

Article

Privacy-Preserving Transaction Admission with Warrant-Based Trace Recovery for Regulated Cryptocurrency Transactions

Yiyin Zhang ^{1,2} , Jiwu Jing ¹, Yuewu Wang ^{1,*} and Chunjing Kou ¹

¹ University of Chinese Academy of Sciences, Beijing 101408, China; zhangyiyin18@mailsucas.ac.cn (Y.Z.); jwjing@ucas.ac.cn (J.J.); kouchunjing@ucas.ac.cn (C.K.)

² Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China

* Correspondence: wangyuewu@ucas.ac.cn

Abstract

Public blockchain transactions need compliance support without exposing stable identity anchors. This paper presents a privacy-preserving transaction admission framework for regulated cryptocurrency transaction paths. It adds an admission and recovery layer to admission-aware applications, contracts, gateways, or chain policies without introducing a new cryptocurrency. In ordinary use, a user submits a public transaction object and a single Groth16 admission proof, which checks registration membership, current state, hidden home-domain membership, hidden home-domain policy compliance, transaction binding, replay control, and trace-object well-formedness while hiding identity-related witnesses. When a valid warrant targets an accepted transaction, recovery committees perform transaction-scoped trace recovery through outer opening and domain-local inner recovery. We implement an end-to-end prototype with off-chain modules and an Ethereum Virtual Machine (EVM)-compatible public admission contract. The evaluation reports a 28,370-constraint circuit, 3.40 s average proving time, about 536k local EVM admission gas, 30/30 replay rejection, 8.46 ms local layered recovery latency, and 3.84 ms domain-dispatch latency.

Keywords: cryptocurrency; public blockchain; transaction admission; privacy-preserving compliance; Groth16 proof; warrant-based trace recovery

1. Introduction

Public blockchain systems make transaction execution publicly verifiable, but this openness exposes behavioral and identity-related signals over time. Regulated transaction paths must also support eligibility checking, policy checking, replay prevention, and exceptional accountability. This paper studies these requirements at the transaction-admission layer and examines how admission-aware applications, contracts, gateways, or chain-level policies can verify regulated transaction conditions without turning ordinary public records into stable identity anchors.

1.1. Background and Motivation

Bitcoin [1] and Ethereum [2] make value transfer and contract execution visible through addresses or accounts. This transparency is useful for decentralized settlement, but it also leaves many signals on the ledger, including addresses, amounts, contract calls, and timing patterns. As shown by Meiklejohn et al. [3] and Kalodner et al. [4], these signals can be enough to support behavioral linkage even when no real-world identity is directly published.



Academic Editor: Mehdi Sookhak

Received: 15 May 2026

Revised: 14 June 2026

Accepted: 23 June 2026

Published: 8 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Privacy-enhancing transaction mechanisms try to reduce this exposure by hiding addresses, amounts, or transfer relations. Zerocash, proposed by Sasson et al. [5], is a representative example of strong payment privacy. Bünz et al. [6] further study confidential payments in account-based smart-contract systems. These mechanisms improve transaction privacy, but they also make compliance harder to express at the public layer. A regulated transaction path may still need to check whether a user is eligible, whether the user's state is current, whether a hidden domain-specific policy is satisfied, whether the same admission value has been replayed, and whether identity recovery is possible in exceptional cases. Buterin et al. [7] show that zero-knowledge proofs can help separate compliance checks from full transaction-history disclosure. Still, transaction-layer values related to identity and recovery must be handled carefully.

One simple way to support compliance is to put real-world identities, certified credentials, or reusable regulatory identifiers into transaction records. This makes admission and attribution easier, but it also creates a stable observation channel. Once a long-term identity anchor, or a reusable derivative of it, appears in admitted records, later transactions can be linked to the same subject.

This paper studies regulated transaction admission without placing such anchors in ordinary admitted records. The framework does not define a new cryptocurrency. It adds an admission and recovery layer to admission-aware applications, contracts, gateways, or chain policies. A user proves registration membership, current state, hidden home-domain membership, hidden home-domain policy compliance, replay binding, and trace-object well-formedness through a single Groth16 proof [8]. The proof hides identity-related witnesses, while a valid warrant can later trigger transaction-scoped dual-layer recovery. The admission-side amount witness is used only for private policy checking. The framework does not make an otherwise transparent asset transfer confidential.

1.2. Gap and Technical Challenge

Garman et al. [9] and Xue et al. [10] study regulated anonymous payments. Camenisch and Lysyanskaya [11], Sonnino et al. [12], and Maram et al. [13] study accountable identity mechanisms and anonymous credentials. Rosenberg et al. [14], Piper et al. [15], and Buterin et al. [7] explore selective-disclosure-based compliance. These works show that privacy and accountability can coexist, but mostly at the payment, identity-authentication, or disclosure-predicate layer. Less attention has been paid to admission as a transaction-layer interface, where a public verifier accepts a transaction-level object only after checking private eligibility, hidden policy compliance, replay control, and recovery preparation.

This gap raises three challenges. First, the verifier must check that a transaction comes from a registered and active user, follows the hidden home-domain policy, and binds public fields to hidden witnesses, without learning the user's identity, home domain, attributes, amount witness, trace token, or rule-level outcomes. Second, admission values, replay-control values, policy-binding handles, and the recovery object must be bound to the same transaction instance. Otherwise, valid-looking components may be mixed across contexts or reused. Third, recoverability must be prepared during admission but must not become ordinary traceability. The trace object should be well formed, transaction-bound, and recoverable only for the accepted transaction named by a valid warrant.

This work treats admission as transaction-layer verification rather than payment-protocol modification or standalone credential presentation. The public verifier checks a transaction-level public object and a single Groth16 admission proof. The proof enforces registration and state validity, hidden home-domain membership, hidden home-domain policy compliance, transaction binding, replay consistency, and trace-object well-formedness. Warrant-based recovery is activated only for a target accepted transaction, keeping recovery

transaction-scoped. The base workflow assumes correct root publication and bounded honest-but-curious recovery committees. Malicious-share robustness, collusion-resistant recovery routing, network anonymity, and complete selective-disclosure security are outside this base setting. Section 5 discusses how these assumptions affect the security claims.

1.3. Contributions

This work treats regulated compliance as transaction admission. The framework binds hidden eligibility, hidden home-domain policy compliance, replay freshness, policy handles, and a warrant-ready trace object to one accepted transaction reference while leaving payment and application semantics to the underlying regulated path.

The main contributions are as follows.

- We formulate privacy-preserving transaction admission as a public-chain problem. The proposed layer checks regulated transaction conditions without placing stable identity anchors in ordinary admitted records.
- We design a transaction-bound admission and recovery mechanism. A single Groth16 proof checks registration, current state, hidden home-domain membership, hidden policy compliance, transaction binding, replay consistency, and trace-object well-formedness, while recovery remains warrant-scoped and transaction-scoped.
- We implement an end-to-end prototype with off-chain modules, a generated TxAdmissionVerifier, and an EVM-compatible PublicAdmission contract. The evaluation reports proof generation, contract gas, replay rejection, recovery latency, domain-scaled recovery, and object-size results.

1.4. Paper Organization

Section 2 reviews related work and positions the proposed admission layer. Section 3 defines the model, assumptions, and goals. Section 4 presents the framework design. Section 5 analyzes the security guarantees and scope. Sections 6 and 7 describe the prototype and evaluation. Sections 8 and 9 give the discussion and conclusion.

2. Related Work

This section reviews work closest to the proposed admission layer. We organize the discussion by protocol boundary: regulated private payments, distributed regulation and regulatable contracts, accountable identity, zero-knowledge compliance, and smart-contract verification patterns. These works motivate our design, but they do not directly provide a transaction-admission interface where one accepted public record binds private eligibility, hidden home-domain policy compliance, replay control, and a warrant-ready trace object to the same transaction instance.

2.1. Regulated Anonymous Payments and Traceable Transactions

Anonymous and confidential payment systems provide the basis for regulated privacy designs. Zerocash [5] hides payment origin, destination, and amount with zk-SNARKs, and Zether [6] brings confidential payment verification into account-based smart contracts. Regulated private-payment systems further add accountability or auditability. Garman et al. [9] support policy enforcement and selective tracing, Xue et al. [10] add regulatory controls and suspicious-transaction recovery, UTT [16] introduces anonymity budgets, and Brugeret et al. [17] study fast regulated payments. These schemes integrate regulation into payment validity, together with ownership, value constraints, double-spending control, and anonymity. Our framework separates regulatory admission from payment validity. The underlying payment system, contract, or application keeps its own semantics, while

the admission proof binds eligibility, current state, hidden home-domain policy compliance, replay freshness, and trace-object well-formedness to one transaction reference.

Traceable anonymous-transaction schemes add tracing or linkability to anonymous transactions [18–20]. Our setting uses a narrower recovery interface. The trace object is checked before acceptance, remains closed during ordinary use, and can be opened only for the warranted accepted reference. This keeps recovery from becoming a reusable user-level tracing handle.

2.2. Distributed Regulation and Regulatable Smart Contracts

Regulated privacy also depends on how tracing authority is distributed and constrained. Platypus [21] and PEReDi [22] design privacy-preserving CBDC systems with regulatory support. DAPCR [23] distributes regulatory power to reduce single-regulator control, and DAPDS [24] considers collaborative regulation with user data sovereignty. These designs mainly address recovery governance, such as who may open, approve, or participate in regulated tracing. Our setting also needs a transaction-level anchor before recovery starts. The accepted record must already bind registration, state, hidden policy compliance, replay control, and trace-object well-formedness to the same reference.

RPSC [25] studies privacy-preserving smart contracts on account-based blockchains and adds a regulatory tracing path to private contract execution. Our work focuses on the admission step before a regulated operation is accepted. A public transaction object is accepted only after the admission proof verifies hidden registration, state, policy, replay, and recovery-preparation conditions under active roots. The PublicAdmission contract then verifies the proof output, checks replay freshness, and records the accepted reference as the anchor for later audit and warrant-based recovery.

2.3. Accountable Identity and Credential-Based Compliance

Anonymous credentials, group signatures, and traceable signatures provide basic tools for anonymous authorization and controlled opening. Camenisch and Lysyanskaya [11] introduce anonymous credentials with unlinkable showings and optional anonymity revocation, Idemix applies this idea to access-control settings [26], group signatures support membership verification with a designated opener [27], and traceable signatures add tracing for repeated misuse [28]. Blockchain identity systems bring related ideas into decentralized settings. DID and VC specifications define decentralized identifiers and verifiable claims [29,30]. Coconut [12], CanDID [13], and zk-creds [14] support credential issuance, selective disclosure, or accountability. These systems can provide enrollment or attribute material for regulated applications. In our framework, identity-side evidence is consumed by the transaction-admission relation, which also checks hidden policy, replay, and recovery constraints for the accepted transaction reference.

2.4. Privacy-Preserving Compliance on Public Blockchains

Recent work checks public-chain compliance predicates without full disclosure. zkKYC [31] supports KYC proofs, DECO [32] proves statements about web data, Privacy Pools [7] proves source-of-funds predicates, Piper et al. [15] study KYC-compliant on-chain permissioning, and zkAt [33] provides policy-private blockchain authentication. These works hide sensitive data while proving KYC, provenance, source-of-funds, access, or authentication predicates. Our setting also controls later opening. Identity resolution remains at the home-domain side, and a warrant targets an accepted transaction reference rather than a reusable user identity. The admission proof therefore separates ordinary compliance checking from domain-local, transaction-scoped recovery.

2.5. Smart-Contract Transaction Verification and Design Patterns

Smart-contract engineering is relevant because PublicAdmission is a transaction-verification endpoint. Kannengiesser et al. [34] summarize development challenges and solutions, Górski [35,36] studies reusable verification patterns for related transaction types, and recent surveys classify security design patterns and their relation to vulnerabilities [37]. These works mainly concern the organization and security of public contract logic. Our contract uses the same transaction-verification setting, but the checked rule is a privacy-preserving admission relation. It checks public arguments, public proof signals, replay freshness, and the Groth16 verifier output and records an accepted anchor only after hidden eligibility, policy compliance, replay binding, and trace-object well-formedness have been proven by the admission proof.

2.6. Position of This Work

The above work addresses privacy, accountability, and compliance at neighboring boundaries. Payment and traceable-transaction systems build accountability into transaction constructions. Distributed-regulation systems organize tracing authority. Credential and compliance systems prove identity, access, or source-of-funds predicates with less disclosure. Smart-contract design-pattern work studies how public verification logic is engineered. This paper addresses the admission layer between these boundaries. The underlying system keeps its own asset-transfer and application semantics, while the public layer checks whether a transaction record is admissible under hidden regulatory witnesses. One Groth16 proof binds registration, current state, hidden home-domain policy compliance, replay freshness, policy handles, and trace-object well-formedness to the same accepted reference. The public layer learns no identity-side witness, and warrant-based recovery remains tied to that reference.

3. System Model and Design Goals

We model the system as an admission layer for regulated transaction paths. It is used when a transaction must be admissible, while the accepted record should not carry identity-side information. The base workflow includes ordinary admission and warrant-based trace recovery. Selective disclosure is outside the base scheme.

The admission check is placed at an existing enforcement point, such as an application, contract, gateway, or chain policy. The model does not require every native transfer to pass through admission. In contract-gated or gateway deployments, the regulated action proceeds only after ρ_j has an accepted $TxRecord_j$. Other transactions may still exist on the underlying chain, but they are not admitted regulated transactions here.

In ordinary admission, the user submits Pub_j and a Groth16 proof Π_j [8]. The verifier checks admissibility under active roots and transaction context, while identity, home domain, attributes, amount witness, trace token, and proof witnesses remain hidden. Recovery is invoked only when a warrant names an accepted transaction. The analysis assumes correct roots and bounded honest-but-curious recovery committees.

3.1. System Entities

We distinguish four functional roles in the base workflow.

- **User.** The user prepares admitted transactions while keeping identity-related witnesses out of the ordinary public record. For each admission request, the user keeps the admission and recovery secrets, obtains the required Merkle witnesses [38], and generates Pub_j together with the admission proof Π_j .
- **Registration Authority and Registries.** The authority handles enrollment before ordinary admission starts. It verifies the user's identity or qualification information,

assigns the home domain, and maintains the off-chain identity mapping used in authorized recovery. The registries publish and update the roots for registration, current state, domain keys, and policy tables. Once the roots are prepared, the authority does not participate in ordinary admission.

- **Public Admission and Ledger Layer.** This layer contains the public roots, the smart-contract interface, and the verification logic. It assembles the statement from the active roots and Pub_j , verifies Π_j , records accepted $TxRecord_j$, maintains $UsedBind$, and stores trace-request anchors and status values when needed. It does not store identity secrets, hidden attributes, trace tokens, Merkle witnesses, proof witnesses, recovery shares, or recovery transcripts.
- **Recovery Committees.** Recovery committees are used only after a valid warrant targets an accepted transaction. The global committee opens the outer trace layer, and domain-local committees try the inner recovery step. The outer opening gives only the inner trace object, and identity resolution succeeds only at the home domain. An implementation-side coordinator may relay artifacts, but it does not hold recovery shares or obtain independent recovery authority.

With this separation, ordinary admission only involves public verification of the admitted record. Identity resolution stays outside that path and requires a target transaction, a valid warrant, and the required committee cooperation.

3.2. Threat Model and Assumptions

We describe the adversarial view and assumptions of the base scheme, focusing on the public transaction layer and warrant-based recovery.

3.2.1. Adversarial Model

Let $\mathcal{U}$ be the users, $\mathcal{R}$ the registration authority and registries, $\mathcal{P}$ the public admission and ledger layer, and $\mathcal{C}$ the recovery committees. Committees are domain-based: $\mathcal{D}$ is the valid domain set in the current epoch, and $\mathcal{C}_d$ is the committee of $d \in \mathcal{D}$. For user u , let $x_u \in \mathbb{F}_r$ be the admission secret, $\tau_u \in \mathbb{F}_r$ the trace-token secret, and d_u the home domain. The user derives $pk_u = \text{Poseidon}(x_u)$ and $I_u = \text{Poseidon}(\tau_u)$ with the Poseidon hash [39]. The value pk_u is used in the registration leaf, and I_u is the home-side lookup token.

For transaction tx_j , the accepted record is

$$TxRecord_j = (Pub_j, \Pi_j). \quad (1)$$

The verifier assembles

$$X_{tx} = \text{Assemble}(root_{reg}, root_{state}, root_{dir}, root_{pol}, Pub_j). \quad (2)$$

The structure of Pub_j is defined in Section 4.

The public ledger view $\mathcal{L}$ contains accepted records, used bindings, epoch parameters, public roots, trace requests, and public transaction fields. It does not contain real-world identities, enrollment records, hidden attributes, trace tokens, Merkle witnesses, recovery shares, Groth16 witnesses, or recovery transcripts.

We consider three adversarial goals.

Observation. The adversary inspects $\mathcal{L}$ and tries to link admitted transactions, infer the home domain, learn hidden attributes or amount witnesses, recover trace tokens, or extract stable identity-related values from Pub_j and Π_j .

Admission soundness. The adversary controls users and chooses transaction contents, public fields, nonces, and witnesses. It tries to make $\mathcal{P}$ accept an invalid $TxRecord_j$ by

using stale roots, violating hidden policies, reusing bindings, mixing contexts, or attaching malformed trace objects.

Unauthorized recovery. The adversary inspects $\mathcal{L}$ and may corrupt committee members below the thresholds but has no valid warrant or home-side database access. It tries to recover I_u , identify the home domain, or resolve the real identity outside the prescribed recovery workflow.

3.2.2. Assumptions and Scope

The base workflow uses the following assumptions.

Enrollment and identity grounding. Real-world identity binding is handled before ordinary admission. The registration authority enrolls users, assigns each user to a home domain, and keeps the off-chain identity mapping used later in authorized recovery. This work does not remove the need for such an identity trust root. The point is to keep this root out of the ordinary admission path after registration, so that public verification does not require identity records or reusable identity-derived values.

Public admission enforcement. The public admission layer accepts an ordinary transaction only under the active root set maintained by the public admission interface, a valid Groth16 proof, and an unused binding value. This active set contains the registration, state, directory, and policy roots for the current epoch. These roots are public verification parameters, while the corresponding leaves, Merkle paths, home-domain witness, attributes, and trace-token material remain private. Revocation and suspension are reflected in the state root. New ordinary admissions must use the current epoch e_{cur} and the current registration, state, directory, and policy roots. The verifier assembles X_{tx} from this active set or checks submitted root signals against it, so stale but once-valid roots are rejected for new ordinary admission.

Epoch directories. The domain and policy directories publish correct epoch roots. The domain root binds each valid domain to its recovery public key, and the policy root binds each domain to its policy table. Root correctness is grounded in the responsible registries and directories, while freshness is enforced by the public admission interface. Root updates replace the active registration, state, directory, and policy roots as one epoch set, so the verifier does not mix roots from different epochs. A monotonic active epoch and the event or digest history of accepted root updates provide rollback protection. Historical roots may be retained for interpreting old accepted records and supporting authorized recovery, but they are not accepted for new ordinary admission. A revocation or suspension takes effect for new admissions after the updated state root becomes active, and it does not retroactively invalidate an already accepted $TxRecord_j$.

Bounded committee corruption. Recovery committees follow threshold assumptions based on Shamir secret sharing [40]. The adversary corrupts fewer than t_N members of the global outer committee and fewer than t_d members of each domain-local committee, including the home-domain committee. Committee members follow the protocol and do not leak intermediate recovery results to unauthorized parties. The base scheme assumes an honest-but-curious threshold backend. Malicious-share robustness and DKG are treated as replaceable backend hardening.

Trusted warrant issuer and warrant-bound recovery. The warrant issuer is the legal authorization trust anchor. A warrant is a signed object

$$W_j = (id_j^w, iss_j, \rho_j, chain_j, e_j^w, [t_j^{from}, t_j^{until}], scope_j, h_j^{legal}, \sigma_j^w). \quad (3)$$

Here, ρ_j identifies the target accepted transaction, h_j^{legal} commits to the legal basis, and σ_j^w is the issuer signature. Before any opening, the workflow checks the issuer signature,

target reference, chain identifier, validity window, and scope. The public layer stores only $h_j^w = H(W_j)$ and records

$$\text{TraceReq}[\rho_j] = (h_j^w, e_j^{\text{req}}, s_j^{\text{req}}). \quad (4)$$

This check binds recovery to the target accepted transaction. It does not decide whether the issuer should have granted the warrant. Warrant misuse is a governance issue for transparency logs, multi-party authorization, or external audit.

Recovery output and visibility. A recovery execution is scoped to the reference ρ_j named in the validated warrant. A warrant for one accepted transaction cannot be reused as a recovery transcript for another transaction. To recover another transaction, the workflow needs another accepted $TxRecord_j$, another valid warrant, and the corresponding threshold openings. After successful recovery, the home domain resolves the identity associated with the warranted transaction. Ordinary admitted records still contain no searchable identity anchor. The public layer records only the trace-request anchor and status for the target reference, while detailed recovery artifacts and recovery transcripts remain off-chain.

This assumption does not control every off-chain consequence of recovery. Once an identity is legally recovered, the protocol cannot govern how an authorized recipient uses that identity outside the system. In the recovery workflow, the inner trace object is dispatched to candidate domain-local recovery modules in parallel. Failed non-home attempts are not written to the public ledger or included in the public recovery result, and non-home domains observe only their own local failures. The guarantee does not make every internal recovery transcript private. If labeled per-domain responses are revealed outside the home-domain side, the successful home domain may still be inferred from the matching response. Hiding this internal recovery-side signal requires private forwarding, anonymous routing, or an aggregation mechanism that does not expose labeled per-domain outcomes outside the authorized recovery result.

Threshold-opening backend. The recovery backend is replaceable. The contribution is the transaction-admission relation and warrant-scoped recovery interface, not a new malicious-secure threshold cryptosystem. The prototype implements a minimal honest-but-curious backend with outer partial openings, threshold combination, domain-local inner recovery, and insufficient-share failures. Deployments can add Feldman VSS [41], Pedersen VSS [42], or secure DKG for discrete-log-based systems [43]. Replacing the backend does not change Pub_j , Π_j , $TxRecord_j$, or the public admission state. Malicious-share robustness, proactive resharing, and committee churn are deployment-hardening.

At each epoch, the global outer key and domain-local inner keys are initialized by DKG or an equivalent trusted setup service, following standard threshold key-generation practice [44]. The scheme abstracts this setup layer and does not address network-layer traffic analysis, full collusion among all entities, deanonymization from auxiliary data, or complete selective-disclosure security.

3.3. Target Properties and Design Goals

Under the above model, the base scheme aims to provide the following properties.

Privacy-preserving ordinary admission. For an accepted record

$$TxRecord_j = (Pub_j, \Pi_j), \quad (5)$$

public observers can verify admissibility without learning the user's identity, home domain d_u , attributes A_u , amount witness, trace token I_u , or proof witnesses. The public object Pub_j should not contain a stable identity anchor. Its tag, replay binding, policy handles, and trace object are generated for the specific transaction instance.

Private-policy admission correctness. An accepted transaction should correspond to a valid enrolled user with a current active state, a valid hidden home domain, and a policy table taken from the active policy root. The admission proof must also show that the transaction satisfies the policy of that hidden home domain. The verifier checks this result without learning which domain policy was used, which internal policy branch accepted the transaction, or the user's hidden attributes and amount witness. The same transaction context must bind the public fields, hidden witnesses, policy handles, replay value, and trace object. Otherwise, the record is rejected.

Replay resistance and checked trace preparation. Each admitted transaction carries a replay binding and a trace object tied to the same context as Π_j . A consumed binding cannot be accepted again. The trace object is checked by Π_j , but it reveals neither I_u nor the home domain during ordinary admission.

Warrant-based recovery and deployment compatibility. Recovery starts only after a valid warrant for ρ_j . Without the warrant and the required committee thresholds, observers, verifiers, and below-threshold committee subsets cannot recover I_u or resolve the user's identity. Successful recovery is transaction-scoped: another transaction requires another accepted record, another warrant, and another set of threshold openings. On-chain components keep only public roots, accepted references, used bindings, warrant digests, and trace-request states. Identity records, witnesses, warrant payloads, detailed recovery artifacts, and recovery transcripts remain off-chain.

These goals rely on the assumptions in Section 3.2. The base setting does not provide malicious-secure threshold recovery, collusion-resistant routing, network anonymity, arbitrary native-transfer enforcement, or complete selective-disclosure security.

4. Framework Design

This section presents the proposed transaction-admission layer. The design addresses three requirements. Ordinary admission must be publicly verifiable without reusable identity-related values. Admission, replay control, policy handles, and recovery objects must refer to the same transaction. Trace recovery must be prepared during admission but activated only for a warranted transaction.

The design follows this structure. For new admission, the public layer obtains the active registration, state, domain-directory, and policy roots from its public state or an authenticated root-registry interface and rejects proofs whose root signals do not match that active set. Public roots anchor registration, state, domain-key, and policy information, while the corresponding leaves and paths remain private witnesses. For each admission request, the user builds a transaction-bound public object and proves, with one Groth16 proof [8], that the hidden witnesses match the active roots and the public transaction context. The recovery object is generated in the same relation, so it is already tied to the accepted record even though it stays closed during ordinary admission. The rest of this section introduces the epoch roots, the ordinary admission path, the warrant-based recovery path, and the public state kept by the ledger layer.

4.1. Design Roadmap

Figure 1 gives the high-level flow of the admission and recovery layer. The user, the registration authority and registries, the public admission and ledger layer, and the recovery committees are connected by two paths. The ordinary path produces an accepted record from Pub_j and Π_j , while the recovery path is entered only after a valid warrant names the accepted reference ρ_j .

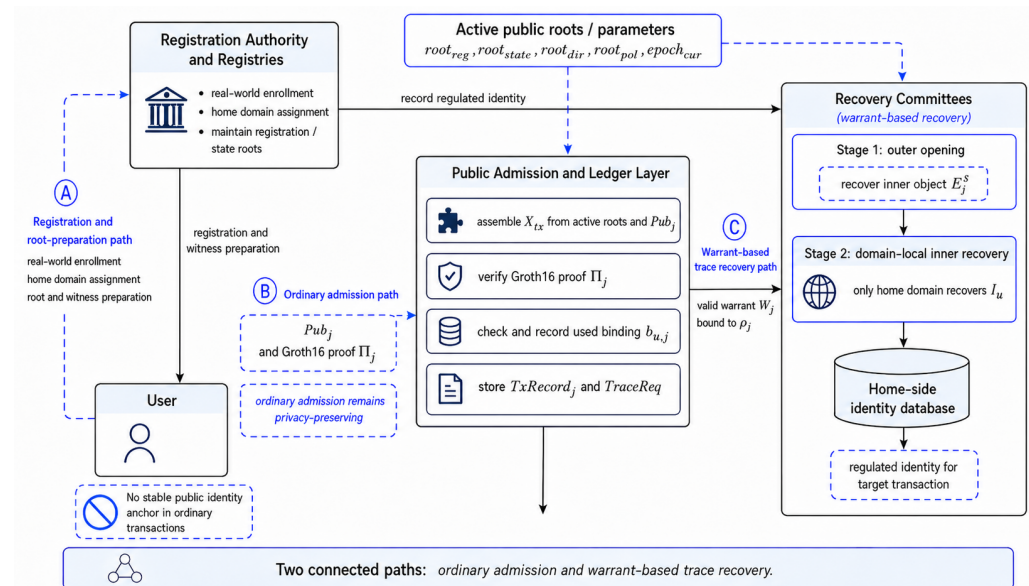


Figure 1. Admission and recovery flow. The public layer checks Pub_j and Π_j under active roots, and the checked trace object is opened only for a warranted accepted transaction.

Enrollment publishes only the anchors needed by later admission. The registration, state, domain-directory, and policy roots become public parameters, while the opened leaves, home domain, attributes, trace token, and identity mapping stay outside ordinary admitted records. For a new ordinary admission, the verifier uses the active root set, and any root public signals must match that set before the accepted reference is recorded. When the user later submits an admission request, Pub_j carries the public transaction-side values, and Π_j proves the root, policy, binding, and trace-object relations against the active roots. The public layer checks this proof and then consults the replay state before accepting the record.

The recovery path relies on the trace object that has already passed the admission check. Since E_j^{tr} is included in Pub_j and constrained by Π_j , recovery does not require the user to provide another consistency proof. Once a valid warrant names ρ_j , the global committee opens the outer trace layer, and candidate domain-local committees run the inner recovery step. A matching identity can be resolved only by the home domain. The ledger side keeps the public state for these operations, including accepted references, consumed bindings, warrant digests, and trace-request states. Identity records, private witnesses, warrant payloads, detailed recovery artifacts, recovery transcripts, and failed non-home attempts remain off-chain unless a deployment adds a separate audit interface.

4.2. Registration and Epoch Roots

Registration provides the off-chain identity basis for the later admission proof. The user first enrolls with the registration authority, which checks the required identity or qualification information and prepares the records used by admission and recovery. After registration, ordinary admission no longer involves the authority directly. The user instead proves registration, state, domain, and policy membership inside the Groth16 admission circuit with Merkle witnesses [8,38].

The user keeps two long-term secrets $x_u, \tau_u \in \mathbb{F}_r$. The value x_u is used to derive admission-side values, including replay bindings. The value τ_u is used on the recovery side. Using Poseidon [39], the user derives $pk_u = \text{Poseidon}(x_u)$ and $I_u = \text{Poseidon}(\tau_u)$. The registration leaf contains pk_u , while I_u later serves as the lookup token in the home-domain identity database. Because the registration leaf and its Merkle path are supplied as private witnesses, I_u is not exposed in ordinary admitted records.

The authority also assigns the home domain d_u and registration serial s_u . It prepares the authorization vector $A_u = (a_u, c_u, r_u, j_u)$, where the entries represent adult status, cross-domain permission, risk tier, and jurisdiction class. The vector is committed as $h_u^A = \text{Poseidon}(A_u)$. During ordinary admission, both the commitment relation and the underlying attributes are checked through hidden witnesses, so the public verifier does not learn A_u .

Table 1 lists the four roots used by ordinary admission. For a new ordinary admission, these roots are active public verification parameters read by the public admission layer or its authenticated root-registry interface, not user-selected values. The registration root commits to enrollment data, the state root reflects the current user status, the domain root maps valid domains to recovery public keys, and the policy root maps domains to their policy tables.

Table 1. Epoch roots used by ordinary admission.

Root	Leaf Form	Main Role
$root_{reg}$	$\text{Poseidon}(pk_u, I_u, d_u, h_u^A, s_u)$	Registration anchor
$root_{state}$	$\text{Poseidon}(s_u, st_u, e_u)$	Current status
$root_{dir}$	$\text{Poseidon}(d, PK_d, e_{cur})$	Domain recovery key
$root_{pol}$	$\text{Poseidon}(d, P_d, e_{cur})$	Domain policy table

The policy table of domain d is $P_d = (R_d^{adult}, R_d^{cross}, R_d^{amt}, R_d^{risk})$. These circuit-friendly templates are evaluated inside the admission circuit. The amount rule uses the admission-side amount witness or its bucketized form. The framework itself does not make an otherwise transparent asset transfer confidential. New ordinary admissions are verified against the active epoch e_{cur} and the active registration, state, directory, and policy roots. If roots are carried as proof public signals, they must match this active root set before $TxRecord_j$ is accepted, so stale roots cannot be reused. Root updates replace the active roots as one epoch set, preventing cross-epoch root mixing. A monotonic active epoch and accepted root-update history support rollback protection. Historical roots may be retained for interpreting old accepted records and supporting authorized recovery but not for new ordinary admission. A revocation or suspension affects new admissions once the updated state root becomes active, without retroactively invalidating accepted records.

The user obtains Merkle witnesses $w_{reg,u}$, $w_{state,u}$, $w_{dir,u}$, and $w_{pol,u}$ for the registration, state, hidden home-domain directory, and hidden home-domain policy leaves. These witnesses are private prover inputs. Public verifiers see only the roots and the proof result.

The home-side identity database of d_u stores the regulated identity record indexed by I_u , written as $DB_{d_u}[I_u] = (ID_u, pk_u, d_u, s_u, meta_u, \dots)$. This database is accessed only during authorized recovery after a valid warrant is recorded for the target reference ρ_j .

4.3. Ordinary Admission

After registration and root preparation, a user prepares an admission request for a canonical regulated transaction body tx_j . The body tx_j is processed by the regulated entry point, and the admission data is generated for this body. The base scheme uses a hidden admission-side amount witness. A public-amount deployment can remove the amount-hiding constraints and expose the amount used for policy checking, but the framework itself does not make an otherwise transparent asset transfer confidential.

Ordinary admission is centered on a transaction-bound public object. For each request, the user derives a transaction context from public transaction-side fields, including the raw-amount commitment, destination domain, chain identifier, policy identifier, transaction category, body digest, and nonce. The body digest is $h_j^{body} = H(\text{Encode}(tx_j))$, where the

canonical encoding includes the regulated entry point, transaction category, public action fields, nonce, and validity window when applicable. The admission-side amount bucket is derived from the hidden amount witnessed under public thresholds and is used only inside the circuit. Under the same context, the user generates a transaction-specific tag, replay binding, policy-binding handles, and dual-layer trace object. One Groth16 proof checks these values together [8].

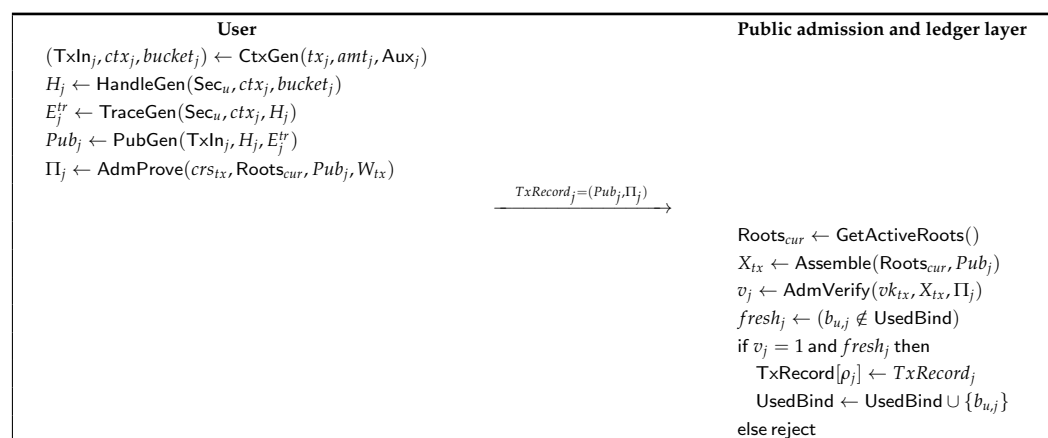
Let $\text{TxIn}_j = (C_j^{\text{raw}}, d_j^{\text{dst}}, ch_j, pid_j, cat_j, h_j^{\text{body}}, v_j, st_u, e_u)$ be the public transaction-side input bundle, and let $\text{Roots}_{\text{cur}} = (root_{\text{reg}}, root_{\text{state}}, root_{\text{dir}}, root_{\text{pol}})$ be the active roots read by the public admission layer for the current epoch. Let Sec_u contain the user's hidden admission and policy witnesses, and let Path_u contain the Merkle witnesses [38]. We write $H_j = (tag_j, b_{u,j}, C_j^{\text{amt}}, C_j^{\text{risk}})$ for the handle vector and $W_{tx} = (\text{Sec}_u, \text{Path}_u, \text{amt}_j, \text{Aux}_j)$ for the proving witness, where Aux_j contains commitment, nonce, and trace-generation randomness.

The public transaction object is

$$\text{Pub}_j = (\rho_j, e_{\text{cur}}, C_j^{\text{raw}}, d_j^{\text{dst}}, ch_j, pid_j, cat_j, h_j^{\text{body}}, v_j, st_u, e_u, b_{u,j}, tag_j, C_j^{\text{amt}}, C_j^{\text{risk}}, E_j^{\text{tr}}). \quad (6)$$

Here, ρ_j is the public transaction reference. The values tag_j and $b_{u,j}$ are transaction-specific and are not reusable user identifiers. The digest h_j^{body} binds the accepted record to tx_j , so changing the regulated entry point, action fields, nonce, or validity window changes the transaction context. The handles C_j^{amt} and C_j^{risk} bind the amount bucket and risk tier to the same transaction instance and reserve stable interfaces for future selective disclosure. The trace object E_j^{tr} is prepared for later warrant-based recovery but remains closed during ordinary admission.

The ordinary admission workflow is shown below. The left side is local to the user, and the right side is the public admission check performed by the ledger layer.



The ledger accepts the record only when the proof is verified and the replay binding has not appeared before. The statement X_{tx} is assembled for Groth16 verification from the active roots obtained by the public admission layer and Pub_j . Thus, the verification roots are not freely chosen by the user. If roots appear as Groth16 public signals, the public layer first checks them against the active root set, so a stale-root proof fails for new ordinary admission. A regulated entry point may verify and execute atomically. If admission and execution are split, the execution entry point checks ρ_j , recomputes the digest of tx_j , compares it with h_j^{body} in Pub_j , and consumes ρ_j or an application-side nonce. The set UsedBind prevents replay of the admission record itself.

Table 2 gives the main relation groups enforced by C_{tx} . They are checked in the same proof, so the accepted record cannot be assembled from pieces prepared for different contexts.

Table 2. Relation groups in the ordinary admission proof.

Relation Group	Core Constraint
Registration and state	Registration leaf opens correctly, and state is current and active
Hidden home domain	Directory key and policy leaf use the same hidden d_u
Transaction context	Public fields, H_j , and E_j^{tr} are derived under one ctx_j
Policy check	$bucket_j$ is valid and $\text{EvalPolicy}(P_{d_u}, A_u, X_j) = 1$
Trace-object binding	E_j^{tr} is formed with the same trace token and transaction context

Here, X_j denotes the policy-facing inputs derived from TxIn_j , such as the transaction category, destination domain, amount bucket, and other public policy parameters. It is different from X_{tx} , which is the public statement assembled for Groth16 verification. The policy check uses the hidden authorization vector A_u and the hidden home-domain policy P_{d_u} . The base circuit uses a fixed-template policy language for transaction-category authorization, cross-domain permission, amount-bucket limits, and risk-tier limits. The verifier learns the final result $ok_j = 1$, but not the home domain, authorization attributes, amount bucket, cross-domain relation, or the internal branch that made the transaction admissible. This policy layer is not a general-purpose regulatory rule engine. Rules that depend on dynamic risk scoring, sanctions-list screening, cross-transaction history, or complex legal interpretation must be preprocessed into registered attributes, state roots, or policy roots, or handled by the surrounding admission-aware application.

The trace check prevents the recovery object from being chosen independently of the admitted transaction. The public field E_j^{tr} must be generated with the same trace token, home-domain recovery key, transaction context, and replay binding used elsewhere in the proof. As a result, the user cannot attach a valid-looking but unrelated trace object. Since E_j^{tr} remains closed in ordinary admission and is tied to this transaction instance, it also does not become a reusable public trace tag.

4.4. Warrant-Based Trace Recovery

Warrant-based trace recovery is an exceptional path. It is not triggered by the mere presence of E_j^{tr} . For an accepted record, Pub_j already contains E_j^{tr} , and Π_j has checked that this object is well formed and bound to the same transaction context. Public verifiers can verify this fact but cannot open the object.

A recovery request must provide a valid warrant W_j for the accepted reference ρ_j . Before any opening, the workflow checks the issuer authorization, signature, target reference, chain identifier, validity window, and scope. The public layer stores only $h_j^w = H(W_j)$ and records $\text{TraceReq}[\rho_j]$. The recovery transcript is derived from ρ_j , h_j^w , the accepted record digest, and the trace-object digest, so a warrant for one transaction cannot authorize recovery for another. These values also support later audit of the recovery scope.

The recovery object has two threshold layers. At each epoch, the system maintains a global outer key $PK_N = [x_N]G$ and domain-local keys $PK_d = [x_d]G$. The secrets x_N and x_d are shared by the corresponding committees under (t_N, n_N) and (t_d, n_d) thresholds. The shares can be initialized by DKG or a deployment-specific trusted setup [44]. Key generation itself is not the contribution of this work.

During ordinary admission, the user first encrypts the trace token I_u under the hidden home-domain key PK_{d_u} and then encapsulates the inner object under the global key PK_N :

$$\begin{aligned} E_j^S &= (R_j^S, c_j^S), \quad R_j^S = [r_j^S]G, \quad c_j^S = I_u + \text{Poseidon}([r_j^S]PK_{d_u}, ctx_j, b_{u,j}, 0), \\ E_j^{tr} &= (R_j^N, C_j^{N,1}, C_j^{N,2}, C_j^{N,3}), \quad R_j^N = [r_j^N]G. \end{aligned} \quad (7)$$

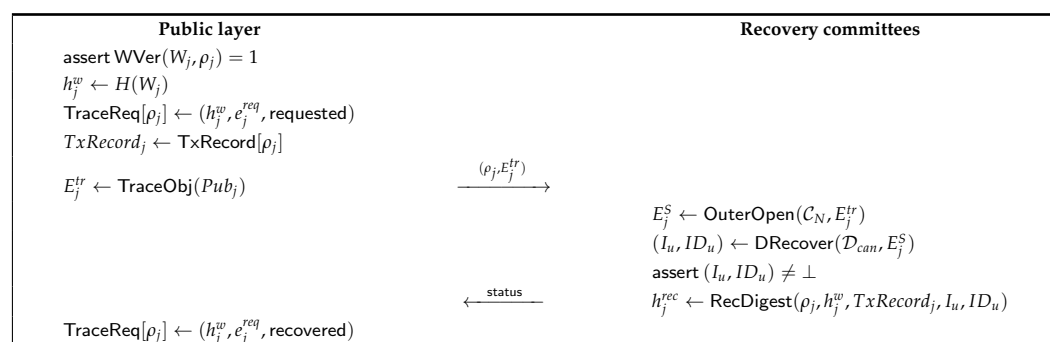
Each $C_j^{N,\ell}$ masks one field component of E_j^S with a value derived from the Poseidon hash [39] over $[r_j^N]PK_N$, ctx_j , $b_{u,j}$, and ℓ . The admission circuit checks that E_j^{tr} uses the same trace token, home-domain recovery key, global recovery key, transaction context, and replay binding. It does not open the object.

When recovery is authorized, the global committee opens only the outer layer. Member i computes $D_{N,i} = [s_{N,i}]R_j^N$ without revealing its share. Given a threshold set T_N , the workflow combines the partial openings and recovers E_j^S . Candidate domain-local committees then try inner recovery. For candidate domain d , members compute $D_{d,i} = [s_{d,i}]R_j^S$ and combine a threshold set T_d :

$$\begin{aligned} D_N &= \sum_{i \in T_N} \lambda_i^{T_N} D_{N,i} = [x_N]R_j^N, \\ D_d &= \sum_{i \in T_d} \lambda_i^{T_d} D_{d,i} = [x_d]R_j^S, \\ I_{j,d} &= c_j^S - \text{Poseidon}(D_d, ctx_j, b_{u,j}, 0). \end{aligned} \quad (8)$$

Only when $d = d_u$ does $I_{j,d} = I_u$ hold. The home domain then resolves I_u in its local database. For a non-home domain, the candidate token does not match a local identity record, and the failure is not written to the public ledger.

The recovery workflow is summarized as follows.



The recovery phase does not ask the user for new openings or consistency proofs. The consistency between the accepted transaction and E_j^{tr} has already been checked by Π_j . Therefore, recovery operates only on the trace object in the accepted $TxRecord_j$ and on the warrant bound to ρ_j .

The public layer stores only recovery-request state: the warrant digest and trace-request status for the accepted reference. Full warrants, recovery transcripts, failed domain-local attempts, identity records, and the recovery digest h_j^{rec} are not written to the public ledger by the base scheme. For auditability, the base scheme defines a minimal interface consisting of an off-chain audit bundle and an optional non-identifying public anchor. The retained bundle is

$$\text{AuditBundle}_j = (W_j, TxRecord_j, E_j^{tr}, t_j^{req}, t_j^{done}, \text{OpenArt}_j, \text{FailCom}_j, h_j^{rec}), \quad (9)$$

where t_j^{req} and t_j^{done} are request and completion timestamps, $OpenArt_j$ contains signed committee-opening artifacts, and $FailCom_j$ commits to failed domain-local attempts without publishing failed-domain labels. The audit digest and optional public anchor are

$$h_j^{audit} = H(\text{Encode}(\text{AuditBundle}_j)), \quad \text{AuditAnchor}[\rho_j] = (h_j^w, t_j^{req}, t_j^{done}, h_j^{audit}). \quad (10)$$

The anchor does not contain the warrant payload, recovered identity, home domain, recovery transcript, or failed-domain labels. An authorized auditor can compare the bundle with h_j^{audit} and check that W_j , $TxRecord_j$, E_j^{tr} , the opening artifacts, failed-attempt commitments, and h_j^{rec} all bind to the same (ρ_j, h_j^w) . The base scheme specifies only these retained and anchored items, while storage and access control for off-chain audit evidence are deployment-specific. The off-chain recovery digest binds the recovery result to ρ_j and h_j^w , but it is not a public identity or trace-token tag.

This design keeps recovery transaction-scoped. A successful recovery identifies the regulated subject associated with the warranted transaction, but it does not create a standing user-level tracing interface. Recovering another transaction requires another accepted record, another valid warrant, and the corresponding threshold openings.

4.5. Public State

The public admission and ledger layer keeps only the state needed for admission, replay prevention, and trace requests. It does not store identity secrets, hidden attributes, Merkle witnesses, trace tokens, warrant payloads, recovery transcripts, recovery artifacts, or failed domain-local attempts.

Table 3 lists the public state. Accepted records are indexed by the transaction reference ρ_j , while the replay binding $b_{u,j}$ is tracked separately in $UsedBind$. This separation gives the two values different roles: ρ_j identifies the accepted transaction, and $b_{u,j}$ prevents the same admission record from being accepted again.

Table 3. Public state maintained by the ledger layer.

State Item	Purpose
Active epoch and roots	Public parameters used to assemble X_{tx} for new admission
$\rho_j \mapsto TxRecord_j$	Accepted transaction lookup
$UsedBind$	Consumed replay bindings $b_{u,j}$
$TraceReq[\rho_j]$	Warrant digest and trace-request status for an accepted reference

During ordinary admission, the public layer assembles X_{tx} from the active roots and Pub_j , verifies Π_j , and checks $b_{u,j} \notin UsedBind$. The active roots may be stored in this layer or obtained from an authenticated root-registry interface. If roots are received as proof public signals, they must match the active roots before acceptance. If the proof is valid and the binding is fresh, it records $TxRecord_j$ under ρ_j and consumes $b_{u,j}$. A copied record carries the same binding and is rejected under the same public state.

Trace request state is updated only after warrant validation. The public layer records the warrant digest and request status, but not the warrant payload, recovery transcript, recovered identity, failed domain-local attempts, or recovery-result digest. After successful recovery, the public layer may update $TraceReq[\rho_j]$ to a recovered status. A deployment that needs post-hoc auditing can additionally publish the non-identifying $\text{AuditAnchor}[\rho_j] = (h_j^w, t_j^{req}, t_j^{done}, h_j^{audit})$ defined in the recovery interface. The corresponding off-chain AuditBundle_j retains the recovery evidence for authorized auditors, while the public anchor reveals no warrant payload, recovered identity, home domain, recovery transcript, or failed-domain labels. This audit anchor is a deployment-specific extension and is not part of the base public admission state in Table 3.

Root management follows the epoch structure. New admission requests are checked against the active roots, while historical registration, state, directory, and policy roots may be kept for interpreting old records and supporting authorized recovery. After a root update, superseded roots are not accepted for new ordinary admission. A revocation or suspension takes effect for new admissions once the updated state root becomes active. With this state layout, the ledger works as an admission and trace-request interface, not as a repository for identity records, recovery transcripts, or recovery-result digests.

5. Security Analysis

We now discuss why the base workflow satisfies the properties stated in Section 3. The arguments follow the admission and recovery relations defined in Section 4, together with the assumptions in Section 3.2. They should be read as theorem-style security arguments for the proposed construction, not as a full composable proof. For each property, we state the guarantee under the stated assumptions and make clear which part of the claim is weakened when an assumption is not met. The analysis focuses on the public transaction-layer view and the prescribed warrant-based recovery workflow. Network traffic, wallet behavior, application metadata, auxiliary off-chain data, and misuse of a legally recovered identity are outside the cryptographic claim, but their effect on the scope of the guarantee is discussed where relevant.

The analysis considers three adversarial views. In the ordinary-admission privacy view, an adversary who sees an accepted public record should not learn which enrolled user or hidden home domain produced it, except for information already implied by public transaction fields or by external metadata outside the admission relation. In the admission-soundness view, a record should not be accepted if it breaks registration, state, hidden-domain, policy, transaction-binding, replay-binding, or trace-object constraints under the active public roots and maintained replay state. These claims depend on the public layer binding new admissions to the active root set. If it accepts stale, rolled-back, or cross-epoch mixed roots, revocation freshness, policy freshness, and domain-key correctness no longer follow from the base workflow. In the unauthorized-recovery view, an adversary without a valid warrant and the required recovery shares should not recover I_u , identify the home domain, or resolve the user's real-world identity. This last claim is conditional on the bounded-threshold recovery assumption. Stronger malicious-share robustness and recovery-routing privacy are treated as backend hardening rather than as ordinary-admission requirements.

5.1. Privacy of Ordinary Admission

The privacy claim concerns the public transaction-layer view. For an accepted record, let $\text{View}_j^{\text{ord}} = (\text{Pub}_j, \Pi_j, \text{Roots}_{\text{cur}}, \mathcal{L})$, where $\text{Roots}_{\text{cur}}$ are the active public roots and $\mathcal{L}$ is the public ledger state. The question is whether this view reveals identity-side information beyond the public transaction fields and admissibility result.

Theorem 1 (Ordinary-admission privacy). *Assume that Groth16 [8] is zero-knowledge, the commitments and trace object hide their witnesses, the hash-based bindings are one-way and collision-resistant, and fresh transaction randomness is used where required. Assume further that $\text{View}_j^{\text{ord}}$ contains only Pub_j , Π_j , $\text{Roots}_{\text{cur}}$, and $\mathcal{L}$. Then $\text{View}_j^{\text{ord}}$ does not reveal the user's identity, home domain, authorization vector, amount witness, trace token, or Merkle witnesses, except for what is implied by Pub_j and the admissibility result. It also gives no stable public identity or trace-token handle for linking the user's other ordinary admitted records at the protocol layer.*

Proof. The public view contains Pub_j , Π_j , active public roots, and ledger-side state. The registration, state, domain, and policy leaves, together with their Merkle paths, are private inputs to the prover. By zero knowledge, Π_j reveals only that these witnesses satisfy the admission relation. It does not reveal the opened leaves, their positions, the home domain, the authorization vector, the amount witness, the trace token, or the Merkle witnesses.

The public values in Pub_j are transaction-specific. The admission tag, replay binding, policy handles, and trace object are bound to the transaction context and generated with fresh randomness where needed. Public transaction fields such as destination domain may be visible, but they are not the hidden home domain or identity-side witnesses. Under the hiding and hash assumptions, these values do not reveal the admission secret, trace-token secret, hidden attributes, or amount witness and do not form a fixed protocol identifier across ordinary admissions.

The policy relation reveals only whether the transaction is admissible under the active roots and context. The verifier does not learn the hidden home-domain policy, internal policy branch, authorization vector, amount bucket, or cross-domain relation.

The trace object only prepares a later recovery path. It is checked for well-formedness by Π_j , remains closed in ordinary admission, and is bound to the transaction context and replay value. Opening it requires the warrant-based recovery workflow and threshold cooperation. Therefore, $View_j^{ord}$ reveals no identity-side information beyond public fields and admissibility. $\square$

If the zero-knowledge, hiding, hash, or freshness assumptions are weakened, the corresponding privacy or unlinkability claim is weakened. If the adversary also observes information outside $View_j^{ord}$, including network traffic, wallet behavior, timing patterns, application metadata, or other auxiliary data, then correlation may still be possible even though the admission object itself contains no stable identity anchor. Such leakage is handled by deployment-level network and application protections, not by the public admission relation.

5.2. Admission Soundness and Replay Resistance

Ordinary admission should not accept a record assembled from stale roots, inconsistent hidden-domain data, failed policy checks, a misbound transaction body, or a malformed trace object. Replay resistance is handled by the public binding state.

Theorem 2 (Admission soundness and replay resistance). *Assume that Groth16 [8] is sound, the Poseidon hash [39] used in commitments and Merkle-root relations [38] is collision-resistant, and the public layer correctly maintains the active root set, UsedBind, and the public admission state. If $TxRecord_j = (Pub_j, \Pi_j)$ is accepted, then it satisfies the ordinary admission relation enforced by C_{tx} under the active roots used by the public layer. After acceptance, no later submission with the same replay binding $b_{u,j}$ can be accepted under that maintained state.*

Proof. The public layer accepts only after verifying $\text{AdmVerify}(vk_{tx}, X_{tx}, \Pi_j) = 1$ on the statement assembled from the active roots and Pub_j . If roots are carried as public signals, they must match the active root set before acceptance. By Groth16 soundness, an accepting proof implies, except with negligible probability, a witness satisfying C_{tx} . The accepted record therefore satisfies the registration, state, hidden-domain, policy, transaction-binding, replay-binding, and trace-object relations encoded in the circuit.

The active-root binding excludes stale or inconsistent admission state from new ordinary admission. The witness must open the registration, current-state, directory, and policy leaves under the active roots, and the directory and policy leaves must use the same hidden home domain. Collision resistance prevents inconsistent openings except with

negligible probability. A prover therefore cannot mix recovery-key and policy domains, or reuse obsolete state, directory, or policy roots for a new admission when the public layer verifies only against the active root set.

The proof also ties all transaction-side values to one context. The tag, replay binding, policy handles, trace object, and public fields are checked under the same witness assignment. The body digest h_j^{body} is part of Pub_j and ctx_j , so changing the transaction body, entry point, operation type, public action fields, nonce, or validity window changes the accepted statement. The private policy relation and trace-object formation are therefore tied to the admitted user and transaction, which prevents component mixing across contexts. A copied record carries the same h_j^{body} and replay binding and cannot be rebound to another action if the regulated entry point checks the body digest.

The replay check is a public-state check. When the record is accepted, the public layer inserts $b_{u,j}$ into UsedBind. Any later submission with the same binding fails the freshness check under the maintained state. Thus, an accepted record satisfies the admission relation and cannot be admitted again with the same binding. $\square$

If active-root maintenance fails, the proof may verify against a root that no longer represents the intended registration, state, directory, or policy view. If the public layer accepts stale, rolled-back, or cross-epoch mixed roots, revocation freshness, policy freshness, and domain-key correctness are no longer guaranteed. If UsedBind is rolled back, reset, or forked without finality protection, replayed bindings may be accepted again. If a split deployment executes tx_j without checking its digest against h_j^{body} in the accepted Pub_j , or executes the same accepted reference more than once, the surrounding regulated path violates the admission interface. Ordering or front-running can at most copy the same bound transaction body at the admission layer. Preventing unauthorized same-body execution depends on entry-point checks such as caller, signature, nonce, deadline, or beneficiary checks. These cases weaken the admission-soundness or replay-resistance claim but do not change the relation enforced by C_{tx} .

5.3. Warrant-Based Recovery

For recovery, the relevant question is not only whether a trace object can be opened but also who can open it and for which accepted transaction. In the ordinary public view, E_j^{tr} should remain a closed object and should not reveal the home domain, the trace token, or a public handle for following the user across other admitted records. Once a valid warrant is recorded, the visibility boundary changes. The authorized workflow may recover the identity for the warranted transaction. Candidate-domain recovery attempts are dispatched in parallel, and failed non-home attempts are not written to the public ledger or included in the public recovery result.

Theorem 3 (Warrant-bound transaction-scoped recovery). *Suppose warrant validation is enforced, the trace object hides the inner token under the outer and inner recovery layers, the recovery committees satisfy the bounded-threshold assumption, and recovery outputs are accepted only through the prescribed workflow for the target reference. Then, the public trace object E_j^{tr} by itself does not allow identity recovery. An adversary without a valid warrant and the required recovery shares cannot recover I_u , identify the home domain, or resolve the user's identity from the public record, except with negligible probability. Any recovery result accepted by the prescribed workflow is tied to both an accepted $TxRecord_j$ and the warrant digest recorded for the same reference ρ_j . The workflow therefore gives transaction-scoped recovery, not a reusable user-level tracing interface.*

Proof. The public value E_j^{tr} is only the outer layer of the trace object. Ordinary verifiers see this value and can check, through Π_j , that it was formed consistently with the admitted transaction. They do not hold the global outer shares, the domain-local inner shares, the hidden home domain, or the trace token. Under the hiding property of the trace construction and the bounded-threshold assumption, this public value does not give enough information to recover I_u or resolve an identity.

A recovery execution must pass the warrant checkpoint before any opening is accepted. The workflow takes the warrant W_j , the accepted record $TxRecord_j$, and the recorded request state $TraceReq[\rho_j]$ as linked inputs. It checks the issuer signature, target reference, chain identifier, validity window, and scope, and the public layer records $h_j^w = H(W_j)$ for the same ρ_j . If the warrant target, recorded request state, and accepted reference do not agree, the recovery attempt is rejected. For post hoc audit, an authorized auditor can compare the retained $AuditBundle_j$ with h_j^{audit} and check that W_j , $TxRecord_j$, E_j^{tr} , opening artifacts, failed-attempt commitments, and h_j^{rec} all bind to the same (ρ_j, h_j^w) . This argument treats the warrant issuer as the legal authorization trust anchor. Whether the issuer should have granted the warrant is a governance question outside the cryptographic workflow.

The consistency between recovery and admission is fixed before recovery starts. During ordinary admission, Π_j already checks that E_j^{tr} is generated from the hidden trace token, the hidden home-domain recovery key, the global recovery key, the transaction context, and the replay binding used in the accepted record. The recovery phase therefore operates on the trace object carried by $TxRecord_j$ and does not rely on a new opening supplied by the user. A transcript built for another context or another replay binding cannot be accepted for the reference ρ_j .

The two-layer recovery path limits what is exposed at each step. The global opening recovers only the inner trace object. Domain-local recovery resolves a matching identity only at the home domain, while a below-threshold coalition cannot derive the required opening value. Failed non-home attempts are not written to the public ledger or included in the public recovery result. Non-home domains only obtain local recovery failures and cannot determine which other domain is the home domain from their own local attempt.

After successful recovery, the home domain resolves the identity associated with the warranted transaction. This result is not a standing decryption key. The base public layer records only the trace-request state for the target reference, while the full recovery evidence remains off-chain. The minimal audit interface may retain this evidence in $AuditBundle_j$ and may publish the non-identifying $AuditAnchor[\rho_j]$. The public anchor contains only digests and timestamps and does not publish the warrant payload, recovered identity, home domain, recovery transcript, or failed-domain labels. Recovering another transaction requires another accepted record, another valid warrant, and the corresponding threshold openings. $\square$

If warrant validation is bypassed or the issuer is abused, recovery may be triggered outside the intended legal scope. If a coalition reaches the relevant threshold, the corresponding trace layer is no longer private. If committee members return malformed partial openings, recovery correctness and availability require verifiable-share hardening, such as verifiable secret sharing, distributed key generation, or verifiable partial openings. These mechanisms are standard threshold-backend hardening techniques and do not change Pub_j , Π_j , $TxRecord_j$, or the public admission state. If a deployment reveals labeled per-domain responses outside the home-domain side, the successful home domain may be inferred from the matching response. Hiding this internal recovery-side signal requires private forwarding, anonymous routing, or an aggregation mechanism that does not expose labeled per-domain outcomes outside the authorized recovery result. If $AuditBundle_j$ is not retained, or if an audit-requiring deployment does not publish the corresponding anchor,

the recovery relation remains transaction-scoped, but later external audit cannot fully verify the warrant, openings, failed-attempt commitments, timestamps, and recovery result.

If batch recovery is supported, the batch warrant must explicitly list the target references, and the public layer should keep one TraceReq state for each target. Broad user-level or pattern-level warrants are outside the base interface. The base workflow also provides transaction-scoped identity recovery rather than reason-dependent selective disclosure. The handles C_j^{amt} and C_j^{risk} are extension points for future disclosure mechanisms, but the current workflow does not reveal different subsets of identity or policy information for different warrant scopes.

5.4. Scope of the Analysis

The above arguments are limited to the base transaction-admission workflow under the assumptions stated in Section 3.2. The scheme relies on epoch-based public roots, so revocation takes effect after the updated state root becomes active rather than instantly. If the public layer accepts stale, rolled-back, or cross-epoch mixed roots, revocation freshness, policy freshness, and domain-key correctness are outside the guarantee. Its policy layer is intentionally circuit-friendly. The base circuit supports transaction-category authorization, cross-domain permission, amount-bucket limits, and risk-tier limits, while more expressive compliance rules require additional templates, preprocessed attributes, state roots, policy roots, or a richer policy circuit. The hidden amount witness only supports private amount-based admission checks and does not make an otherwise transparent asset transfer confidential.

The recovery path should also be understood as a protocol-level interface, not a complete governance or production threshold system. The prototype uses an honest-but-curious threshold-opening backend and does not implement malicious-share robustness, publicly verifiable partial openings, proactive resharing, or committee churn. These components can be added with VSS, DKG, or verifiable-opening techniques without changing Pub_j , Π_j , $TxRecord_j$, or the public admission state. The base public state keeps only trace-request anchors. Full warrants, recovery transcripts, signed opening artifacts, failed-attempt commitments, timestamps, and recovery-result digests remain off-chain audit artifacts. The minimal audit interface in Section 4 defines $AuditBundle_j$ and the optional non-identifying $AuditAnchor[\rho_j]$ for retaining and anchoring these items. External audit depends on retaining them, while storage and access rules for off-chain evidence are deployment-specific.

The base scheme does not claim complete selective-disclosure security. The handles C_j^{amt} and C_j^{risk} are extension points, not a full disclosure protocol. The analysis also covers only the transaction-layer public view. Network metadata, wallet behavior, repeated transaction patterns, application metadata, and auxiliary data may still support correlation attacks, as shown by Biryukov et al. [45]. These deployment-level risks are outside the public admission relation and should be handled by network, wallet, and application-layer protections.

6. Implementation

This section presents an end-to-end research prototype of the base workflow in Section 4. The prototype combines an EVM-compatible public admission contract with off-chain modules for enrollment, root preparation, transaction-object construction, Groth16 proving and verification [8], replay checking, warrant validation, and committee-based trace recovery. Policy-binding handles are implemented only as hooks for future selective-disclosure extensions.

6.1. Prototype Instantiation

We implemented a hybrid prototype of the base workflow. The on-chain part consists of a Groth16 verifier and a PublicAdmission contract, which verifies admission proofs, records accepted transaction references and digests, maintains UsedBind, and stores TraceReq anchors and status values. Off-chain modules handle enrollment, root and witness preparation, construction of Pub_j , Groth16 proof and verification [8], warrant validation, threshold recovery, and home-side identity lookup. The prototype fixes one root bundle for each experimental run and does not implement a production RootRegistry or root-update governance. Such deployments can connect PublicAdmission to the active-root interface in Section 4. Identity records, Merkle witnesses, warrant payloads, recovery transcripts, failed non-home recovery attempts, and recovery-result digests remain off-chain unless a deployment adds a separate audit interface. The prototype uses Groth16 over the bn128 curve, following common EVM zkSNARK-verification deployments based on bn128 arithmetic and pairing precompiles [46,47].

The prototype is designed to exercise the admission and recovery layer itself, rather than a full contract-gated application, gateway service, or chain-level enforcement policy. The circuit includes a field-level honest-but-curious trace-object relation that binds the public trace object to the hidden trace token, home-domain recovery key, transaction context, replay binding, and trace randomness. This check validates the admission-time consistency needed by the base workflow, but it should not be read as a malicious-secure threshold-encryption circuit.

6.2. Contract and Recovery Realization

Figure 2 shows the implementation-level realization of the prototype. (In Figure 2, “(...)” denotes omitted function arguments for brevity). Different from the framework overview in Section 4, this figure describes the executable modules and the data exchanged among them. The user-side modules prepare Pub_j , the witness bundle, and the Groth16 proof Π_j , while the witness bundle remains off-chain. The public-admission path is implemented by the generated TxAdmissionVerifier and the PublicAdmission contract on a local EVM chain. The recovery path is mainly off-chain, but it is anchored by the on-chain accepted-record and trace-request states.

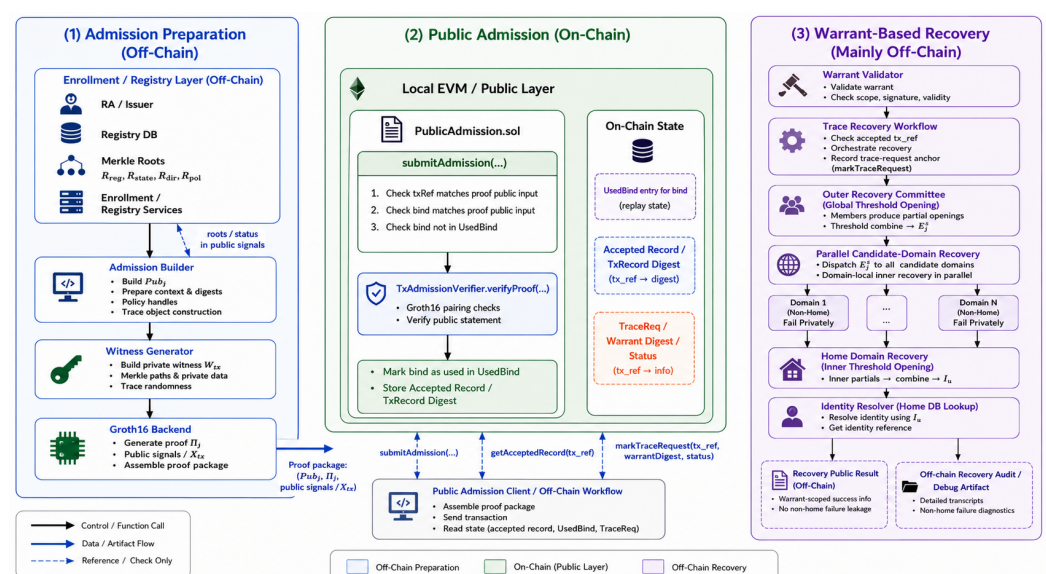


Figure 2. Prototype realization and module interactions.

The PublicAdmission contract realizes the public decision boundary of ordinary admission. For a submitted admission package, the caller invokes submitAdmission with the transaction reference, replay-binding value, canonical digests, Groth16 proof elements, and the public signal vector assembled by the off-chain workflow. The prototype treats this call as admission anchoring. It verifies the proof-bearing package and records the accepted reference and canonical digests but does not implement a regulated business executor. The signal vector is generated under a fixed experimental root bundle. A production contract can check root signals against an active-root registry before proof verification. The contract checks that the transaction reference and replay-binding value match the corresponding public signals, calls the generated TxAdmissionVerifier to verify the Groth16 proof, and rejects the submission if the replay-binding value already appears in UsedBind. If the checks pass, the contract consumes the binding and records the accepted-record entry. The on-chain state therefore contains only admission anchors and replay-control state, including accepted-record digests and UsedBind. Identity records, private witnesses, Merkle witnesses, and full transaction-side artifacts remain off-chain.

The Solidity-side logic is summarized in Algorithm 1. The algorithm abstracts the implemented contract checks rather than listing Solidity source code line by line. It shows that proof verification, replay-state update, accepted-record storage, and trace-request marking are separated at the contract boundary. A contract-gated deployment can verify and execute atomically, or it can keep execution split and require the execution entry point to check ρ_j , recompute the digest of tx_j , compare it with h_j^{body} in Pub_j , and consume ρ_j or an application-side nonce before execution.

Algorithm 1 Solidity-side admission and trace-request checks in PublicAdmission

```

1: procedure SUBMITADMISSION( $\rho_j, b_{u,j}, d_{stmt}, d_\pi, d_{rec}, p_A, p_B, p_C, signals$ )
2:   reject if  $\rho_j = \perp$  or  $b_{u,j} = \perp$  or  $d_{stmt} = \perp$  or  $d_\pi = \perp$  or  $d_{rec} = \perp$ 
3:   reject if  $acceptedRecords[\rho_j].exists$ 
4:   reject if  $usedBindings[b_{u,j}] \neq \perp$ 
5:   reject if  $signals[5] \neq \rho_j$  or  $signals[15] \neq b_{u,j}$ 
6:    $ok \leftarrow TxAdmissionVerifier.verifyProof(p_A, p_B, p_C, signals)$ 
7:   reject if  $ok = 0$ 
8:    $acceptedRecords[\rho_j] \leftarrow (\rho_j, b_{u,j}, d_{stmt}, d_\pi, d_{rec}, block.timestamp, 1)$ 
9:    $usedBindings[b_{u,j}] \leftarrow \rho_j$ 
10:  emit TxAdmitted( $\rho_j, b_{u,j}, d_{stmt}, d_\pi, d_{rec}$ )
11: end procedure

12: procedure MARKTRACEREQUEST( $\rho_j, h_j^w, s_j^{req}$ )
13:  reject if  $\rho_j = \perp$  or  $h_j^w = \perp$  or  $s_j^{req} = 0$ 
14:  reject if  $\neg acceptedRecords[\rho_j].exists$ 
15:   $traceRequests[\rho_j] \leftarrow (\rho_j, h_j^w, s_j^{req}, block.timestamp, 1)$ 
16:  emit TraceRequested( $\rho_j, h_j^w, s_j^{req}$ )
17: end procedure

```

The trace-request entry point is deliberately separate from ordinary admission. It records only the warrant digest and status for an already-accepted transaction reference. The current prototype contract does not store recovery transcripts, identity records, failed domain attempts, recovery-result digests, or a dedicated recovery-audit mapping. These artifacts remain off-chain. In deployments that enable the minimal audit interface, they can form $AuditBundle_j$, while $AuditAnchor[\rho_j]$ can be published as an optional non-identifying anchor. The prototype does not implement this dedicated audit-anchor mapping, and supplementary experiment artifacts retain recovery materials only for testing and benchmarking.

For warranted recovery, the workflow checks that the target reference ρ_j names an accepted transaction, validates the warrant for that accepted reference, records or updates the trace-request state through `markTraceRequest`, and runs the off-chain recovery backend. The recovery backend performs the outer threshold opening to recover E_j^S and dispatches E_j^S to all candidate domain-local recovery modules in parallel. Only the home domain can complete the inner recovery and resolve the trace token I_u through its local identity resolver. Non-home domains fail locally, and their failures are not returned as part of the public recovery result.

The public recovery result is restricted to warrant-scoped success information and does not contain failed non-home domains or per-domain attempt records. Detailed recovery results, off-chain audit artifacts, recovery-result digests, and non-home failure diagnostics remain off-chain. In deployments with the minimal audit interface, these materials can be retained in `AuditBundlej`, while public observers see only non-identifying anchors such as the warrant digest, trace-request status, timestamps, and an optional audit digest. This implementation boundary keeps ordinary admission separate from exceptional accountability. The on-chain contract records admission and trace-request anchors, while identity resolution remains a home-domain off-chain operation. The prototype source code is available in a public GitHub repository at <https://github.com/zhangyiyin/regulated-tx-admission-prototype> (accessed on 9 June 2026), with the evaluated version identified by commit 129777de4dfa05622acffc7976a2a7cf879eade7. Generated Groth16 artifacts, generated proof artifacts, raw logs, and complete experiment artifacts are provided separately as supplementary material.

7. Evaluation

The evaluation focuses on whether the proposed admission and recovery layer can be realized with concrete proof, contract, replay-control, and recovery components, and on the overhead introduced by these components. The measurements cover ordinary admission, warrant-based recovery, and the functional boundary with representative baselines. For ordinary admission, we measure transaction-object generation, witness generation, proof generation, public verification, local EVM submission, replay rejection, and object size. For recovery, we measure trace-request recording, local layered recovery, and parallel candidate-domain dispatch. The goal is not to match transparent transactions, which do not provide hidden-domain admission, private policy checking, checked trace preparation, or warrant-scoped recovery. Instead, the experiments quantify the cost of adding these functions to regulated public-chain transaction paths and identify the dominant sources of overhead. Unless otherwise stated, on-chain measurements are obtained on a local EVM testbed, and recovery scalability is reported as a local recovery-backend microbenchmark rather than as production-distributed-domain deployment latency.

7.1. Experimental Setup

All experiments ran on Ubuntu 22.04.5 LTS in a VirtualBox virtual machine with four vCPUs of an Intel Core i9-10885H and 7.8 GiB RAM. The prototype used Python 3.11.11, Node.js 24.14.1, npm 11.11.0, Hardhat 3.4.5, solc 0.8.28, circom 2.2.3, and snarkjs 0.7.6 [48,49] on a local Hardhat EVM chain with chain ID 31337. Before measurement, we fixed the code version, circuit artifacts, Groth16 keys, verification key, Solidity verifier, PublicAdmission contract, and benchmark configuration. Recovery benchmarks use an optimized Poseidon helper for repeated recovery-side hash evaluations. The results should therefore be interpreted as local prototype measurements, not public-chain deployment results.

The benchmarks record off-chain runtime, local EVM gas and latency, and artifact sizes under fixed public-root, policy, transaction-category, and recovery settings. Each run uses fresh bindings and isolated runtime state where replay or trace-request state may affect results. Recovery and domain-scaled recovery discard five warm-up runs and report mean and standard deviation over 30 kept runs, following the same presentation style as the ordinary-admission measurements. Negative-path checks cover invalid proofs, replayed bindings, invalid warrants, unaccepted references, insufficient shares, and non-home recovery failures. Recovery benchmarks use a persistent Poseidon worker to avoid measuring repeated Node/circomlibjs process startup. The original subprocess backend is retained only as a consistency-tested fallback.

7.2. Ordinary Admission Performance

This subsection reports the cost of ordinary admission. In each run, the user constructs Pub_j , prepares W_{tx} , generates Π_j , and submits $TxRecord_j$ to the public layer. The verifier assembles X_{tx} , verifies Π_j , checks UsedBind, and records the transaction when the binding is fresh. These measurements characterize one proof-bearing transaction record that jointly covers registration, state, domain-directory, and policy-root membership checks, hidden home-domain consistency, fixed-template policy evaluation, amount-bucket computation, transaction binding, replay binding, policy handles, and trace-object well-formedness. Table 4 reports the resulting circuit size and proof-artifact sizes.

Table 4. Admission circuit and proof artifacts.

Item	Value
Proving system/curve	Groth16/bn128
Wires/constraints	28,383/28,370
Public/private inputs	20/90
Witness size	908,332 bytes
Raw proof size	806 bytes
Public signals size	1260 bytes
Wrapped Π_j size	3770 bytes

Table 5 reports per-transaction costs over 30 independent runs. The builder-core row is a local microbenchmark with proof generation replaced by a stub. It covers public-object construction, policy handles, trace-object construction, statement-digest preparation, and witness-bundle packaging. It is reported separately from Groth16 proof because the prototype keeps reproducibility-oriented JSON artifacts and debug bundles during preparation. Reproducibility-oriented workflow preparation is not treated as optimized online prover latency.

Table 5. Cost of ordinary admission.

Metric	Value
Builder-core object construction	4705.7 ± 445.0 ms
Witness generation	244.4 ± 36.9 ms
Groth16 proving	3398.7 ± 418.8 ms
Groth16 verification	925.7 ± 170.5 ms
Local EVM admission latency	3032.7 ± 390.7 ms
Local EVM admission gas	535,986.2 ± 21.1
Replay rejection latency	1357.9 ± 162.5 ms
Replay rejection result	rejected 30/30
Size of Pub_j	1441.0 bytes
Size of $TxRecord_j$	5462.9 bytes

The results indicate that Groth16 proof is the main off-chain cryptographic cost, while local EVM admission is mainly affected by proof verification and public-state updates. The reported gas is the cost of placing the admission decision on an EVM-compatible public layer and does not include the underlying asset transfer. The public objects remain moderate in size: Pub_j is about 1.4 KB and $TxRecord_j$ is about 5.5 KB. Replayed submissions were rejected in all 30 runs, confirming that the local EVM path enforces the replay-binding state. These results support the intended role of ordinary admission: the public layer can verify admissibility under active roots and reject repeated submissions without learning the user's identity, home domain, hidden attributes, amount witness, trace token, or Merkle witnesses.

7.3. Recovery Performance

This subsection reports the cost of warrant-based trace recovery. Unlike ordinary admission, recovery is activated only when a valid warrant targets an accepted transaction, so its cost is interpreted as the overhead of the exceptional accountability path. In each test, the workflow checks that the target reference ρ_j names an accepted transaction, validates the warrant for that accepted reference, records the trace-request anchor through `markTraceRequest`, and then runs the off-chain recovery backend. The recovery backend extracts E_j^{tr} , performs the outer threshold opening to recover E_j^S , and dispatches E_j^S to all candidate domain-local recovery modules in parallel. Only the home domain can complete the inner recovery and resolve the trace token through its local identity resolver, while non-home attempts fail locally and remain off-chain.

Table 6 reports the recovery measurements under parallel candidate-domain dispatch. Results are reported as mean and standard deviation over 30 kept runs after discarding five warm-up runs. The full workflow row measures the prototype recovery path, including accepted-record checking, warrant handling, trace-request processing, recovery execution, benchmark-artifact handling, and result packaging. The local layered recovery and domain-dispatch rows isolate the local recovery backend. Trace-request recording is listed separately because it is the only recovery-stage operation executed by the `PublicAdmission` contract.

Table 6. Warrant-based recovery cost.

Metric	Latency/Cost
Full workflow recovery latency	3177.002 ± 988.375 ms
Local layered recovery latency	8.458 ± 2.722 ms
Domain dispatch wall latency	3.836 ± 1.259 ms
Domain-attempt max latency	1.940 ± 0.871 ms
Home-domain attempt latency	1.879 ± 0.890 ms
TraceReq marking gas	$137,693.0 \pm 5.5$ gas
Dispatch mode	Parallel
Non-home failures in public result	No
Runs	30

The results indicate that the end-to-end recovery workflow is dominated by orchestration overhead, while the on-chain part is limited to recording the trace-request anchor. The trace-request call does not perform threshold opening, domain-local recovery, identity lookup, or non-home failure reporting on-chain. The local layered recovery backend takes 8.458 ± 2.722 ms, and the parallel candidate-domain dispatch wall time takes 3.836 ± 1.259 ms. This shows that the recovery backend itself is small compared with the full workflow overhead, which includes local EVM interaction, benchmark artifacts, and orchestration. The outer opening recovers only E_j^S , not the user's identity or home domain.

Inner recovery succeeds only at the home domain, and the home-domain identity resolver produces the authorized recovery result.

The recovery workflow sends E_j^S to all candidate domains in parallel, waits for all attempts to complete, and then emits the public result. The public recovery result does not contain failed non-home domains or per-domain attempt records. In the prototype, detailed recovery transcripts, recovery-result digests, and non-home failure diagnostics remain off-chain and are retained in experiment artifacts only for testing and benchmarking. The benchmark uses a persistent Poseidon worker, so the reported local recovery and dispatch latencies do not include repeated Node/circomlibjs process startup for each hash call. These measurements support the intended separation between ordinary admission and exceptional recovery. Ordinary admission only carries a checked trace object, while identity recovery is performed later under a valid warrant and remains bound to the accepted reference ρ_j .

7.4. Scalability and Comparison

We first evaluate recovery scalability under parallel candidate-domain execution. Table 7 reports a recovery-only local microbenchmark with 1, 2, 4, 8, and 16 candidate domains. In each configuration, the outer opening result E_j^S is dispatched to all candidate domains in parallel. Non-home domains fail locally, and only the home domain resolves the trace token. Results are reported as the mean and standard deviation over 30 kept runs after discarding five warm-up runs. The measurements should be read as local wall-clock prototype costs, not as distributed-domain deployment latency.

Table 7. Domain-scaled recovery cost.

Candidate Domains	Workflow Latency (ms)	Local Layered Latency (ms)	Dispatch Wall Latency (ms)
1	7.829 ± 1.690	4.662 ± 1.156	1.524 ± 0.262
2	7.442 ± 0.923	4.751 ± 0.720	2.258 ± 0.468
4	8.820 ± 1.947	6.196 ± 1.811	3.727 ± 0.814
8	16.366 ± 4.621	12.615 ± 3.778	9.634 ± 3.191
16	25.851 ± 13.334	21.649 ± 11.519	18.614 ± 9.610

The domain-scaled measurements show that the local recovery backend remains in the millisecond range as the number of candidate domains increases to 16. Because the workflow dispatches all candidate domains in parallel and waits for all attempts to complete, the dispatch wall-clock latency mainly reflects the slowest local attempt and local scheduling overhead. The workflow latency grows from 7.829 ± 1.690 ms with one candidate domain to 25.851 ± 13.334 ms with 16 candidate domains. These larger-domain runs oversubscribe the 4-vCPU prototype environment, so the table should be read as the measured local prototype range under parallel all-attempt completion, not as an analytical scaling bound for distributed deployment.

We then compare the proposed scheme with representative baselines from private payment systems, regulated anonymous payment systems, and collaborative regulation schemes. These works target different protocol tasks and trust boundaries, so the comparison in Table 8 clarifies functionality and design trade-offs rather than providing an apples-to-apples runtime benchmark. Zerocash [5] and Zether [6] protect transaction privacy but do not provide template-based private admission or warrant-based identity recovery for admitted public-chain transaction records. Accountable and regulated private-payment schemes add policy checks, tracing, or accountability to anonymous payments [9,10,16,17], but these functions are usually embedded in specific payment constructions. Collaborative

regulation schemes distribute or separate regulatory authority [21–24], but do not necessarily bind private policy checking, replay control, and trace-object well-formedness to one accepted transaction record.

Table 8. Functionality comparison.

Representative Baseline	Hidden tx/ Admission Data	Private Admission	Warrant Recovery	Tx-Scoped Recovery	Admission Replay Binding
Zerocash/Zether [5,6]	Tx data	No	No	No	No
Accountable private payments [9,10,16,17]	Tx data	Partial	Yes	Partial	Partial
Collaborative regulation schemes [21–24]	Partial	Partial	Partial	Partial	No
Proposed scheme	Admission-side	Yes	Yes	Yes	Yes

In the table, “Tx data” refers to addresses, transfer relations, or amounts hidden inside a private-payment construction. “Admission-side” means that the proposed scheme hides identity-side witnesses, the home domain, authorization attributes, the admission-side amount witness, and the trace token during admission, but does not make an otherwise transparent asset transfer confidential. “Admission replay binding” refers to replay control for the proposed admission record rather than to payment-layer double-spending control or nullifier mechanisms. “Partial” denotes category-level support, where the property appears only in some schemes or under a specific payment, CBDC, or regulator model. The comparison should therefore be read as a boundary summary rather than a ranking of individual systems.

The costs reported in Sections 7.2 and 7.3 should be read under this boundary. Private-payment systems mainly pay for hiding payment data inside a payment protocol. Our prototype measures a different workload in which a public admission layer checks hidden home-domain policy compliance, replay binding, policy handles, and a well-formed trace object as part of the same accepted *TxRecord_j*. The recovery measurements also belong to the exceptional warrant path, not to ordinary transaction submission. The overhead therefore reflects the price of adding publicly verifiable regulated admission and transaction-scoped accountability while keeping identity-side witnesses out of the ordinary public record.

As a gas-based public-chain execution-cost estimate, the measured EVM gas can be converted by multiplying it by the assumed gas price. At a gas price of g gwei, ordinary admission costs approximately $535,986.2 \cdot g \cdot 10^{-9}$ ETH, and trace-request marking costs approximately $137,693.0 \cdot g \cdot 10^{-9}$ ETH. These estimates are derived from the measured EVM gas of the prototype contract calls and exclude the underlying asset-transfer transaction, block-inclusion delay, and chain-specific fee components outside EVM execution gas.

8. Discussion

8.1. Regulated Entry Points on Open Public Chains

On an open public chain, it is unrealistic to assume that miners, validators, or ordinary users will enforce a new admission rule for every native transfer. A user can always stay outside the admitted path and continue to send ordinary on-chain transactions. The more practical leverage comes from regulated entry points and regulated counterparties around the chain. Exchanges, custodians, bridges, payment gateways, regulated contracts, and institutional users may already face compliance pressure from law, business policy, or risk control. These parties can require an accepted *TxRecord_j* before they process a deposit, withdrawal, bridge operation, payment, or other regulated interaction. Under this view, the framework does not try to control all blockchain activity. It defines what

it means for a transaction to be accepted by a regulated path. A user may choose not to use the framework but may also lose access to services or counterparties that require admitted transactions.

8.2. Privacy Within the Admitted Path

Once a regulated path asks for admission, a direct implementation might attach a KYC identity, account identifier, credential identifier, or reusable regulatory tag to the transaction. That design would simplify checking and later attribution, but it would also give observers a stable handle across admitted records. The admission layer in this paper takes a different route. The public layer checks that the user is registered, active, policy-compliant, replay-fresh, and trace-prepared, while the user's identity, home domain, attributes, amount witness, trace token, and proof witnesses stay hidden in ordinary admission. This privacy claim is tied to the admitted path. It does not claim to hide all behavior of a user on the underlying chain. It aims to prevent regulated admission itself from becoming a public-identity-linking mechanism.

Cross-transaction linkability may still arise in realistic public-chain deployments. Wallet-side behavior can link admitted records through address reuse, shared funding sources, deterministic fee or payment patterns, or repeated use of the same application account. Network metadata can link submissions observed by RPC providers, relayers, gateways, or peer-to-peer monitors. Timing, transaction category, destination domain, and public amount information may shrink the anonymity set. Application-layer interactions may add identifiers, sessions, device fingerprints, or business records that correlate admitted transactions outside the admission relation. These signals are not eliminated by the base proof system. The admission object itself does not add a reusable identity, home-domain, or trace-token handle. Deployments that require stronger cross-transaction anonymity should combine the admission layer with wallet separation, batching or delayed submission, network-layer privacy, amount bucketing or padding, and application-level identifier separation.

8.3. Recovery and Deployment Choices

For many regulated entry points, a purely anonymous admission result would be difficult to accept. Some form of exceptional recovery is usually needed before such services can rely on privacy-preserving admission. This is why the trace object is prepared during ordinary admission and opened only under a warrant for the target accepted transaction. The recovery path gives the regulated side an accountability mechanism, while the ordinary public record still avoids a stable identity anchor. The prototype uses fixed policy templates and an honest-but-curious threshold-opening backend to validate the full workflow. A real deployment would need richer policy rules, selective disclosure, malicious-robust threshold recovery, private recovery routing, and governance or audit rules. These extensions affect cost and governance, but they build on the same admission and recovery interface. A regulated path can require public admission without placing a reusable identity tag in the ordinary transaction record. Extending policy expressiveness would require either new fixed templates with updated policy roots or a more flexible proving layer, such as a universal policy circuit, zkVM-style policy execution, or authenticated external compliance oracles. These choices would increase proving cost and key or policy management complexity.

9. Conclusions

This paper presents a privacy-preserving transaction-admission framework for regulated cryptocurrency transaction paths on public blockchains. The framework adds an

admission and recovery layer to admission-aware applications, contracts, gateways, or chain policies without introducing a new cryptocurrency. A public transaction object and a single Groth16 proof allow the public layer to verify registration, current state, hidden home-domain membership, fixed-template policy compliance, transaction binding, replay consistency, and trace-object well-formedness while hiding the user's identity, home domain, authorization attributes, admission-side amount witness, and trace token. The checked trace object remains closed during ordinary use and can be activated only by a valid warrant for a target accepted transaction, keeping recovery transaction-scoped rather than user-scoped.

We implemented an end-to-end prototype with an EVM-compatible public admission contract and off-chain admission and recovery modules. The admission circuit contains 28,370 constraints and 20 public inputs. In 30 ordinary-admission runs, Groth16 proof took 3.40 s on average, local EVM admission consumed about 536,000 gas, and all 30 replayed submissions were rejected. For warrant-based recovery, local layered recovery took 8.46 ms on average, parallel candidate-domain dispatch took 3.84 ms on average, and trace-request marking consumed about 138,000 gas on the local EVM testbed. These results show the overhead of adding public admissibility, hidden identity-side checking, replay-enforced acceptance, and warrant-scoped accountability to regulated public-chain transaction paths. Future work includes richer policy templates, selective disclosure, proof and gas optimizations, and stronger recovery governance.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/electronics15142990/s1>, including the prototype source-code package, generated Groth16 artifacts, generated proof artifacts, raw logs, and complete experiment artifacts.

Author Contributions: Conceptualization, Y.Z., J.J. and Y.W.; methodology, Y.Z.; software, Y.Z.; validation, Y.Z.; formal analysis, Y.Z.; investigation, Y.Z.; resources, J.J. and Y.W.; data curation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, J.J., Y.W. and C.K.; visualization, Y.Z.; supervision, J.J. and Y.W.; project administration, J.J. and Y.W.; funding acquisition, Y.W. and C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2023YFB3105803. The APC was funded by the National Key Research and Development Program of China.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are included in the article and the supplementary material. The prototype source code is available in a public GitHub repository at <https://github.com/zhangyiyin/regulated-tx-admission-prototype> (accessed on 9 June 2026), commit 129777de4dfa05622acffc7976a2a7cf879eade7. The repository contains the public implementation for ordinary admission and warrant-based trace recovery. Generated Groth16 artifacts, generated proof artifacts, raw logs, and complete experiment artifacts are provided separately as supplementary material. Additional materials are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT based on GPT-5.5 Thinking for language polishing, structural editing, readability improvement, and assistance in drafting Figures 1 and 2 based on the system architecture described in the manuscript and the prototype code repository. The authors reviewed and edited the AI-assisted output, verified the correctness of the figures and text, and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 22 June 2026).
2. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
3. Meiklejohn, S.; Pomarole, M.; Jordan, G.; Levchenko, K.; McCoy, D.; Voelker, G.M.; Savage, S. A fistful of bitcoins: Characterizing payments among men with no names. In Proceedings of the 2013 Internet Measurement Conference, Barcelona, Spain, 23–25 October 2013; pp. 127–140.
4. Kalodner, H.; Möser, M.; Lee, K.; Goldfeder, S.; Plattner, M.; Chator, A.; Narayanan, A. BlockSci: Design and applications of a blockchain analysis platform. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Virtual, 12–14 August 2020; pp. 2721–2738.
5. Sasson, E.B.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. Zerocash: Decentralized anonymous payments from bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*; IEEE: New York, NY, USA, 2014; pp. 459–474.
6. Bünz, B.; Agrawal, S.; Zamani, M.; Boneh, D. Zether: Towards privacy in a smart contract world. In *Proceedings of the International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 423–443.
7. Buterin, V.; Illum, J.; Nadler, M.; Schär, F.; Soleimani, A. Blockchain privacy and regulatory compliance: Towards a practical equilibrium. *Blockchain Res. Appl.* **2024**, *5*, 100176.
8. Groth, J. On the size of pairing-based non-interactive arguments. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 305–326.
9. Garman, C.; Green, M.; Miers, I. Accountable privacy for decentralized anonymous payments. In *Proceedings of the International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 81–98.
10. Xue, L.; Liu, D.; Ni, J.; Lin, X.; Shen, X.S. Enabling Regulatory Compliance and Enforcement in Decentralized Anonymous Payment. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 931–943.
11. Camenisch, J.; Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 93–118.
12. Sonnino, A.; Al-Bassam, M.; Bano, S.; Meiklejohn, S.; Danezis, G. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. *arXiv* **2018**, arXiv:1802.07344.
13. Maram, D.; Malvai, H.; Zhang, F.; Jean-Louis, N.; Frolov, A.; Kell, T.; Lobban, T.; Moy, C.; Juels, A.; Miller, A. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2021; pp. 1348–1366.
14. Rosenberg, M.; White, J.; Garman, C.; Miers, I. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2023; pp. 790–808.
15. Piper, F.; Wolf, K.; Heiss, J. Privacy-Preserving On-chain Permissioning for KYC-Compliant Decentralized Applications. In *Proceedings of the 2025 7th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*; IEEE: New York, NY, USA, 2025; pp. 1–10.
16. Tomescu, A.; Bhat, A.; Applebaum, B.; Abraham, I.; Gueta, G.; Pinkas, B.; Yanai, A. Utt: Decentralized ecash with accountable privacy. *Cryptol. Eprint Arch.* **2022**, preprint.
17. Brugerres, M.; Languille, V.; Kuznetsov, P.; Zarfaoui, H. Fast, private and regulated payments in asynchronous networks. In Proceedings of the 7th Conference on Advances in Financial Technologies (AFT 2025), Pittsburgh, PA, USA, 8–10 October 2025.
18. Duan, J.; Wang, L.; Wang, W.; Gu, L. TRCT: A traceable anonymous transaction protocol for blockchain. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 4391–4405. [[CrossRef](#)]
19. Xue, L.; Huang, H.; Xiao, F.; Li, Q.; Wang, Z. A privacy-enhanced traceable anonymous transaction scheme for blockchain. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 1176–1191. [[CrossRef](#)]
20. Yu, Q.; Liao, S.; Wang, L.; Yu, Y.; Zhang, L.; Zhao, Y. A regulated anonymous cryptocurrency with batch linkability. *Comput. Stand. Interfaces* **2024**, *87*, 103770.
21. Wüst, K.; Kostianen, K.; Delius, N.; Capkun, S. Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 2947–2960.

22. Kiayias, A.; Kohlweiss, M.; Sarencheh, A. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1739–1752.
23. Zhaolu, T.; Wan, Z.; Wang, H. Division of Regulatory Power: Collaborative Regulation for Privacy-Preserving Blockchains. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 2533–2548.
24. Ke, Z.; Meng, B.; Zuo, C.; Wang, D. Separation of regulatory power considering users: Collaborative regulation with user data sovereignty compliance in anonymous privacy-preserving traceable blockchain. *Blockchain Res. Appl.* **2025**, 100413. [\[CrossRef\]](#)
25. Jiang, Z.L.; Xie, M.; Chen, H.; Pan, Y.; Lyu, J.; Au, M.H.; Fang, J.; Liu, Y.; Wang, X. RPSC: Regulatable privacy-preserving smart contracts on account-based blockchain. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 4822–4835.
26. Camenisch, J.; Van Herreweghen, E. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2002; pp. 21–30.
27. Boneh, D.; Boyen, X.; Shacham, H. Short group signatures. In *Proceedings of the Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 41–55.
28. Kiayias, A.; Tsounis, Y.; Yung, M. Traceable signatures. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 571–589.
29. W3C. Decentralized Identifiers (DIDs) v1.0. W3C Recommendation, 2022. Available online: <https://www.w3.org/TR/did-core/> (accessed on 22 June 2026).
30. W3C. Verifiable Credentials Data Model v1.1. W3C Recommendation, 2022. Available online: <https://www.w3.org/TR/vc-data-model-1.1/> (accessed on 22 June 2026).
31. Pauwels, P. zkKYC: A solution concept for KYC without knowing your customer, leveraging self-sovereign identity and zero-knowledge proofs. *Cryptol. Eprint Arch.* **2021**, preprint.
32. Zhang, F.; Maram, D.; Malvai, H.; Goldfeder, S.; Juels, A. Deco: Liberating web data using decentralized oracles for tls. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1919–1938.
33. Kryptos Chalkias, K.; Maram, D.; Roy, A.; Wang, J.; Yadav, A. Zero-Knowledge Authenticator for Blockchain: Policy-Private and Obliviously Updateable. In *Proceedings of the 7th Conference on Advances in Financial Technologies (AFT 2025)*; Leibniz International Proceedings in Informatics (LIPIcs); Schloss Dagstuhl–Leibniz-Zentrum für Informatik: Dagstuhl, Germany, 2025; Volume 354, pp. 2:1–2:23. [\[CrossRef\]](#)
34. Kannengiesser, N.; Lins, S.; Sander, C.; Winter, K.; Frey, H.; Sunyaev, A. Challenges and common solutions in smart contract development. *IEEE Trans. Softw. Eng.* **2021**, *48*, 4291–4318. [\[CrossRef\]](#)
35. Górski, T. AdapT: A reusable package for implementing smart contracts that process transactions of congruous types. *Softw. Impacts* **2024**, *21*, 100694. [\[CrossRef\]](#)
36. Górski, T. Smart contract design pattern for processing logically coherent transaction types. *Appl. Sci.* **2024**, *14*, 2224. [\[CrossRef\]](#)
37. Azimi, S.; Golzari, A.; Ivaki, N.; Laranjeiro, N. A systematic review on smart contracts security design patterns. *Empir. Softw. Eng.* **2025**, *30*, 95. [\[CrossRef\]](#)
38. Merkle, R.C. A digital signature based on a conventional encryption function. In *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1987; pp. 369–378.
39. Grassi, L.; Khovratovich, D.; Rechberger, C.; Roy, A.; Schofnegger, M. Poseidon: A new hash function for Zero-Knowledge proof systems. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*; USENIX Association: Berkeley, CA, USA, 2021; pp. 519–535.
40. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [\[CrossRef\]](#)
41. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*; IEEE: New York, NY, USA, 1987; pp. 427–438.
42. Pedersen, T.P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 129–140.
43. Gennaro, R.; Jarecki, S.; Krawczyk, H.; Rabin, T. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 295–310.
44. Pedersen, T.P. A threshold cryptosystem without a trusted party. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 522–526.
45. Biryukov, A.; Khovratovich, D.; Pustogarov, I. Deanonymisation of clients in Bitcoin P2P network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 15–29.

46. Buterin, V.; Reitwiessner, C. EIP-196: Precompiled Contracts for Addition and Scalar Multiplication on the Elliptic Curve alt_bn128. Ethereum Improvement Proposals, 2017. Available online: <https://eips.ethereum.org/EIPS/eip-196> (accessed on 22 June 2026).
47. Buterin, V.; Reitwiessner, C. EIP-197: Precompiled Contracts for Optimal Ate Pairing Check on the Elliptic Curve alt_bn128. Ethereum Improvement Proposals, 2017. Available online: <https://eips.ethereum.org/EIPS/eip-197> (accessed on 22 June 2026).
48. iden3. Circom 2 Documentation. Software Documentation, 2024. Available online: <https://docs.circom.io/> (accessed on 22 June 2026).
49. iden3. snarkjs: zkSNARK Implementation in JavaScript and WASM. Software Documentation, 2024. Available online: <https://github.com/iden3/snarkjs> (accessed on 22 June 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.