKs puede promover la inclusión financiera al permitir a más personas acceder a servicios financieros de forma segura y privada. Esto es especialmente relevante en regiones donde el acceso a servicios financieros tradicionales es limitado o inexistente.

Innovación y diversidad de aplicaciones
La implementación de zk-SNARKs y zk-STARKs en Ethereum puede fomentar la diversidad de aplicaciones y soluciones desarrolladas en la plataforma al expandir las posibilidades de uso en áreas donde la privacidad es esencial. Esto puede conducir a una mayor innovación y a la creación de soluciones más inclusivas y equitativas.

1.5. Metodología

El enfoque metodológico de este trabajo se basa en la exploración y análisis de la privacidad en Ethereum, tomando en consideración la relevancia y la aplicabilidad en el contexto actual de las transacciones en línea. Inicialmente, el método adopta una revisión exhaustiva de literaturas relevantes extraídas de bases de datos académicas y blogs especializados. Este primer paso busca establecer un marco teórico robusto y obtener una comprensión profunda del estado actual de la privacidad en Ethereum, así como de los protocolos y proyectos más destacados en este ámbito.

Una vez establecido este fundamento, la metodología se dirige hacia la identificación y análisis de protocolos y proyectos que se centran específicamente en la privacidad en Ethereum. Este estudio se enfoca particularmente en aquellos protocolos y proyectos que emplean zk-SNARK y zk-STARK. El objetivo principal de esta fase es entender las estrategias actuales que se están implementando para abordar los problemas de privacidad en Ethereum y las soluciones que se están proponiendo.

En lugar de enfocarse en experimentos prácticos, este trabajo concentra sus esfuerzos en un análisis meticuloso, reflexionando sobre las implicaciones de los hallazgos, la mejora de las soluciones existentes y las posibles direcciones para la investigación futura. En suma, este enfoque metodológico proporciona un panorama comprensivo del estado actual de la privacidad en Ethereum y sirve como un punto de partida sólido para futuros trabajos e investigaciones en este campo.

1.6. Planificación del trabajo

El desarrollo del trabajo se estructuró en diversas fases para garantizar un enfoque sistemático y eficiente. Cada etapa del proyecto fue diseñada para avanzar de manera incremental en la complejidad y profundidad del tema en estudio.

El trabajo inició con una fase de investigación y planificación, que permitió definir con claridad las tareas a abordar, la metodología a seguir, y los objetivos a alcanzar. El proceso se vio complementado por un estudio detallado del estado del arte y la redacción continua de la memoria.

A continuación, se llevó a cabo una investigación sobre el estado de privacidad en Ethereum. Durante este periodo, se estudiaron en detalle las características de Ethereum, su actualización 2.0, los problemas de privacidad inherentes y las posibles soluciones de privacidad existentes.

El siguiente paso consistió en un estudio más detallado de las pruebas de conocimiento nulo, zk-SNARKs y zk-STARKs. Esta fase requirió una inmersión en estas tecnologías, e incluyó una comparación entre ambas. Cabe señalar que la comprensión de estas tecnologías presentó ciertos desafíos debido a la naturaleza técnica de la información y la falta de recursos fácilmente disponibles.

La fase final de la investigación se centró en el estudio de las herramientas disponibles para zk-SNARKs y zk-STARKs. Durante esta etapa, se llevó a cabo una recopilación de información sobre las herramientas existentes y un análisis de diversos proyectos basados en estas tecnologías. La redacción y refinamiento continuos de la memoria final fueron aspectos cruciales para consolidar los hallazgos y logros de la investigación.

El trabajo culmina con la preparación de una presentación en vídeo, que resume y explica los resultados de la investigación. En conjunto, el proceso de preparación del trabajo, guiado por la estructura de entrega de la UOC, permitió

una evolución constante del entendimiento y conocimientos, favoreciendo un enfoque disciplinado.

La planificación temporal detallada de este trabajo puede consultarse en la Tabla 1, la cual desglosa cada fase del trabajo en tareas específicas y los periodos de tiempo asignados para cada una de ellas.

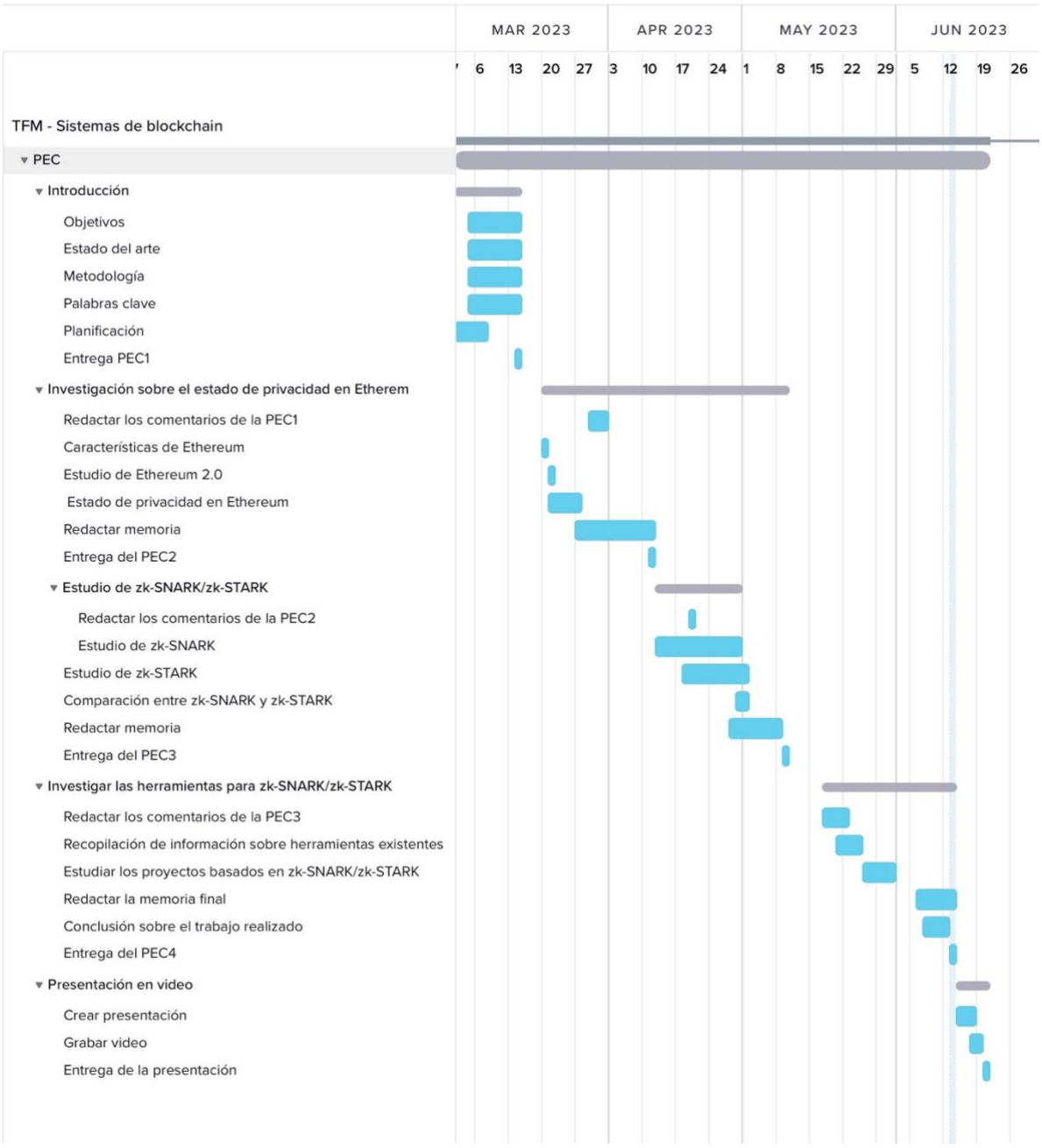

Tabla 1. La planificación temporal detallada

2. Privacidad en Ethereum

2.1. Descripción de blockchain Ethereum

Ethereum es una plataforma blockchain para aplicaciones descentralizadas y es la segunda criptomoneda por capitalización de mercado (ETH), a fecha de mayo 2023. La mayoría de los proyectos populares en el ámbito de DeFi (finanzas descentralizadas) y NFT (tokens no fungibles) funcionan en la red Ethereum. Ethereum es conocido por su capacidad para ejecutar contratos inteligentes, que son programas informáticos que permiten realizar transacciones sin la intervención de un tercero.

2.1.1. Características de Ethereum y sus aplicaciones

Ethereum es una blockhain poderosa y versátil, con una serie de características distintivas que la han convertido en la elección principal para muchas aplicaciones descentralizadas. A continuación, exploraremos algunas de estas características clave de Ethereum, incluyendo los contratos inteligentes, los diferentes tipos de tokens y las diversas aplicaciones que se han desarrollado en su red.

- Contratos inteligentes: Los contratos inteligentes son programas que se ejecutan automáticamente en la red Ethereum cuando se cumplen ciertas condiciones preestablecidas. Estos contratos permiten a los usuarios crear aplicaciones descentralizadas y automatizar procesos sin la necesidad de intermediarios.

- Tokens: Ethereum permite a los usuarios crear e intercambiar tokens personalizados mediante el uso de estándares como ERC-20 [42] y ERC-721 [43]. Los tokens pueden representar cualquier tipo de activo o valor, como monedas, puntos de lealtad o incluso objetos coleccionables digitales. Hay diferentes categorías de tokens. Según un estudio [27] existen cuatro categorías que incluyen tokens de pago, tokens no fungibles (NFT), tokens de utilidad y tokens de seguridad.

    - Tokens de pago: Incluyen criptomonedas y stablecoins (criptomonedas vinculadas al valor de una moneda normal). Aunque no son emitidos por gobiernos o bancos, estos tokens existen en redes descentralizadas y suelen utilizar el estándar ERC-20. La regulación a nivel mundial es mínima, pero está en desarrollo en varios países.

    - NFT (tokens no fungibles): Los NFT son tokens únicos e irreemplazables que contienen información única, como imágenes, archivos multimedia o datos del juego. Ethereum ganó popularidad como blockchain para NFT gracias al estándar ERC-721. Estos tokens pueden ser intercambiados y vendidos en mercados especializados y se han vuelto populares en áreas como el arte, los juegos y los coleccionables digitales. En 2022, los tokens

Soulbound (SBT) también comenzaron a ganar popularidad. Los SBTs son básicamente fichas de identidad digital, vinculados a un usuario específico y representan las características, rasgos y logros de una persona o entidad [25,26].

- Tokens de utilidad: Son emitidos por las plataformas en las que se van a utilizar, como monedas nativas en metaversos u otras aplicaciones descentralizadas. Suelen utilizarse para canjear productos y servicios en lugar de ser considerados activos apreciables. Los tokens utilizables son utilizados principalmente por las empresas para aumentar el interés en sus productos y para aplicaciones del ecosistema blockchain. Estos tokens generalmente se basan en el estándar ERC-20 o variantes de este.

- Tokens de seguridad: Son similares a los NFT emitidos por empresas de artículos deportivos para combinarlos con productos físicos, pero están reservados para combinarse con valores financieros, como acciones, bonos o derivados. También llamado "token de inversión" o "token de capital", en el ámbito de la tecnología blockchain, un token de seguridad es un token criptográfico vinculado a una oferta de valores.

En lugar de proporcionar un beneficio tangible al inversor, como el acceso al ecosistema, el token de seguridad representa una participación en la empresa que emitió el token. Los inversores que compran estos tokens esperan obtener un rendimiento de su inversión.

Para garantizar el cumplimiento de las regulaciones financieras, los tokens de seguridad utilizan diferentes estándares en función de sus propiedades y requisitos legales. Entre ellos, se destaca el estándar ERC-1400 [58], creado específicamente para tokens de seguridad con el objetivo de tener una estructura unificada para cualquier token de este tipo y mejorar el cumplimiento de la seguridad de los tokens en la red de Ethereum.

Sin embargo, el ERC-1400 ha evolucionado hacia el estándar ERC-3643 [45] (anteriormente conocido como protocolo T-REX), el primer y único estándar de token con permiso reconocido oficialmente por la comunidad Ethereum. Este estándar proporciona un mayor control al emisor y los activos pueden ser recuperados incluso si se pierde la clave privada de la wallet, debido a que la propiedad de los activos está garantizada por la identidad digital.

- Aplicaciones de Ethereum:

  - DeFi (finanzas descentralizadas): son instrumentos financieros en forma de servicios y aplicaciones basados en blockchain, es un ecosistema financiero descentralizado que ofrece servicios como préstamos, intercambios y gestión de activos sin la necesidad de intermediarios financieros tradicionales. Las aplicaciones DeFi se

basan en contratos inteligentes y tokens para facilitar la creación de productos financieros descentralizados, como plataformas de préstamos (Aave, Compound), intercambios descentralizados (Uniswap, SushiSwap) y gestión de activos (Yearn Finance, Balancer).

- Metaversos y juegos: Ethereum se ha utilizado para desarrollar metaversos y juegos basados en blockchain, donde los usuarios pueden poseer y comerciar activos digitales como tierras virtuales, objetos de juego y avatares. Ejemplos de metaversos y juegos en Ethereum incluyen Decentraland, The Sandbox y Axie Infinity.

- Identidad digital y verificable: Ethereum también se utiliza en aplicaciones de identidad digital y verificación de credenciales, como Veramo [46], donde los usuarios pueden almacenar y compartir información de identidad de manera segura y verificable en la blockchain.

- Gobernanza descentralizada y DAOs: Ethereum ha sido el hogar de muchas organizaciones autónomas descentralizadas (DAOs), que permiten la toma de decisiones colectiva y la gobernanza de proyectos o comunidades mediante la utilización de contratos inteligentes y tokens de gobernanza. Ejemplos de DAOs incluyen Aragon y Gnosis.

Estas aplicaciones y muchas más demuestran la versatilidad y el potencial de Ethereum como plataforma de desarrollo para una amplia variedad de casos de uso y soluciones descentralizadas.

2.1.2. Ethereum 2.0

Ethereum 2.0 es una actualización de la plataforma Ethereum que busca mejorar significativamente su escalabilidad, eficiencia energética y capacidad de procesamiento de transacciones. Esta actualización se centra en dos enfoques principales: las soluciones de segundo nivel (Layer 2) y las mejoras de Ethereum 2.0 en sí, que incluyen la transición al algoritmo de consenso Proof-of-Stake (PoS), la implementación de la tecnología de sharding, la introducción de nueva máquina virtual llamada eWASM y la Beacon Chain.

- Soluciones de segundo nivel (Layer 2). Estas soluciones buscan aumentar la capacidad de Ethereum y reducir las comisiones por transacción. Las soluciones de segundo nivel se implementan mediante tecnologías como Rollups, que permiten incluir cientos de transacciones de segundo nivel en una sola transacción de primer nivel. Ejemplos de aplicaciones L2 incluyen Arbitrum, Optimism, dYdX y StarkNet.

- Proof-of-Stake (PoS). Ethereum ha cambiado su algoritmo de consenso de Proof-of-Work (PoW) a Proof-of-Stake (PoS) para reducir el consumo de energía y aumentar la seguridad de la red. PoS permite a los usuarios participar en la validación de bloques y la generación de recompensas en

función de la cantidad de ETH que poseen y ponen en garantía ("stake") en la red.

- Sharding. El sharding es una tecnología que divide la cadena de bloques de Ethereum en segmentos manejables (shards) y permite la ejecución paralela de operaciones en cada uno de ellos. Esto mejora la escalabilidad de Ethereum al aumentar la cantidad de transacciones que se pueden procesar simultáneamente en la red.

- Beacon Chain. La Beacon Chain es una cadena de bloques separada que se ejecuta en paralelo con Ethereum y coordina el proceso de consenso PoS y el sharding. La Beacon Chain es responsable de la selección de validadores, la distribución de recompensas y la comunicación entre los shards.

2.2. Problemas de privacidad en Ethereum

La transparencia es la esencia de Ethereum, permite la verificación y trazabilidad de las transacciones en blockchain. Sin embargo, esta naturaleza transparente también conlleva problemas de privacidad, ya que las direcciones y las transacciones pueden ser rastreadas y vinculadas a las identidades de los usuarios [16].

La blockchain de Ethereum es pública, y cualquier persona puede ver las transacciones y las direcciones asociadas. A pesar de que esta característica es útil en términos de verificación y transparencia, también plantea preocupaciones sobre la privacidad de los usuarios. Cuando se trata de privacidad en las transacciones de criptomonedas, hay tres tipos de información privada que pueden filtrarse: el remitente, el receptor y la cantidad transferida. La privacidad perfecta se logra cuando se ocultan con éxito estos tres aspectos a observadores externos [17].

La privacidad en las transacciones de criptomonedas no es un concepto binario [18], sino que se extiende a lo largo de un espectro desde totalmente público hasta completamente privado. Por lo tanto, es importante considerar qué aspectos de la privacidad son más importantes para los usuarios, si están dispuestos a pagar por ella en términos de costo y esfuerzo, y cuáles son las compensaciones para lograr transacciones privadas.

Matt Shapiro y Ryan Gentry [17] sostienen que la privacidad debería ser una característica inherente en todos los sistemas, incluido Ethereum. Aunque Ethereum ha avanzado en la implementación de soluciones de privacidad, como protocolos de capa de aplicación y tokens de privacidad, la falta de privacidad por diseño en Ethereum es un problema, ya que las soluciones actuales no están integradas de forma nativa en el protocolo principal. Esto puede limitar su adopción y efectividad en la protección de la privacidad del usuario. Ferenc Béres y otros investigadores han demostrado [8] cómo se pueden desanonimizar usuarios de Ethereum analizando patrones de comportamiento y vinculando direcciones Ethereum a identidades específicas.

Además, la vinculación de direcciones y metadatos puede permitir a los atacantes crear perfiles de usuarios y revelar información privada, como la ubicación geográfica, el comportamiento de gasto y las conexiones con otras direcciones.

La resistencia a la censura no puede lograrse sin privacidad, lo que hace que la privacidad sea un componente clave de las finanzas abiertas, el dinero global libre de estado y la Web3.

La dependencia de servicios centralizados, como los intercambios de criptomonedas, también puede poner en riesgo la privacidad de los usuarios de Ethereum. Estos servicios pueden almacenar información personal y financiera y, en algunos casos, estar sujetos a regulaciones gubernamentales que pueden requerir la divulgación de datos del usuario.

Las demandas de privacidad entre empresas y consumidores varían significativamente. Por un lado, las empresas necesitan proteger información sensible en las transacciones, como nombres de productos, cantidades, precios, direcciones y datos de identificación financiera personal. Por otro lado, los consumidores suelen mostrar menos preocupación por la privacidad, a menudo sacrificándola por comodidad o acceso gratuito.

En el ámbito de las criptomonedas, la falta de adopción de tecnologías de privacidad, como el almacenamiento de ZEC utilizando SNARKs en Zcash, indica que la mayoría de los usuarios pueden no estar dispuestos a pagar por la privacidad. A pesar de que Zcash existe desde hace casi 7 años, solo el 6% de ZEC se almacena utilizando SNARKs, y aproximadamente el 93% se encuentra en direcciones transparentes con poca privacidad. [19]

La falta de privacidad en Ethereum puede, por lo tanto, socavar la adopción generalizada de la plataforma y erosionar la confianza en la tecnología de las criptomonedas.

2.3. Soluciones de privacidad en Ethereum

En blockchain, la privacidad es particularmente difícil de lograr debido a su diseño de que todas las transacciones son transparentes y el suministro de monedas es verificable. Las soluciones de privacidad deben garantizar que estos mecanismos se conserven mientras protegen la privacidad.

En este capítulo, exploramos las soluciones de privacidad disponibles en Ethereum.

2.3.1. Uso de protocolos zk-SNARK y zk-STARK

Los protocolos zk-SNARK y zk-STARK son técnicas criptográficas avanzadas que permiten mejorar la privacidad y la escalabilidad en Ethereum. Ben-Sasson y otros autores en 2013 [21] presentan zk-SNARK, un protocolo que utiliza pruebas de conocimiento nulo para permitir que los usuarios demuestren la validez de las transacciones sin revelar información sensible. La aplicación de

zk-SNARK en Ethereum permitiría mejorar la privacidad y reducir el tamaño de las transacciones, lo que a su vez mejoraría la escalabilidad de la plataforma.

La primera implementación significativa de zk-STARK fue introducida por Eli Ben-Sasson, Iddo Bentov, Yinon Horesh y Michael Riabzev en un artículo titulado "Scalable, transparent, and post-quantum secure computational integrity", publicado en 2018 [22]. Zk-STARK es una evolución del protocolo zk-SNARK que aborda algunas de las limitaciones de este último, como la dependencia de un conjunto de parámetros de confianza y la necesidad de operaciones criptográficas de alto costo. Zk-STARK utiliza pruebas de conocimiento nulo sin interacción en lugar de las pruebas interactivas utilizadas en zk-SNARK, lo que permite una mayor escalabilidad y seguridad. Al implementar zk-STARK en Ethereum, se pueden mejorar aún más la privacidad y la escalabilidad, permitiendo un mayor número de transacciones y contratos inteligentes en la plataforma.

2.3.2. Anonimato

Estas soluciones se centran en proteger la identidad de los usuarios y ocultar las relaciones entre las direcciones de las carteras y las transacciones realizadas en la cadena de bloques. El anonimato en Ethereum se puede lograr a través de varios enfoques, como el uso de direcciones diferentes para cada transacción, mezcladores de monedas y protocolos de privacidad en la capa de aplicación. Estos enfoques buscan principalmente garantizar que las transacciones no puedan vincularse a usuarios o entidades específicas.

2.3.2.1. Direcciones sigilosas (Stealth Addresses)

Las direcciones sigilosas son una solución que permite a los usuarios recibir criptomonedas en direcciones únicas y no relacionadas entre sí, lo que dificulta el rastreo de las transacciones y la identificación de los usuarios. Al utilizar direcciones sigilosas, los usuarios pueden mantener un mayor anonimato al evitar que sus transacciones se vinculen a una dirección específica de la cartera. En Ethereum, se utiliza ECDSA (Elliptic Curve Digital Signature Algorithm) en la generación de estas direcciones sigilosas.

2.3.2.2. Mezcladores de monedes (Coin Mixers)

Los mezcladores de monedas son servicios que mezclan transacciones de diferentes usuarios, lo que hace que sea difícil rastrear el flujo de fondos entre las direcciones de las carteras. Estos servicios pueden utilizarse para mejorar el anonimato en Ethereum al agregar una capa adicional de protección para las transacciones y dificultar la identificación de los usuarios.

2.3.2.3. Protocolos de capa de aplicación y transacciones confidenciales

Los protocolos de capa de aplicación, como Tornado Cash, pueden implementarse en Ethereum para mejorar el anonimato de las transacciones. Estos protocolos utilizan técnicas criptográficas avanzadas, como las pruebas de conocimiento nulo, para permitir que los usuarios realicen transacciones

privadas sin revelar información sobre sus direcciones de cartera o el monto de las transacciones.

Por ejemplo, Tornado Cash es un protocolo de mezclado de monedas que utiliza zk-SNARK para permitir a los usuarios depositar y retirar criptomonedas de un grupo común sin revelar sus identidades. Dado que las retiradas se realizan desde las reservas de liquidez de los contratos inteligentes del proyecto, es imposible saber quién es el remitente original. Esto oscurece el flujo de fondos y dificulta su rastreo. La vinculación anónima de las direcciones del remitente y el destinatario es posible gracias a la tecnología Zero-knowledge proof.

Las transacciones confidenciales son otro enfoque para mejorar el anonimato en Ethereum. Estas transacciones ocultan el monto involucrado en una transacción, lo que dificulta el análisis de la cadena de bloques y la identificación de los usuarios.

Aztec Protocol es otro protocolo de capa de aplicación L2 que utiliza pruebas de conocimiento nulo para garantizar la confidencialidad en las transacciones de Ethereum. Una transacción confidencial es una transferencia de valor entre dos o más entidades, en la que los valores que se transfieren no son visibles para los observadores.

El protocolo funciona mediante el uso de notas AZTEC, que representan valores ocultos. Para realizar una transacción confidencial, los usuarios combinan y dividen las notas, creando nuevas notas con diferentes valores sin revelar los montos involucrados. El protocolo AZTEC también admite direcciones sigilosas, lo que permite una mayor privacidad en la propiedad y transferencia de notas.

2.3.2.4. Metatransacciones

Las metatransacciones son transacciones que permiten a los usuarios interactuar con contratos inteligentes sin tener que pagar directamente por el gas, lo que mejora la privacidad. La red de retransmisión de gas abierto (OpenGSN) [29,30] y las soluciones de OpenZeppelin permiten a los desarrolladores implementar metatransacciones en sus dApps. Al usar metatransacciones, los usuarios pueden realizar transacciones sin revelar su dirección de Ethereum, lo que mejora el anonimato en la plataforma.

2.3.3. Criptografía utilizada en Ethereum

El cifrado es una técnica criptográfica que se utiliza para proteger la información mediante la conversión de datos legibles en texto cifrado que solo puede ser descifrado por aquellos que poseen la clave de descifrado correspondiente.

Wüst y Gervais [20] argumentan que el cifrado es esencial para garantizar la integridad y la confidencialidad de las transacciones en línea. En el contexto de las criptomonedas y Ethereum, las soluciones de cifrado pueden aplicarse a contratos inteligentes y transacciones para proteger los datos y mantener la privacidad de los usuarios. Estas soluciones pueden incluir el uso de algoritmos de cifrado simétrico y asimétrico para garantizar la confidencialidad de los datos.

2.3.3.1. Cifrado asimétrico

El cifrado asimétrico, también conocido como criptografía de clave pública, utiliza un par de claves, una pública y una privada. La clave pública se utiliza para cifrar la información, mientras que la clave privada se utiliza para descifrarla. Los algoritmos de cifrado asimétrico, como RSA, Elliptic Curve Cryptography (ECC) y Lattice-based Cryptography, ofrecen mayor seguridad en comparación con los algoritmos de cifrado simétrico, pero son más lentos en términos de rendimiento. En Ethereum, el cifrado asimétrico se utiliza para proteger las transacciones y garantizar que solo los usuarios con la clave privada correspondiente puedan acceder a los fondos o ejecutar acciones específicas en los contratos inteligentes.

2.3.3.2. Funsiones hash

Las funciones hash desempeñan un papel crucial en la criptografía y la seguridad de Ethereum. Una función hash es una función matemática que toma una entrada y produce un valor fijo y único de tamaño fijo, conocido como hash. En Ethereum, las funciones hash se utilizan en varios aspectos, como la creación de direcciones a partir de claves públicas, la construcción del árbol Merkle y la minería.

Ethereum utiliza principalmente la función hash Keccak-256 [47]. Keccak-256 es un algoritmo de hashing seguro y unidireccional que proviene de la familia SHA-3. Convierte una entrada en una salida hash de longitud fija de 256 bits, sin posibilidad de revertir el proceso. El hashing consiste en codificar datos en un hash utilizando un algoritmo específico. Los algoritmos de hashing seguros ocultan la lógica interna, lo que garantiza la protección de los datos.

Ethereum utiliza Keccak-256 en múltiples aspectos, como firmas digitales, validación de transacciones y protección de datos almacenados en la blockchain. Al convertir los datos en un hash, se garantiza que la información se almacene de manera segura y privada. Solo aquellos que poseen las claves correspondientes pueden acceder y leer los datos reales.

2.4. Comparación de soluciones de privacidad

A continuación, se presenta una tabla comparativa que resume las principales soluciones de privacidad, sus descripciones, ventajas y desventajas.

Solución de privacidad	Descripción	Ventajas	Desventajas
Direcciones sigilosas	Permite a los usuarios recibir criptomonedas en direcciones únicas y no relacionadas entre sí.	Mejora el anonimato al dificultar el rastreo de transacciones y la identificación de usuarios.	No oculta el monto de las transacciones.

Mezcladores de monedas	Mezcla transacciones de diferentes usuarios para dificultar el rastreo del flujo de fondos	Mejora el anonimato y dificulta la identificación de usuarios.	Puede tener costos adicionales y no garantiza la privacidad absoluta.
Protocolos de capa de aplicación (como Tornado Cash y Aztec Protocol)	Utiliza técnicas criptográficas avanzadas para permitir transacciones privadas sin revelar información sensible	Mejora el anonimato y la privacidad, y puede reducir el tamaño de las transacciones.	Puede ser complejo de implementar y tener costos adicionales.
Metatransacciones	Permite a los usuarios interactuar con contratos inteligentes sin pagar directamente por el gas.	Mejora la privacidad al no revelar la dirección de Ethereum del usuario.	No garantiza la privacidad absoluta y puede requerir una infraestructura adicional.
Zk-SNARK	Protocolo de pruebas de conocimiento nulo que permite demostrar la validez de las transacciones sin revelar información sensible.	Mejora la privacidad y reduce el tamaño de las transacciones.	Requiere un conjunto de parámetros de confianza y operaciones criptográficas de alto costo.
Zk-STARK	Evolución del protocolo zk-SNARK que utiliza pruebas de conocimiento nulo sin interacción.	Mayor escalabilidad y seguridad en comparación con zk-SNARK.	Más complejo, alta demanda de recursos computacionales y falta de estandarización.
Cifrado asimétrico	Criptografía de clave pública que utiliza un par de claves para cifrar y descifrar información.	Mayor seguridad en comparación con el cifrado simétrico.	Más lento en términos de rendimiento.
Funciones hash	Funciones matemáticas que toman una	Garantiza la protección de los datos y la	No proporciona privacidad ni

	entrada y producen un valor fijo y único de tamaño fijo, conocido como hash.	integridad de la información.	anonimato por sí mismas.

Tabla 2. Comparación de soluciones de privacidad

3. Casos de uso de la privacidad en Ethereum

Este capítulo explora los distintos casos de uso de la privacidad en la blockchain de Ethereum. Inicialmente, se examina el papel de las criptomonedas anónimas, explorando las tácticas que emplean y el funcionamiento de las más destacadas. Seguidamente, el foco se dirige hacia la privacidad en el ecosistema DeFi, donde se discute el uso de contratos inteligentes confidenciales y la tecnología de mezcla, con un particular énfasis en el funcionamiento del algoritmo de Tornado Cash. Por último, el capítulo se sumerge en el mundo de los ZK-Rollups. De esta forma, se ofrece una visión holística sobre cómo la privacidad se aplica en diversas situaciones y usos en la blockhain de Ethereum.

3.1. Criptomonedas anónimas

Las criptomonedas anónimas son una clase de criptomoneda que impone transacciones privadas y anónimas en la blockchain ocultando su origen y destino. Algunas de las técnicas utilizadas incluyen ocultar el saldo real del usuario y la dirección del monedero y mezclar múltiples transacciones entre sí para evitar el análisis de la cadena.

La transparencia está en el corazón del espíritu de Ethereum y otras blockchains no confidenciales. Todo el mundo puede ver las direcciones públicas y las transacciones en su red, lo que hace relativamente fácil rastrear los depósitos y retiradas de alguien. Sin embargo, las monedas anónimas abordan dos problemas muy diferentes: el anonimato y la imposibilidad de rastreo. El anonimato oculta la identidad detrás de la transacción, mientras que la imposibilidad de rastreo hace prácticamente imposible encontrar los extremos a través de los servicios de análisis de blockchain.

3.1.1. Estrategias utilizadas por las monedas anónimas

Para preservar eficazmente el anonimato y la imposibilidad de rastreo, las monedas anónimas utilizan muchas estrategias diferentes, las más populares de las cuales son: direcciones ocultas, firmas de anillo, CoinJoin y zk-SNARK.

- Las direcciones ocultas requieren que el remitente genere una nueva dirección para cada transacción enviada, para evitar estar vinculado al destinatario. Monero (XMR), una de las mejores monedas anónimas, utiliza una versión de la dirección oculta denominada protocolo de dirección oculta de dos claves (DKSAP) [48].

- Firma de anillo. Un esquema de firma de anillo permite a un miembro del grupo firmar una transacción en nombre del grupo sin revelar quién es el remitente del pago. El grupo se forma aleatoriamente. No hay ningún gestor que pueda revelar la identidad del verdadero firmante. La firma única garantiza el anonimato del remitente.

- CoinJoin. Una estrategia de anonimización que requiere que varias partes firmen conjuntamente un contrato inteligente digital para mezclar

monedas en una nueva transacción. Como resultado de la transacción, los participantes reciben el mismo número de monedas, pero las direcciones se mezclan para dificultar el seguimiento externo. Realizar CoinJoin manualmente es muy difícil, pero este servicio lo ofrecen algunos monederos anónimos como Wasabi y Samourai.

- Zk-SNARK permite a los titulares de criptomonedas demostrar la validez de una transacción sin revelar información identificativa importante, como las partes implicadas y el saldo de la cuenta.

3.1.2. Criptomonedas anónimas más populares

- Monero: hasta ahora la única moneda con la que todas las transacciones son anónimas por defecto. Monero es una criptomoneda centrada en la privacidad que utiliza tres técnicas principales para mantener la confidencialidad de las transacciones: direcciones sigilosas, firmas de anillo y transacciones confidenciales de anillo (RingCT).

- Dash: los desarrolladores de la criptomoneda utilizaron el mecanismo de mezcla de tokens de PrivateSend [49], que permite enviar monedas de forma anónima. La mezcla de Dash se produce a través de masternodes aleatorios que ocultan la conexión entre el remitente y el destinatario. Un masternode es un tipo de nodo que se encarga de cifrar los datos en las transacciones de monedas.

- Zcash: La ZEC permite tanto las transferencias privadas, llamadas transacciones blindadas, como las transacciones públicas. Las pruebas de conocimiento nulo permiten verificar las transacciones sin revelar detalles del remitente, el destinatario o el importe del pago. Las características de divulgación selectiva de Zcash permiten a un participante compartir sólo ciertos detalles de las transacciones si es necesario para cumplir con cualquier requisito o auditoría.

3.2. Privacidad en el ecosistema DeFi

Los participantes interactúan con el ecosistema DeFi a través de contratos inteligentes. Debido a la transparencia de todos los datos almacenados en la blockchain, no es posible almacenar datos sensibles dentro de los contratos inteligentes. En consecuencia, los datos sobre las cantidades y los términos del contrato son visibles para todo el mundo.

La privacidad en DeFi significa que no es posible obtener datos que puedan utilizarse para identificar al propietario del monedero.

3.2.1. Contratos inteligentes confidenciales

En febrero de 2019, un equipo de científicos de la Universidad de Stanford creó un mecanismo para habilitar contratos inteligentes privados en la red Ethereum, Zether. El protocolo está implementado en la red Ethereum como un contrato

inteligente Zether Smart Contract (ZSC). El mecanismo Zether permite crear nuevos tipos de contratos inteligentes en los que los saldos de las cuentas se cifran y almacenan hasta que se cumple el depósito, y los fondos se bloquean en el contrato y permanecen allí bloqueados.

Utilizando el mecanismo Zether, los usuarios envían ETH a un contrato ZSC y a cambio reciben una cantidad equivalente en tokens ZTH, que se utilizan para transacciones ocultas. Zether también ha implementado la capacidad de ocultar información sobre los participantes de la transacción a discreción del iniciador de la transacción.

3.2.2. Mezcladores. Tornado Cash

Una forma de garantizar la privacidad es utilizar servicios de mezcla de monedas. En el proceso de mezcla, los fondos de varios usuarios se mezclan y luego llegan a su destino. Es casi imposible rastrear quién posee qué fondos una vez que se han mezclado. Una de las soluciones más populares en este sentido es un servicio para mezclar ETH - Tornado Cash, que permite mezclar transacciones usando pruebas de conocimiento nulo, en ingles Zero-Knowledge proof, y cifrar los fondos de los usuarios.

Tornado Cash es un mezclador de criptomonedas descentralizado, lo que significa que utiliza contratos inteligentes que aceptan criptomonedas y luego permiten retirarlas a otras direcciones. Tras enviar un depósito al contrato inteligente Tornado Cash, las monedas pueden retirarse a una nueva dirección Ethereum. Este proceso garantiza que los fondos retirados no puedan vincularse a la fuente de depósito, asegurando así la privacidad y el anonimato de los activos [50]. Tornado Cash utiliza el protocolo zk-SNARK.

3.2.2.1. Algoritmo de Trabajo de Tornado Cash

En Tornado Cash, los fondos de varios usuarios se mezclan en un contrato inteligente y luego se retiran a otras direcciones, dificultando el rastreo del flujo de fondos y protegiendo la identidad de los usuarios. Al depositar cantidades fijas de criptomonedas en el contrato, los usuarios pueden ocultar sus transacciones dentro de un grupo de transacciones similares, lo que dificulta aún más el rastreo de los fondos.

Por ejemplo, un usuario deposita 0,1 ETH y otras 19.999 personas hacen lo mismo. Dado que el proceso de transferencia de fondos a un contrato inteligente es de conocimiento público, cuando un usuario deposita 0,1 ETH, esos 0,1 ETH pueden ser rastreados hasta el grupo de 20.000 personas, pero no se puede identificar directamente a la persona que hizo el depósito.

Los conjuntos de anonimato en este escenario tienen un tamaño aproximado de 20.000 miembros cada uno. Cabe destacar que cada denominación tiene un conjunto distinto, por lo tanto, cualquier disminución del tráfico en una denominación específica puede llevar a una reducción de la privacidad, ya que disminuiría el tamaño del conjunto de anonimato.

Depósito
Para depositar una moneda, un usuario sigue estos pasos:

1. Generar dos números aleatorios $k,\ r\ \in\ B^{248}$ y calcular $\mathtt{C}=\ H_1\left(k\ \|\ r\right)$

Aquí, H_1es la función de hash de Pedersen [51] que asigna una secuencia de bits a un punto comprimido en una curva elíptica. Llamamos k el "anulador" (nullifier) y r la "aleatoriedad" (randomness).

2. Enviar una transacción de Ethereum con N ETH al contrato $\mathbb{C}$ con los datos $\mathtt{C}$ interpretados como un entero sin signo de 256 bits. Si el árbol no está lleno, el contrato acepta la transacción y añade $\mathtt{C}$ al árbol como una nueva hoja distinta de cero. Si el árbol está lleno, la transacción no es aceptada.

Retiro
Para retirar una moneda (κ, r) con posición l en el árbol, un usuario procede de la siguiente manera:

1. Seleccionar una dirección de destinatario A y un valor de comisión $f\ \leq\ N$

2. Seleccionar una raíz R entre las almacenadas en el contrato y calcular la apertura $\mathcal{O}\ (l)$ que termina con R. Aquí, $\mathcal{O}\ (l)$ es el valor de los nodos hermanos en el camino desde la hoja l hasta la raíz en el árbol de Merkle.

3. Calcular el hash del anulador $h\ =\ H_1(k)$. Nuevamente, H_1 es la función hash de Pedersen.

4. Calcular la prueba P llamando a $Prove$ en d_p

La declaración de conocimiento es la siguiente:

$$S\ [R,h,A,f,t]\ =\ \{SABEMOS\ k,r,l,\mathcal{O}\ TAL\ QUE\ =\ H_1(k)$$
$$Y\ \mathcal{O}\ \text{es la apertura de}\ H_2\ (k\ \|\ r)\ en\ la\ posición\ l\ a\ R\}$$

Aquí H_2 es la función hash MiMC [0]. Podemos construir un zk-SNARK para la declaración de conocimiento anterior que resultará en $(d_{p,}d_v)$, que son el constructor de prueba y el verificador de prueba.

5. Realizar el retiro de una de las siguientes maneras:

- Enviar una transacción de Ethereum al contrato $\mathbb{C}$ proporcionando R,h,A,f,t,P en los datos de la transacción.

- Enviar una solicitud al Relayer proporcionando los datos de la transacción R,h,A,f,t,P. Relayer es una entidad que toma la transacción de un

usuario y la publica en la blockchain por él. Luego, el Relayer realizará una transacción al contrato $\mathbb{C}$ con los datos suministrados.

El contrato verifica la prueba y la singularidad del hash del anulador. En caso de éxito, envía $(N - f)$ a A y f al Relayer t y añade h a la lista de hashes de anuladores.

Cuando se trata de retirar fondos, es fundamental no comprometer la identidad del usuario. Aquí es donde entra en juego zk-SNARK. La idea es permitir que el usuario demuestre que conoce el secreto y el nulificador correspondientes al hash almacenado en el contrato inteligente sin revelar estos números. Esto se logra mediante el uso de pruebas de conocimiento nulo.

Para generar una prueba zk-SNARK en Tornado Cash, se deben realizar las siguientes comprobaciones dentro de la prueba:

1. Que hash (secreto, nulificador) esté en el contrato inteligente.
2. Que hash de nulificador sea igual a hash (nulificador).

Contratos Inteligentes

$\mathbb{C}$ es el contrato inteligente que tiene las siguientes funcionalidades:

- Almacena los últimos $n = 100$ valores de raíz en el historial del array. Para el último árbol de Merkle T, también almacena los valores de los nodos en el camino desde la última hoja añadida hasta la raíz que son necesarios para calcular la siguiente raíz.

- Acepta pagos por N ETH con datos C. El valor C se añade al árbol de Merkle, se recalcula el camino desde el último valor añadido y la última raíz. La raíz anterior se añade al historial del array.

- Verifica la supuesta prueba P en función de los valores públicos enviados R, h, A, f, t. Si la verificación tiene éxito, el contrato libera $(N - f)$ ETH a la dirección A y la comisión f ETH a la dirección del Relayer t.

- Verifica que la moneda no haya sido retirada antes comprobando que el hash del anulador de la prueba no ha aparecido antes y, si es así, lo añade a la lista de hashes de anuladores.

Tornado cash permite retirar el ETH a una cuenta directamente (siempre que el aceptante tenga suficiente ETH para cubrir los gastos de gas) o también a un nuevo monedero de ETH (a través de nodos repetidores). Esto garantiza un nivel adicional de privacidad.

Según los creadores de Tornado cash, los usuarios necesitan esperar entre un depósito y la retirada para garantizar el anonimato (ya que aumenta el conjunto de anonimato). Sin embargo, esto plantea problemas potenciales cuando buscamos aplicaciones instantáneas de la privacidad, lo que es necesario para muchos escenarios de escalado de criptomonedas. [50]

3.3. ZK-Rollups

Un zk-Rollup es una solución de escalado Layer-2 (L2) para Ethereum (y otros protocolos de blockchain) que utiliza las pruebas de conocimiento nulo para mejorar la eficiencia y el rendimiento de las transacciones.

A alto nivel, los zk-Rollups toman múltiples operaciones en la cadena de bloques, como las transacciones, y las "agrupan" en una sola prueba de conocimiento nulo, que luego se publica en la cadena de bloques de Ethereum (L1). Esta prueba, aunque mucho más pequeña en términos de tamaño de datos que las transacciones originales, sirve como prueba criptográfica de que las transacciones agrupadas son válidas y no alterarán incorrectamente el estado de la cadena de bloques.

En cuanto a la privacidad, aunque la tecnología de conocimiento nulo puede teóricamente permitir transacciones privadas (al ocultar detalles sobre los participantes o los montos de las transacciones, por ejemplo), la mayoría de las implementaciones actuales de zk-Rollups no utilizan esta característica de las ZKPs. En su lugar, se centran en utilizar ZKPs para mejorar la escalabilidad y eficiencia.

Esto significa que las transacciones realizadas a través de zk-Rollups no son inherentemente privadas.

El motivo de esta falta de privacidad en muchos zk-Rollups es que agregar privacidad a través de ZKPs puede ser bastante complejo y aumentar los requisitos de computación puede contrarrestar algunos de los beneficios de escalado que ofrecen los zk-Rollups. Por lo tanto, muchos proyectos optan por concentrarse en la escalabilidad antes que en la privacidad.

4. Protocolos zk-SNARK y zk-STARK

4.1. Pruebas de conocimiento nulo

La prueba de conocimiento nulo o ZKP es un protocolo criptográfico en el que intervienen dos partes: la parte que prueba y la parte que verifica (el verificador). El propósito del protocolo es permitir que el verificador se asegure de que la parte demostradora conoce el parámetro secreto. Sin embargo, el parámetro secreto en sí no debe revelarse al verificador ni a nadie más [31].

Esto puede representarse como un programa con dos entradas $C(x, a)$. La entrada x es abierta, a es un parámetro secreto (testigo). La salida del programa es binaria (verdadero o falso). Se establece una x pública particular. La tarea consiste en demostrar que la parte que prueba conoce el parámetro secreto a, de modo que $C(x, a) = TRUE$.

Una prueba de conocimiento nulo debe, por definición, satisfacer las tres propiedades siguientes:

1. Exhaustividad: si la afirmación es verdadera y ambas partes siguen el mismo protocolo, el verificador puede convencerse de la verdad de la afirmación.
2. Persistencia: si la afirmación es falsa, no es probable que el verificador esté convencido de su verdad.
3. Conocimiento nulo: el verificador no recibe información adicional.

El concepto de sistemas de prueba interactivos con conocimiento nulo se introdujo por primera vez por los autores Goldwasser S., Micali S. y Rackof C. [32]. A lo largo de los años de investigación sobre las pruebas de conocimiento nulo, los sistemas basados en este método han ido mejorando gradualmente, centrándose en optimizar su eficacia para aplicaciones específicas. Esto ha llevado a la aparición de algoritmos que han reducido significativamente el número de rondas de interacción entre los participantes en el protocolo.

Las características de la tecnología blockchain imponen una serie de limitaciones a los protocolos criptográficos utilizados, en particular la prueba de conocimiento nulo. Como blockchain es un sistema distribuido, los usuarios pueden no estar en línea al mismo tiempo. Sin embargo, la prueba debe estar disponible para todos los participantes. Una vez proporcionada la prueba, cualquier usuario debe poder verificar su exactitud en cualquier momento. Esto hace que el uso de protocolos de prueba interactivos y de conocimiento nulo en los sistemas blockchain sean difíciles de implementar.

En el trabajo de los autores Blum M., Feldman P. y Micali S [33] se propuso por primera vez un protocolo no interactivo de prueba de conocimiento nulo. La aparición de zk-SNARK [21], que permitió el uso eficaz de protocolos no interactivos de prueba de conocimiento nulo en sistemas blockchain, puede considerarse un avance significativo en esta dirección.

4.2. zk-SNARK

zk-SNARK es un protocolo criptográfico para la prueba no interactiva de conocimiento nulo [34]. El acrónimo zk-SNARK significa:

ZK	**Z**ero-**K**nowledge
S	**S**uccinct
N	**N**on-Interactive
AR	**AR**gument
K	of **K**nowledge

- Zero-Knowledge (Conocimiento nulo): Este es un tipo de prueba en el que una parte puede probar a otra que sabe un valor o que se cumple una afirmación, sin revelar ninguna información aparte de la verdad de la afirmación en sí.
- Succinct (Succincto): Las pruebas que son cortas y rápidas de verificar.
- Non-Interactive (No Interactivo): Este término se refiere a un tipo de prueba en la que la interacción entre el demostrador (la parte que quiere probar algo) y el verificador (la parte que verifica la prueba) es mínima o inexistente.
- ARgument (Argumento): Un argumento, en este contexto, es una afirmación o declaración que se utiliza como parte de la prueba.
- of Knowledge (de Conocimiento): Indica que la prueba se centra en demostrar el conocimiento de algo, en lugar de solo demostrar la verdad de una afirmación. En una prueba de conocimiento nulo, el demostrador prueba que conoce cierta información sin revelarla.

zk-SNARK es una representación concisa que evidencia la posesión de un conocimiento específico, sin desvelar ningún detalle sobre dicho conocimiento y sin necesitar la interactividad de un verificador para confirmar su validez.

Es uno de los componentes básicos de la privacidad en la blockchain. Permite probar que algunos datos privados satisfacen un sistema de restricciones expresado como un esquema aritmético *C* sin revelar esos datos.

4.2.1. Características zk-SNARK

El protocolo zk-SNARK consta de 3 funciones: $G, P\ y\ V$.

La función G (generador de claves), toma el parámetro λ (también llamado "residuo tóxico"), programa C (la función parámetro necesaria para calcular W). A continuación, se generan dos claves: la "clave de comprobación" (proving_key) y la "clave de verificación" (verifing_key).

$$(proving\_key, verifying\_key) = G(\lambda, \mathrm{C})$$

La función P toma tres parámetros como entrada: la clave de prueba (clave_de_prueba), el valor aleatorio y público x y el conocimiento de prueba Witness (que no se revela). La salida de P crea y devuelve una prueba “proof”.

$$Proof = P(proving\_key, x, witness)$$

La función V toma tres parámetros como entrada: la clave del verificador (verifing_key), el valor disponible x y la prueba. La salida de V es $True$ o $False$, indicando si la prueba es verdadera o falsa.

$$\{True \mid False\} = V(vk, x, prf)$$

El parámetro λ utilizado en la función G es el llamado “residuo tóxico”, que debe mantenerse confidencial y, si es posible, destruirse inmediatamente junto con el portador, ya que poseerlo permitiría falsificar cualquier prueba. Las pruebas falsificadas devuelven VERDADERO independientemente de si son realmente válidas y de si la persona que las prueba conoce el valor secreto del testigo.

En términos generales, zk-SNARK consta de los componentes que se muestran esquemáticamente a continuación:

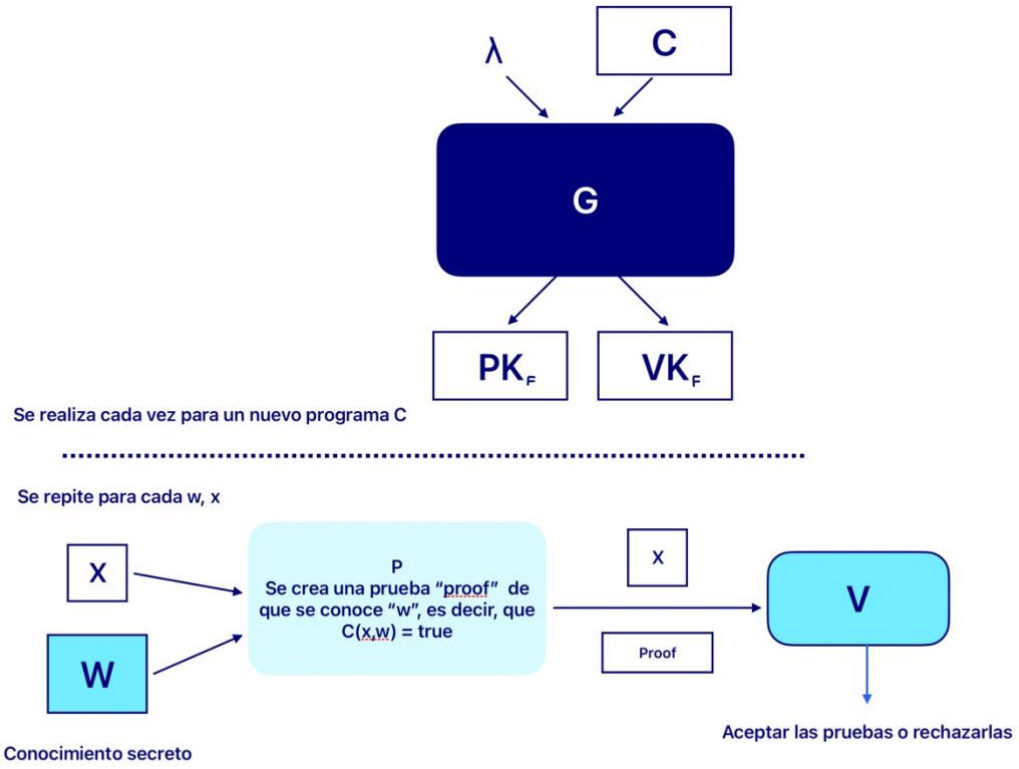

Figura 1. Componentes y funcionamiento del protocolo zk-SNARK. Fuente: elaboración propia.

En zk-SNARK, el procedimiento de comprobación de pruebas consiste en operaciones sobre curvas elípticas. En particular, el verificador requiere

multiplicación escalar y suma en un grupo de puntos de curva elíptica, así como una operación computacionalmente más compleja, el emparejamiento bilineal.

Ethereum proporciona una implementación de estas operaciones en forma de contratos precompilados. Con estos, es posible implementar esquemas basados en pruebas con conocimiento nulo en el código del contrato inteligente [35,36].

Los algoritmos de generación y verificación de pruebas zk-SNARK no están implementados en Ethereum. Esto plantea una serie de problemas a la hora de utilizar esquemas basados en zk-SNARK en Ethereum:

- No hay posibilidad de crear esquemas complejos. Todos los algoritmos tienen que implementarse en contratos inteligentes, que tienen limitaciones estrictas en cuanto al tamaño del código, y los esquemas criptográficos suelen requerir un gran número de operaciones.

- Todos los algoritmos tienen que implementarse manualmente.

- Hay que generar parámetros independientes para cada nuevo contrato.

4.3. zk-STARK

Zk-STARKs es una versión mejorada de protocolo zk-SNARK y una forma más rápida y conveniente de implementar tipos similares de técnicas [54]. El acrónimo zk-STARK significa:

ZK	**Z**ero-**K**nowledge
S	**S**calable
T	**T**ransparent
AR	**AR**gument
K	of **K**nowledge

• Zero-Knowledge (Conocimiento nulo): Al igual que en zk-SNARKs, este término se refiere a una prueba donde un participante puede demostrar a otro que conoce un valor o que una afirmación se cumple, sin revelar ninguna información adicional aparte de la veracidad de la afirmación en sí.

• Scalable (Escalable): En el contexto de las pruebas de conocimiento nulo, una prueba escalable es aquella que puede manejar una gran cantidad de datos sin aumentar significativamente el tiempo de cálculo o el tamaño de la prueba.

• Transparent (Transparente): En el contexto de las pruebas de conocimiento nulo, la transparencia se refiere a la capacidad de los verificadores para revisar y validar las pruebas sin necesidad de información adicional del proponente. La transparencia permite a los verificadores confirmar que las pruebas son correctas y justas sin la necesidad de confiar ciegamente en el proponente.

• ARgument (Argumento): Al igual que en zk-SNARKs, un argumento en este contexto es una declaración o afirmación utilizada como parte de la prueba.

• of Knowledge (de Conocimiento): Este término indica que la prueba se centra en demostrar el conocimiento de algo, en lugar de solo demostrar la verdad de una afirmación. En una prueba de conocimiento nulo, el proponente prueba que conoce cierta información sin revelarla.

Los primeros artículos que detallan zk-STARK fueron publicados en 2018 por Eli Ben-Sasson, Iddo Bentov, Yinon Horeshi y Mikhail Ryabtsev [53].

Siendo otro tipo de prueba de conocimiento nulo no interactiva, zk-STARK es menos popular que su homóloga zk-SNARK, principalmente porque es un modelo más reciente [37].

Utilizando zk-STARKs, es posible calcular varios miles de transacciones en lotes fuera de la cadena y presentar una única prueba zk-STARK para validar las transacciones en la cadena.

Con la T que significa “transparente”, los ZK-STARK resuelven una de las principales debilidades de los ZK-SNARK, su dependencia de una “configuración de confianza”. También vienen con suposiciones criptográficas mucho más simples, evitando la necesidad de curvas elípticas, emparejamientos y el conocimiento de la suposición del exponente y, en su lugar, confiando puramente en los hashes y la teoría de la información; esto también significa que son seguros incluso contra atacantes con computadoras cuánticas [55].

4.3.1. Características zk-STARK

El protocolo zk-STARKs tiene dos ventajas principales:

- Transparencia: el sistema funciona sin una instalación de confianza (es decir, elimina los “residuos tóxicos” que permite zk-SNARKs).

- Escalabilidad: son más escalables con respecto a la velocidad y el tamaño del cálculo, ya que el tiempo necesario para probar se escala de forma casi lineal y, lo que es más importante, el tiempo necesario para la verificación completa y el preprocesamiento se escala de forma polilogarítmica.

El protocolo zk-STARK está compuesto por 3 funciones principales: $G, P\ y\ V$, similares a zk-SNARK.

La función G (generador de claves) toma como entrada un programa C (la función de parámetro necesaria para calcular W) y genera dos claves: la “clave de prueba” (proving_key) y la “clave de verificación” (verifying_key).

$$(proving\_key, verifying\_key) = G$$

La función P toma tres parámetros como entrada: la clave de prueba (clave_de_prueba), el valor aleatorio y público x y el conocimiento de prueba Witness (que no se revela). La salida de P crea y devuelve una prueba “proof”.

$$Proof = P(proving\_key, x, witness)$$

La función V toma tres parámetros como entrada: la clave del verificador (verifying_key), el valor disponible x y la prueba. La salida de V es $True$ o $False$, indicando si la prueba es verdadera o falsa.

$$\{True \mid False\} = V(vk, x, prf)$$

A diferencia de zk-SNARK, zk-STARK no utiliza un parámetro λ "residuo tóxico", lo que significa que no hay necesidad de confiar en un conjunto de confianza para garantizar la seguridad y privacidad de las transacciones.

Desde un punto de vista técnico, los zk-STARKs presentan una ventaja significativa en términos de confianza y eficiencia gracias a su algoritmo de cifrado simétrico basado en funciones hash resistentes a colisiones. A diferencia de los zk-SNARKs, que necesitan una cadena de referencia común (CRS) para funcionar y pueden requerir una gran cantidad de potencia informática para su configuración, los zk-STARKs eliminan la necesidad de estas configuraciones de confianza gracias a su algoritmo de cifrado.

El término "Argumento" representado por "AR" en zk-SNARK se refiere a la configuración que debe realizarse antes de que se pueda utilizar el protocolo. Estos argumentos pueden ser intensivos en términos de computación y representan un potencial punto de vulnerabilidad para futuras computadoras cuánticas.

Por el contrario, los zk-STARKs utilizan funciones hash resistentes a colisiones. Estas funciones, que son esenciales para la seguridad criptográfica, tienen la propiedad de que es computacionalmente difícil encontrar dos entradas distintas que den como resultado el mismo hash. Esta característica es la que elimina la necesidad de configuraciones de confianza en los zk-STARKs, haciéndolos más eficientes y seguros frente a amenazas futuras de la computación cuántica.

Esos argumentos cuestan una gran cantidad de potencia informática y plantean la posibilidad de verse amenazados por futuras computadoras cuánticas [56].

El principal problema de zk-STARK es su tamaño. Actualmente, las pruebas que utiliza son demasiado grandes para ser utilizadas en la mayoría de las cadenas de bloques en su forma actual. Según Vitalik Buterin, zk-STARKs dará lugar a pruebas de varios cientos de kilobytes de tamaño, frente a los 288 bytes de zk-SNARKs.

4.4. Comparación entre zk-SNARK y zk-STARK

Aunque ambos protocolos comparten objetivos similares, existen diferencias y similitudes importantes en sus propiedades y aplicaciones. Este capítulo proporcionará un análisis de las diferencias y similitudes entre estos dos métodos, así como sus ventajas y desventajas.

1. Configuración confiable y transparencia

ZK-SNARKs requieren una fase de configuración inicial llamada “ceremonia de generación de parámetros”, en la que se generan ciertos parámetros públicos y secretos. Es crucial que los secretos generados durante esta fase sean destruidos, ya que, de lo contrario, un atacante podría generar pruebas fraudulentas. Esta dependencia en una configuración confiable puede ser problemática, ya que se necesita confiar en la honestidad de los participantes en la ceremonia.

Por otro lado, zk-STARKs no requieren una configuración confiable y utilizan aleatoriedad pública en lugar de secretos compartidos. Esto significa que no hay necesidad de confiar en la honestidad de terceros y, por lo tanto, se considera una solución más transparente y segura.

2. Resistencia cuántica

La seguridad de zk-SNARK se basa en suposiciones criptográficas específicas, como el problema de residuos cuadráticos cortos y la hipótesis de Diffie-Hellman extendida. Estas suposiciones son vulnerables a ataques de computadoras cuánticas, lo que hace que zK-SNARKs sean potencialmente inseguros en el futuro.

En contraste, ZK-STARKs son resistentes a ataques cuánticos, ya que su seguridad se basa en funciones de hash resistentes a colisiones y un modelo de oráculo aleatorio. La resistencia cuántica de ZK-STARKs los convierte en una opción más sólida a largo plazo en términos de seguridad.

3. Complejidad y escalabilidad

zk-STARKs ofrecen una mayor escalabilidad en comparación con zk-SNARKs. A medida que aumenta el tamaño de la computación, la complejidad asociada con la comunicación, la generación de la prueba por parte del probador, y la verificación de la prueba por parte del verificador, crece a un ritmo mucho más lento en el caso de los zk-STARKs. Este aumento controlado de la complejidad se atribuye en parte al uso que hacen los zk-STARKs de los códigos de corrección de errores y los árboles de Merkle, mecanismos que ayudan a minimizar la complejidad de las pruebas.

Sin embargo, hay una compensación en términos de eficiencia de verificación. ZK-SNARKs generalmente tienen tiempos de verificación de pruebas más rápidos después de la fase de configuración, mientras que ZK-STARKs pueden tener tiempos de verificación más largos debido a su mayor complejidad de comunicación.

4. Aplicaciones y adopción

Las pruebas zk-SNARKs han sido adoptados en varias criptomonedas, como Zcash, para ofrecer transacciones privadas y protección de identidad. Las zk-STARKs, en mismo tiempo, son una tecnología más reciente y, aunque aún

no se han adoptado ampliamente, tienen el potencial de ser utilizados en una variedad de aplicaciones debido a sus propiedades de transparencia y resistencia cuántica. Se espera que ZK-STARKs encuentren adopción en áreas como votaciones, sistemas de identidad digital y contratos inteligentes, donde la transparencia y la resistencia a la corrupción son críticas.

Se puede llegar a las siguientes conclusiones, que las zk-SNARK y las zk-STARK difieren en cuatro puntos principales:

- Transparencia:
  - Zk-SNARK: Requiere configuración de confianza inicial y es vulnerable a pruebas falsas si los parámetros caen en malas manos.
  - Zk-STARK: No requiere configuración de confianza y es más transparente al utilizar criptografía resistente a colisiones.

- Seguridad:
  - Zk-SNARK: Computacionalmente robusto, pero vulnerable a ataques de computación cuántica.
  - Zk-STARK: Es resistente a la computación cuántica.

- Escalabilidad:
  - Zk-SNARK: Más pequeño, pero más lento en generar pruebas; consume menos gas y verifica pruebas más rápido.
  - Zk-STARK: Aunque las pruebas generadas son más grandes, el proceso de generación de pruebas es más rápido y escala mejor. La verificación de las pruebas puede ser más lenta en períodos de bajo ancho de banda

- Estructura:
  - Zk-SNARK: Basado en curvas elípticas.
  - Zk-STARK: Utiliza funciones hash resistentes a colisiones.

A continuación, se presenta una tabla comparativa entre zk-SNARKs y zk-STARKs, que presenta una visión rápida y simplificada de las diferencias y similitudes entre estos dos protocolos:

	zk-SNARKs	**zk-STARKs**
Transparencia	Baja	Alta
Resistencia cuántica	No	Sí
Escalabilidad	Baja	Alta
Verificación	Rápida	Lenta
Configuración inicial	Necesaria	No necesaria

Adopción actual	Alta	Menor (emergiendo)
Estructura criptográfica	Curvas elípticas	Funciones hash resistentes a colisiones

Tabla 3. Comparación entre zk-SNARK y zk-STARK. Fuente: elaboración propia

4.5. Limitaciones

Vitalik Buterin señala [57] que los zk-SNARK son efectivos al establecer sistemas donde los usuarios poseen un estado privado. No obstante, los zk-SNARK no pueden mantener un estado privado desconocido para todos. Para generar una prueba sobre cierta información, quien la demuestre debe conocerla en su forma original.

Un ejemplo de lo que resulta difícil privatizar es Uniswap. En Uniswap, hay una cuenta única centralizada en términos lógicos, que es la cuenta del creador de mercado. Esta cuenta no tiene propietario y cada transacción en Uniswap se ejecuta en oposición a la cuenta del creador de mercado. Ocultar el estado de esta cuenta es imposible, ya que alguien tendría que sostener ese estado en texto legible para generar pruebas, y su implicación activa sería necesaria en cada transacción.

Se podría implementar un Uniswap centralizado, seguro y privado utilizando circuitos ofuscados con zk-SNARK, aunque no está claro si los beneficios de hacerlo compensan los costos. Puede que ni siquiera haya un beneficio real: el contrato necesitaría informar a los usuarios sobre los precios de los activos y los cambios en los precios de bloque en bloque revelarían gran parte de la actividad comercial.

Aunque las blockchains permiten que la información del estado sea global y los zk-SNARKs logren que la información del estado sea privada, no se cuentan con un método efectivo para combinar ambas características.

Es importante mencionar que se puede emplear la computación *multi-party* para lograr un estado privado compartido. Sin embargo, esto implica una suposición de umbral de mayoría honesta, que posiblemente sea inestable en la práctica, ya que (a diferencia de los ataques del 51%) una mayoría malintencionada podría conspirar para violar la privacidad sin ser descubierta.

5. Herramientas zk-SNARK y zk-STARK en Ethereum

En este capítulo se llevará a cabo un estudio de las diversas herramientas disponibles para la implementación de pruebas zk-SNARK, incluyendo bibliotecas y lenguajes específicos. A continuación, se realizará un análisis similar para las pruebas zk-STARK.

Además, se presentará un recorrido por varios proyectos basados en Ethereum que han adoptado estos protocolos.

5.1. Herramientas para Zk-SNARKs

Este apartado proporciona una visión general de las herramientas para implementar pruebas zk-SNARK y zk-STARK.

Estas herramientas pueden ser clasificadas en tres categorías:

1. Bibliotecas
2. Compiladores
3. Lenguajes específicos

5.1.1. Bibliotecas para zk-SNARKs

Las bibliotecas para zk-SNARKs son herramientas que facilitan la creación y manejo de pruebas zk-SNARK. En la Tabla 4, se ofrece un resumen de algunas de las bibliotecas disponibles para escribir pruebas SNARK.

Nombre de biblioteca	Lenguaje
libsnark	C++
bellman	Rust
jsnark	Java
gnark	Go
snarkjs	JavaScript

Tabla 4. Bibliotecas para zk-SNARK

Cada una de estas bibliotecas ofrece capacidades y ventajas únicas, tal y como se describen a continuación:

- **libsnark**: Es una de las más antiguas en términos de implementación de zk-SNARKs, particularmente de Groth16, un algoritmo popular en este campo. Libsnark es conocida por su velocidad y la amplia gama de funcionalidades que ofrece, incluyendo una API robusta para la creación

de circuitos. Aunque su uso puede ser complejo, libsnark es notablemente completa y es muy apreciada en la comunidad de desarrolladores.

- **bellman**: Esta biblioteca permite a los usuarios crear circuitos a nivel muy bajo y utilizar abstracciones de alto nivel. Bellman es la biblioteca elegida para la implementación de Zcash, un importante proyecto de criptomoneda centrado en la privacidad.

- **jsnark**: Es una biblioteca en Java que utiliza libsnark como backend. También puede funcionar como backend para xJsnark [59], un marco de alto nivel para la creación de circuitos. Aunque principalmente se utiliza con fines académicos y experimentales, jsnark es valiosa por su accesibilidad y compatibilidad con Java.

- **gnark:** Esta biblioteca en Go es especialmente destacable por su velocidad en la generación de pruebas zk-SNARK y por su soporte tanto para Groth16 como para Plonk. Estos son dos algoritmos diferentes que se utilizan en la construcción de pruebas zk-SNARK. Groth16 es conocido por ser uno de los esquemas zk-SNARK más eficientes en términos de tamaño de prueba y tiempo de verificación, mientras que Plonk es un esquema zk-SNARK universal que es compatible con todo tipo de cálculos. Además de su compatibilidad con estos dos esquemas, gnark también destaca por su flexibilidad, permitiendo a los desarrolladores la creación y verificación de pruebas personalizadas.

- **snarkjs:** Es una biblioteca de JavaScript que implementa los algoritmos zk-SNARK Groth16 y Plonk. Esta biblioteca contiene las herramientas necesarias para realizar la configuración de confianza, cubriendo tanto la fase universal como la parte específica del circuito. Las especificaciones del circuito para snarkjs pueden escribirse en el lenguaje Circom. Lo que distingue a snarkjs de otras bibliotecas es su enfoque en facilitar la configuración de confianza, haciendo el proceso más accesible y manejable para los desarrolladores.

Es importante señalar que la mayoría de estas bibliotecas son académicas o experimentales y, por lo tanto, no están preparadas para su uso en producción.

La biblioteca libsnark, reconocida por su rapidez y completitud, es la opción más utilizada en la comunidad de desarrollo actualmente (mayo 2023). Su reputación se debe no solo a su desempeño eficiente, sino también a su amplio conjunto de funcionalidades que la convierten en una herramienta versátil y completa para trabajar con zk-SNARKs.

5.1.2. Lenguajes para zk-SNARKs

El panorama de los lenguajes específicos de dominio para crear zk-SNARKs ha experimentado una notable evolución en los últimos años, con la aparición de diversas opciones como ZoKrates y Circom. Estos lenguajes, cada uno con sus particularidades y ventajas, presentan diferencias sustanciales en cuanto a su nivel de abstracción, oscilando entre propuestas de alto y bajo nivel.

ZoKrates

Es un conjunto de herramientas para usar zk-SNARK en la red de Ethereum. ZoKrates permite a los desarrolladores crear pruebas y luego validarlas con Solidity, lo que permite la validación de pruebas en aplicaciones descentralizadas (Dapps) basadas en la blockchain de Ethereum.

El lenguaje ZoKrates admite operadores aritméticos y de comparación estándar, y aunque permite los bucles “for”, es necesario especificar un límite superior para el número de iteraciones. No admite la recursión.

ZoKrates proporciona un lenguaje de programación de alto nivel y un compilador que permiten a los desarrolladores diseñar programas que se traducen en pruebas zk-SNARK, eliminando la necesidad de interactuar directamente con la complejidad de Libsnark. Así, ZoKrates facilita el uso de zk-SNARKs, haciendo estas pruebas más accesibles para las aplicaciones de la blockchain.

Circom

Circom es un lenguaje de programación de circuitos y un compilador que permite a los programadores diseñar y crear sus propios circuitos aritméticos para ZKP. La salida del compilador es un archivo JSON que describe las restricciones. Después, podemos usarlo como entrada para la configuración de confianza, creando el testigo y la construcción de los pasos de prueba llevados a cabo por la librería snarkjs.

Dado que Circom es un lenguaje de bajo nivel, permite a los desarrolladores ser más flexibles en el nivel de precisión para describir un circuito. Como resultado, Circom es bastante popular entre los desarrolladores, especialmente como herramienta para la primera experiencia práctica con pruebas de conocimiento nulo.

Un ejemplo destacado es su uso en el juego de estrategia en tiempo real Dark Forest, que utiliza pruebas de conocimiento nulo para mantener el estado del mundo del juego en secreto hasta que los jugadores decidan revelarlo. Además, Circom se ha utilizado en el mezclador Tornado Cash.

5.2. Herramientas para zk-STARK

Este capítulo presenta una visión general de las bibliotecas y lenguajes utilizados para implementar las pruebas de conocimiento nulo zk-STARK y zk-SNARK, y además, revisaremos algunos de los proyectos más notables que han aplicado estas tecnologías en sus sistemas.

5.2.1. Bibliotecas para zk-STARK

Para proporcionar una visión más completa de las bibliotecas que implementan STARKs, la Tabla 5 ofrece un panorama general. En caso de zk-STARK, también es importante destacar que, en su mayoría, estos proyectos tienen un carácter más académico y aún no se encuentran en fase de producción.

Nombre de biblioteca	Lenguaje
libSTARK	C++
OpenZKP	Rust
genSTARK	Javascript

Tabla 5. Bibliotecas para zk-STARK

- **Libstark**: Fue desarrollado por los autores del artículo “Scalable Zero Knowledge with No Trusted Setup” [60]. Este artículo incluye un ejemplo de uso de zk-STARKs para el caso de uso de Perfiles de ADN (DPM, por sus siglas en inglés). Este caso de uso ilustra cómo los zk-STARKs pueden aplicarse para resolver problemas en el campo de la genética forense y la privacidad del ADN, demostrando así la versatilidad y aplicabilidad de esta tecnología en una amplia gama de contextos. Libstark proporciona las herramientas y recursos necesarios para que los desarrolladores puedan experimentar con zk-STARKs y adaptarlos a sus propios proyectos y necesidades.

- **OpenZKP.** Esta biblioteca facilita una interfaz sencilla para desarrollar los zk-STARK, tanto desde el punto de vista del que proporciona la prueba como el que verifica la prueba. La biblioteca contiene ejemplos de pruebas para la secuencia de Fibonacci, MiMC, Pedersen hash y algunos artefactos para operaciones en curvas elípticas.

- **GenSTARK.** Esta biblioteca cuenta con ejemplos de STARKs para las funciones hash compatibles con STARK Rescue, MiMC y Poseidon. Además, se puede utilizar en combinación con los lenguajes Airscript y AirAssembly para una expresión más directa de las declaraciones.

5.2.2. Lenguajes para zk-STARK

Cairo

Cairo fue presentado en 2020 como un lenguaje de programación Turing-completo diseñado para la creación eficaz de programas que pueden ser demostrados mediante el uso de STARK. El enfoque implementado en el Cairo consiste en tener un conjunto universal de restricciones que pueden expresar la

ejecución de cualquier programa por una máquina virtual. Cairo consta de los siguientes componentes:

- Bytecode de programa Cairo
- Ensamblaje de Cairo
- Cairo Runner

El proceso de creación y prueba de una prueba STARK utilizando Cairo consta de los siguientes pasos:

1. Escribe un programa de Cairo para el cálculo. Se puede usar el ensamblaje de Cairo directamente o algún otro lenguaje que pueda compilarse en bytecode de Cairo.

2. Compila el programa en bytecode de Cairo.

3. Ejecuta el programa usando Cairo Runner para obtener la traza de ejecución S, la función de memoria parcial $m*$ y la función de memoria completa m.

4. Utiliza un demostrador STARK para el Cairo AIR para generar una prueba para la afirmación: “La máquina de Cairo no determinista acepta dado la entrada S, $m*$, m (salida del Cairo Runner)”.

Cairo tiene dos conceptos interesantes:

- Hints. Las pistas son fragmentos de código insertados entre las instrucciones de Cairo. Representan la información solo conocida por el Prover.

- Builtins. Son unidades de ejecución de bajo nivel predefinidas utilizadas para realizar cálculos comunes que serían costosos usando el enfoque general de Cairo, como verificaciones de rango, la función hash de Pedersen, o ECDSA. Podemos imaginarlos como columnas adicionales para el Cairo AIR universal.

Cairo ya es un proyecto de grado de producción: es la columna vertebral de varios sistemas de criptomonedas en la cadena de bloques Ethereum. Sin embargo, dado que la mayoría de los desarrolladores están acostumbrados al lenguaje Solidity para escribir proyectos de contratos inteligentes, se inventó un conversor de Solidity a Cairo llamado Warp. Su principal caso de uso es convertir proyectos existentes en Solidity a Cairo. Para proyectos futuros, la implementación directamente en Cairo debería ser más ventajosa.

5.3. Proyectos basados en zk-SNARK

Zcash

Anteriormente conocido como ZeroCash en alusión a la prueba de conocimiento nulo que respalda sus transacciones que preservan la privacidad, Zcash es uno de los activos criptográficos más antiguos y ha contribuido a impulsar el uso de la tecnología de pruebas de conocimiento nulo en la industria.

Loopring

Loopring es un exchange descentralizado (DEX) construido en Ethereum que admite operaciones como libro de órdenes sin tomar la custodia de los activos de los usuarios. Alimentado por Chainlink Price Feeds, ha servido a más de cien mil usuarios y ha facilitado miles de millones en volumen de trading.

zkSync 1.0

zkSync 1.0, una realización de la tecnología zk-Rollup en la red Ethereum, es un protocolo que permite la ejecución de transferencias y swaps de tokens, aunque no admite contratos inteligentes. Este protocolo fue diseñado y desarrollado por Matter Labs, y se caracteriza por su enfoque en la escalabilidad y eficiencia, procurando una integración efectiva con la red Ethereum.

zkSync 2.0

Similar a StarkNet, zkSync 2.0 es una solución de escalabilidad de Ethereum de segunda capa que utiliza una arquitectura de volición que admite contratos inteligentes. ZkSync utiliza zk-SNARKs para validar transacciones y usa zkPorter, un sistema de prueba de participación, para la disponibilidad de datos. La principal diferencia entre zkSync 2.0 y StarkNet, además de su prueba de validez, es que zkSync 2.0 es compatible con EVM.

ZigZag

El protocolo ZigZag es un exchange descentralizado que promete más privacidad y más rendimiento, y utiliza un libro de órdenes para pares de negociación ERC-20, a diferencia de la mayoría de los DEX, que utilizan diseños de creadores de mercado automáticos (AMM). Es la escalabilidad de los zk-rollups lo que hace viable el diseño del libro de órdenes. Cualquier token en el registro de zkSync puede ser listado en ZigZag. El protocolo actualmente opera en zkSync 1.0 pero tiene planes de lanzarse tanto en zkSync 2.0 como en StarkNet.

Mina

El protocolo Mina es un proyecto de blockchain de prueba de conocimiento nulo más ligero del mundo. Mina utiliza zk-SNARKs para diseñar una blockchain completa que mide aproximadamente 22kB. Es la primera capa L1 que permite una implementación eficiente y una programabilidad sencilla de contratos

inteligentes de conocimiento nulo (zkApps). Gracias a sus distintivas funcionalidades de privacidad y su habilidad para enlazarse con cualquier página web, Mina está diseñando una conexión segura entre el mundo criptográfico y el mundo real.

5.4. Proyectos basados en zk-STARK

StarkEx

StarkEx es una solución de escalabilidad de segunda capa L2, construido sobre Ethereum, que aprovecha las pruebas STARK para validar las transacciones en custodia propia, habilitando la construcción de aplicaciones de trading y pagos. Variados proyectos como DeversiFi, Sorare y dYdX, que se han desarrollado sobre StarkEx, han logrado generar centenares de millones de transacciones y volúmenes de trading valorados en cientos de miles de millones de dólares. A pesar de estas ventajas, StarkEx carece de la funcionalidad de contratos inteligentes requerida para el funcionamiento de dApps plenamente desarrolladas.

StarkNet

StarkNet es una plataforma de uso general diseñada para permitir a los desarrolladores implementar contratos inteligentes en un zk-rollup basado en Ethereum. Se espera que Aave y Maker, dos de las dApps más destacadas de Ethereum, hagan su lanzamiento en StarkNet. Importante también mencionar que los zk-rollups de StarkEx pueden ser desplegados sobre StarkNet, incrementando así la escalabilidad de una aplicación.

Con el fin de explotar al máximo las capacidades avanzadas de computación y escalabilidad ofrecidas por STARKs, StarkWare ha introducido Cairo, un lenguaje de programación de Turing altamente eficiente para generar pruebas STARK. Esto implica que StarkWare se encuentra en proceso de establecer un ecosistema de desarrolladores, proporcionando documentación, marcos de trabajo y herramientas de soporte.

Immutable X

Immutable X es una plataforma NFT que utiliza un zk-rollup específico de la aplicación con StarkEx para facilitar la creación y el intercambio de NFTs y tokens. A pesar de los periodos de alta congestión en la red Ethereum, la plataforma ha logrado soportar decenas de millones de creaciones y transacciones de NFT a bajas tarifas.

Con su potente motor de escalabilidad StarkEx, ImmutableX ofrece a los desarrolladores y jugadores la capacidad de realizar transacciones a alta velocidad y con costos de gas reducidos, todo sin comprometer la seguridad y la descentralización inherentes a Ethereum. La implementación de pruebas STARK permite la comprobación segura de las transacciones, manteniendo a la vez una experiencia de juego de alta calidad.

ApeX Pro

ApeX Pro es una plataforma de trading sin custodia que se centra en ofrecer operaciones ilimitadas de contratos perpetuos con margen cruzado. El objetivo primordial de ApeX Pro es abordar y superar varios obstáculos prevalentes en el mundo del comercio de criptomonedas, como la restricción en el acceso global, el rendimiento comercial que no cumple con las expectativas y la deficiente protección de la seguridad y privacidad.

Aprovechando las pruebas criptográficas de StarkEx y Validium, ApeX Pro valida de manera segura los lotes de transacciones. Las pruebas STARK de StarkWare aseguran la disponibilidad e integridad de los datos en la blockchain, lo que refuerza la seguridad de este protocolo sin custodia. Esta arquitectura de capa 2, construida sobre la red Ethereum, publica pruebas de conocimiento nulo directamente en los contratos inteligentes de Ethereum para su verificación. Las transacciones se agrupan de una forma única antes de su publicación en la cadena, de manera que solo los cargos por saldo sean visibles.

6. Conclusiones

En resumen, la integración de privacidad y transparencia en una blockchain plantea un desafío inherentemente contradictorio. Sin embargo, reconocer la necesidad de un enfoque equilibrado es fundamental para garantizar el éxito y la adopción generalizada de la tecnología blockchain. La mejora de la privacidad en Ethereum ha sido identificada por Vitalik Buterin, cofundador de Ethereum, como una de las transiciones críticas necesarias para el futuro de la plataforma [61].

A raíz del análisis llevado a cabo, se constata que los protocolos de conocimiento nulo se encuentran en una fase activa de investigación y desarrollo por parte de los agentes implicados en el ámbito de los registros distribuidos. Esto se debe a su prometedor potencial para fortalecer la confidencialidad en estos sistemas. No obstante, es importante señalar que la mayoría de estos proyectos tienen un enfoque predominantemente académico y existe una notoria escasez de productos de software desarrollados para crear y utilizar sistemas basados en pruebas de conocimiento nulo.

Entre los diferentes protocolos de conocimiento nulo, zk-SNARK y zk-STARK se han alzado como soluciones prometedoras para mejorar la privacidad en Ethereum. Aunque presentan diferencias en términos de características y disponibilidad de herramientas, ambos representan avances significativos en la protección de la información sensible en la red.

En el caso de zk-SNARK, se ha logrado un mayor avance y se ha utilizado en proyectos reales, como Tornado Cash y Zcash. Estos protocolos han demostrado su eficacia en la protección de la privacidad de los usuarios, lo cual es fundamental en un entorno donde la mayoría de los usuarios valora la transparencia y la trazabilidad de las transacciones. Sin embargo, es importante tener en cuenta que la adopción de tecnologías de privacidad puede estar limitada por el costo percibido de pagar por la privacidad. Esto se evidencia en el caso de Zcash, donde a pesar de existir desde hace casi 7 años, solo una pequeña fracción de ZEC se almacena utilizando protocolos de privacidad, y la gran mayoría se encuentra en direcciones transparentes con poca privacidad.

Las zk-STARKs están en una etapa más temprana de desarrollo y adopción. Aunque se están realizando esfuerzos por parte de organizaciones como StarkWare para investigar y desarrollar zk-STARK, aún queda trabajo por hacer para mejorar su escalabilidad y descentralización. Además, la falta de herramientas disponibles y la necesidad de una mayor madurez tecnológica pueden limitar su adopción en la actualidad.

Es importante destacar que la conciencia de los usuarios sobre la importancia de la privacidad en las transacciones y registros blockchain está en una etapa incipiente. Muchos usuarios aún no están familiarizados con los riesgos asociados con la divulgación de información personal y financiera en un entorno público y abierto. Esto puede influir en la demanda y adopción de tecnologías de

conocimiento nulo, ya que los usuarios pueden no percibir la necesidad o el valor de utilizar estas soluciones.

En conclusión, si bien los protocolos de conocimiento nulo como zk-SNARK y zk-STARK ofrecen soluciones prometedoras para mejorar la privacidad en Ethereum, su adopción está influenciada por varios factores. La disponibilidad de herramientas y la madurez tecnológica, el costo percibido de pagar por la privacidad y la conciencia de los usuarios son elementos clave que afectan la adopción de estas tecnologías. A medida que se realicen avances en el desarrollo de soluciones, se fomente la conciencia sobre la importancia de la privacidad y se reduzcan las barreras de adopción, es probable que veamos un mayor uso de tecnologías de conocimiento nulo en el futuro. Sin embargo, es importante reconocer que el camino hacia la adopción generalizada de la privacidad en blockchain aún presenta desafíos significativos que requieren una atención continua por parte de la comunidad y los desarrolladores.

7. Bibliografía

1. Antonopoulos, G.Wood, Mastering Ethereum, O´Reilly, 2018

2. *What are zk-snarks?* (2022) *Zcash*. Disponible en: https://z.cash/technology/zksnarks/ (Accedido el 02/03/2023).

3. Deleuze, G. (1991). "Posdata sobre las sociedades de control", en Ferrer, C. (Comp.), El lenguaje literario, Tomo 2, Nordan, Montevideo.

4. inEducationInfrastructureResearch, P. et al. (2022) The zero knowledge frontier: On snarks, Starks, and future applications, The Tie Research. Disponible en: https://research.thetie.io/zero-knowledge-starks-snarks/ (Accedido el 25/02/2023).

5. Gabizon, A. (2018) Explaining snarks part VII: Pairings of elliptic curves, Electric Coin Company. Disponible en: https://electriccoin.co/blog/snark-explain7/. (Accedido el 15/03/2023).

6. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. (2014). Scalable Zero Knowledge via Cycles of Elliptic Curves.

7. Binance Academy (2023) Что такое zk-snarks и zk-Starks?, Binance Academy. Binance Academy. Disponible en: https://academy.binance.com/ru/articles/zk-snarks-and-zk-starks-explained (Accedido el 20/02/2023).

8. F. Beres, I. A. Seres, A. A. Benczur, & M. Quintyne-Collins (2021). Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users. In 2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) (pp.69-78).IEEE Computer Society.

9. G. Almashaqbeh, and R. Solomon 2022. SoK: Privacy-Preserving Computing in the Blockchain Era. In 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) (pp. 124-139). IEEE Computer Society.

10. The State of Privacy on ethereum (2019) ConsenSys. Disponible en: https://consensys.net/blog/developers/the-state-of-privacy-on-ethereum/ (Accedido el 10/03/2023).

11. Understanding plonk (no date) Vitalik Buterin's website. Disponible en: https://vitalik.ca/general/2019/09/22/plonk.html (Accedido el 12/03/2023).

12. Boneh, D., Drake, J., Fisch, B., and Gabizon, A. 2021. Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments. In Advances in

Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I (pp. 649–680). Springer-Verlag.

13. Hackernoon. (2019). Zero-Knowledge Proof Algorithm: ZK-STARK & FRI Protocol. Hackernoon. Disponible en: https://hackernoon.com/zero-knowledge-proof-algorithm-zk-stark-fri-protocol (Accedido el 15/03/2023).

14. Bowe, S. (2020) Explaining halo 2, Electric Coin Company. Disponible en: https://electriccoin.co/blog/explaining-halo-2/ (Accedido el 15/03/2023).

15. Malwa, S. (2022) At ethseoul, Ethereum developers turn attention to privacy and users, CoinDesk Latest Headlines RSS. CoinDesk. Disponible en: https://www.coindesk.com/tech/2022/08/10/at-ethseoul-ethereum-developers-turn-attention-to-privacy-and-users/ (Accedido el 16/03/2023).

16. Kumar, E.S. Preserving Privacy in Ethereum Blockchain. *Ann. Data. Sci.* **9**, 675–693 (2022). https://doi.org/10.1007/s40745-020-00279-9

17. Multicoin Capital: Privacy is a feature, not a product (no date) Multicoin Capital RSS. Disponible en: https://multicoin.capital/2019/09/24/privacy-is-a-feature/?mc_cid=6947a61a2c&mc_eid=655d7850ca (Accedido el 23/03/2023).

18. ConsenSys (2020) The State of Privacy on ethereum, Medium. ConsenSys Media. Disponible en: https://media.consensys.net/the-state-of-privacy-on-ethereum-96b42f3109e6 (Accedido el 23/03/2023: March 23, 2023).

19. Zchain - Zcash blockchain explorer (no date). Disponible en: https://explorer.zcha.in/statistics/value (Accedido el 28/03/2023).

20. Wüst, K., and Gervais, A. 2018. Do you Need a Blockchain?. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 45-54).

21. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., & Virza, M. (2014). SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Advances in Cryptology - CRYPTO 2013 (pp. 90-108). Springer, Berlin, Heidelberg.

22. Bürgel, D.S. (2022) A post-merge vision for WEB3, Medium. HOPR. Disponible en: https://medium.com/hoprnet/a-post-merge-vision-for-web3-7b44f4308e79 (Accedido el 15/03/2023).

23. Liu, B. (2022) What to expect from Ethereum Security and privacy after the merge, Blockworks. Disponible en: https://blockworks.co/news/what-to-expect-from-ethereum-security-and-privacy-after-the-merge (Accedido el 01/04/2023).

24. Ethereum is grappling with the risk of censorship after the merge (no date) The Block. Disponible en: https://www.theblock.co/post/169508/ethereum-is-grappling-with-the-risk-of-censorship-after-the-merge (Accedido el 15/03/2023).

25. Caria, W. 2023, ValueWalk: Soulbound Tokens (SBTs) Are Better Than NFTs, Newstex, Chatham

26. Binance Academy (2023) Sybil attacks explained, Binance Academy. Binance Academy. Disponible en: https://academy.binance.com/en/articles/sybil-attacks-explained (Accedido el 05/04/2023).

27. "Tokens here and everywhere", 2023, Shoe Intelligence, vol. 34, no. 13.

28. Zhou, Y., Wu, J., and Zhang, S. 2021. Anonymity Analysis of Bitcoin, Zcash and Ethereum. In 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE) (pp. 45-48).

29. #Ethereum Gas Station Network (GSN) (no date) v3.0.0-beta.3 pre-release. Disponible en: https://docs.opengsn.org/#the-problem (Accedido el 08/04/2023).

30. Servillo, P. (2022) Gas-free transactions: Meta Transactions explained, Medium. Coinmonks. Disponible en: https://medium.com/coinmonks/gas-free-transactions-meta-transactions-explained-f829509a462d (Accedido el 08/04/2023).

31. Schneier, B. (1996). Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth). John Wiley & Sons, Inc.

32. Goldwasser S., Micali S., and Rackof C. The knowledge complexity of interactive proof systems. STOC'85. Proc. 17th Ann. ACM Symp. Theory of Computing, Providence, Rhode Island, USA, 1985, pp. 291-304.

33. Blum M., Feldman P., and Micali S. Non-interactive zero-knowledge proof systems and applications // STOC'88. Proc. 20th Ann. ACM Symp. Theory of Computing. Chicago, USA, 1988. P. 103-112.

34. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. (2014). Zerocash: Decentralized Anonymous Payments from Bitcoin.

35. Hisham S. Galal, and Amr M. Youssef. (2018). Verifiable Sealed-Bid Auction on the Ethereum Blockchain.

36. Eberhardt, J., and Tai, S. 2018. ZoKrates - Scalable Privacy-Preserving Off-Chain Computations. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and

Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 1084-1091).

37. Team, P. (2023) ZK-Starks vs ZK-snarks - differences in zero-knowledge technologies, Panther Protocol Blog. Panther Protocol Blog. Disponible en: https://blog.pantherprotocol.io/zk-snarks-vs-zk-starks-differences-in-zero-knowledge-technologies/ (Accedido el 29/04/2023).

38. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. (2016). ZKBoo: Faster Zero-Knowledge for Boolean Circuits.

39. Virza, M. (2019) ZK-Sharks, MIT Digital Currency Initiative. Disponible en: https://dci.mit.edu/zksharks (Accedido el 02/05/2023).

40. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. In 2018 IEEE Symposium on Security and Privacy (SP) (pp. 315-334).

41. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. (2019). Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings.

42. Estándar de token ERC-20 (no date) ethereum.org. Disponible en: https://ethereum.org/es/developers/docs/standards/tokens/erc-20/ (Accedido el 02/05/2023).

43. Estándar de token no fungible ERC-721 (no date) ethereum.org. Disponible en: https://ethereum.org/es/developers/docs/standards/tokens/erc-721/ (Accedido el 02/05/2023).

44. Estándar de token ERC-777 (no date) ethereum.org. Disponible en: https://ethereum.org/es/developers/docs/standards/tokens/erc-777/ (Accedido el 02/05/2023).

45. Joachim Lebrun (@Joachim-Lebrun), T.M.(@T.M. (2021) ERC-3643: T-rex - token for regulated exchanges [draft], Ethereum Improvement Proposals. Disponible en: https://eips.ethereum.org/EIPS/eip-3643 (Accedido el 06/05/2023).

46. Singh, A. (2023) Create decentralized identifiers and verifiable credentials with Veramo framework, Infrablok. Disponible en: https://infrablok.com/create-decentralized-identifiers-and-verifiable-credentials-with-veramo-framework/ (Accedido el 06/05/2023).

47. Introduction to ethereum's keccak-256 algorithm (2022) RugDoc Wiki. Disponible en: https://wiki.rugdoc.io/docs/introduction-to-ethereums-keccak-256-algorithm/ (Accedido el 07/05/2023).

48. IoTeX (2018) Blockchain privacy-enhancing technology series - stealth address (I), HackerNoon. Disponible en: https://hackernoon.com/blockchain-privacy-enhancing-technology-series-stealth-address-i-c8a3eb4e4e43 (Accedido el 07/05/2023).

49. Autor Washington Gómez CISO (Chief Information Security Officer) Washington Gómez es un profesional de la seguridad de la información que cuenta con amplia experiencia en el campo de las Tecnologías y de la Seguridad de la Información. Actualm et al. (2023) ¿Qué es privatesend? pagos anónimos con dash, Bit2Me Academy. Disponible en: https://academy.bit2me.com/que-es-privatesend-dash/ (Accedido el 08/05/2023).

50. Roy Rinberg, and Nilaksh Agarwal. (2022). Privacy when Everyone is Watching: An SOK on Anonymity on the Blockchain.

51. Pedersen hash¶ (no date) Pedersen Hash - iden3 0.1 documentation. Disponible en: https://iden3-docs.readthedocs.io/en/latest/iden3_repos/research/publications/zkproof-standards-workshop-2/pedersen-hash/pedersen.html (Accedido el 08/05/2023).

52. Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. (2016). MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity.

53. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. (2018). Scalable, transparent, and post-quantum secure computational integrity.

54. Gong, Y., Jin, Y., Li, Y., Liu, Z., and Zhu, Z. 2022. Analysis and comparison of the main zero-knowledge proof scheme. In 2022 International Conference on Big Data, Information and Computer Network (BDICN) (pp. 366-372)

55. Starks, part I: Proofs with polynomials (no date) Vitalik Buterin's website. Disponible en: https://vitalik.ca/general/2017/11/09/starks_part_1.html (Accedido el 30/04/2023).

56. Gong, Y., Jin, Y., Li, Y., Liu, Z., and Zhu, Z. 2022. Analysis and comparison of the main zero-knowledge proof scheme. In 2022 International Conference on Big Data, Information and Computer Network (BDICN) (pp. 366-372).

57. Some ways to use ZK-snarks for privacy (no date) Vitalik Buterin's website. Disponible en: https://vitalik.ca/general/2022/06/15/using_snarks.html (Accedido el 30/04/2023).

58. Anon, (2020). ERC-3643 vs ERC-1400 - Tokeny. Disponible en: https://tokeny.com/erc3643-vs-erc1400/ (Accedido el 21/05/2023).

59. Kosba, A., Papamanthou, C., & Shi, E. (2018). xJsnark: A Framework for Efficient Verifiable Computation. 2018 IEEE Symposium on Security and Privacy (SP), 944–961. https://doi.org/10.1109/SP.2018.00018

60. Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2019). Scalable Zero Knowledge with No Trusted Setup. Advances in Cryptology – CRYPTO 2019, 11694, 701–732. https://doi.org/10.1007/978-3-030-26954-8_23

61. Morgan, N. (2023) 'ethereum fails' without these 3 changes, says Vitalik Buterin, Decrypt. Disponible en: https://decrypt.co/143991/ethereum-fails-without-these-3-changes-says-vitalik-buterin (Accedido el 10/06/2023).