Identity-Private Healthcare Data Sharing on Blockchain via Zero-Knowledge Proofs and Account Abstraction

Kun Li, Ankur Lohachab, and Visara Urovi
Department of Advanced Computing Sciences
Maastricht University
Maastricht, The Netherlands
{k.li, ankur.lohachab, v.urovi}@maastrichtuniversity.nl

***Abstract*—Blockchains are considered for healthcare data sharing due to their immutability, decentralization, and auditability. However, ledger transparency exposes on-chain identifiers and activity metadata, enabling linkage across pseudonyms and inference over user behavior. Prior work has primarily focused on content confidentiality and access control, while leaving identity unlinkability insufficiently addressed. To this end, we present an approach that integrates Account Abstraction (AA), zero-knowledge proofs (Groth16), and Pedersen commitments. The approach embeds proof- and commitment-based verification into programmable smart contract accounts (SCAs), enabling authentication without disclosing identifiers and decoupling transactions from static keys. We develop a proof-of-concept on the Polygon Amoy testnet using `Circom` and `Solidity`, and evaluate privacy under a *global, passive, external, static, and computationally bounded* attacker. For the ERC-4337 comparison, the attacker is assumed to know user–SCA mappings; for the account-shuffling comparison, the attacker knows one SCA per user. Using entropy metrics and clustering-based inference over on-chain metadata, our approach achieves the maximum entropy of $\log_2(10) \approx 3.32$ in a ten-user setting (versus 0 for ERC-4337 as specified, i.e., without privacy extensions) and substantially reduces clustering accuracy relative to address shuffling (ARI $0.468 \rightarrow 0.038$, NMI $0.653 \rightarrow 0.177$), while maintaining the auditability required for healthcare governance.**

***Index Terms*—Blockchain, Identity Privacy, Zero-Knowledge Proofs, Account Abstraction, Healthcare Data Sharing**

I. Introduction

The digitization of healthcare demands secure and interoperable data sharing [1]–[3]. Blockchain offers decentralized and immutable healthcare data sharing but simultaneously exposes user identities and metadata, raising serious privacy concerns under regulations like the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR) [4]. Existing blockchain-based privacy-preserving methods—such as encryption and access control—focus on transaction content [5], [6], yet often rely on externally owned accounts (EOAs), leaving identity traces visible on-chain [7]–[12]. Consequently, adversaries can perform attacks, for instance, inference attacks based on blockchain activities, compromising patient privacy [13], [14].

This identity and metadata exposure poses significant risks in healthcare, where data sensitivity and regulatory compliance are critical [15]. Beyond the immediate privacy threats to individuals, linkage of on-chain identifiers and metadata can enable longitudinal profiling of patients, re-identification of de-identified records, and discriminatory treatment based on inferred health conditions [16]. Such inferences may reveal not only disease status but also treatment timelines, care providers, and patterns of service usage, thereby exposing patients to social stigma, insurance discrimination, or undesired surveillance by third parties [17]. Furthermore, regulatory frameworks (e.g., GDPR, HIPAA) impose obligations on data controllers and processors to prevent re-identification and unjustified data disclosures; failures at the identity and metadata level can therefore translate into legal and compliance risks for healthcare platforms [15].

Despite advances in privacy-preserving techniques, current works lack identity unlinkability and behavioral untraceability, leaving users vulnerable to inference attacks [13], [14]. While mitigation strategies, such as *account shuffling*, where users cycle through multiple accounts to obscure behavioral linkability [18], [19], offer partial obfuscation, they lack formal unlinkability guarantees and can be defeated by advanced clustering techniques leveraging on-chain metadata.

This raises the question: How can blockchain-based healthcare data sharing preserve user identity privacy without sacrificing transparency and auditability? To this, while prior efforts secured data content and access policies, few tackled the challenge of unlinking user identities from on-chain behavior [7], [8], [11], [20].

Though, as a more recent solution, Account Abstraction (AA) [21] replaces externally owned accounts (EOAs) with programmable smart contract accounts (SCAs). Unlike EOAs, which are rigidly bound to a single ECDSA key pair, SCAs support custom verification logic, including zero-knowledge proof (ZKP)-based authentication [22]. This flexibility breaks the deterministic link between users and long-term static addresses, thereby has the potential to disrupt traceability and resisting inference. In the healthcare context, AA has been explored for accountability and compliance [23], providing a foundation for verifiable participation while enabling unlinkability. However, AA by itself does not ensure identity

unlinkability or behavioral untraceability, as on-chain metadata and transaction patterns remain vulnerable to inference attacks.

Building on this, we propose a novel privacy-preserving approach that combines AA, Groth16-based ZKPs, and Pedersen commitment schemes [24]. Our approach eliminates the reliance on EOAs, enabling anonymous, unlinkable, and verifiable user identities through programmable SCAs. Through entropy-based anonymity analysis and behavioral clustering simulations, we show that the proposed method mitigates identity traceability and protects against inference attacks, outperforming the ERC-4337 [21] and account shuffling baselines.

Specifically, this work makes the following key contributions:

- We propose a novel privacy-preserving approach that integrates Account Abstraction, Groth16-based ZKPs, and Pedersen commitments, enabling unlinkable and verifiable identities for healthcare data sharing.
- We develop and deploy a prototype implementation on the Polygon Amoy testnet using `Circom` and `Solidity`, demonstrating the practicality of our approach in a blockchain environment.
- We conduct an empirical evaluation using entropy-based anonymity analysis and behavioral clustering, showing that our approach substantially improves resistance to identity traceability and inference attacks compared to default AA and account-shuffling baselines.

The remainder of this paper is structured as follows: Section II reviews technical background and related work; Section III introduces the proposed approach and Section IV details its implementation; Section V presents the evaluations settings and results. Section VI discusses the findings, limitations, and future directions. Finally, Section VII concludes the paper.

II. Background

This section outlines the foundations of our approach: Account Abstraction, zero-knowledge proofs, and Pedersen commitments.

A. Account Abstraction

AA, formalized in ERC-4337 [21], replaces EOAs with programmable SCAs. In EOA-based systems, all transactions are bound to a single ECDSA key pair, creating a static identity trace. AA introduces *UserOperations (UOs)*, which are submitted by *Bundlers* to the `EntryPoint` contract for validation and execution. This indirection enables SCAs to implement custom authentication logic beyond ECDSA, such as multi-signatures, social recovery, or ZKP-based verification. The programmability of AA enables embedding privacy-preserving authentication directly into account logic, which we leverage to achieve unlinkable and verifiable participation.

B. Zero-Knowledge Proofs

ZKPs allow a prover to convince a verifier of the validity of a statement without revealing the underlying witness [22]. A ZKP protocol satisfies three properties: *completeness* (an honest prover convinces the verifier), *soundness* (a dishonest prover cannot convince the verifier of a false claim), and *zero-knowledge* (the verifier learns nothing beyond the validity of the claim). In practice, we rely on Groth16 [25], a zk-SNARK construction that generates constant-size proofs and supports efficient on-chain verification. These properties make Groth16 well-suited for healthcare applications, where attributes such as patient identity, consent status, or regulatory clearance must be verified succinctly and without disclosure.

C. Pedersen Commitments

Commitment schemes [24] allow a value to be fixed at one point in time and later revealed or proven without changing it. A secure scheme must satisfy *hiding* (the committed value remains secret) and *binding* (the committer cannot open the commitment to a different value). In our construction, We adopt a hash-based commitment instantiated with a SNARK-friendly Pedersen hash. While this construction is collision-resistant and computationally hiding for high-entropy secrets, it is not additively homomorphic like classical elliptic-curve Pedersen commitments. Our security relies on the hash's collision resistance and on choosing a sufficiently high-entropy secret with domain separation.

D. Related Work

Blockchain-based healthcare data sharing has been studied extensively, with most approaches emphasizing *transaction-content confidentiality*. Typical strategies include encryption, fine-grained access control, or hybrid on/off-chain storage (e.g., BlocHIE [7], MedBloc [9], Hocbs [8]). While effective in protecting medical data, these systems are built on externally owned accounts (EOAs), which inherently expose persistent addresses and interaction metadata. This design choice makes user activities linkable across transactions. Recent work has introduced zero-knowledge–based mechanisms to strengthen privacy. zkPass [26] enables verification of off-chain credentials, Semaphore [27] provides anonymous signaling primitives, and zkLogin [28] supports blockchain authentication with existing credentials. These systems address specific privacy aspects but do not simultaneously achieve identity unlinkability, resistance to behavioral inference, and integration with account abstraction, which limits their suitability for healthcare data sharing. Table I summarizes representative approaches along four privacy objectives: *identity unlinkability* (preventing persistent address exposure), *behavioral untraceability* (resisting inference from metadata such as gas usage or timing), *transaction-content confidentiality* (restricting unauthorized access to data), and *on-chain verifiability* (ensuring enforcement is auditable through smart contracts). None of the prior approaches achieve all four objectives simultaneously. In particular, identity unlinkability and behavioral untraceability remain unaddressed across existing designs. Our approach closes this gap by integrating account abstraction, zero-knowledge proofs, and Pedersen commitments to jointly realize unlinkability, anonymity, confidentiality, and verifiability.

TABLE I
COMPARISON ACROSS REPRESENTATIVE BLOCKCHAIN-BASED PRIVACY-PRESERVING HEALTHCARE DATA SHARING SOLUTIONS. ONLY "✓"/"✗" JUDGMENTS ARE SHOWN FOR CLARITY.

Papers	Identity privacy	Behavioural-linkability	Transaction-content privacy	On-chain verifiability
Jiang et al., 2018 [7]	✗	✗	✓	✓
Zhou et al., 2019 [29]	✗	✗	✓	✓
Miyachi and Mackey,2021 [8]	✗	✗	✓	✓
Huang et al., 2019 [9]	✗	✗	✓	✗
Hussien et al., 2021 [10]	✗	✗	✓	✓
Zhang et al., 2018 [30]	✗	✗	✓	✗
Sun et al., 2024 [12]	✗	✗	✓	✗
Urovi et al., 2022 [31]	✗	✗	✓	✗
Sarode et al., 2022 [11]	✗	✗	✓	✗
zkPass Team, 2025 [26]	✗	✗	✓	✓
PSE Team, 2021 [27]	✓	✗	✗	✓
Baldimtsi et al., 2024 [28]	✗	✗	✗	✓
Our approach	✓	✓	✓	✓

III. PROPOSED APPROACH

Our approach addresses the challenge of preserving identity unlinkability in blockchain-based healthcare data sharing while mitigating inference attacks.

A. Overview

As described in Section II, our solution integrates three complementary components:

- **AA:** EOAs are replaced with programmable SCAs, which embed custom verification logic and decouple transaction execution from static identities.
- **ZKPs:** Users generate succinct proofs of knowledge of hidden attributes (e.g., identity credentials or consent) without revealing them, enabling efficient privacy-preserving authentication.
- **Pedersen Commitments:** Users commit to a secret value that remains hidden but binding, allowing ZKPs to be constructed over committed data while preserving secrecy.

B. Modules

To support secure and unlinkable participation in healthcare data sharing, our approach is organized into four modules. Table II summarizes the four modules, their responsibilities, and main components.

1) Interface: This module acts as the entry point for all user interactions. It is responsible for: 1) generating commitment values; 2) constructing ZKPs, and 3) assembling the required data for registration and transaction execution. By generating all privacy-critical data off-chain, the interface ensures that sensitive user information is never exposed during communication with the blockchain.

2) User Management Module (UMM): This module governs identity registration and verification. It incorporates three smart contracts: **Registry Smart Contract (RegC)**, **Commitment Verifier Contract (CVC)**, and **Registration Verifier Contract (RVC)**.

Registry Smart Contract (RegC). The RegC serves as a global directory of user registrations. Each new registration generates a *registration commitment*, constructed from the user's SCA address to attest to the registration of that address. This commitment is inserted into a Merkle tree, whose root is maintained on-chain by RegC. By storing only commitments rather than SCA addresses, RegC mitigates the risk of personal identity exposure. Registered users can later prove they are registered by providing a zero-knowledge Merkle membership proof. The contract exposes two primary functions:

- `registerUser(_registration_commitment)`: Inserts a registration commitment into the Merkle tree and emits a `UserRegistered` event with the leaf index.
- `verify(_proof)`: Validates the zero-knowledge Merkle membership proof, confirming that a given commitment is present in the tree without revealing the actual user data.

Commitment Verifier Contract (CVC). The CVC handles zero-knowledge proof verification for user secrets. Whenever a SCA needs to confirm ownership, it invokes the CVC with a proof that demonstrates knowledge of the preimage (secret) behind its on-chain commitment. The `verifyProof(_proof)` in CVC method checks a user's zero-knowledge proof attesting that the user-supplied secret matches the stored commitment, without revealing the secret itself.

Registration Verifier Contract (RVC). The RVC provides on-chain functions to verify proofs of registration. When users interact with certain system features, they are required to prove that their SCA address is already registered in `RegC`. The `verifyProof(_proof)` method in RVC checks whether a given Merkle proof is valid against the current `RegC` root, thereby ensuring only authenticated SCAs can participate in operations.

3) Account Abstraction Module (AAM): This module enables users to manage transactions without EOAs by leveraging AA. It comprises the following components:

- **Smart Account Factory Contract (SAFC):** Deploys SCAs embedded with the user's commitment and ZKP-based validation logic.

TABLE II
SUMMARY OF SYSTEM MODULES AND THEIR KEY RESPONSIBILITIES.

Module	Key Responsibilities	Main Components
Interface	Generate commitments and zero-knowledge proofs off-chain; assemble registration and transaction data for submission.	Client-side tool
User Management Module (UMM)	Handle identity registration and verification; maintain registry of commitments; validate Merkle membership proofs.	**RegC**, **CVC**, **RVC**
Account Abstraction Module (AAM)	Deploy SCAs with embedded ZKP-based validation logic; manage user operations within AA framework.	**SAFC**, **RFC**, per-user **SCAs**
Runner Management Module (RMM)	Coordinate execution of user operations; verify registration proofs and enforce authorization logic.	**RCs**

- **Runner Factory Contract (RFC):** Deploys Runner Contracts (RCs), which validate and execute user-submitted operations in compliance with AA.
- **AA handler:** Manages communication with the EntryPoint contract in AA and coordinates UO submission.

By embedding ZK logic within SCAs and decoupling execution into RCs, the module enables programmable, verifiable, and unlinkable transaction processing.

Smart Contract Account (SCA). Each user controls a SCA that includes:

1) *Commitment*: A Pedersen hash of the user's off-chain secret.
2) Custom `validateUserOp()` logic, compatible with AA to verify a ZK proof in place of a traditional ECDSA signature.

SCAs are deployed through the SAFC, ensuring consistent initialization and secure creation process. In combining with the registration mechanism provided by RegC, an SCA can participate in transactions without revealing the underlying user identity. Furthermore, by replacing ECDSA signatures with proof-based ownership checks, the SCA safeguards user anonymity and prevents direct linkage between transactions and personal identities.

Runner Contract (RC). The RC is a specialized SCA deployed via RFC by the administrator. It is responsible for executing aggregated UOs within the AA framework and performs the following functions:

- **Operation Validation**: Implements a custom `validateUserOp()` function that checks a zero-knowledge proof in place of a traditional digital signature before executing the transaction payload.
- **User Operation Execution**: Bundlers submit UOs to the EntryPoint contract, which delegates execution to the RC once the proof is validated.

4) Runner Management Module (RMM): This module is responsible for coordinating the execution of UOs. When a UO is submitted, it is forwarded to a RC, which verifies the accompanying registration proof and, if valid, proceeds to execute the operation. This mechanism allows user identities to be authenticated without revealing sensitive information, while enforcing authorization logic on-chain.

5) Component Relationships: Figure 1 illustrates the component relationships used in our approach. Off-chain components (Interface and Bundler) communicate with the on-chain architecture consisting of the SAFC, per-user SCAs, the RegC, verifiers and the RC. The SAFC initializes SCAs with user commitments, and the RegC records registration commitments for subsequent inclusion proofs. Unlike traditional EOAs, user operations are never submitted directly from user-bound addresses. Instead, they are processed via the Bundler and EntryPoint, which forward execution to the RC. The RC verifies registration and ownership proofs. Because all transactions appear to originate from the RC, no persistent user identifier is exposed, ensuring unlinkability and behavioral untraceability.

Algorithm 1 Merkle Membership Proof Generation (as implemented)

Require: Secret s, SCA address $addr_{SCA}$, nullifier ν, Registry Contract **RegC**
Ensure: Merkle membership proof p_{mm}

```
c_r ← H_Ped(s ‖ addr_SCA ‖ ν) {compute registration commitment}
(root, path, index) ← RegC.getProof(c_r) {obtain Merkle proof from registry}
if path = ∅ then
    throw "Commitment not found in registry"
end if
public ← { root } {public input to circuit}
private ← { s, addr_SCA, ν, path, index } {private inputs to circuit}
p_mm ← prove(public, private, "registration_circuit") {generate Groth16 proof}
return p_mm
```

C. System Workflows

Our approach operates through a three-phase workflow that governs how users securely register, authenticate, and interact with the blockchain without exposing their identities.

1) Commitment and Registration: The workflow begins with user onboarding. Each user selects a private secret s that serves as the basis for future authentication and registration. Two cryptographic commitments are computed:

- The **user commitment** $c_u = \text{hash}(s)$, which is submitted to the SAFC by the administrator to initialize a SCA embedded with the commitment.

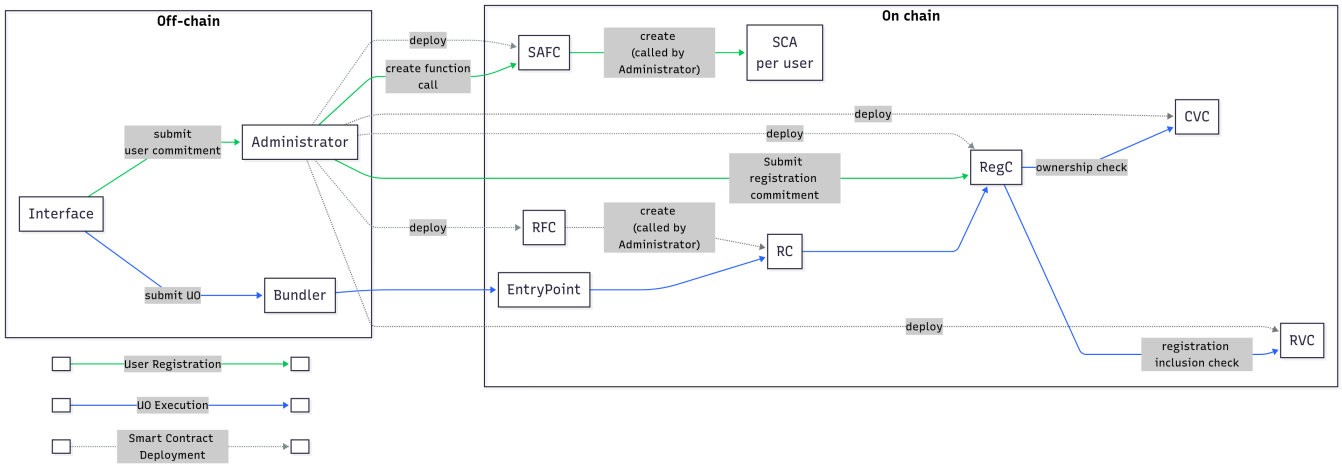

Fig. 1. Component relationships among smart contracts and off-chain entities in our approach. User operations are always executed via the RC, ensuring unlinkability.

- The **registration commitment** $c_r = \text{hash}(s, \text{addr}_{\text{SCA}})$, which binds s (secret) to $addr_{SCA}$ (the deployed SCA address). This commitment is submitted to the RegC, which inserts it into the Merkle Tree representing the registry of authorized users.

In this two-level commitment mechanism, the **user commitment** conceals the user's secret, while the **registration commitment** preserves the privacy of the user's SCA. Together, these commitments enable verifiable registration and support future zero-knowledge authentication.

This process is illustrated in Figure 2, where the user computes a commitment and deploys a SCA containing the commitment. Subsequently, Figure 3 depicts the submission of the registration commitment to the RegC and its insertion into the on-chain Merkle Tree.

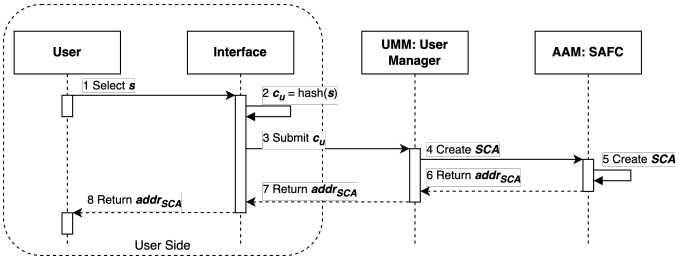

Fig. 2. Illustration of SCA creation using the user commitment c_u

2) Registration Verification: To authenticate without revealing identity details, a user generates a Merkle membership proof p_{mm} demonstrating that their registration commitment c_r is included in the Merkle tree maintained by the RegC. The process is shown in Algorithm 1. Off-chain, the user recomputes

$$c_r = \mathsf{H}_{\text{Ped}}(s \,\|\, addr_{SCA} \,\|\, \nu),$$

where s is the user's secret, $addr_{SCA}$ is the deployed smart contract account address, and ν is a user-chosen nullifier.

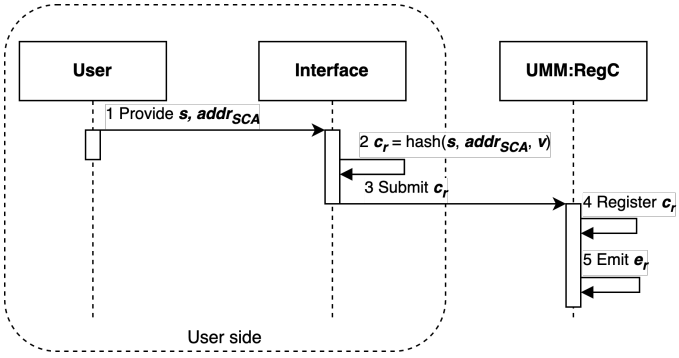

Fig. 3. Submission of the registration commitment $c_r = \text{hash}(s, \text{addr}_{\text{SCA}})$ to the RegC

Unlike randomized commitments that require storing an additional blinding factor, this construction relies only on reproducible inputs: the user can always recompute c_r locally from $(s, addr_{SCA}, \nu)$. The nullifier ν ensures that each commitment is unique and, if rotated, prevents replay of old registrations.

The user then queries RegC to obtain the current Merkle root, the corresponding path, and the leaf index. These values are supplied as private inputs (s, $addr_{SCA}$, ν, path, index) and public input ($root$) to the `registration_circuit`, which generates p_{mm}.

Finally, p_{mm} is submitted to the **RVC**, which verifies the proof against the root stored in RegC. This confirms that the user's SCA is registered without revealing s, $addr_{SCA}$, or ν. As shown in Figure 4, the user generates a Merkle membership proof and submits it to the RegC in UMM, which validates inclusion in the Merkle Tree.

3) User Operation Execution: To initiate an on-chain action (e.g., data request, consent management), the user constructs a UO that includes:

- The address of the RC,
- The intended function call,

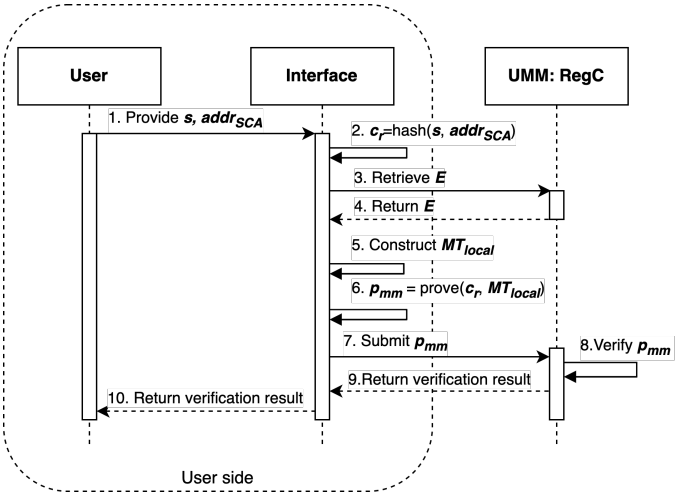

Fig. 4. Merkle membership proof verification to confirm a user's registration commitment is included in the RegC.

- A zero-knowledge proof that validates the user's registered identity, namely the Merkle membership proof p_{mm}.

A pre-verification step is performed by submitting p_{mm} to the RC, which records the verification status. The UO is then forwarded by the *AA Handler* to a bundler node, which subsequently submits it to the EntryPoint contract. The EntryPoint then forwards the UO to the RC for validation.

The RC invokes a custom `validateUserOp()` function that checks the embedded zero-knowledge proof and consults the pre-verification result. If the proof is valid, the requested function is executed; otherwise, the transaction is reverted.

This design eliminates the need for digital signatures or static user addresses, thereby breaking linkability between different transactions initiated by the same user.

The overall execution sequence is summarized in Figure 5, which shows how the UO is verified through zero-knowledge proofs and executed via the RC within the AA framework.

IV. Implementation

This section details the implementation of our approach, including the deployment environment, toolchain, and zero-knowledge circuits.

A. Deployment Environment and Toolchain

We implemented and tested the system on the Polygon Amoy test network (chain ID 80002), which is EVM-compatible and supports ERC-4337 account abstraction. This environment allows us to validate functional correctness and evaluate proof-based privacy mechanisms. Smart contracts are written in `Solidity` 0.8.0 and developed with `Hardhat` for compilation, deployment, and testing. Zero-knowledge circuits are implemented in `circom2` 0.2.19 and compiled to R1CS. We use `snarkjs` 0.7.5 with the Groth16 proving system [25] to generate proving/verification keys and Solidity verifier contracts. For SNARK-friendly primitives, we rely on `circomlib` 2.0.5 (Pedersen hash), and we use `circomlibjs` off-chain to compute leaves and prepare witness inputs before transactions are submitted.

B. Circuit Design

We implement two circuits that constitute the cryptographic core of our approach: a user-commitment circuit and a registration (Merkle membership) circuit. Both circuits use the same hash function and leaf format as the on-chain registry.

1) Commitment Circuit: This circuit proves knowledge of a secret s such that its Pedersen hash equals a published user commitment.

- **Private input:** s.
- **Public input:** $c_u = \mathsf{H}_{\mathrm{Ped}}(s)$.
- **Constraint:** recompute $\mathsf{H}_{\mathrm{Ped}}(s)$ and enforce equality with c_u.

2) Registration Circuit: This circuit proves that the registration commitment c_r is included in the Merkle tree maintained by RegC, enabling anonymous registration checks.

- **Private inputs:** s, $addr_{SCA}$, nullifier ν, Merkle path elements, leaf index.
- **Public input:** registry root $root$.
- **Constraint:** reconstruct

$$c_r = \mathsf{H}_{\mathrm{Ped}}(s \,\|\, addr_{SCA} \,\|\, \nu)$$

and verify that c_r is a leaf under $root$ via the supplied Merkle path and index.

Both circuits are compiled to R1CS with `circom2`; verifier contracts generated by `snarkjs` are deployed on-chain. Proofs are produced off-chain and attached to transactions for efficient on-chain verification with minimal disclosure.

3) Trusted Setup and Verifier Generation: The Groth16 setup proceeds in two phases:

1) *Circuit Compilation.* Each circuit is compiled into an R1CS format describing its constraints.
2) *Phase 1 (Powers of Tau).* An updatable universal ceremony produces a `.ptau` file that can be reused across circuits.
3) *Phase 2 (Circuit-Specific).* From the `.ptau` file, a proving key and verification key are generated for each circuit.
4) *Verifier Contract Generation.* The verification key is compiled into a Solidity verifier contract to enable on-chain proof verification.

Phase 1 is reusable across circuits. Phase 2 must be run for each circuit, but once completed, the resulting verifier and keys are reusable for all users of that circuit.

V. Evaluation and Results

We evaluate privacy under simulated adversarial conditions and quantify anonymity. We (i) specify the attacker model, (ii) define privacy metrics, and (iii) describe experimental settings and results comparing our approach against ERC-4337 as specified (i.e., without privacy extensions) and account shuffling.

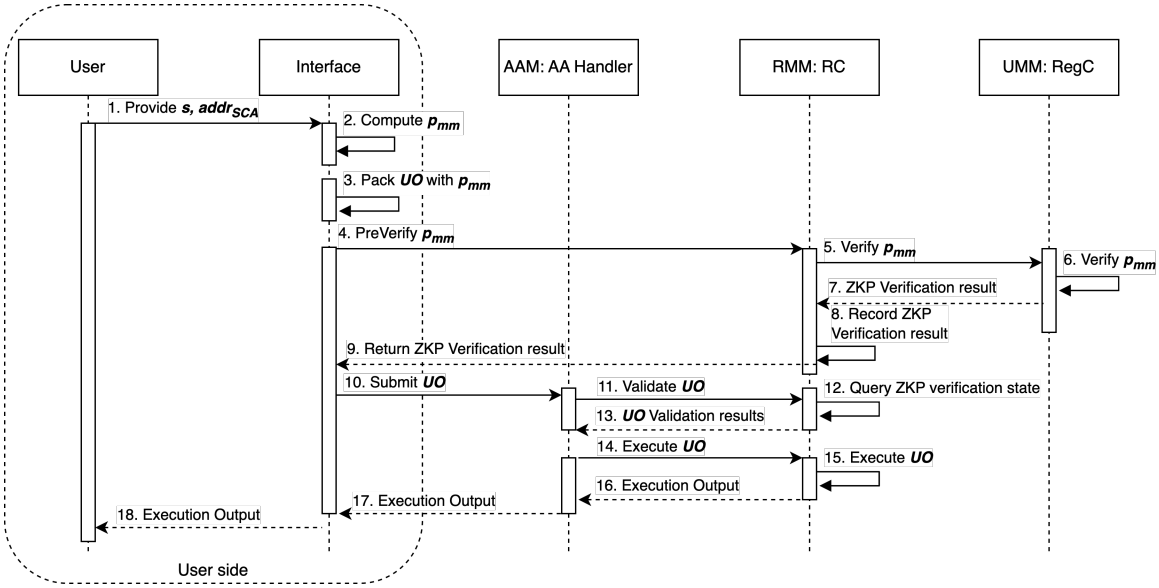

Fig. 5. Workflow of UO execution

A. Attacker Model

We adopt the taxonomy of Wagner and Eckhoff [32] and consider an attacker with the following capabilities:

- **Global.** The attacker has full visibility of the blockchain ledger, including all transactions, smart contract state, and emitted events.
- **Passive.** The attacker only observes; it does not inject, modify, or block transactions. Timing and gas-cost metadata are observable but no active probes are issued.
- **External.** The attacker is not a registered participant, holds no secrets, and cannot produce valid zero-knowledge witnesses.
- **Static.** The attacker relies on a fixed set of inference heuristics rather than adaptive learning during execution.
- **Computationally bounded.** The attacker is limited to polynomial-time analysis and cannot break underlying cryptographic primitives (e.g., commitments, Groth16 soundness).

Prior knowledge. We distinguish two settings corresponding to our evaluation scenarios: (i) **ERC-4337 setting.** The attacker is assumed to know the complete mapping between users and their SCAs, as well as the total number of users. (ii) **Account shuffling setting.** The attacker is assumed to know exactly one SCA for each user and the total number of users, but must infer additional addresses through clustering.

B. Metrics

We quantify privacy as the adversary's residual uncertainty under the leakage assumed in our attacker model (Section V-A). Following anonymity metrics (e.g., [33]), we use information-theoretic measures over the adversary's posterior for each transaction:

- **Shannon entropy** H: given a posterior $\{p_i\}$ over candidate users,
$$H \;=\; -\sum_i p_i \log_2 p_i,$$
which quantifies uncertainty in identifying the true originator.
- **Effective anonymity set size** 2^H: interprets H as the size of an equally likely set that induces the same entropy (the *scaled anonymity set size* [33]).
- **Anonymity set size**: the cardinality of the users that remain plausible under the adversary's view (independent of probability weights).

C. Simulation Settings

We instantiate the attacker model in two evaluation scenarios.

a) ERC-4337 (as specified) vs. Proposed Approach: We simulate ten users under two configurations:

- **ERC-4337 (as specified).** User operations follow the standard ERC-4337 signature-based flow via `EntryPoint` with no ZK/commitment extensions. The adversary is assumed to know the full user – SCA mapping and cohort size; hence, for each transaction the adversary's posterior places probability 1 on the true originator (entropy $H = 0$).
- **Proposed approach.** User operations are executed via the RC. Authentication uses a ZK membership proof against RegC; on-chain observations expose RC interactions rather than user-bound addresses.

Execution traces were collected from the Polygon Amoy testnet (chain ID 80002) via RPC and trace APIs; the adversary view is constructed from these traces and evaluated using the metrics in Section V-B.

b) Account Shuffling vs. Proposed Approach: We also simulate an account-shuffling configuration in which each user controls multiple SCAs and selects one uniformly at random per transaction. The adversary is assumed to know one SCA per user and the cohort size and attempts to recover additional addresses via clustering (privileged-operation heuristic). To keep evaluation tractable while preserving behavior heterogeneity, we simulate four users with distinct roles and workloads (Table III); each user executes the same workload under both configurations.

Clustering attack and entropy instantiation. Let C denote a cluster recovered by the adversary and let $a^* \in C$ be a known SCA of the target user U. Define $C' = C \setminus \{a^*\}$ as the set of candidate addresses whose ownership is unknown to the adversary. Under the *uniform prior* assumption (attacker has no additional contextual information beyond clustering), the adversary assigns equal probability to each $a_i \in C'$:

$$P(a_i \in A_U \mid C') \;=\; \frac{1}{|C'|}, \qquad \forall a_i \in C',$$

where A_U denotes the set of all addresses controlled by user U.

Applying Shannon entropy to this uniform posterior yields

$$H \;=\; -\sum_{i=1}^{|C'|} \frac{1}{|C'|} \log_2 \frac{1}{|C'|} \;=\; \log_2 |C'|.$$

Consequently the effective anonymity set size is

$$2^H \;=\; |C'|,$$

i.e., the number of indistinguishable candidate addresses from the adversary's perspective under the stated assumptions.

TABLE III
EXPERIMENT II USER CONFIGURATION (ACCOUNT SHUFFLING VS. PROPOSED).

User	Role	# SCAs (shuffling)	Transaction types
User 1	Analyst	9	Deploy (1); Update (9)
User 2	Doctor	9	Deploy (2); Update (9)
User 3	Insurer	6	Deploy (2); Update (6)
User 4	Researcher	6	Deploy (2); Update (6)

The attacker applies the following heuristic for address clustering: If an account performs a privileged operation (e.g., `update`) on a contract, it is assumed to be controlled by that contract's owner. This heuristic reflects the assumption that modification rights are exclusive to the owner. Clustering quality is measured by Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI), both standard clustering metrics ranging from 0 (random clustering) to 1 (perfect recovery).

D. Results

1) ERC-4337 (as specified) vs. Proposed Approach: Under the stated attacker model and ten-user cohort, ERC-4337 yields near-zero entropy and an effective anonymity set size of one, since transactions remain linkable to known user–SCA pairs (Table IV). Our approach attains the cohort's maximum entropy, $\log_2 10 \approx 3.32$ bits, corresponding to an effective anonymity set size of 10. This indicates that, in this setting, transactions are indistinguishable among the ten participants.

TABLE IV
PRIVACY METRICS: ERC-4337 (AS SPECIFIED) VS. PROPOSED APPROACH.

Configuration	H	2^H	Avg. anonymity set
ERC-4337 (as specified)	0.00	1	1
Proposed approach	3.32	10	10

2) Account Shuffling vs. Proposed Approach: With account shuffling, the attacker reconstructs user clusters with moderate accuracy (ARI $= 0.468$, NMI $= 0.653$) under the privileged-operation heuristic. In our approach, user operations are executed via the RC, so the observable destination remains the RC; clustering degrades toward chance (ARI $= 0.038$, NMI $= 0.177$). Entropy-based analysis shows effective anonymity set sizes between 3 and 6 under shuffling (Table V), whereas our approach produces no meaningful clusters, increasing uncertainty and reducing inferred linkability under the attacker model.

TABLE V
ENTROPY ANALYSIS OF CLUSTERS (ACCOUNT SHUFFLING).

Cluster	# candidates	H	2^H
C1	6	2.58	6
C2	4	2.00	4
C3	3	1.58	3
C4	3	1.58	3

VI. DISCUSSION

This work addresses four privacy objectives important for secure healthcare data sharing on public blockchains: *unlinkability*, *anonymity*, *on-chain verifiability*, and *compliance-aligned data minimization*.

A. Unlinkability

By replacing EOAs with SCAs and delegating execution to RCs, our approach dissociates user identities from observable transactions. In a ten-user evaluation, the approach reached the entropy bound of $\log_2 10 \approx 3.32$ bits, equivalent to an effective anonymity set size of ten. In contrast, ERC-4337 as specified produces zero entropy, enabling deterministic linkage. These results show that our approach achieves unlinkability under the stated attacker model.

B. Anonymity

User identities remain hidden through Pedersen commitments and ZKPs. No identifiers are published on-chain, preventing behavioral linkability that arises from fixed pseudonymous addresses. Even when the attacker knows one SCA address of a target user, experiments indicate that additional addresses cannot be linked with statistical confidence. This suggests that anonymity is preserved for transactions beyond the attacker's prior knowledge, within the limits of our model.

C. On-chain Verifiability

All checks are enforced through smart contracts. Registration commitments are inserted into the RegC, and membership proofs are verified against its Merkle root. Subsequent interactions embed ZKP verification within AA validation flows. This ensures publicly auditable verification of operations while preserving privacy. Although proof generation occurs off-chain, correctness is enforced by the on-chain verifier, eliminating the need for additional off-chain trust beyond the ZKP setup.

D. Compliance-Aligned Data Minimization

The approach ensures that no personally identifiable information is transmitted or stored on-chain. Sensitive computations occur off-chain, and only non-invertible commitments and Merkle roots are recorded. This aligns with regulatory principles of data minimization and de-identification emphasized in frameworks such as GDPR and HIPAA, supporting privacy-by-design without requiring raw health data on-chain.

E. Threats and Limitations

Our analysis assumes a global passive adversary with full visibility of on-chain state but no active interference or access to side channels. Within this model, we demonstrated unlinkability and anonymity guarantees. Several threats and limitations, however, remain beyond scope. First, timing- and gas-cost side channels could still leak behavioral information even when addresses are unlinkable. Second, miner extractable value (MEV) strategies such as transaction reordering or frontrunning could undermine privacy if validators collude with adversaries. Third, denial-of-service or censorship of registration or verification transactions is not addressed. Fourth, collusion between registered users and external parties remains outside the model. Beyond threat assumptions, our implementation has additional limitations. Reliance on Groth16 entails a trusted setup. The current design supports static registrations and one-time verification, without revocable or dynamic credentials. Gas costs of proof verification and contract execution are not yet quantified. Finally, while our approach aligns with GDPR and HIPAA principles (e.g., data minimization), a formal article-level compliance mapping remains future work. Addressing these limitations calls for transparent proof systems (e.g., PLONK or STARKs), support for revocable and delegable credentials, evaluation under stronger adversarial models, systematic gas-cost benchmarking, and deployment studies with explicit regulatory alignment.

VII. Conclusion

We proposed a privacy-preserving approach for healthcare data sharing on public blockchains that integrates ERC-4337 account abstraction, Groth16 zero-knowledge proofs, and Pedersen hash–based commitments. By replacing EOAs with SCAs and verifying user operations through on-chain proofs, the approach decouples transaction execution from static keys and mitigates behavioral linkability. Our simulation on the Polygon Amoy testnet shows that, under a global passive attacker model and a ten-user cohort, the approach achieves unlinkability and anonymity close to the theoretical bound for the cohort while preserving on-chain verifiability and auditability. Comparative evaluations demonstrate stronger privacy guarantees than account shuffling and ERC-4337 as specified (i.e., without privacy extensions). These findings indicate that identity privacy can be realized without compromising transparency – one of the key requirement in healthcare, where protecting patient identities is as critical as safeguarding medical content. By aligning with data-minimization principles in GDPR and HIPAA, this work contributes toward decentralized healthcare platforms that are both trustworthy and regulation-aware. While limitations remain in credential dynamics, formal compliance analysis, and adversarial scope, the proposed approach establishes a foundation for future works that integrate transparent-proof constructions, dynamic and revocable credentials, and rigorous performance and compliance evaluations in real-world deployments.

References

[1] Mahsa Shabani and Luca Marelli. Re-identifiability of genomic data and the gdpr: Assessing the re-identifiability of genomic data in light of the eu general data protection regulation. *EMBO reports*, 20(6):e48316, 2019.

[2] World Health Organization. Global strategy on digital health 2020-2025, 2021. Licence: CC BY-NC-SA 3.0 IGO.

[3] Kun Li, Ashish Rajendra Sai, and Visara Urovi. Do you need a blockchain in healthcare data sharing? a tertiary review. *Exploration of Digital Health Technologies*, 2(3):101–123, 2024.

[4] Aashima Sharma, Sanmeet Kaur, and Maninder Singh. A comprehensive review on blockchain and internet of things in healthcare. *Transactions on Emerging Telecommunications Technologies*, 32(10):e4333, 2021.

[5] Jorge Bernal Bernabe, Jose Luis Canovas, Jose L Hernandez-Ramos, Rafael Torres Moreno, and Antonio Skarmeta. Privacy-preserving solutions for blockchain: Review and challenges. *Ieee Access*, 7:164908–164940, 2019.

[6] K Sabiri, F Sousa, and T Rocha. A systematic review of privacy-preserving blockchain applications in healthcare. *Multimedia Tools and Applications*, pages 1–56, 2025.

[7] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. Blochie: a blockchain-based platform for healthcare information exchange. In *2018 ieee international conference on smart computing (smartcomp)*, pages 49–56. IEEE, 2018.

[8] Ken Miyachi and Tim K Mackey. hocbs: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design. *Information processing & management*, 58(3):102535, 2021.

[9] Jack Huang, Yuan Wei Qi, Muhammad Rizwan Asghar, Andrew Meads, and Yu-Cheng Tu. Medbloc: A blockchain-based secure ehr system for sharing and accessing medical data. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 594–601. IEEE, 2019.

[10] Hassan Mansur Hussien, Sharifah Md Yasin, Nur Izura Udzir, and Mohd Izuan Hafez Ninggal. Blockchain-based access control scheme for secure shared personal health records over decentralised storage. *Sensors*, 21(7):2462, 2021.

[11] Rashmi P Sarode, Yutaka Watanobe, and Subhash Bhalla. A blockchain-based approach for audit management of electronic health records. In *International Conference on Big Data Analytics*, pages 86–94. Springer, 2022.

[12] Lianshan Sun, Diandong Liu, Yang Li, and Danni Zhou. A blockchain-based e-healthcare system with provenance awareness. *IEEE Access*, 2024.

[13] Aristodemos Paphitis, Nicolas Kourtellis, and Michael Sirivianos. Graph analysis of blockchain p2p overlays and their security implications. In *International Symposium on Security and Privacy in Social Networks and Big Data*, pages 167–186. Springer, 2023.

[14] Chaitanya Rahalkar and Anushka Virgaonkar. Summarizing and analyzing the privacy-preserving techniques in bitcoin and other cryptocurrencies. *arXiv preprint arXiv:2109.07634*, 2021.

[15] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[16] Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):3069, 2019.

[17] Brent Daniel Mittelstadt and Luciano Floridi. The ethics of big data: current and foreseeable issues in biomedical contexts. *The ethics of biomedical big data*, pages 445–480, 2016.

[18] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[19] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *arXiv preprint arXiv:1708.04748*, 2017.

[20] Vikas Jaiman and Visara Urovi. A consent model for blockchain-based health data sharing platforms. *IEEE access*, 8:143734–143745, 2020.

[21] Vitalik Buterin, Yoav Weiss, Dror Tirosh, Shahaf Nacson, Alex Forshtat, Kristof Gazso, and Tjaden Hess. ERC-4337: Account abstraction using alt mempool [draft]. https://eips.ethereum.org/EIPS/eip-4337, September 2021. Ethereum Improvement Proposal, no. 4337.

[22] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, pages 203–225. 2019.

[23] Ankur Lohachab and Visara Urovi. A blockchain-based approach for model card accountability and regulatory compliance. In *International Conference on Advanced Information Systems Engineering*, pages 37–48. Springer, 2024.

[24] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[25] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

[26] zkPass Team. zkpass. https://zkpass.org/, 2025. Accessed 5 Jun 2025.

[27] Privacy & Scaling Explorations (PSE). Semaphore. https://semaphore.pse.dev/, 2021. Accessed: 2025-06-06.

[28] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. zklogin: Privacy-preserving blockchain authentication with existing credentials. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3182–3196, 2024.

[29] Tong Zhou, Xiaofeng Li, and He Zhao. Med-ppphis: blockchain-based personal healthcare information system for national physique monitoring and scientific exercise guiding. *Journal of medical systems*, 43(9):305, 2019.

[30] Peng Zhang, Jules White, Douglas C Schmidt, Gunther Lenz, and S Trent Rosenbloom. Fhirchain: applying blockchain to securely and scalably share clinical data. *Computational and structural biotechnology journal*, 16:267–278, 2018.

[31] Visara Urovi, Vikas Jaiman, Arno Angerer, and Michel Dumontier. Luce: A blockchain-based data sharing platform for monitoring data license accountability and compliance. *Blockchain: Research and Applications*, 3(4):100102, 2022.

[32] Isabel Wagner and David Eckhoff. Technical privacy metrics: a systematic survey. *ACM Computing Surveys (Csur)*, 51(3):1–38, 2018.

[33] Christer Andersson and Reine Lundin. On the fundamentals of anonymity metrics. In *IFIP International Summer School on the Future of Identity in the Information Society*, pages 325–341. Springer, 2007.