Programmable cryptography for privacy preservation in blockchain verification protocols

Juan Augusto Pose[1]

[1] Laboratorio de Investigación, Desarrollo e Innovación
jaugusto.pose@gmail.com,
[2] C & S Informática S.A

Abstract. Programs for clients and users typically require in-person verification or the repeated exposure of sensitive identifiers to prove eligibility, which increases the risk of data leakage and makes it difficult to audit errors or abuse. In decentralized environments, validation without efficient coordination can lead to invasive practices (such as exposing documents) or costly centralized schemes, compromising both privacy and scalability. Furthermore, the misuse of a "right" may demand additional controls that end up generating correlation between events or leaking operational metadata. This work proposes replacing identity-disclosure-based verification with Zero-Knowledge Proofs (ZKPs), so that eligibility can be validated without revealing who the beneficiary is. The contribution focuses on enabling cryptographically consistent verifications through commitments and Merkle-like structures, with controlled consumption mechanisms to prevent double-spending within a defined temporal and operational context. In this regard, the project uses programmable cryptography to abstract away from the typical challenges of classical cryptography, while maintaining blockchain verifiability and traceability with minimal disclosure.

Keywords: Blockchain, zk-SNARKs, Zero Knowledge, Privacy, Programmable Cryptography

Criptografía programable para la preservación de privacidad en protocolos de verificación blockchain

Resumen. Existen programas para clientes y usuarios que suelen requerir verificaciones presenciales o la exposición repetida de identificadores sensibles para demostrar elegibilidad, lo que incrementa el riesgo de filtración de datos y dificulta la comprobación ante errores o abusos. En entornos descentralizados, la validación sin coordinación eficiente puede derivar en prácticas invasivas (mostrar documentos, por ejemplo) o en esquemas centralizados costosos, comprometiendo privacidad y escalabilidad. Además, el uso indebido de un “derecho” puede exigir controles adicionales que terminan generando correlación entre eventos o

filtrando metadatos operativos. Este trabajo propone sustituir la verificación basada en divulgación de identidad por pruebas de Conocimiento Cero (ZKPs), de modo que la elegibilidad pueda validarse sin revelar quién es el beneficiario. La contribución se enfoca en habilitar verificaciones criptográficamente consistentes mediante compromisos y estructuras tipo Merkle, con mecanismos de consumo controlado para prevenir dobles usos dentro de un contexto temporal y operacional definido. En este sentido, el proyecto utiliza criptografía programable para abstraerse de dificultades típicas de la criptografía clásica, manteniendo verificabilidad y trazabilidad blockchain con mínima divulgación.

Palabras clave: Blockchain, zk-SNARKs, Conocimiento Cero, Privacidad, Criptografía Programable

1 Introducción

La gestión de beneficios y programas de elegibilidad ha evolucionado junto con la digitalización de los servicios. Sin embargo, la mayoría de los sistemas actuales continúan dependiendo de la exposición directa de datos personales como condición necesaria para la verificación. En la práctica, esto se traduce en que un usuario que desea acceder a un descuento, un servicio exclusivo o cualquier beneficio restringido debe presentar su identidad de forma explícita —ya sea mediante un documento físico, un número de identificación o un código asociado a su perfil— al operador que realiza la verificación.

1.1 Problemática de la privacidad en la verificación de elegibilidad

Este modelo introduce al menos tres problemas estructurales. En primer lugar, el operador obtiene información sobre el usuario que excede la necesaria para el propósito de la validación: saber si alguien es elegible no debería requerir saber quién es (Chaum, 1985; Camenisch and Lysyanskaya, 2001). En segundo lugar, los registros acumulados de esas verificaciones pueden construir perfiles de comportamiento de los usuarios sin su consentimiento explícito, incluso cuando el sistema no fue diseñado con esa intención (Narayanan and Shmatikov, 2008). En tercer lugar, la dependencia de bases de datos centralizadas introduce un único punto de falla: si el servidor de elegibilidad no está disponible o es comprometido, el sistema completo se ve afectado.

Este último punto merece especial atención en el contexto de la ciberseguridad. Un repositorio centralizado de datos de identidad asociados a hábitos de consumo podría representar un blanco atractivo para actores maliciosos. Pero incluso en ausencia de una brecha explícita, la mera existencia de esos datos en manos de terceros habilita usos derivados que el usuario no autorizó (Zuboff, 2019): segmentación por algoritmos de recomendación, ajuste dinámico de precios en función del historial de uso (Hannak et al., 2014), o correlación con bases de datos externas para inferir información sensible (Kosinski et al., 2013). Según

lo interpretado en Lavin et al., 2024, estos riesgos no son exclusivos de un dominio particular, sino que se repiten en cualquier sistema donde la ve-rificación de una condición requiera la divulgación de la identidad del verificado.

1.2 Las pruebas de conocimiento cero como alternativa

Las técnicas de conocimiento cero (*Zero-Knowledge Proofs*, ZKPs) ofrecen una alternativa formal a este modelo. Desde los trabajos fundacionales de Goldwasser, Micali y Rackoff Goldwasser et al., 1989, las ZKPs permiten demostrar el conocimiento de un secreto —o el cumplimiento de una condición— sin revelar el secreto en sí. En los últimos años, la aparición de implementaciones eficientes como los zk-SNARKs (*Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge*) ha hecho que esta clase de técnicas sea prácticamente viable en contextos reales, incluyendo aplicaciones sobre redes blockchain Ben-Sasson et al., 2014; Hopwood et al., 2016.

A diferencia de los esquemas clásicos de verificación, los zk-SNARKs permiten que un verificador confirme la validez de una afirmación —por ejemplo, "este usuario pertenece al conjunto de elegibles"— sin obtener ningún dato adicional sobre el usuario. El resultado es un desplazamiento del paradigma: en lugar de "demostrar quién eres", el usuario demuestra "qué sabe". Esta distinción no es solo técnica; implica que el usuario recupera el control sobre su propia información y que ningún operador acumula datos que no necesita para cumplir su función.

Descripción general del sistema propuesto. Este trabajo describe el diseño e implementación de ZkResident, un protocolo basado en zk-SNARKs que permite verificar la elegibilidad de un beneficiario sobre una red blockchain pública sin revelar su identidad. El sistema es completamente descentralizado: cada punto de verificación opera de manera autónoma contra un estado publicado *on-chain* [3], y los beneficiarios no necesitan interactuar con ningún servidor en tiempo real para hacer uso de su derecho. La elegibilidad se acredita mediante una credencial en formato QR que el beneficiario presenta al verificador, quien genera la prueba criptográfica y la valida directamente en la blockchain.

2 Marco teórico y trabajos relacionados

2.1 Pruebas de conocimiento cero

Una prueba de conocimiento cero es un protocolo entre un *prover* y un *verifier* donde el primero convence al segundo de que conoce algún valor w (llamado *testigo* o *witness*) que satisface una relación $R(x, w) = 1$, sin revelar ninguna información sobre w más allá de su existencia Goldwasser et al., 1989.

Los zk-SNARKs son una subclase especialmente atractiva de ZKPs: son no-interactivos (la prueba se genera una sola vez y puede ser verificada por

[3] datos o procesos ejecutados y registrados directamente en una blockchain, sujetos a su consenso y con garantías de inmutabilidad y verificabilidad pública.

cualquiera), concisos (la prueba tiene tamaño constante y se verifica en tiempo eficiente) y de conocimiento cero. Su uso en contratos inteligentes fue popularizado por Zcash Hopwood et al., 2016 y adoptado luego en múltiples proyectos DeFi y de identidad descentralizada. Una propiedad clave para este trabajo es la *simulability*: todo lo que un verificador puede computar a partir de una prueba ZKP, podría haberlo computado sin ella, lo que garantiza formalmente que no hay fuga de información.

2.2 Criptografía programable

El término criptografía programable hace referencia a la capacidad de expresar condiciones arbitrarias como circuitos aritméticos que luego son probados mediante zk-SNARKs. Lenguajes como Noir (más en Community, 2025) permiten escribir circuitos criptográficos con una sintaxis similar a Rust, abstrayendo al desarrollador de la aritmética de campos finitos y de la construcción manual de circuitos. Esta capa de abstracción habilita a equipos sin experiencia profunda en criptografía teórica a construir sistemas con garantías formales de privacidad, tal como también se menciona en Lavin et al., 2024. En la práctica, significa que es posible implementar la lógica "este usuario tiene derecho al beneficio" como un programa verificable, sin que dicha lógica requiera la divulgación de la identidad del usuario en ningún paso.

2.3 Árboles Merkle e inclusión

Un árbol Merkle es una estructura de datos donde cada nodo interno es el hash de sus nodos hijos, y la raíz (*root*) es un resumen compacto de todos los datos de las hojas Merkle, 1988. Los árboles Merkle incrementales (*Incremental Merkle Trees*, IMT) permiten insertar hojas de forma eficiente manteniendo un estado actualizable, lo cual es especialmente útil en contratos inteligentes donde el costo computacional debe minimizarse Semaphore Community, 2025.

La prueba de inclusión Merkle es un vector de nodos hermanos que permite verificar que una hoja pertenece al árbol sin conocer todas las demás hojas. Combinada con una ZKP, esta propiedad permite demostrar pertenencia a un conjunto de forma completamente anónima: el verificador no obtiene ni el índice ni el valor de la hoja, solo la certeza de que existe.

2.4 Poseidon2

Poseidon2 es una función hash diseñada específicamente para uso eficiente en circuitos aritméticos sobre campos finitos Grassi et al., 2023. A diferencia de SHA-256, que requiere miles de restricciones en un circuito R1CS, Poseidon2 puede computarse con pocas decenas, reduciendo el costo de generación de pruebas de forma drástica. Esta función se ha consolidado como un estándar de facto para aplicaciones ZKP sobre la curva BN254.

2.5 Trabajos relacionados

Varios trabajos han explorado la aplicación de ZKPs a sistemas de identidad y membresía privada. Semaphore (ver Semaphore Community, 2025) es un protocolo de señalización anónima basado en árboles Merkle que permite demostrar membresía sin revelar identidad. MACI (*Minimal Anti-Collusion Infrastructure*) extiende estos principios a votaciones resistentes a colisión (ver MACI Project, 2025). En el ámbito de la identidad, proyectos como Polygon ID aplican ZKPs a la verificación de credenciales en contextos comerciales (más en Polygon ID, 2025).

ZkResident adopta la misma responsabilidad de proteger la privacidad del usuario y se diferencia de estos enfoques por su aplicación a un caso concreto —la verificación de elegibilidad en programas de beneficios distribuidos por comercio— e incorpora la noción de períodos y nullifiers por local. Esto permite reutilizar el mismo árbol Merkle de elegibles a lo largo del tiempo sin sacrificar el control de doble gasto, una combinación no presente en los protocolos mencionados.

3 Arquitectura del sistema

3.1 Visión general

ZkResident se estructura en torno a tres actores principales: el operador (quien administra el árbol de elegibles), el beneficiario (quien posee una credencial privada) y el verificador (el comercio o punto de validación). La interacción entre estos actores se articula mediante contratos inteligentes *on-chain* y un circuito zk-SNARK ejecutado *off-chain*. Desde el punto de vista de la privacidad, el di-seño fue concebido para minimizar la cantidad de actores —incluyendo al operador que habilitó al beneficiario— que pueda rastrear cuándo, dónde o con qué frecuencia ese beneficiario hace uso de su derecho.

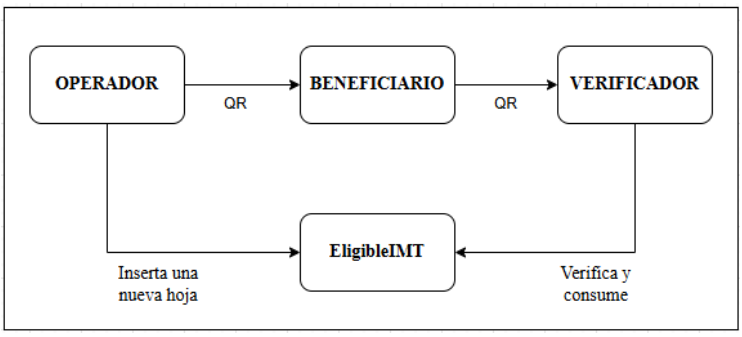

Fig. 1. Flujo general del sistema ZkResident con sus tres actores principales.

3.2 Componentes principales

Contrato EligibleIMT. EligibleIMT implementa un árbol Merkle incremental *on-chain* con Poseidon2 como función de hash y profundidad fija de 10 niveles (capacidad para hasta 1024 hojas). Sus operaciones principales son:

- startNewPeriod(uint64 period): inicia un nuevo período, reseteando el árbol al estado vacío.
- insertLeaf(bytes32 leaf): inserta un *commitment* en la siguiente posición disponible y actualiza la raíz.
- currentRoot() → bytes32: retorna la raíz vigente del árbol.
- currentPeriod() → uint64: retorna el período activo.

El operador es el único autorizado a llamar estas funciones. Nótese que el contrato almacena únicamente *commitments* criptográficos, no datos personales: no existe información en el estado *on-chain* que, por sí sola, permita identificar a qué persona corresponde cada hoja del árbol.

Contrato LocalVerifier. LocalVerifier es un contrato independiente por cada punto de verificación. Su función principal es *verifyAndConsume(bytes proof, bytes32[] publicInputs, bytes32 nullifier)*, que verifica la prueba zk contra la raíz actual de EligibleIMT y, si es válida, registra el nullifier como consumido para el período vigente. Si el nullifier ya fue registrado, la transacción revierte con *NullifierAlreadyUsed*.

Circuito Noir. El circuito, implementado en Noir, recibe como entradas privadas el secreto (*secret*) y el nullifier del beneficiario, y como entradas públicas la raíz del árbol y el path de inclusión. El circuito verifica que:

1. *commitment* = Poseidon2(*secret*, *nullifier*) es una hoja del árbol con la raíz provista.
2. El path de inclusión es consistente con la raíz.

Las entradas privadas nunca abandonan el dispositivo del beneficiario: solo la prueba resultante —un objeto de tamaño constante sin contenido semántico legible— es enviada al contrato. Esta separación entre datos privados y prueba pública es el núcleo del modelo de privacidad del sistema.

Credencial del beneficiario (QR). El operador genera para cada beneficiario un par aleatorio (*secret*, *nullifier*), calcula *commitment* = Poseidon2(*secret*, *nullifier*) e inserta el *commitment* en EligibleIMT. El par se entrega al beneficiario como un código QR. Crucialmente, este par no es almacenado por ningún servidor: el sistema no posee una tabla de correspondencia entre beneficiarios y credenciales. El beneficiario es el único custodio de su secreto, lo que elimina la posibilidad de que un tercero lo revele, lo venda o lo use en su nombre sin consentimiento.

3.3 Flujo de verificación

1. El beneficiario presenta su código QR al verificador.
2. El verificador lee (*secret*, *nullifier*) del código QR.
3. El frontend calcula el *commitment* y consulta al Merkle Path Server para obtener el path de inclusión.
4. Con estos datos, el frontend genera la prueba zk usando el circuito Noir compilado.
5. El frontend llama a *LocalVerifier.verifyAndConsume(proof, [root], nullifier)*.
6. El contrato verifica la prueba *on-chain* y, si es válida y el nullifier no fue usado, registra el uso y emite un evento de verificación exitosa.

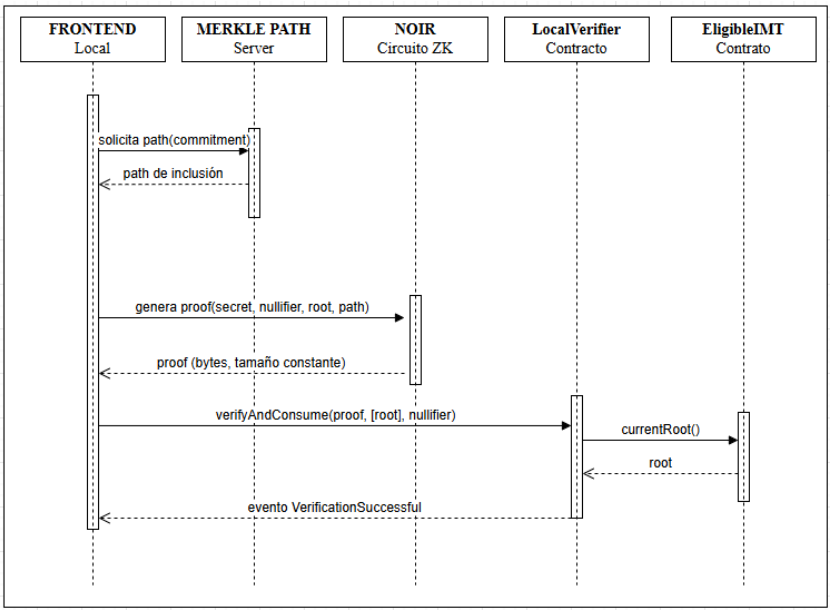

Fig. 2. Diagrama de secuencia del flujo de verificación *on-chain*.

3.4 Merkle Path Server

Para construir la prueba, el verificador necesita el *path* o ruta de inclusión del *commitment* dentro del árbol. El Merkle Path Server es un servicio auxiliar que escucha los eventos *LeafInserted* de EligibleIMT, reconstruye el árbol en memoria y expone un endpoint HTTP `/merkle-proof?commitment=0x...` con el *path* correspondiente.

Este servidor no introduce confianza adicional en el sistema: un *path* incorrecto produce una prueba inválida que el contrato rechaza. La corrección del *path*

es verificada criptográficamente por el circuito, por lo que el servidor puede ser reemplazado o replicado sin afectar las garantías de seguridad ni de privacidad del protocolo.

4 Implementación

4.1 Stack tecnológico

Table 1. Stack tecnológico del sistema ZkResident

Componente	**Tecnología**
Circuito zk	Noir 1.0.0-beta.17 (Aztec Labs)
Verificador on-chain	Barretenberg 0.58.0
Contratos inteligentes	Solidity 0.8.24
Hash on-chain	Poseidon2
Backend	Node.js v20, Express 5.2.1, ethers.js 6.16.0
Frontend	React 18.3.1, Vite 5.4.21, ethers.js 6.15.0
Red local (testing)	Anvil (Foundry v1.5.0)

4.2 Compatibilidad criptográfica entre capas

Uno de los desafíos más relevantes durante la implementación fue garantizar que la función Poseidon2 produjera resultados idénticos en las tres capas del sistema: el contrato Solidity, el código JavaScript del frontend y el circuito Noir. Una inconsistencia en cualquiera de estas capas implicaría que los *commitments* calculados *off-chain* no coincidieran con las hojas insertadas *on-chain*, haciendo al sistema inutilizable.

Se desarrollaron tres scripts de verificación independientes para confirmar esta compatibilidad:

- *verify-poseidon-compatibility.js*: compara *Poseidon2Hasher.hash()* (Solidity) contra *@zkpassport/poseidon2* (JavaScript).
- *verify-noir-js-poseidon2.js*: compara *poseidon2Hash* en *bb.js* (Barretenberg) contra la implementación JavaScript.
- *test-municipio-flow.js*: prueba el flujo completo de generación de *commitment* e inserción en EligibleIMT, verificando consistencia de la raíz resultante.

Los tres scripts confirmaron compatibilidad completa entre las implementaciones, condición necesaria para que el sistema funcione de extremo a extremo.

4.3 Prevención de doble gasto

El mecanismo de nullifiers está implementado en LocalVerifier mediante un mapping anidado que asocia cada período con el conjunto de nullifiers consumidos:

```
mapping(uint64 => mapping(bytes32 => bool))
    public usedNullifiersByPeriod;
```

Cuando una verificación es exitosa, el contrato registra el nullifier:

```
usedNullifiersByPeriod[currentPeriod][nullifier] = true;
```

El nullifier es un valor derivado del secreto del beneficiario que no revela a qué *commitment* corresponde. Un observador externo puede constatar que algún beneficiario consumió su derecho en ese período, pero no puede correlacionar ese evento con la inserción original ni con ningún otro uso en otro local. Al inicio de un nuevo período, los nullifiers anteriores dejan de ser relevantes, permitiendo que el mismo QR sea reutilizado si el beneficiario continúa siendo elegible.

5 Análisis de seguridad y privacidad

5.1 Privacidad del beneficiario

El sistema garantiza que un verificador que realiza una validación exitosa no obtiene ninguna información sobre la identidad del beneficiario. Lo único que el contrato registra es que un nullifier fue consumido: un valor que no puede ser correlacionado con el *commitment* ni con ningún identificador personal sin conocer el secreto privado del beneficiario.

Esta propiedad tiene implicancias que van más allá de la seguridad técnica. En los sistemas convencionales, cada acto de verificación deja un rastro asociado a la identidad del usuario. Ese rastro puede ser explotado para construir perfiles de comportamiento, inferir patrones de consumo, ajustar precios de forma discriminatoria o ser vendido a terceros sin que el usuario tenga mecanismos de control. En ZkResident, no existe rastro vinculable: cada verificación es criptográficamente indistinguible de cualquier otra desde el punto de vista de un observador externo (Goldwasser et al., 1989; Chaum, 1985).

5.2 Desvinculación de transacciones

Dos usos del mismo QR en distintos comercios producen nullifiers distintos y sin relación observable entre sí. Incluso si un atacante controlara múltiples puntos de verificación, no podría determinar que las pruebas recibidas provienen del mismo beneficiario. Esta propiedad de no vinculación es fundamental en contextos donde la correlación de eventos podría revelar información sensible (frecuencia de uso, distribución geográfica, hábitos temporales).

5.3 Resistencia a brechas de datos

Dado que el sistema no almacena datos personales en ninguna capa —ni *on-chain* ni en servidores auxiliares—, una brecha de seguridad en cualquier componente no expone información sobre los beneficiarios. El estado *on-chain* contiene únicamente hashes de compromisos; el Merkle Path Server almacena únicamente hojas del árbol (también hashes); y el secreto del beneficiario reside exclusivamente en su dispositivo. Este modelo se alinea con el principio de *data minimization* establecido en regulaciones de protección de datos como el RGPD [4], que establece que solo deben recopilarse los datos estrictamente necesarios para el fin declarado.

5.4 Solidez (*soundness*)

La solidez del esquema está garantizada por el sistema de pruebas subyacente (Barretenberg / UltraPlonk) **aztec-labs**. Un beneficiario que no posea un par $(secret, nullifier)$ cuyo *commitment* sea una hoja del árbol vigente no puede generar una prueba válida que el circuito acepte, salvo con probabilidad despreciable. Esto cierra el vector de ataque de presentar pruebas falsas para obtener beneficios indebidos.

5.5 Integridad del árbol

El árbol Merkle *on-chain* es inmutable a nivel de estado: una vez que una hoja es insertada, no puede modificarse ni eliminarse. El operador puede iniciar un nuevo período (reseteando el árbol), pero no puede alterar el historial de inserciones, que queda registrado en los eventos *on-chain* de forma permanente y auditable por cualquier parte.

5.6 Confianza en el Merkle Path Server

El servidor de *paths* es un componente auxiliar que no altera las garantías de seguridad ni de privacidad del protocolo. Un *path* incorrecto produce una prueba inválida que el contrato rechaza. El servidor puede ser reemplazado, replicado o eliminado sin comprometer el sistema. En particular, el servidor no aprende nada sobre la identidad del beneficiario al responder una consulta: solo conoce el valor del *commitment* solicitado, que es un hash sin información personal.

5.7 Custodia del QR

El secreto del beneficiario no es almacenado por ningún servidor del sistema. Esto implica que la pérdida del QR equivale a la pérdida del derecho para ese período, lo que constituye una decisión de diseño que prioriza la privacidad en detrimento de la recuperabilidad: un sistema que almacene el secreto para recuperación introduciría necesariamente un punto centralizado de confianza, erosionando las garantías descritas en esta sección.

[4] Reglamento General de Protección de Datos de la Unión Europea, aplicable como referencia de buenas prácticas también fuera de su jurisdicción formal.

6 Discusión

6.1 La privacidad como ventaja estructural

En el diseño de sistemas de verificación, la privacidad suele tratarse como un requisito regulatorio a cumplir, más que como una propiedad técnica que aporta valor en sí misma. ZkResident propone una perspectiva diferente: la privacidad puede ser una ventaja estructural que otorga un beneficio a todos los actores del sistema.

Para el beneficiario, supone independencia frente a actores que de otro modo podrían usar su historial de verificaciones como insumo para tomar decisiones que lo afectan —desde la fijación de precios hasta la denegación de servicios—. Para el operador, elimina la responsabilidad legal y operativa de custodiar datos personales sensibles, reduciendo su superficie de ataque. Para el verificador, simplifica el cumplimiento normativo al no acumular información que no necesita. Y para el sistema en su conjunto, la ausencia de datos centralizados elimina el incentivo económico de atacarlo.

Esta alineación de incentivos es característica de los sistemas diseñados con privacidad por defecto Lavin et al., 2024, donde la protección de datos no se agrega como capa posterior sino que emerge de las propiedades fundamentales del protocolo.

6.2 Aplicabilidad del modelo

Aunque ZkResident fue desarrollado en el contexto de programas de beneficios, el modelo es aplicable a cualquier escenario donde sea necesario verificar que un usuario pertenece a un conjunto sin revelar cuál. Ejemplos directos incluyen programas de fidelidad, acceso a servicios premium, votaciones restringidas a miembros, o control de acceso a eventos. En todos estos casos, el patrón común es el mismo: existe una condición binaria (“pertenece” / “no pertenece”) cuya verificación no requiere revelar la identidad de quien la cumple.

6.3 Escalabilidad

El árbol Merkle de profundidad 10 soporta hasta 1024 beneficiarios por período. Esta capacidad puede incrementarse ajustando la profundidad, con el costo de aumentar el tamaño del circuito y el tiempo de generación de pruebas. Para casos de uso con mayor volumen de usuarios, arquitecturas basadas en inserciones por lotes o árboles de mayor profundidad son vías de extensión viables.

6.4 Limitaciones actuales

La principal limitación es la generación de pruebas en el lado del cliente, que puede ser costosa en tiempo para dispositivos de baja capacidad. El uso de sistemas de prueba más rápidos o de pruebas delegadas a un servidor de confianza son alternativas a explorar, aunque esta última opción introduce un componente

centralizado que debe evaluarse cuidadosamente en función del modelo de amenaza.

Adicionalmente, el sistema no verifica que el par $(secret, nullifier)$ fue generado de forma honesta por el operador. Un operador malicioso podría generar múltiples credenciales para un mismo beneficiario. Este aspecto se aborda como trabajo futuro mediante el uso de credenciales verificables (VCs) como capa adicional de autenticación, lo que permitiría vincular el proceso de emisión a una identidad verificada sin comprometer la privacidad en el momento de la verificación.

7 Conclusiones y trabajo futuro

Este trabajo presentó ZkResident, un protocolo basado en pruebas de conocimiento cero para la verificación de elegibilidad en entornos descentralizados. El sistema permite que un beneficiario demuestre su pertenencia a un conjunto de elegibles sin revelar su identidad, y que un verificador autónomo pueda confirmar esa pertenencia *on-chain* sin depender de infraestructura centralizada.

La principal contribución técnica es la integración coherente de criptografía programable (Noir), contratos inteligentes (Solidity/Foundry) y componentes de frontend, garantizando compatibilidad en la función Poseidon2 a través de todas las capas del sistema. El mecanismo de nullifiers por período permite reutilizar el mismo árbol de elegibles a lo largo del tiempo con control de doble gasto por contexto, sin revelar la identidad del beneficiario en ningún paso del proceso.

Desde la perspectiva de la ciberseguridad, la principal contribución consiste en evidenciar que es posible construir un sistema de verificación funcional que minimice la acumulación de datos personales a lo largo de su arquitectura, reduciendo así la disponibilidad de información susceptible de ser objeto de robo, correlación o comercialización no autorizada. La privacidad no es aquí un objetivo secundario, sino una consecuencia directa de las propiedades matemáticas del protocolo.

El sistema fue validado en un entorno local con un stack completo, incluyendo la verificación de compatibilidad criptográfica entre implementaciones. Como trabajo futuro, se propone explorar la reducción del tiempo de generación de pruebas en dispositivos de baja capacidad, la incorporación de credenciales verificables como capa de autenticación del operador, y la adaptación del protocolo a redes de producción con múltiples operadores y verificadores.

Bibliografía

Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Succinct non-interactive zero knowledge for a von neumann architecture. *Proceedings of the 23rd USENIX Security Symposium*, 781–796.

Camenisch, J., & Lysyanskaya, A. (2001). An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Advances in Cryptology — EUROCRYPT 2001*.

Chaum, D. (1985). Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*.

Community. (2025). Noir language documentation. Retrieved January 1, 2025, from https://noir-lang.org/docs

Goldwasser, S., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, *18*(1), 186–208.

Grassi, L., Khovratovich, D., & Schofnegger, M. (2023, March). Poseidon2: A faster version of the poseidon hash function [IACR ePrint Archive. Visitado: 2025].

Hannak, A., Soeller, G., Lazer, D., Mislove, A., & Wilson, C. (2014). Measuring price discrimination and steering on e-commerce web sites. *Proceedings of the 2014 Conference on Internet Measurement Conference*, 305–318. https://doi.org/10.1145/2663716.2663747

Hopwood, D., Bowe, S., Hornby, T., & Wilcox, N. (2016). *Zcash protocol specification*. Retrieved January 1, 2025, from https://zips.z.cash/protocol/protocol.pdf

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, *110*(15), 5802–5805. https://doi.org/10.1073/pnas.1218772110

Lavin, R., Liu, X., Mohanty, H., Norman, L., Zaarour, G., & Krishnamachari, B. (2024). A survey on the applications of zero-knowledge proofs. *arXiv preprint arXiv:2408.00243*.

MACI Project. (2025). *Minimal anti-collusion infrastructure*. Retrieved January 1, 2025, from https://maci.pse.dev

Merkle, R. C. (1988). A digital signature based on a conventional encryption function. *Advances in Cryptology – CRYPTO '87*, 369–378.

Narayanan, A., & Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. *2008 IEEE Symposium on Security and Privacy*.

Polygon ID. (2025). *Polygon id documentation*. Retrieved January 1, 2025, from https://0xpolygonid.github.io/tutorials

Semaphore Community. (2025). *Semaphore documentation*. Retrieved January 1, 2025, from https://semaphore.appliedzkp.org

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. PublicAffairs.