

Provenance Graphs for AML/CFT: Asset Tracing and Freezing in Private Payments

Maxence Brugerres

LTCI, Telecom Paris, Institut Polytechnique de Paris
Paris, France
Forvis Mazars
Paris, France
maxence.brugerres@telecom-paris.fr

Petr Kuznetsov

LTCI, Telecom Paris, Institut Polytechnique de Paris
Paris, France
petr.kuznetsov@telecom-paris.fr

Abstract

To comply with Anti-Money-Laundering and Counter-Terrorist-Financing (AML/CFT) frameworks, payment systems must allow designated authorities to infer the *provenance* of any transaction of interest, i.e., to trace back the funds that directly or indirectly enabled it. We show that a supervisor can be given the investigative capability the obligation requires, and no more.

We first fix the investigative semantics on a transparent ledger by defining the (k, t) -*provenance graph* of a target account: the subgraph of transfers lying on a time-respecting path to the target of length at most k or spanning at most t blocks. The disjunction is what makes the notion robust to some AML/CFT evasion patterns, such as layering and peeling chains, which inflate hop counts cheaply but cannot compress elapsed time. We give an extraction algorithm based on Pareto-frontier pruning for transparent ledgers.

We then address the challenge of inferring transaction provenance in a *private* payment system. We propose a generic construction that can extend any private payment system whose construction is similar to Zerocash’s. Every coin carries an *embedded* (k, t) -provenance graph such that every payment leaves a *trace* of the sender’s identity in the provenance graph of the created coins.

A designated *regulator* may blacklist a user, which simultaneously prevents her from transferring funds and freezes every coin whose active graph carries one of her traces. Upon voluntary disclosure by a coin holder, the regulator may reconstruct that coin’s provenance graph in clear text and conduct a standard investigation. Without such disclosure, the regulator learns only the sender of each payment: not the receiver, not the amount, and not the link from an output coin to its later spend. Compliance reduces to coprimality of two polynomials, certified by a Bezout identity that validators verify *outside* the zero-knowledge circuit.

We report an implementation showing low transaction proving and verification times, making our approach practical with very little impact on the base blockchain’s throughput.

CCS Concepts

• **Applied computing** → **Digital cash**; • **Security and privacy** → *Public key (asymmetric) techniques*; **Distributed systems security**.

Keywords

AML/CFT compliance, Provenance analysis, Privacy-preserving payments, Regulatory compliance, CBDC, Zero-knowledge proofs, Asset freezing

1 Introduction

Anti-money laundering and counter-terrorist financing (AML/CFT) rules require payment service providers to assess incoming funds’ *provenance*: for a target account, which earlier transfers contributed to it, how closely, and through which intermediaries? On *transparent* ledgers such as Bitcoin or Ethereum, this reduces to a graph traversal over the public transaction history. In private payment systems such as Zerocash [5] or Tornado Cash Nova [43], coins are burnt and created at each payment, while the sender, receiver, and amount are hidden. Provenance analysis is then impossible: the cryptography protects ordinary and illicit users alike. This creates an apparent deadlock, not merely a trade-off.

We avoid this deadlock by giving a *regulator* a *specific, bounded* investigative capability rather than full visibility or none. Our construction extends Zerocash but applies naturally to most private systems with the same spend-and-create coin model (Tornado Cash [43], Railgun [37], Lelantus [27]). The capability is backward provenance within a bounded neighbourhood of a transfer, together with the power to freeze funds derived from a designated illicit source, and to reconstruct a full clear-text provenance graph when a holder chooses to disclose. Everything outside that capability remains hidden from the supervisor and from every other party: receivers, amounts, the link from an output to its later spend, and the provenance of any coin that has not been disclosed.

1.1 Positioning of our work

The provenance question in transparent ledgers. We model a transparent ledger as a *multigraph* with users as vertices and transactions as edges. Because the full graph is too large to analyze in full, an investigation needs a subgraph defined by *proximity* to the target user.

We therefore need proximity criteria that keep only the most relevant transactions and users. Hop distance between the target user and other users in the provenance graph is a natural first criterion. However, the following patterns can inflate it artificially. In AML terminology, *layering* [20] moves illicit funds rapidly through many addresses to obscure their origin. In a *peeling chain*, each transfer sends (“peels”) a small amount to a destination and forwards the remainder to a fresh change address controlled by the same entity; repeating this process forms a chain [33]. Both patterns can thus evade a hop bound by artificially increasing hop counts. We therefore add a temporal constraint: an adversary can cheaply add hops, but transfers within a short time window should all remain inside the provenance boundary. We express this constraint

as *block distance*. The resulting (k, t) -provenance graph contains every transaction that is either (1) within k hops of the target or (2) on a path spanning at most t blocks. Because we trace money flows, we consider only *time-respecting* paths, whose edges have increasing block numbers.

Ideal compliance in private payments. Ideally, the *regulator* would retain transparent-ledger capabilities while preserving user privacy. We therefore seek these operational properties:

- (1) the regulator and users learn no more information per payment than Zerocash exposes;
- (2) when a user *discloses* her coin’s content, the regulator can rebuild its full provenance graph as in a transparent ledger;
- (3) the regulator can blacklist a user, making her coins and any coin whose provenance graph contains a blacklisted identity *frozen*, i.e., unspendable.

Goal (1) directly conflicts with (2) and (3). Our construction gives up a small amount of privacy to meet the other two goals. Without user *disclosure*, the regulator sees each payment’s sender and can group payments by sender, but learns neither the receiver nor the amount and cannot link an output to its later spend. A user can also learn whether some of her coins’ provenance graphs share users. The limited impact of this leakage is discussed in Section 8.

1.2 Technical overview

Provenance Graph. We meet goals (1)–(3) by equipping every coin with an *embedded* provenance multigraph. The coin commitment binds the graph and lets the sender prove compliance without revealing it. Its vertices are *identity traces*, not user identities. A user u derives one fresh trace per emitted coin as $\text{IDT}_u(\text{tc}_u, j) = \text{PRF}(\text{idk}_u; \text{tc}_u + j)$, using her registered *identity key* idk_u and per-output *transaction counter* tc_u . Each vertex also encrypts idk_u under the regulator’s public key, letting the regulator identify the emitter after coin disclosure while revealing nothing to a public observer. Pseudorandom *source handles* derived from the emitter’s identity key encode edges. They are opaque to coin holders but reconstructible by the regulator after recovering vertex identities.

Transaction Compliance. The regulator can blacklist users. Following AML/CFT semantics, blacklisting freezes the user’s own coins *and* every coin whose provenance graph contains a trace of her identity. This freezes tainted coins even when their holder is honest and not blacklisted, preventing illicit users from laundering funds through honest users. We do so efficiently as follows. The regulator knows the identity key idk_u of each user u and her transaction counter tc_u . The regulator maintains two append-only blacklists: a *versioned identity trace blacklist* and a *user blacklist*. When blacklisting u , the regulator uses her identity key and transaction counter to recompute all of her past traces and append them to the identity trace blacklist. It also adds the unique user identifier $\text{PRF}(\text{idk}_u; 0)$ to the user blacklist.

We encode the multiset of identity traces in $\mathcal{G}$ as $P_{\mathcal{G}}(X) = \prod_{\text{idt} \in \text{Traces}(\mathcal{G})} (X - \text{idt})$. Compliance means the coin’s traces do not intersect the identity trace blacklist, i.e., $\gcd(P_{\mathcal{G}}, Q) = 1$. For this coprimality test, the prover gives (U, W) satisfying the Bezout

identity $U \cdot P_{\mathcal{G}} + W \cdot Q = 1$, checked at one Fiat–Shamir [17] evaluation point. A *delta-blacklist polynomial* $Q_{a \rightarrow b}(X)$ encodes the difference between a coin’s version a (the coin is compliant with blacklist version a) and the current public version b . Versioning lets the user prove coprimality against the smaller delta polynomial $Q_{a \rightarrow b}$ instead of the full identity trace blacklist. Separately, each sender proves in zero knowledge that the reserved user identifier $\text{uid} := \text{PRF}(\text{idk}_u; 0)$ is absent from the user blacklist. To do so, the user blacklist is structured as a sorted Merkle tree, and the sender proves that $\text{PRF}(\text{idk}_u; 0)$ lies strictly between two consecutive leaves.

One non-interactive zero-knowledge (NIZK) proof, instantiated as a SNARK [24], enforces most transaction obligations: well-formed traces and source handles, the active-window update, and direct evaluation of the output provenance polynomial. The Bezout identity is checked outside the NIZK using KZG [29] commitments to its witnesses and a masked provenance polynomial to preserve privacy. As in Zerocash, transaction details remain hidden from regular users: the regulator sees only each payment’s sender, as needed to maintain user transaction counters and the blacklist. Still, we want the regulator to rebuild the provenance graph of a coin when investigations must be conducted, e.g., when an honest user has coins frozen because a malicious user appears in their graphs. We thus introduce a selective-disclosure mechanism: a user can open her coin commitment to the regulator, who decrypts each vertex’s identity in the *embedded* provenance graph and matches source handles to reconstruct the clear-text provenance graph (the one that would be accessible in a transparent ledger).

1.3 Contributions and roadmap

Contributions.

- *Formalization and extraction of (k, t) -provenance as an investigation tool for AML/CFT.* We define the (k, t) -provenance graph of a target as a subgraph of all ledger edges that lie on at least one time-respecting path to the target under the disjunctive hop (length of the path) or time (duration of the path) constraint. We give a backward-traversal algorithm for efficiently extracting the (k, t) -provenance graph of a target in a public ledger.
- *Private payments with embedded provenance and blacklisting.* In our extension of Zerocash, we equip each coin with an embedded (k, t) -provenance graph of identity traces, with every emitter encrypted under the regulator’s public key. The regulator can blacklist a user, which both prevents her from transferring funds *and* freezes every coin carrying one of her traces.
- *Efficient compliance checks.* We encode each coin’s provenance traces as a polynomial and evaluate it directly from the embedded graph inside the NIZK. Compliance reduces to a proof of polynomial coprimality with the Bezout identity, proved succinctly via KZG commitments opened at a Schwartz–Zippel challenge point.
- *Selective disclosure.* Selective disclosure lets the regulator reconstruct exactly the (k, t) -provenance graph from our first contribution when the holder of a coin opens its content to the regulator, while the holder cannot reconstruct the clear-text provenance graph of her coins.

- *Generic construction.* The construction inherits Zerocash’s coin syntax and transposes naturally to most private payment systems that spend and create coins at every transfer [27, 37, 43].
- *Security analysis with explicit residual leakage.* We give game-based definitions for compliance soundness, disclosure integrity, and regulator indistinguishability (RegIND). We extend Zerocash ledger indistinguishability with an explicit leakage function that enumerates every additional bit a public observer learns over Zerocash. We prove the construction meets this limited leakage under standard assumptions.
- *Implementation.* We have implemented our construction and demonstrated low proving and verification times, which ensure the practicality of our scheme.

Roadmap. Section 2 covers related work; Section 3 defines (k, t) -provenance graphs on transparent blockchains with an efficient extraction algorithm; Sections 4–6 present our Zerocash extension with embedded provenance and blacklisting; Section 7 reports its implementation; and Section 8 analyzes security.

2 Related Work

Temporal graphs. Our work is rooted in the temporal-graph literature, where paths must follow chronological edge order. Kempe et al. [30] introduced *time-respecting* paths in temporal graphs. We adopt the strict variant (timestamps strictly increase), as recommended when edge traversal takes nonzero time [34].

Prior work ranges from temporal DFS/BFS and reachability [26] to optimal temporal-path search [44], but does not study subgraph extraction under our combined hop-or-time proximity rule.

Blockchain transaction and forensic analysis. On transparent ledgers, early forensics (Bitloline [41]) and analytics platforms (BlockSci [28], GraphSense [25]) parse, cluster, label, and explore blockchain activity. Extensive measurement and learning work also studies transaction graphs [11, 31, 33]. Recent work adds time to fund tracing: TRacer [45] propagates personalized-PageRank scores on Ethereum’s temporal graph, while TPGraph [12] accelerates queries by partitioning temporal graphs into subgraphs for distinct transaction-time windows.

A related line studies taint and contamination propagation, including Bitcoin risk scoring and FIFO-style tracing [1, 35]. Another major direction treats AML as a transaction-graph learning problem, notably through the Elliptic dataset and recent subgraph-learning methods [4]. These works are relevant to forensics and AML, but produce rankings, labels, or faster queries rather than exact semantics for backward fund provenance.

To our knowledge, we are the first to define AML/CFT investigation semantics by extracting a blockchain temporal subgraph under disjunctive hop and time bounds.

Privacy-preserving payment systems. Zerocash [5] pioneered private blockchain payments: it hides the sender, receiver, and amount, and makes transactions *unlinkable*, so one cannot tell whether two transactions share any details. It commits coins in a Merkle tree, and each transaction carries an NIZK certifying correct coin consumption and creation. Later similar designs include Lelantus [27], Tornado Cash [43], Railgun [37], PaxPay [7],

PARScoin [40], and UTT [42]. The last three rely on weaker assumptions, without consensus.

Reconciling privacy and compliance. Privacy creates a natural compliance challenge because AML/CFT investigations are no longer possible. One line of work seeks to reconcile these goals. Garman et al. [22] were among the first to add policies such as user spending limits to Zerocash. PRCash [46] also limits each user’s receipts per epoch, a system-defined time window. Its privacy is weaker than Zerocash’s: validators can link two transactions from the same sender in an epoch, and receivers necessarily learn the sender’s identity. UTT [42] offers spending limits. Platypus [47] adds holding (assets held at once), spending, and receiving limits.

A second line adds oversight after the fact instead of enforcing it during transfers. zkLedger [36] and MiniLedger [10] let an auditor query per-bank aggregates, answered in zero knowledge. PGC [13] keeps balances under each user’s key, so policies are proved on request rather than enforced. DART [38] escrows each transaction to an asset-specific auditor who can unilaterally recover its sender, receiver, and value. Privacy Pools [8] gives no decryption power to anyone: when withdrawing from a mixing pool, a user proves that her own deposit belongs to a list of deposits considered legitimate, a list compiled by a third party rather than computed from the ledger.

PEReDi [39] and PARScoin [40] trace through threshold decryption by validators, requiring trust in them. Recipients also see sender identities, reducing privacy. PaxPay [7] matches Zerocash privacy and uses a sorted Merkle-tree blacklist to freeze a user’s own assets. Our work retains this sender check and goes further by freezing coins held by non-blacklisted users when they carry traces of blacklisted users. All these systems evaluate a predicate on one transfer or aggregated limits; none propagates a verdict from a coin to its descendants. To our knowledge, none supports regulator fund tracing or freezes assets tainted by illicit users. Instead, we enforce a predicate over a multi-hop provenance graph at spend time. Our scheme hides the sender from the receiver and, without disclosure, reveals only each payment’s sender identity to the regulator.

AML/CFT context. Our motivation follows the broader AML/CFT literature on provenance analysis, layering, and risk-based investigations. Guidance from FATF, FinCEN, Chainalysis, Elliptic, and CipherTrace highlights the need for tracing tools that handle short-hop layering [14, 16, 18], bounded time windows [9, 16], and cross-service movement [9, 14, 18, 19].

In this sense, our contribution is not another detection score, but a formal tracing primitive that provides real-time guarantees and investigation capabilities even in private payment systems. Our construction could, for instance, support Central Bank Digital Currencies (CBDCs), which also require AML/CFT compliance. That setting also requires compliance with traditional financial sanctions issued by central banks or intergovernmental organizations (e.g., the UN¹).

¹<https://main.un.org/securitycouncil/en/sanctions/information>

3 Introducing Provenance Graphs in Transparent Blockchains

We first define, on a transparent ledger, the graph semantics reproduced by our private construction.

3.1 Definitions

We represent blockchain transactions as a multigraph $G = (U, E, \text{src}, \text{dst}, \text{blc}, \text{val})$, where

- U is the set of users/addresses (vertices),
- E is the set of transactions (edges), including multiple transactions between the same addresses. Each transaction is a tuple $(\text{source}, \text{destination}, \text{block}, \text{value})$ containing its sender, receiver, block number, and value,
- $\text{src} : E \rightarrow U$ maps each transaction to its sender,
- $\text{dst} : E \rightarrow U$ maps each transaction to its receiver,
- $\text{blc} : E \rightarrow \mathbb{N}$ maps each transaction to its block, and
- $\text{val} : E \rightarrow \mathbb{N}$ maps each transaction to its value.

Time-respecting incoming set. For $u \in U$, $b \in \mathbb{N}$, $\text{In}(u, b) := \{e \in E : \text{dst}(e) = u \wedge \text{blc}(e) < b\}$.

Time-respecting path. A *time-respecting path* from user u to v is a sequence of transactions $P = (e_1, \dots, e_n)$, where $\forall i \in [1, n-1]$, $e_i \in \text{In}(\text{src}(e_{i+1}), \text{blc}(e_{i+1}))$. The *time span* of P , denoted $\Delta(P)$, is defined as $\text{blc}(e_n) - \text{blc}(e_1)$.

3.2 Provenance graph

Given a transaction multigraph G , a target address $\text{target} \in U$, a hop budget $k \in \mathbb{N}$, and a time budget $t \in \mathbb{N}$ (in blocks), define the *provenance graph* $S_{k,t} = (U_{k,t}, E_{k,t})$ as a subgraph of G . Its edge set $E_{k,t}$ consists of all transactions in E that lie on at least one time-respecting path $P = (e_1, \dots, e_n)$ to target (i.e., $\text{dst}(e_n) = \text{target}$) satisfying

$$|P| \leq k \vee \Delta(P) \leq t. \quad (1)$$

The vertex set is induced by these edges:

$$U_{k,t} := \{\text{src}(e), \text{dst}(e) : e \in E_{k,t}\}.$$

Thus, the path either has at most k hops or begins within t blocks of the target delivery. The two parameters interact as a disjunction: k guarantees a minimum exploration depth regardless of timing, while t extends exploration beyond k hops for transactions that are temporally close to the target delivery.

Let $b_p = \text{blc}(e_n)$ denote the block number of the last transaction on the path P (the transaction that delivers funds to target). Condition (1) is equivalent to checking, for every $i \in [1, n]$,

$$n - i < k \vee (b_p - \text{blc}(e_i) \leq t). \quad (2)$$

3.3 Algorithm for provenance graph extraction

Backward traversal. We extract the provenance graph by traversing G backward from target : compute its time-respecting incoming set, recurse on each source, and stop when both budgets are exhausted. Each recursive call is given a block bound b and remaining hop budget h : considering only edges with $\text{blc}(e) < b$ ensures time-respectingness. For an edge e entering target , the initial call uses $b = \text{blc}(e)$ and $h = \max(k - 1, 0)$ because this edge consumes

one hop. Within a call, an incoming edge e is kept and its source explored iff $h > 0$ or $b_p - \text{blc}(e) \leq t$, as in (2).

Pareto-frontier tracking. Different paths may reach the same address u with different parameters (b, h, b_p) . To avoid repeated recursive explorations, we maintain its *Pareto frontier* $\text{PF}(u) \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, which records all non-dominated labels. A label (b_1, h_1, b_{p_1}) *dominates* (b_2, h_2, b_{p_2}) , written $(b_1, h_1, b_{p_1}) \succeq (b_2, h_2, b_{p_2})$, iff

$$b_1 \geq b_2 \wedge h_1 \geq h_2 \wedge b_{p_1} \leq b_{p_2}.$$

An exploration call (b, h, b_p) is *redundant* iff some $(b', h', b'_p) \in \text{PF}(u)$ dominates it. This is sound because the dominating exploration covers a superset of edges: $b' \geq b$ gives $\text{In}(u, b) \subseteq \text{In}(u, b')$; $h' \geq h$ permits at least the same depth; and $b'_p \leq b_p$ makes the budget check at least as permissive.

This pruning principle comes from multi-objective shortest-path algorithms [32], but our labels track exploration budgets, not path costs, and our goal is exhaustive edge collection, not an optimal path. Thus, instead of pruning suboptimal paths, dominance prunes explorations whose edge sets a prior exploration provably subsumes. Appendix A gives the full pseudocode (Algorithm 1), comprising EXPLORE and the outer loop over edges entering target , and proves correctness and termination.

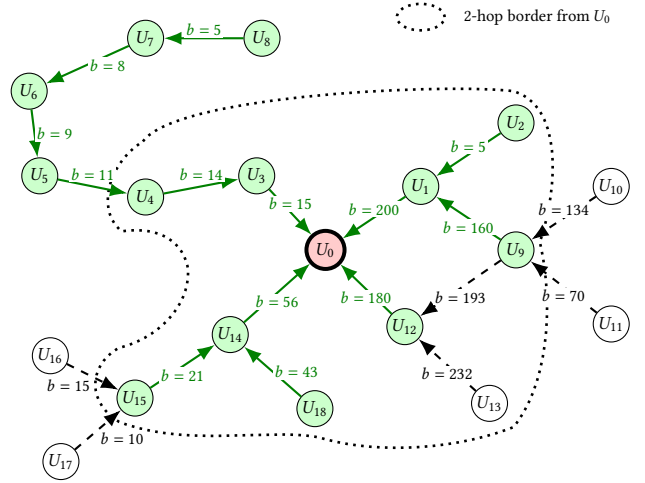


Figure 1: Transaction graph around U_0 . The dotted contour marks nodes two hops from U_0 ; green nodes and edges form its $(k=2, t=20)$ -provenance graph. U_{13} is excluded because its transaction at block $b=232$ is not on a time-respecting path to U_0 .

4 Preliminaries for our private payment construction

The remaining sections equip a private payment system with an embedded (k, t) -provenance graph, identity trace and user blacklists, and selective regulator disclosure. We build on Zerocash, but the construction can be adapted to most similar private payment systems [27, 37, 43].

4.1 Cryptographic building blocks

We assume the following standard primitives, whose security properties appear in Section 8.

Hash function H . We model $H : \{0, 1\}^* \rightarrow \mathbb{F}$ as a random oracle [3] and instantiate it with Poseidon2 [23], which has an efficient R1CS representation for the NIZK.

Private communication channels. As in Zerocash, we assume authenticated private channels between any two registered users and between users and the regulator.

Field and bilinear group. Let $\mathbb{F}$ be a finite field. All polynomial computations are in $\mathbb{F}[X]$. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of prime order $p = |\mathbb{F}|$, with generators g_1, g_2 and a non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We write $[x]_1 = g_1^x$ and $[x]_2 = g_2^x$.

KZG polynomial commitments. We use KZG [29] over $\mathbb{F}$ and the groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$. It provides the following algorithms:

- $\text{KZG.Setup}(1^\lambda, d) \rightarrow \text{pp}$ takes a security parameter λ and a degree bound d , and outputs public parameters

$$\text{pp} = ([\tau^0]_1, [\tau]_1, \dots, [\tau^d]_1, [\tau^0]_2, [\tau]_2).$$

Here $[\tau^0]_1 = g_1$ and $[\tau^0]_2 = g_2$. The setup trapdoor τ , used to generate pp , must then be destroyed.

- $\text{KZG.Com}_{\text{pp}}(P) \rightarrow C$ commits to $P \in \mathbb{F}[X]$ of degree at most d .
- $\text{KZG.Open}_{\text{pp}}(P, z) \rightarrow (y, \omega)$ outputs $y = P(z)$ and an opening proof ω for $z \in \mathbb{F}$.
- $\text{KZG.Verify}_{\text{pp}}(C, z, y, \omega) \in \{0, 1\}$ checks that the committed polynomial evaluates to y at z .

Randomized public-key encryption. We assume randomized public-key encryption $(\text{EncGen}, \text{Enc}, \text{Dec})$, where $\text{Enc}(m; \text{pk}; \eta) \rightarrow \text{cyph}$ uses randomness η and $\text{Dec}(\text{cyph}, \text{sk}) \rightarrow m$ or $\perp$. The regulator runs $\text{EncGen}(1^\lambda)$ once to obtain $(\text{pk}_{\text{Reg}}, \text{sk}_{\text{Reg}})$ for security parameter λ . We instantiate it with standard ElGamal encryption [15] in $\mathbb{G}_1$.

Pseudorandom function. We assume a family $\text{PRF} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ such that $\text{PRF}(k; \cdot)$ is indistinguishable from a random function to any adversary not knowing $k \in \mathcal{K}$. The spending secret key is the PRF key, and our analysis uses the PRF properties inherited from Zerocash. In particular, we require *collision resistance*: for independently sampled distinct keys $k \neq k'$ and any x, x' , even when the keys are known, $\Pr[\text{PRF}(k; x) = \text{PRF}(k'; x')] \leq \epsilon_{\text{PRF}}^{\text{CR}}$.

Commitment scheme. We assume a computationally *hiding* and *binding* commitment $\text{Com}(r; m) \rightarrow c$ with uniform $r \in \mathcal{R}$. We instantiate it from the PRF, using the randomness as key.

Non-interactive zero-knowledge. We assume a complete, sound, and zero-knowledge non-interactive proof system NIZK for an NP relation $\mathcal{R}$. It consists of NIZK.Setup , NIZK.Prove , and NIZK.Verify . We instantiate it as the Groth16 SNARK [24].

4.2 Zerocash construction

We recall Zerocash [5] and fix notation for our generalization.

Addresses, coins, commitments. A user has keys (pk, sk) with $\text{pk} = \text{PRF}(\text{sk}; 0)$. A *coin* is $c = (v, \text{pk}, \rho)$, where v is its value, pk the recipient address, and $\rho \in \mathbb{F}$ a uniform seed. Only its commitment $\text{cm} = \text{Com}(r; v, \text{pk}, \rho)$ appears on-chain, as a leaf in the append-only Merkle tree $\mathcal{T}_{\text{cm}}$ with public root rt .

Nullifiers. Each coin has a unique *nullifier* $\text{sn} = \text{PRF}(\text{sk}; \rho)$ derived from its owner's secret key and seed. To spend it, the owner publishes sn , which is added to $\mathcal{T}_{\text{sn}}$; validators reject a reused nullifier, preventing double spending. By PRF security, sn reveals nothing about the coin's value, recipient, or history.

Transactions and NIZK. A *transaction* (private payment) publishes nullifiers, fresh coin commitments, and an NIZK proof π . Following [7], we generalize Zerocash from two inputs and two outputs to n input and m output coins. With inputs $[(v_i^{\text{in}}, \text{pk}_i^{\text{in}}, \rho_i^{\text{in}})]_{i=1}^n$ committed as $[\text{cm}_i^{\text{in}}]_{i=1}^n$ and outputs $[(v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \rho_j^{\text{out}})]_{j=1}^m$ committed as $[\text{cm}_j^{\text{out}}]_{j=1}^m$, the sender proves:

- (Z1) **Ownership.** $\forall i \in [n], \text{pk}_i^{\text{in}} = \text{PRF}(\text{sk}; 0)$.
- (Z2) **Coin opening.** $\forall i \in [n], \text{cm}_i^{\text{in}} = \text{Com}(r_i^{\text{in}}; v_i^{\text{in}}, \text{pk}_i^{\text{in}}, \rho_i^{\text{in}})$.
- (Z3) **Merkle membership.** $\forall i \in [n], \text{cm}_i^{\text{in}}$ is a leaf under rt .
- (Z4) **Nullifier correctness.** $\forall i \in [n], \text{sn}_i = \text{PRF}(\text{sk}; \rho_i^{\text{in}})$.
- (Z5) **Value conservation.** $\sum_i v_i^{\text{in}} = \sum_j v_j^{\text{out}}$.

Validators accept the transaction if π verifies and all published nullifiers are fresh and pairwise distinct. They then insert fresh commitments into $\mathcal{T}_{\text{cm}}$ and nullifiers into $\mathcal{T}_{\text{sn}}$.

Privacy and our extension. Because the proof is zero-knowledge and coins appear only as opaque commitments, a public observer learns only the nullifiers (which break links to the original commitments), new coin commitments, and balance data intentionally exposed by the protocol. We add an embedded provenance payload to each coin and the obligations in Section 6.3 to its transaction proof, while retaining this privacy guarantee for all other fields.

5 Parties and Data Structures

5.1 Parties and storage

The system consists of three types of parties described below.

Users. A set $\mathcal{U}$ of users sends and receives private payments. Each registered user u stores

$$(\text{sk}_u, \text{pk}_u), \quad \text{idk}_u, \quad \text{ct}_u.$$

$(\text{sk}_u, \text{pk}_u)$ is the spending key pair. idk_u is the user's unique *identity key*, known to the user and issued by the regulator through Know Your Customer (KYC).

$\text{ct}_u = (\text{idk}_u, \text{pk}_u, \text{tc}_u, \rho)$ is the user's unique *compliance token*, where tc_u is a monotone counter initialized to 0 at registration and ρ is the Zerocash coin seed. Each valid transaction consumes the sender's current token and creates a fresh one for that sender with an updated counter. For m output coins, the new token stores $\text{tc}_u + m$. As in [7, 42], this token prevents concurrent transactions from reusing a counter and ensures counter correctness.

Regulator. The *regulator* is a trusted authority that can blacklist users and rebuild provenance graphs on request. It manages KYC and issues identity keys. It holds sk_{Reg} for the public encryption key pk_{Reg} . The regulator maintains the identity trace blacklist B_{idt} ,

the sorted user-blacklist as a Merkle tree $\mathcal{T}_{\text{user}}$, and the registry

$$[(\text{idk}_1, \text{tc}_1), (\text{idk}_2, \text{tc}_2), \dots, (\text{idk}_{|\mathcal{U}|}, \text{tc}_{|\mathcal{U}|})].$$

Section 5.4 defines both blacklists. The registry records every user's identity key and transaction counter. At each payment, the regulator decrypts the sender's identity key and updates its counter accordingly.

Validators (consensus). Validators verify transactions and record them on-chain. In the case of Zerocash, they are called “miners”.

Coins. A coin is $\mathbf{c} = (v, \text{pk}, \rho, \text{bv}, \mathcal{G})$, where v, pk, ρ retain their Zerocash roles, bv is the identity trace blacklist version against which the coin is certified (Section 5.4), and $\mathcal{G}$ is the embedded provenance graph defined in the next section. The coin commitment binds this graph and all other fields.

5.2 Embedded provenance graph $\mathcal{G}$

In a coin, $\mathcal{G}$ lists *provenance vertices*. For privacy, vertices do not name users: each carries only an *identity trace* of a past owner, while *source handles* encode edges such that only the regulator can resolve them. We define these notions next.

Provenance vertex. A provenance vertex is a tuple

$$V = (\text{idt}, \text{eik}, h, b, \text{src}),$$

where $\text{idt} \in \mathbb{F}$ is the emitter's identity trace, $\text{eik} = \text{Enc}(\text{idk}; \text{pk}_{\text{Reg}}; \eta)$ encrypts the emitter's identity key with randomness η under the regulator's public key, $h \in \mathbb{N}$ is the hop count, $b \in \mathbb{N}$ is the block number at which the vertex was created, and $\text{src} = (\sigma_1, \dots, \sigma_r)$ is a list of source handles. During a transaction, output coins inherit the input coins' provenance vertices and add one vertex whose identity trace is derived from the sender's identity.

Identity trace. Suppose user u spends n input coins to create m outputs, with current counter tc_u (initialized to 0 and advanced per output). For each $j \in \{1, \dots, m\}$, the output graph receives a new vertex with fresh *identity trace*

$$\text{IDT}_u(\text{tc}_u, j) := \text{PRF}(\text{idk}_u; (\text{tc}_u + j)). \quad (3)$$

Distinct output coins receive distinct identity traces. No identity trace is derived from a transaction counter equal to 0. Such a trace is used as a user identifier, as explained in the next section.

Source handles. Source handles encode provenance-graph edges. For a transfer by u , let $\text{idt}_1^{\text{in}}, \dots, \text{idt}_r^{\text{in}}$ be the traces of the spent input graphs' 1-hop vertices, added when those coins were created. The fresh vertex in output graph j receives $\text{src}_j = (\sigma_{j,1}, \dots, \sigma_{j,r})$, where

$$\sigma_{j,\ell} := \text{PRF}(\text{idk}_u; \text{idt}_\ell^{\text{in}} \parallel \text{IDT}_u(\text{tc}_u, j)) \quad \ell \in \{1, \dots, r\}. \quad (4)$$

After a coin is disclosed, the regulator decrypts the identity key in each vertex $(\text{IDT}_u(\text{tc}_u, j), \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta), h, b, \text{src})$ and uses it to resolve that vertex's trace and handles. For each handle $\sigma_{j,\ell}$ on a vertex with trace idt_j , it tests candidate parent traces idt_ℓ by recomputing $\text{PRF}(\text{idk}_u; \text{idt}_\ell \parallel \text{idt}_j)$. At most one present parent matches. A handle may lack a matching active trace: handles are fixed at vertex creation and never pruned, while a parent, whose hop count is one above its child's, may leave the (k, t) -active window before the child. The regulator can therefore reconstruct

$\mathcal{G}$ by recursively following handles, as in the transparent-ledger provenance semantics from Section 3.

5.3 Active vertex and provenance polynomial

Active provenance vertex. For policy (k, t) and a coin created at b_{now} , a vertex $V = (\text{idt}, \text{eik}, h, b, \text{src})$ is (k, t) -active if $h \leq k$ or $b_{\text{now}} - b \leq t$. Each transfer drops vertices no longer active from $\mathcal{G}$.

Provenance polynomial. Denote the multiset² of traces in $\mathcal{G}$ by

$$\text{Traces}(\mathcal{G}) := \{\{\text{idt} : (\text{idt}, \text{eik}, h, b, \text{src}) \in \mathcal{G}\}\}.$$

The provenance polynomial is defined as

$$P_{\mathcal{G}}(X) := \prod_{\text{idt} \in \text{Traces}(\mathcal{G})} (X - \text{idt}). \quad (5)$$

Repeated traces yield repeated roots. The coin does not store this polynomial; the transaction circuit computes the evaluation needed for the blacklist check directly from the witnessed graph.

5.4 Identity trace and user blacklists

The regulator maintains two complementary public blacklist structures. The *identity trace blacklist* freezes coins carrying a blacklisted user's traces. The *user blacklist* prevents blacklisted users from making later transfers, regardless of their input coins' versions.

Identity trace blacklist. Let $B_{\text{idt}}^{(0)} \subseteq B_{\text{idt}}^{(1)} \subseteq \dots \subseteq B_{\text{idt}}^{(d)}$ be an append-only family of sets of identity traces. For versions $a \leq b$, define the identity trace blacklist delta

$$\text{DeltaBlack}(a, b) := B_{\text{idt}}^{(b)} \setminus B_{\text{idt}}^{(a)}.$$

User blacklist. For each registered user u , define the user identifier

$$\text{uid}_u := \text{PRF}(\text{idk}_u; 0).$$

This unique user identifier is added to the user blacklist if u is blacklisted. As in PaxPay's sorted-list non-membership method [7], the user blacklist is a public sorted Merkle tree $\mathcal{T}_{\text{user}}$ whose current root is rt_{user} . Leaves are ordered, and a user identifier is absent exactly when it lies strictly between two consecutive leaves.³

Blacklisting a user u . To blacklist u , the regulator reads its stored transaction counter tc_u^* and appends

$$\{\text{PRF}(\text{idk}_u; i) : 1 \leq i \leq \text{tc}_u^* + 1\}$$

to the identity trace blacklist. As explained in Section 6.2, a transfer must prove disjointness between the provenance graph of spent coins and the blacklist, which freezes coins tainted with u 's identity. The regulator also inserts $\text{uid}_u = \text{PRF}(\text{idk}_u; 0)$ into $\mathcal{T}_{\text{user}}$, which blocks later transfers by u . It then publishes both the updated identity trace blacklist and the new user-blacklist root.

Delta-blacklist polynomial. Given two identity trace blacklist versions $a < b$, the corresponding *delta-blacklist polynomial* is

$$Q_{a \rightarrow b}(X) := \prod_{\beta \in \text{DeltaBlack}(a, b)} (X - \beta). \quad (6)$$

²A coin may have two vertices with the same identity trace.

³The tree starts with sentinel leaves uid_{min} and uid_{max} , the system's smallest and largest possible user identifiers.

Coins have a field bv that guarantees compliance with the identity trace blacklist at version bv . To transact at current version b with a coin certified at a , the sender need only prove trace disjointness from $\text{DeltaBlack}(a, b)$. This keeps proofs short as the full blacklist grows. It does not replace the user-blacklist proof, which always uses the current root rt_{user} .

6 Transactions

At time b_{now} , a transaction consumes input coins $c_1^{\text{in}}, \dots, c_n^{\text{in}}$ and the sender's compliance token ct_u^{in} . It then creates output coins $c_1^{\text{out}}, \dots, c_m^{\text{out}}$ and an updated compliance token ct_u^{out} . We now describe its construction.

6.1 Provenance graph update

Fresh provenance vertices for the output. For each output index $j \in \{1, \dots, m\}$, define

$$\text{idt}_j^{\text{out}} := \text{IDT}_u(tc_u, j), \quad \text{eik}_j^{\text{out}} := \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta_j^{\text{out}}),$$

where $\text{idt}_j^{\text{out}}$ is defined above and η_j^{out} is randomness unique to that output. Create the fresh vertex

$$V_j^{\text{out}} = (\text{idt}_j^{\text{out}}, \text{eik}_j^{\text{out}}, 1, b_{\text{now}}, \text{src}_j^{\text{out}}),$$

where $\text{src}_j^{\text{out}}$ contains exactly the handles from (4) derived from the spent inputs' 1-hop vertices.

Inherited vertices. Outputs also inherit the input coins' provenance vertices. Let

$$\mathcal{G}^{\text{in}} := \mathcal{G}_1^{\text{in}} \uplus \dots \uplus \mathcal{G}_n^{\text{in}}$$

be the multiset union of their graphs. For each $V \in \mathcal{G}^{\text{in}}$, write $V = (\text{idt}, \text{eik}, h, b, \text{src})$ and update it to

$$V^+ = (\text{idt}, \text{eik}, h + 1, b, \text{src}).$$

$\text{Update}(\mathcal{G}^{\text{in}})$ denotes the provenance graph after updating all the vertices of $\mathcal{G}^{\text{in}}$ as shown above and dropping the vertices no longer active under the (k, t) -active window at time b_{now} .

Output provenance graph and version inheritance. To conclude the last two paragraphs, for each output coin j , the output provenance graph $\mathcal{G}_j^{\text{out}}$ contains the vertices of $\text{Update}(\mathcal{G}^{\text{in}})$ and the fresh vertex V_j^{out} :

$$\mathcal{G}_j^{\text{out}} = \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\}. \quad (7)$$

The NIZK proves this correct update from the input graphs. For input-coin versions $bv_1^{\text{in}}, \dots, bv_n^{\text{in}}$, the spender chooses $0 \leq bv^{\text{in}} \leq \min\{bv_1^{\text{in}}, \dots, bv_n^{\text{in}}\}$. When bv^{in} equals the minimum of input versions, it gives the smallest delta blacklist and thus the shortest computation for the transaction proof. Yet, an observer can infer that spent coins were all generated at or after bv^{in} . Hence choosing an older bv^{in} improves privacy. The NIZK in Section 6.3 proves every output compliant with the current public identity trace blacklist version $bv^{\text{cur}} \geq bv^{\text{in}}$ and sets $bv_1^{\text{out}} = \dots = bv_m^{\text{out}} = bv^{\text{cur}}$. It also proves the sender absent from the current user blacklist.

Compliance-token update. The spent compliance token stores tc_u and the new one stores $tc_u + m$, giving each fresh output a unique counter-derived trace.

6.2 Proof of non-blacklisting

Semantic goal. A compliant transaction meets two conditions. First, the sender's user identifier $\text{uid}_u = \text{PRF}(\text{idk}_u; 0)$ is absent from the user blacklist under the current root rt_{user} . Second, for the inherited identity trace blacklist version bv^{in} and the current version bv^{cur} , every output $j \in \{1, \dots, m\}$ satisfies

$$\text{Traces}(\mathcal{G}_j^{\text{out}}) \cap \text{DeltaBlack}(bv^{\text{in}}, bv^{\text{cur}}) = \emptyset, \quad (8)$$

i.e., no created coin c_j^{out} can have a (k, t) -active provenance graph with an identity trace newly blacklisted between versions bv^{in} and bv^{cur} .

Sender non-membership in the user blacklist. Let β_u^- and β_u^+ be leaves at consecutive positions γ and $\gamma + 1$ in $\mathcal{T}_{\text{user}}$. The prover privately supplies the two leaves, their positions, and their Merkle authentication paths. The NIZK recomputes $\text{uid}_u = \text{PRF}(\text{idk}_u; 0)$ from the idk_u bound to the spent compliance token, verifies both paths against rt_{user} , and enforces $\beta_u^- < \text{uid}_u < \beta_u^+$ under the leaf encodings' canonical integer order. Leaf membership, positional adjacency, and strict inequalities prove that uid_u is absent from the sorted tree. Validators require the current rt_{user} .

Bezout identity on first output. Let $P_{\mathcal{G}_1^{\text{out}}}$ be the provenance polynomial from (5) and $Q_{bv^{\text{in}} \rightarrow bv^{\text{cur}}}$ the delta-blacklist polynomial from (6). For each j , condition (8) is equivalent to

$$\text{gcd}(P_{\mathcal{G}_1^{\text{out}}}, Q_{bv^{\text{in}} \rightarrow bv^{\text{cur}}}) = 1. \quad (9)$$

By (7), output graphs differ only in their fresh vertex, whose trace derives from the sender's identity key and monotone counter. Because the sender is proved absent from the user blacklist, checking $\mathcal{G}_1^{\text{out}}$ against the delta blacklist ensures every output complies with the current identity trace blacklist. Write

$$P := P_{\mathcal{G}_1^{\text{out}}}, \quad Q := Q_{bv^{\text{in}} \rightarrow bv^{\text{cur}}}.$$

By Bezout's identity over $\mathbb{F}[X]$, coprimality of P and Q is equivalent to the existence of two polynomials U and W satisfying:

$$U(X)P(X) + W(X)Q(X) = 1. \quad (10)$$

The following mask makes the published Bezout values independent of the private provenance polynomial, preserving Zerocash privacy.

Masked provenance polynomial. The prover samples a degree-1 polynomial $M(X) = \mu_0 + \mu_1 X$ and sets $\tilde{P}(X) := P(X) + M(X)Q(X)$. Reduction modulo Q gives $\tilde{P} \equiv P \pmod{Q}$, hence $\text{gcd}(\tilde{P}, Q) = \text{gcd}(P, Q)$.

Randomized Bezout witnesses. The prover uses the extended Euclidean algorithm to find reduced (U_0, W_0) satisfying

$$U_0(X)\tilde{P}(X) + W_0(X)Q(X) = 1.$$

It then samples a degree-1 polynomial $S(X)$ and computes randomized witnesses satisfying (10) with $\tilde{P}$ and Q :

$$U := U_0 + SQ, \quad W := W_0 - S\tilde{P}.$$

Commitments and Fiat-Shamir challenge. The prover commits to the randomized witnesses as $C_U := \text{KZG.Com}(U)$ and $C_W := \text{KZG.Com}(W)$, and to the provenance mask as $\text{cm}_{\text{mask}} :=$

$\text{Com}(r_{\text{mask}}; \mu_0, \mu_1)$ with fresh randomness r_{mask} . These commitments enter the public Fiat–Shamir context

$$\text{ctx} := (\{\text{sn}_i^{\text{in}}\}_i, \text{sn}_{\text{ctu}}^{\text{in}}, \{\text{cm}_j^{\text{out}}\}_j, \text{cm}_{\text{ctu}}^{\text{out}}, \text{rt}, \text{rt}_{\text{user}}, b_{\text{now}}, \text{bv}^{\text{in}}, \text{bv}^{\text{cur}}, \text{eik}^{\text{pub}}, \text{cm}_{\text{mask}}, C_U, C_W). \quad (11)$$

Validators recompute the challenge $z := H(\text{ctx})$, and the prover sets $q := Q(z)$. If $q = 0$, it resamples all mask and commitment randomness and derives a fresh context and challenge.

Published evaluations and validator checks. The prover publishes

$$(\text{cm}_{\text{mask}}, C_U, C_W, p', u', \pi_U, \pi_W), \quad p' := \tilde{P}(z), \quad u' := U(z).$$

Here π_U and π_W are KZG evaluation proofs. Validators derive z from the context in (11). They evaluate the public polynomial Q at z and reject if $q = 0$; otherwise, they compute

$$w' := \frac{1 - u'p'}{q} \quad (12)$$

and verify

$$\begin{aligned} \text{KZG.Verify}_{\text{pp}}(C_U, z, u', \pi_U) &= 1, \\ \text{KZG.Verify}_{\text{pp}}(C_W, z, w', \pi_W) &= 1. \end{aligned}$$

Equation (12) gives the scalar identity $u'p' + w'q = 1$. The second opening forces $w' = W(z)$ for the polynomial committed before sampling z , enforcing the random-point Bezout test. The NIZK verifies the correct computation of p' from the provenance polynomial as described in Section 6.3.

6.3 Transaction proofs

A valid transaction at b_{now} consumes coins $c_1^{\text{in}}, \dots, c_n^{\text{in}}$ and the sender's compliance token ct_u^{in} . It produces coins $c_1^{\text{out}}, \dots, c_m^{\text{out}}$ and an updated token ct_u^{out} . Concretely, a transaction is the tuple of all the public fields listed below together with the NIZK proof π certifying the relation $\mathcal{R}$ defined in this section.

Beyond Zerocash, $\mathcal{R}$ enforces sender non-membership in the current user blacklist, the full graph update, and the link between the published masked evaluation p' and the first output graph. To speed NIZK proving, randomized-Bezout KZG checks remain outside the circuit: validators recompute the public challenge and delta-blacklist evaluation, derive the required W evaluation, and verify two KZG openings.

Public inputs. The statement x_{NIZK} contains:

- *Base payment layer.* Input coin nullifiers $\text{sn}_1^{\text{in}}, \dots, \text{sn}_n^{\text{in}}$; spent compliance token nullifier $\text{sn}_{\text{ctu}}^{\text{in}}$; fresh output coin commitments $\text{cm}_1^{\text{out}}, \dots, \text{cm}_m^{\text{out}}$; updated compliance token commitment $\text{cm}_{\text{ctu}}^{\text{out}}$; coin commitment Merkle root rt ; and current block b_{now} .
- *Blacklist state.* Inherited identity trace blacklist version bv^{in} , current public version bv^{cur} , and current user-blacklist root rt_{user} .
- *Identity-key encryption.* Ciphertext eik^{pub} , which decrypts under sk_{Reg} to idk_u .
- *Compliance link.* Mask commitment cm_{mask} , validator-derived challenge z and delta-blacklist evaluation $q = Q(z)$, and published masked provenance evaluation p' .

The KZG commitments C_U, C_W , evaluation u' , and opening proofs π_U, π_W are public transaction fields checked outside the NIZK. Validators use them to reconstruct (11), then supply z and q as NIZK public inputs. The regulator publishes on-chain $\text{pk}_{\text{Reg}}, \text{PRF}, \text{Enc}$, and the policy (k, t) . Validators also use the public KZG parameters pp and the Fiat–Shamir hash H .

Private witness. The prover holds:

- The sender's secret key sk_u , registered identity key idk_u , and current counter tc_u .
- For each $i \in \{1, \dots, n\}$, an opening $(v_i^{\text{in}}, \text{pk}_u, \rho_i^{\text{in}}, \text{bv}_i^{\text{in}}, \mathcal{G}_i^{\text{in}}, r_i^{\text{in}})$ of c_i^{in} and its Merkle path to rt .
- The spent compliance token opening $(\text{idk}_u, \text{pk}_u, \text{tc}_u, \rho_{\text{ct}}^{\text{in}}, r_{\text{ct}}^{\text{in}})$ and its Merkle path to rt , and the output compliance token opening $(\text{idk}_u, \text{pk}_u, \text{tc}_u + m, \rho_{\text{ct}}^{\text{out}}, r_{\text{ct}}^{\text{out}})$ of ct_u^{out} .
- For each $j \in \{1, \dots, m\}$, an opening

$$(v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \rho_j^{\text{out}}, \text{bv}_j^{\text{out}}, \mathcal{G}_j^{\text{out}}, r_j^{\text{out}})$$

of c_j^{out} , including the fresh vertex

$$V_j^{\text{out}} = (\text{idt}_j^{\text{out}}, \text{eik}_j^{\text{out}}, 1, b_{\text{now}}, \text{src}_j^{\text{out}}).$$

- Randomness η_j^{out} for each $\text{eik}_j^{\text{out}}$ and η^{pub} for eik^{pub} .
- Consecutive user-blacklist leaves β_u^-, β_u^+ at positions $\gamma, \gamma + 1$, and their Merkle paths to rt_{user} .
- Mask coefficients μ_0, μ_1 and opening randomness r_{mask} satisfying

$$\text{cm}_{\text{mask}} = \text{Com}(r_{\text{mask}}; \mu_0, \mu_1).$$

Relation. The statement and witness satisfy $\mathcal{R}$ iff:

- (R1) **Base payment correctness.** $\text{pk}_u = \text{PRF}(\text{sk}_u; 0)$ and, for each $i \in \{1, \dots, n\}$, $\text{cm}_i^{\text{in}} = \text{Com}(r_i^{\text{in}}; v_i^{\text{in}}, \text{pk}_u, \rho_i^{\text{in}}, \text{bv}_i^{\text{in}}, \mathcal{G}_i^{\text{in}})$ is a leaf under the root rt and $\text{sn}_i^{\text{in}} = \text{PRF}(\text{sk}_u; \rho_i^{\text{in}})$. For each $j \in \{1, \dots, m\}$, $\text{cm}_j^{\text{out}} = \text{Com}(r_j^{\text{out}}; v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \rho_j^{\text{out}}, \text{bv}_j^{\text{out}}, \mathcal{G}_j^{\text{out}})$, and $\sum_{i=1}^n v_i^{\text{in}} = \sum_{j=1}^m v_j^{\text{out}}$. This matches (Z1)–(Z5) of Section 4.2.
- (R2) **Compliance-token consumption and update.** The spent compliance token commitment satisfies

$$\text{cm}_{\text{ctu}}^{\text{in}} = \text{Com}(r_{\text{ctu}}^{\text{in}}; \text{idk}_u, \text{pk}_u, \text{tc}_u, \rho_{\text{ct}}^{\text{in}})$$

and is a leaf of the tree under the root rt . Its nullifier is

$$\text{sn}_{\text{ctu}}^{\text{in}} = \text{PRF}(\text{sk}_u; \rho_{\text{ct}}^{\text{in}}).$$

The fresh commitment satisfies

$$\text{cm}_{\text{ctu}}^{\text{out}} = \text{Com}(r_{\text{ctu}}^{\text{out}}; \text{idk}_u, \text{pk}_u, \text{tc}_u + m, \rho_{\text{ct}}^{\text{out}}).$$

- (R3) **Fresh identity trace correctness.** For each $j \in \{1, \dots, m\}$, $\text{idt}_j^{\text{out}} = \text{PRF}(\text{idk}_u; \text{tc}_u + j)$, matching (3).
- (R4) **Fresh regulator encryption.** For each $j \in \{1, \dots, m\}$, V_j^{out} satisfies

$$\text{eik}_j^{\text{out}} = \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta_j^{\text{out}}).$$

The public ciphertext satisfies

$$\text{eik}^{\text{pub}} = \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta^{\text{pub}}).$$

(R5) **Source-handle consistency.** Let $\mathcal{G}^{\text{in}} := \mathcal{G}_1^{\text{in}} \uplus \dots \uplus \mathcal{G}_n^{\text{in}}$, and let $(\text{idt}_1, \dots, \text{idt}_r)$ be its 1-hop traces, with multiplicity, in canonical order. For each $j \in \{1, \dots, m\}$, $\text{src}_j^{\text{out}} = (\sigma_{j,1}, \dots, \sigma_{j,r})$ with $\sigma_{j,\ell} = \text{PRF}(\text{idk}_u; \text{idt}_\ell \parallel \text{idt}_j^{\text{out}})$ for every $\ell \in \{1, \dots, r\}$, matching (4).

(R6) **Provenance inheritance and active-window filtering.** For each $j \in \{1, \dots, m\}$,

$$\mathcal{G}_j^{\text{out}} = \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\},$$

where $\text{Update}(\mathcal{G}^{\text{in}})$ increments each hop count by one and discards vertices not (k, t) -active at b_{now} . The fresh vertex is

$$V_j^{\text{out}} = (\text{idt}_j^{\text{out}}, \text{eik}_j^{\text{out}}, 1, b_{\text{now}}, \text{src}_j^{\text{out}}),$$

so its hop count is exactly 1 and its timestamp is exactly b_{now} .

(R7) **Identity trace version correctness.**

$$\text{bv}^{\text{in}} \leq \min\{\text{bv}_1^{\text{in}}, \dots, \text{bv}_n^{\text{in}}\}.$$

For each j , $\text{bv}_j^{\text{out}} = \text{bv}^{\text{cur}}$.

(R8) **Sender user-blacklist non-membership.** Compute $\text{uid}_u = \text{PRF}(\text{idk}_u; 0)$ from the spent compliance token's identity key. Verify that β_u^-, β_u^+ are leaves at consecutive positions $\gamma, \gamma + 1$ under the root rt_{user} , and enforce $\beta_u^- < \text{uid}_u < \beta_u^+$.

(R9) **Mask-commitment opening.**

$$\text{cm}_{\text{mask}} = \text{Com}(r_{\text{mask}}; \mu_0, \mu_1).$$

(R10) **Masked provenance evaluation at z .** Compute

$$y_P = \prod_{V \in \mathcal{G}_1^{\text{out}}} (z - V.\text{idt})$$

directly from the output graph fixed by (R1) and (R6), and enforce

$$p' = y_P + (\mu_0 + \mu_1 z)q.$$

Validator checks. On receipt of a transaction, validators verify its validity as follows:

- *Nullifier freshness and distinctness.* Every published nullifier is absent from $\mathcal{T}_{\text{sn}}$, and the n input-coin nullifiers and compliance token nullifier are pairwise distinct.
- *Ledger consistency.* rt is an accepted historical root of $\mathcal{T}_{\text{cm}}$, b_{now} is the current block, and $(\text{bv}^{\text{cur}}, \text{rt}_{\text{user}})$ matches the identity trace blacklist version and user-blacklist root.
- *Challenge and blacklist evaluation.* Reconstruct ctx as in (11), set $z = H(\text{ctx})$, evaluate $q = Q_{\text{bv}^{\text{in}} \rightarrow \text{bv}^{\text{cur}}}(z)$ from the public identity trace blacklist, and reject if $q = 0$.
- *External Bezout checks.* Set $w' = (1 - u'p')/q$ and verify:
 - $\text{KZG.Verify}_{\text{pp}}(C_U, z, u', \pi_U) = 1$
 - $\text{KZG.Verify}_{\text{pp}}(C_W, z, w', \pi_W) = 1$
- *NIZK verification.* Reconstruct x_{NIZK} using

$$(\text{rt}_{\text{user}}, \text{cm}_{\text{mask}}, z, q, p')$$

and check

$$\text{NIZK.Verify}(\text{crs}, x_{\text{NIZK}}, \pi) = 1.$$

On acceptance, validators insert the published nullifiers into $\mathcal{T}_{\text{sn}}$ and the output-coin and compliance token commitments into $\mathcal{T}_{\text{cm}}$.

Regulator update. The regulator decrypts eik^{pub} to recover idk_u , then increments user u 's counter tc_u by the number of outputs m .

6.4 Selective disclosure and regulator reconstruction

A coin is frozen if its provenance polynomial shares a root with the current delta-blacklist polynomial $Q_{\text{bv}^{\text{in}} \rightarrow \text{bv}^{\text{cur}}}$: its holder cannot produce Section 6.2's Bezout witness and thus cannot spend it. An honest holder who considers such a coin legitimate may use an escape hatch by opening it to the regulator, who reconstructs its graph, performs a standard AML/CFT investigation, and whitelists it if the holder is cleared. We next describe this disclosure and reconstruction.

Coin opening. Over an authenticated private channel, the holder sends the regulator the frozen coin's full opening:

$$\mathbf{c} = (v, \text{pk}, \rho, \text{bv}, \mathcal{G}), \quad \text{cm} = \text{Com}(r; \mathbf{c}).$$

The regulator recomputes cm and checks its on-chain presence and non-compliance with $B_{\text{idt}}^{(\text{bv}^{\text{cur}})}$. This authenticates the opening and limits disclosure to one coin: it reveals neither the user's other coins nor transfers outside the (k, t) -active window.

Vertex resolution. Once the coin is disclosed, consider each vertex

$$V = (\text{idt}, \text{eik}, h, b, \text{src}) \in \mathcal{G}.$$

The regulator computes

$$\text{idk} \leftarrow \text{Dec}(\text{eik}, \text{sk}_{\text{Reg}})$$

and finds in its registry the emitter who added V during a payment.

Edge reconstruction. By (4), each handle $\sigma \in \text{src}$ on vertex V_j with trace idt_j equals $\text{PRF}(\text{idk}_u; \text{idt}_\ell \parallel \text{idt}_j)$ for at most one parent trace idt_ℓ in $\mathcal{G}$, with idk_u the emitter key recovered for V_j above. The regulator tests each candidate $\text{idt}'_\ell \in \text{Traces}(\mathcal{G})$ by recomputing $\text{PRF}(\text{idk}_u; \text{idt}'_\ell \parallel \text{idt}_j)$. If a matching trace remains, it is unique and becomes σ 's parent. Repeating this for every $\sigma \in \text{src}$ and $V_j \in \mathcal{G}$ recovers all directed edges whose endpoint vertices remain active.

Investigation and whitelisting. The reconstructed graph gives the regulator the same information as a transparent ledger: identified users at each vertex, directed prior-transfer edges, hop distances, and block-level timestamps. Layering detection, peeling-chain analysis, and counterparty risk assessment apply unchanged. If the coin is cleared, the regulator issues a credential that permits spending despite the blacklist hit. A straightforward design is a signed certificate supplied privately to the NIZK to bypass the compliance check. To keep the paper as light as possible, we do not formalize this straightforward whitelisting mechanism.

6.5 Minting coins

We also adapt Zerocash minting. At registration, the regulator creates the user's initial compliance token with counter $\text{tc}_u = 1$; index 0 remains reserved for uid_u . Minting takes as input a value v , a recipient key pk , and the current identity trace blacklist version bv^{cur} . The user samples fresh $\rho \xleftarrow{\$} \mathbb{F}$ and $r \xleftarrow{\$} \mathcal{R}$ and sets the initial graph to the empty multiset $\mathcal{G}_{\text{mint}} = \emptyset$ (so $P_{\mathcal{G}_{\text{mint}}}(X) = 1$). The user then publishes $\text{cm}_{\text{mint}} = \text{Com}(r; v, \text{pk}, \rho, \text{bv}^{\text{cur}}, \emptyset)$ with an

NIZK proof that the coin is well-formed. Thus, the first transaction spending a minted coin uses $\mathcal{G}^{\text{in}} = \emptyset$ and produces output graphs containing only the minter’s fresh identity trace vertex.

7 Implementation and benchmarks

Implementation. We implemented our construction in Go using Gnark [6]. The NIZK uses Groth16 [24] over the BN254 curve.

Deployment parameters. We use $t = 48$ hours as the main temporal window. Global Ledger [2] reports that stolen funds start moving after about 17 hours on average, while public disclosure occurs after roughly 36 hours. Thus, 48 hours would typically cover detection and reporting delays plus time to publish the blacklist. We test a conservative $t = 7$ days because stolen funds take about 5.2 days on average to reach a first mixer or virtual-asset service provider [2]. For normal activity, we assume approximately $p := 1.55$ payments per user per day, based on the Federal Reserve’s estimate of 47 monthly consumer payments [21]. Finally, we set $k = 6$, the maximum traversal depth in the Elliptic2 AML dataset [4]; this is a practical benchmark, not an optimal regulatory threshold, which may be lower.

To relate the payment rate to graph size, let C be a user’s average number of spendable coins and m the average inputs per payment. For p payments per user per day, a coin’s average turnover rate is $\text{rate} := mp/C$. In an illustrative model, $p = 1.55$, $m = 2$, and $C = 7$ give $\text{rate} \approx 0.44$ spends per coin per day, a mean holding time of approximately 2.3 days, and about 3.1 provenance generations over seven days. With conservative binary fan-in and disjoint ancestries, three generations contain 16 vertices. Graphs of 64, 128, and 256 vertices capture higher turnover at about 3, 6, and 12 payments per day.

Benchmark. We evaluated the implementation on an Ubuntu 22.04 laptop with a 2.6-GHz Intel i7 CPU. Figure 2 reports total transaction crafting and verification times (including the NIZK and Bezout checks) and the NIZK constraint count for the input graph sizes above. Crafting time grows linearly with graph size because the constraint count is linear in the number of provenance vertices and NIZK proving, itself linear in the constraint count, dominates crafting. As expected, verification time remains constant: NIZK verification is independent of the constraint count, and KZG commitments make the Bezout checks independent of the graph’s vertex count. All runs on the table are done with empty blacklist. We measured with a delta-blacklist containing 10,000 traces, which is the equivalent of 20 years of traces emitted by a blacklisted user at 1.5 payments per day, which is what we can expect in practice. It adds only 0.2 ms to the verification time, which is negligible compared to the 2 ms baseline, and 1.2 s to the transaction crafting time, which is perfectly acceptable, especially since blacklist updates are sporadic events.

Validator workload. “Validators” denotes the Zerocash miners, or more generally the consensus layer that verifies transactions and adds them to the blockchain. For example, in an Ethereum deployment extending Railgun, the smart contract would be the validator. Our construction does not limit throughput because verification is

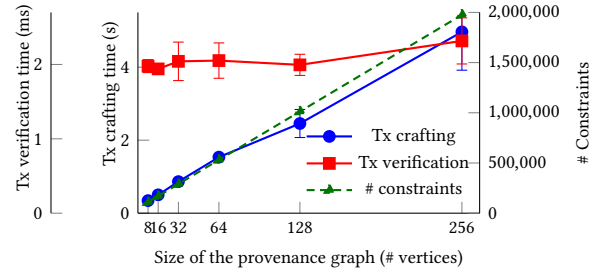


Figure 2: Transaction crafting time, verification time, and R1CS constraints with respect to the provenance-graph size, for $n = 2$ inputs and $m = 2$ outputs. Average over 10 runs.

much faster than the underlying block time: Zcash typically processes about 20 transactions per second, while a laptop running our implementation verifies over 250 transactions per second.

Regulator workload. The regulator can remain passive and compute asynchronously: as transactions enter the ledger, it decrypts each transaction’s public ciphertext eik^{pub} to recover the sender’s identity key and increments that sender’s compliance counter. With ElGamal, eik decryption (one point multiplication and one addition) is faster than transaction verification, which performs at least the same work to compute the NIZK public input. The main regulator cost is blacklisting, which requires computing every trace emitted by a user. On the same laptop, computing one million traces, corresponding to one million transactions, takes under 8 seconds, fast enough to update the blacklist in real time.

Coin size. In our implementation, a coin occupies 2.5–40 KB for provenance graphs of 16–256 vertices, which poses no user-storage issue. On-chain storage matches Zerocash: the coin commitment and nullifier remain 32 bytes each.

8 Security

The construction inherits the *balance* integrity guarantee of Zerocash, and adds *compliance soundness* and *disclosure integrity*. We do not claim transaction non-malleability as defined by Zerocash. This limitation does not affect the balance, compliance-soundness, or disclosure-integrity guarantees established here. If full transaction non-malleability is required, for example to obtain non-malleable transaction identifiers, it can be achieved by applying Zerocash’s mechanism of transaction signing. We omit this mechanism to simplify the construction. On the privacy side, our scheme does not inherit Zerocash *ledger indistinguishability* as is: every transaction exposes additional public fields and every coin carries an embedded provenance graph that its holder can inspect. We characterize this relaxation through an explicit *leakage profile* (Section 8.2) and capture it in a transaction indistinguishability game TIND (Definition 5), which recovers Zerocash ledger indistinguishability exactly when the adversary holds no coins. Here we summarize the model and main theorem. Appendix C gives formal definitions, theorem statements, and proof sketches.

8.1 Threat model

Trust assumptions. The NIZK CRS and the KZG parameters pp come from a trusted setup whose trapdoors are destroyed. The regulator is trusted for KYC, for the registry, and for the correctness of the two published blacklists, but *not* for privacy (cf. the next paragraph). Validators are assumed to enforce the checks of Appendix B and to keep the ledger consistent and available. Consensus-level attacks are orthogonal to our construction. Users communicate over authenticated private channels, as in Zerocash. Corruption is static: a corrupted user hands the adversary (sk_u, idk_u) and the openings of her coins.

Adversaries. We consider three PPT adversaries, each capturing a distinct privacy boundary. $\mathcal{A}_{pub}$ is a public observer with the full ledger view but no regulator key. $\mathcal{A}_{hold}$ has the same view, plus the openings of a set C_{hold} of coins she controls (across one or several corrupted users), including the embedded provenance graphs $\mathcal{G}^{(i)}$ stored in those coins. $\mathcal{A}_{reg}$ is an honest-but-curious regulator that holds sk_{reg} and the identity registry, follows the protocol, and may receive disclosure openings at a user’s request. Integrity (balance, compliance soundness and disclosure integrity) is instead proved against a fully malicious sender.

Out of scope. Network-level metadata, off-chain identification of counterparties, and censorship or denial of service by validators lie outside the model and affect Zerocash identically. Whether a blacklisting is justified is a governance question and depends on how the system is used rather than its design. Appendix C.1 gives the full model and the assumptions used in the proofs.

8.2 Leakage profile

Compared to Zerocash, a public observer additionally sees, per transaction, the inherited and current blacklist versions (bv^{in}, bv^{cur}) and an IND-CPA-encrypted regulator ciphertext eik^{pub} that opens only under sk_{reg} to the sender’s identity key, together with the masked compliance values $(cm_{mask}, C_U, C_W, p', u', \pi_U, \pi_W)$, which are rerandomized per transaction and reveal nothing about its content. We capture this as a public leakage function $Leak_{pub}$ (Definition 3).

A holder of multiple coins additionally learns, for every identity trace occurring in her coins, its creation block and the multiset of hop counts at which it occurs in each of them. In particular, she learns when two of her coins share an active ancestor, and with what multiplicity. We capture this as a holder-side leakage function $Leak_{hold}$ (Definition 4), recomputed over the provenance graphs of all coins in C_{hold} at every point of the game. This leakage is structural in the sense that it does not depend on the actual identity keys of the users behind the vertices, but only on the graph structure of coins a holder owns. Pseudorandomness of PRF and IND-CPA security of Enc prevent her from identifying the user behind a vertex whenever the candidate emitters are structurally interchangeable in her view. Section 8.4 states what she can infer otherwise.

The transaction indistinguishability game TIND parameterizes indistinguishability by the pair $(Leak_{pub}, Leak_{hold})$ and is the right

generalization of Zerocash ledger indistinguishability for our setting.

For the regulator, define $Leak_{reg}(tx) := (idk_u, \{idk_j^{out}\}_{j=1}^m)$. The RegIND game gives the adversary the regulator’s secret key and registry, and admits only challenge pairs that agree on both $Leak_{pub}$ and $Leak_{reg}$. It therefore asks whether the regulator learns anything beyond the public leakage, the sender’s identity, and the fresh output traces derivable from that identity and its counter.

8.3 Main theorem

THEOREM 1. *Under the standard cryptographic assumptions listed in Appendix C.1, the construction satisfies the following properties against the adversaries of Section 8:*

- (1) balance (Theorem 5);
- (2) compliance soundness (Theorem 6);
- (3) disclosure integrity (Theorem 7);
- (4) regulator indistinguishability in RegIND (Theorem 11);
- (5) holder identity-hiding (Corollary 12);
- (6) transaction indistinguishability with respect to $(Leak_{pub}, Leak_{hold})$ (Theorems 9 and 10).

The remainder of this section reviews each property informally. Appendix C provides the formal definitions and proof sketches.

Balance. Balance is inherited from Zerocash unchanged: the extended commitment input $(v, pk, \rho, bv, \mathcal{G})$ leaves the base-layer obligations intact.

Compliance soundness. No accepted transaction has an output coin whose embedded (k, t) -active provenance graph contains an identity trace currently in $\Delta\text{Black}(bv^{in}, bv^{cur})$, and the on-chain commitments contain exactly the provenance graph prescribed by the update rule of Section 6.1.

The proof combines NIZK weak simulation extractability, KZG knowledge soundness in the AGM (which extracts the randomized Bezout witnesses behind C_U and C_W), commitment binding (which ties the masked evaluation p' to the committed output graph), the Schwartz–Zippel lemma for the random-point Bezout identity at $z = H(ctx)$, and PRF collision resistance for fresh output traces.

Disclosure integrity. A corrupted holder cannot supply the regulator with an opening of an on-chain commitment whose embedded provenance graph differs from the one prescribed by the update rule.

The proof is direct: NIZK weak simulation extractability recovers the witness graph; commitment binding then forces the disclosed graph to equal the witness graph; constraint (R6) of $\mathcal{R}$ forces the witness graph to equal the prescribed update.

Regulator indistinguishability. In RegIND the regulator receives sk_{reg} and the identity registry, yet cannot distinguish challenge transactions agreeing on $Leak_{reg}$. Thus, on non-disclosed coins, its additional knowledge over a public observer is limited to the per-transaction sender identity (recovered via decryption of eik^{pub}) and the derivable fresh output identity traces. It does not thereby learn receivers, amounts, or links from an output to its later spend.

Holder identity-hiding. For two honest users whose contributions to the adversary’s view are structurally interchangeable, formally for challenge executions admissible under (A1)–(A3) of Definition 5, $\text{Leak}_{\text{hold}}$ does not reveal which of them contributed a given provenance vertex (Corollary 12).

The requirements are necessary rather than technical. Users who transacted at different blocks or at different lineage depths differ in $\text{Leak}_{\text{hold}}$ itself and are separated by inspection. Section 8.4 states what a coin holder with common ancestors in their provenance graphs can infer in that case.

Transaction indistinguishability. TIND uses the leakage pair $(\text{Leak}_{\text{pub}}, \text{Leak}_{\text{hold}})$. It captures privacy against $\mathcal{A}_{\text{pub}}$ and $\mathcal{A}_{\text{hold}}$. The regulator’s privacy property is captured separately by RegIND (Theorem 11), because $\mathcal{A}_{\text{reg}}$ can decrypt eik^{pub} and thereby learn the sender of each transaction.

The proof goes through standard hybrids that simulate the NIZK proof, simulate the published compliance values using the provenance mask and the Bezout-witness randomization, replace the regulator-key ciphertexts by encryptions of zero, and replace all PRF outputs keyed by honest users’ secret material by uniform values, then reduces the residual gap to base Zerocash ledger indistinguishability.

When $C_{\text{hold}} = \emptyset$, $\text{Leak}_{\text{hold}}$ is empty and TIND reduces to Theorem 9, i.e., indistinguishability against Leak_{pub} alone, i.e., Zerocash ledger indistinguishability augmented with public exposure of $(\text{bv}^{\text{in}}, \text{bv}^{\text{cur}}, b_{\text{now}})$ and IND-CPA hiding of eik^{pub} . Note that bv^{in} could be set by the user to 0 to prevent leaking this last piece of information.

8.4 What holder-side leakage means in practice

Corollary 12 shows that $\text{Leak}_{\text{hold}}$ hides *who* stands behind a provenance vertex. It does not make $\text{Leak}_{\text{hold}}$ harmless. We now explain how a payment recipient can use it. Every component of $\text{Leak}_{\text{hold}}$ comes from a coin opening and is available only to a payee: regardless of resources, a passive ledger observer learns only Leak_{pub} and no provenance.

$\text{Leak}_{\text{hold}}$ over a single coin. Every active vertex records the exact block in which it was created, so a holder learns, for each ancestor of her coin, the block of the transaction that produced it and its lineage distance. She can then restrict that ancestor to the shielded transactions of that single block which match the public fields of Leak_{pub} . It also composes with side information: anyone who learns one sender active in that block can restrict the possible identities of the sender among a small set of candidates.

$\text{Leak}_{\text{hold}}$ over multiple coins. The matrix $\mathbf{H}$ of Definition 4 in Appendix C.5 reveals which of a holder’s coins share an ancestor and with what multiplicity. By itself this is structural linkage. As above, it can be combined with off-chain information. For example, a payment recipient usually knows the sender, although anonymous donations remain possible. She may thus identify the user behind the identity traces at the 1-hop vertices of received coins, then test whether those traces occur in her other coins’ graphs.

Yet this actually reveals nothing about the payer’s other activity, since such a trace, being only 1-hop, cannot appear in any coin other than the one on which it was received. It may appear only in

coins that the holder herself will create and in their descendants, as a vertex at hop 2 or greater. Linkage identified through such an identity trace is in this sense confined to the holder’s own outgoing lineage and does not extend to the payer’s other activity.

Extractable information. Many distinct transaction histories may induce the same $\text{Leak}_{\text{hold}}$ view. From inspection of common ancestors, a holder learns that her coins fall into common-origin classes, not what those origins are, nor through which intermediaries the funds passed. For instance, if two of her coins carry a common trace at hop 3, she learns that both descend from the single transaction that spent the coin carrying that trace, at lineage distance two, but not who emitted it, nor whether the two paths passed through the same intermediary. A hop-2 vertex common to two coins only states that the two coins were created during the same transaction, which provides no more information than Zerocash does, since the receiver can deduce it from the transaction that added the coins’ commitments to the on-chain Merkle tree.

Mitigations. Larger values of (k, t) improve AML coverage but also increase the amount of information a holder can extract from her coins. Compliance strength and holder-side privacy are therefore not independent design axes. The same parameters also bound the leakage: vertices are discarded once they leave the (k, t) -active window, so a holder sees nothing beyond that horizon. As a consequence, the leakage is temporary in some sense. Also note that a freshly minted coin induces no leakage at all, since its provenance graph is empty.

To minimize the leakage described in the first paragraph, the system can rely on a wide timestamp granularity, e.g., by defining t as a number of epochs rather than a number of blocks, with each epoch composed of several blocks. AML/CFT does not need high granularity; an epoch could be a day, for instance.

9 Conclusion and future work

We presented a framework for AML/CFT provenance analysis in both transparent and private payment systems. On transparent ledgers, we formalized the (k, t) -provenance graph and gave a backward-traversal algorithm. We then extended Zerocash so that every coin embeds such a graph, with compliance reduced to a SNARK-linked polynomial-coprimality check and selective disclosure enabling regulator reconstruction. Our implementation confirms the practicality of the scheme. The provenance polynomial is evaluated directly from the committed graph inside the SNARK, while the two KZG openings needed for the randomized Bezout test are verified by validators. Compared to a solution with KZG openings made inside the circuit, this removes pairing checks from the circuit at the cost of two additional validator-side opening checks. Future work could explore more sophisticated provenance graph structures, such as weighted edges or multi-asset graphs, and investigate the integration of our framework with other privacy-preserving protocols.

Acknowledgments

Maxence Brugeres was supported by Forvis Mazars Group. Petr Kuznetsov was supported by CHIST-ERA-22-SPiDDS-05 (REDONDA

project) and Agence Nationale de la Recherche (ANR-23-CHR4-0009).

References

- [1] Ross Anderson, Ilia Shumailov, and Mansoor Ahmed. 2018. Making Bitcoin Legal. In *Security Protocols XXVI (Lecture Notes in Computer Science, Vol. 11286)*. Springer, 243–253. doi:10.1007/978-3-030-03251-7_29
- [2] Anastasia Arashkevich. 2026. *The 2026 Crypto Laundering Roadmap: 5 Trends Redefining Global Financial Crime*. Global Ledger. <https://blog.globalledger.io/blog/financial-crime-trends>
- [3] Mihir Bellare and Phillip Rogaway. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 62–73.
- [4] Claudio Bellei, Muhua Xu, Ross Phillips, Tom Robinson, Mark Weber, Tim Kaler, Charles E Leiserson, Jie Chen, et al. 2024. The shape of money laundering: Subgraph representation learning on the blockchain with the elliptic2 dataset. *arXiv preprint arXiv:2404.19109* (2024).
- [5] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 459–474. doi:10.1109/SP.2014.36
- [6] Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. 2024. ConsenSys/gnark: v0.11.0.
- [7] Maxence Brugeret, Victor Languille, Petr Kuznetsov, and Hamza Zarfaoui. 2025. Fast, Private and Regulated Payments in Asynchronous Networks. In *7th Conference on Advances in Financial Technologies (AFT 2025) (Leibniz International Proceedings in Informatics (LIPIcs))*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2025.3
- [8] Vitalik Buterin, Jacob Illum, Matthias Nadler, Fabian Schär, and Ameen Soleimani. 2023. Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium. *Blockchain: Research and Applications* 5 (12 2023), 100176. doi:10.1016/j.bcr.2023.100176
- [9] Chainalysis. 2026. The 2026 Crypto Crime Report. Industry report. <https://www.chainalysis.com/reports/crypto-crime-2026/>
- [10] Panagiotis Chatzigiannis and Foteini Baldimtsi. 2021. MiniLedger: Compact-sized Anonymous and Auditable Distributed Payments. *Cryptology ePrint Archive*, Paper 2021/869. <https://eprint.iacr.org/2021/869>
- [11] Ting Chen, Zihao Li, Yuxiao Zhu, Jiachi Chen, Xiapu Luo, John Chi-Shing Lui, Xiaodong Lin, and Xiaosong Zhang. 2020. Understanding ethereum via graph analysis. *ACM Transactions on Internet Technology (TOIT)* 20, 2 (2020), 1–32.
- [12] Xiangao Chen, Tianyu Wang, Kecheng Huang, and Zili Shao. 2024. TPGraph: A Highly-scalable Time-partitioned Graph Model for Tracing Blockchain. In *Proceedings of the 17th ACM International Systems and Storage Conference (Virtual, Israel) (SYSTOR '24)*. Association for Computing Machinery, New York, NY, USA, 25–38. doi:10.1145/3688351.3689161
- [13] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. 2020. PGC: Decentralized Confidential Payment System with Auditability. In *ESORICS*.
- [14] CipherTrace. 2021. Cryptocurrency Crime and Anti-Money Laundering Report. Industry report. <https://ciphertrace.com/wp-content/uploads/2021/05/CipherTrace-Cryptocurrency-Crime-and-Anti-Money-Laundering-Report-May-2021-051221b.pdf>
- [15] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.
- [16] Elliptic. 2024. Typologies Report 2024: Preventing Financial Crime in Cryptocurrencies: Identifying Evolving Criminal Behavior. Industry report. <https://www.elliptic.co/hubfs/Elliptic%20Typologies%20Report%202024.pdf>
- [17] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*. Springer, 186–194.
- [18] Financial Action Task Force. 2021. Updated Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Providers. Guidance report. <https://www.fatf-gafi.org/content/dam/fatf-gafi/guidance/Updated-Guidance-VA-VASP.pdf>
- [19] Financial Crimes Enforcement Network. 2013. Application of FinCEN’s Regulations to Persons Administering, Exchanging, or Using Virtual Currencies. Interpretive guidance. FIN-2013-G001. <https://www.fincen.gov/system/files/shared/FIN-2013-G001.pdf>
- [20] Financial Crimes Enforcement Network. 2026. History of Anti-Money Laundering Laws. <https://www.fincen.gov/resources/history-anti-money-laundering-laws>. Accessed: 2026-04-25.
- [21] Kevin Foster and Marcin Hitzencen. 2026. *2025 Survey and Diary of Consumer Payment Choice*. Technical Report. Federal Reserve Bank of Atlanta. <https://www.atlantafed.org/research-and-data/surveys/survey-and-diary-of-consumer-payment-choice>
- [22] Christina Garman, Matthew Green, and Ian Miers. 2016. Accountable Privacy for Decentralized Anonymous Payments. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 9603)*, Jens Grossklags and Bart Preneel (Eds.). Springer, 81–98.
- [23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. 2023. Poseidon2: A Faster Version of the Poseidon Hash Function. *Cryptology ePrint Archive*, Paper 2023/323. <https://eprint.iacr.org/2023/323>
- [24] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II* 35. Springer, 305–326.
- [25] Bernhard Haslhofer, Rainer Stütz, Matteo Romiti, and Ross King. 2021. Graphsense: A general-purpose cryptoasset analytics platform. *arXiv preprint arXiv:2102.13613* (2021).
- [26] Silu Huang, James Cheng, and Huanhuan Wu. 2014. Temporal graph traversals: Definitions, algorithms, and applications. *arXiv preprint arXiv:1401.1919* (2014).
- [27] Aram Jivanyan. 2019. Lelantus: A New Design for Anonymous and Confidential Cryptocurrencies. *Cryptology ePrint Archive*, Paper 2019/373. <https://eprint.iacr.org/2019/373>
- [28] Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. {BlockSci}: Design and applications of a blockchain analysis platform. In *29th USENIX Security Symposium (USENIX Security 20)*. 2721–2738.
- [29] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. 2010. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*. Springer, 177–194.
- [30] David Kempe, Jon Kleinberg, and Amit Kumar. 2000. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. 504–513.
- [31] Dan Lin, Jiajing Wu, Qi Yuan, and Zibin Zheng. 2020. T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis. *Frontiers in Physics* 8 (2020), 204.
- [32] Ernesto Queirós Vieira Martins. 1984. On a multicriteria shortest path problem. *European Journal of Operational Research* 16, 2 (1984), 236–245. doi:10.1016/0377-2217(84)90077-8
- [33] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*. 127–140.
- [34] George B Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. 2013. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages, and Programming*. Springer, 657–668.
- [35] Malte Möser, Rainer Böhme, and Dominic Breuker. 2014. Towards Risk Scoring of Bitcoin Transactions. In *Proceedings of the 1st Workshop on Bitcoin Research*. Barbados, 1–16.
- [36] Neha Narula, Willy Vasequez, and Madars Virza. 2018. zkLedger: Privacy-Preserving Auditing for Distributed Ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, 65–80. <https://www.usenix.org/conference/nsdi18/presentation/narula>
- [37] RAILGUN Project. 2021. *RAILGUN: Privacy System Technical Whitepaper*. Technical Report. RAILGUN. Accessed: 2026-01-28. <https://docs.railgun.org/>
- [38] Amirreza Sarencheh, Hamidreza Khoshakhlagh, Alireza Kavousi, and Aggelos Kiayias. 2025. DART: Decentralized, Anonymous, and Regulation-friendly Tokenization. *ePrint 2025/239*.
- [39] Amirreza Sarencheh, Aggelos Kiayias, and Markulf Kohlweiss. 2022. PEReDi: Privacy-Enhanced, Regulated and Distributed Central Bank Digital Currencies. *Cryptology ePrint Archive*, Paper 2022/974. <https://eprint.iacr.org/2022/974>
- [40] Amirreza Sarencheh, Aggelos Kiayias, and Markulf Kohlweiss. 2024. PARscoin: A Privacy-preserving, Auditable, and Regulation-friendly Stablecoin. In *Cryptology and Network Security: 23rd International Conference, CANS 2024, Cambridge, UK, September 24–27, 2024, Proceedings, Part I* (Cambridge, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 289–313.
- [41] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. Bitfodine: Extracting Intelligence from the Bitcoin Network. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science, Vol. 8437)*. Springer, 457–468. doi:10.1007/978-3-662-45472-5_29
- [42] Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. 2022. UTT: Decentralized Ecash with Accountable Privacy. *IACR Cryptol. ePrint Arch.* (2022). <https://eprint.iacr.org/2022/452>
- [43] Tornado Cash Team. 2021. Tornado Cash Nova: Arbitrary Amounts & Shielded Transfers. <https://github.com/tornadocash/tornado-nova>. Accessed: 2026-01-28.
- [44] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path problems in temporal graphs. *Proceedings of the VLDB Endowment* 7, 9 (2014), 721–732.

- [45] Zhiying Wu, Jieli Liu, Jiajing Wu, Zibin Zheng, and Ting Chen. 2023. TRacer: Scalable graph-based transaction tracing for account-based blockchain trading systems. *IEEE Transactions on Information Forensics and Security* 18 (2023), 2609–2621.
- [46] Karl Wüst, Kari Kostiainen, Vedran Capkun, and Srdjan Capkun. 2018. PRCash: Fast, Private and Regulated Transactions for Digital Currencies. *Cryptology ePrint Archive*, Paper 2018/412. <https://eprint.iacr.org/2018/412>
- [47] Karl Wüst, Kari Kostiainen, Noah Delius, and Srdjan Capkun. 2021. Platypus: A Central Bank Digital Currency with Unlinkable Transactions and Privacy Preserving Regulation. *Cryptology ePrint Archive*, Paper 2021/1443. doi:10.1145/3548606.3560617

A Detailed algorithm for provenance graph extraction

Algorithm. The algorithm implementing an efficient subgraph extraction is described in this section as Algorithm 1.

Algorithm 1 Temporal Subgraph Extraction

Require: $\mathcal{G} = (U, E, \text{src}, \text{dst}, \text{blc}, \text{val})$, target , k , t
Ensure: Edge set $E_{k,t}$ of the extracted subgraph

```

1: PF  $\leftarrow$  map with PF[u] =  $\emptyset$  by default;  $E_{k,t} \leftarrow \emptyset$ 
2: for each  $e \in \text{In}(\text{target}, \infty)$  do ▷ Each entry into target
3:    $E_{k,t} \leftarrow E_{k,t} \cup \{e\}$ 
4:   EXPLORE(src(e), blc(e), max(k - 1, 0), blc(e))
5: end for
6: return  $E_{k,t}$ 

7: procedure EXPLORE( $u, b, h, b_p$ )
8:   if ISDOMINATED(PF[u],  $b, h, b_p$ ) then return
9:   end if
10:  PF[u]  $\leftarrow$  UPDATEFRONTIER(PF[u],  $b, h, b_p$ )
11:  for each  $e \in \text{In}(u, b)$  do
12:    if  $h > 0 \vee b_p - \text{blc}(e) \leq t$  then ▷ Budget check (2)
13:       $E_{k,t} \leftarrow E_{k,t} \cup \{e\}$ 
14:      EXPLORE(src(e), blc(e), max(h - 1, 0),  $b_p$ )
15:    end if
16:  end for
17: end procedure

18: function ISDOMINATED( $F, b, h, b_p$ )
19:   return  $\exists (b', h', b'_p) \in F : b' \geq b \wedge h' \geq h \wedge b'_p \leq b_p$ 
20: end function

21: function IN( $u, b$ )
22:   return  $\{e \in E : \text{dst}(e) = u \wedge \text{blc}(e) < b\}$ 
23: end function

24: function UPDATEFRONTIER( $F, b, h, b_p$ )
25:   return  $\{(b', h', b'_p) \in F : \neg (b \geq b' \wedge h \geq h' \wedge b_p \leq b'_p)\} \cup \{(b, h, b_p)\}$ 
26: end function

```

THEOREM 2 (SOUNDNESS). *Every transaction $e \in E_{k,t}$ lies on a time-respecting path to target satisfying (2) at every position along the path.*

PROOF SKETCH. A transaction e is added to $E_{k,t}$ either directly in the outer loop (as an edge into *target*) or inside a call EXPLORE(u, b ,

h, b_p) where $e \in \text{In}(u, b)$ and the budget check $h > 0 \vee b_p - \text{blc}(e) \leq t$ passed. In either case, we construct the *witness path* for e : the sequence of transactions obtained by following the recursion chain from e back to *target*. Concretely, if e was found in a call triggered by transaction e' (with $\text{src}(e') = u = \text{dst}(e)$), which was itself found via e'' , and so on up to the edge e^* entering *target*, then the witness path is $P = (e, e', e'', \dots, e^*)$. By construction, $\text{blc}(e) < \text{blc}(e') < \text{blc}(e'') < \dots < \text{blc}(e^*)$, so P is time-respecting. The per-transaction condition (2) was verified for each edge before it was added to $E_{k,t}$, with $b_p = \text{blc}(e^*)$. $\square$

THEOREM 3 (COMPLETENESS). *If a transaction e lies on a time-respecting path to target satisfying (2), then $e \in E_{k,t}$.*

PROOF SKETCH. Let $e = e_j$ be a transaction on a valid path $P = (e_1, \dots, e_n)$ to *target* satisfying (2). We argue by induction from e_n to e_j that each edge is added to $E_{k,t}$. The base case is immediate: e_n enters *target* and is added in the outer loop, which calls EXPLORE($\text{src}(e_n)$, $\text{blc}(e_n)$, max($k-1$, 0), $\text{blc}(e_n)$).

For the inductive step, once e_{i+1} is added, the algorithm calls EXPLORE on $\text{src}(e_{i+1})$. Since P is time-respecting, $\text{blc}(e_i) < \text{blc}(e_{i+1}) \leq b$, so e_i is explored. Since P satisfies (2), the budget check passes and e_i is added. If the call was pruned by ISDOMINATED, a dominating label (b', h', b'_p) with $b' \geq b, h' \geq h, b'_p \leq b_p$ belongs to a call that has either already completed or is currently on the recursion stack. The former has processed a superset of edges with a more permissive budget check. The latter will process that superset before returning. Thus e_i has already been added or will be added by the dominating call. In both cases the valid-path exploration is covered and e_i is added to the graph. By induction, every $e_j \in E_{k,t}$. $\square$

THEOREM 4 (TERMINATION). *Algorithm 1 terminates.*

PROOF SKETCH. Each non-pruned call to EXPLORE(u, b, h, b_p) inserts a new non-dominated label (b, h, b_p) into PF(u). The values are drawn from finite domains: b and b_p range over the finitely many distinct block numbers in E , and $h \in \{0, 1, \dots, k\}$. Since the Pareto frontier of any finite subset of $\mathbb{N}^3$ is finite, each node admits only finitely many non-pruned calls. Combined with the finite number of nodes reachable from *target*, the total number of calls is finite. $\square$

In-degree cap. In practice, $\delta_{\max}$ can be very large for addresses with high in-degree, such as exchange hot wallets. The implementation therefore caps the in-degree at a configurable threshold $d_{\max}$: if $|\text{In}(u, b)| > d_{\max}$, node u is flagged as *skipped* and its incoming transactions are not explored. This replaces $\delta_{\max}$ by $d_{\max}$ in the bound above.

B Transaction details

This appendix fixes the exact on-chain payload and the validator checks, and records the abort and restart conditions that Section 6 leaves implicit. The spender is a user u with keys $(\text{sk}_u, \text{idk}_u, \text{pk}_u)$ and the compliance token $\text{ct}_u^{\text{in}} = (\text{idk}_u, \text{pk}_u, \text{tc}_u, \rho_{\text{ct}}^{\text{in}})$ committed under $r_{\text{ct}}^{\text{in}}$, the input coins are $\text{c}_i^{\text{in}} = (v_i^{\text{in}}, \text{pk}_u, \rho_i^{\text{in}}, \text{bv}_i^{\text{in}}, \mathcal{G}_i^{\text{in}})$ opened under r_i^{in} . We write H for the Fiat–Shamir hash.

B.1 Spender computation

The spender constructs a transaction through steps T1–T12 below. Steps T1–T7 instantiate the base transaction, graph update, and blacklist-delta construction described in Sections 6.1 and 6.2, whereas T8–T12 construct the non-blacklisting witnesses, the randomized Bezout transcript, and the NIZK proof.

T1. For each input index i , compute the input-coin nullifier and the compliance-token nullifier

$$\begin{aligned} \text{sn}_i^{\text{in}} &\leftarrow \text{PRF}(\text{sk}_u; \rho_i^{\text{in}}), \\ \text{sn}_{\text{ctu}}^{\text{in}} &\leftarrow \text{PRF}(\text{sk}_u; \rho_{\text{ct}}^{\text{in}}). \end{aligned}$$

T2. Form the multiset union

$$\mathcal{G}^{\text{in}} := \bigcup_i \mathcal{G}_i^{\text{in}},$$

and compute $\text{Update}(\mathcal{G}^{\text{in}})$ by incrementing every hop count and discarding the vertices outside the active window. Constraint (R6) checks this operation directly on the witnessed multiset.

T3. For each output index $j \in \{1, \dots, m\}$, compute the identity trace and source handles of (3) and (4), sample fresh independent encryption randomness η_j^{out} , and set

$$\text{eik}_j^{\text{out}} \leftarrow \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta_j^{\text{out}}).$$

Form the fresh vertex

$$V_j^{\text{out}} = (\text{idt}_j^{\text{out}}, \text{eik}_j^{\text{out}}, 1, b_{\text{now}}, \text{src}_j^{\text{out}}).$$

Independently sample η^{pub} and compute

$$\text{eik}^{\text{pub}} \leftarrow \text{Enc}(\text{idk}_u; \text{pk}_{\text{Reg}}; \eta^{\text{pub}}).$$

T4. For each $j \in \{1, \dots, m\}$, construct the output provenance graph

$$\mathcal{G}_j^{\text{out}} := \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\}.$$

Set $P := P_{\mathcal{G}_1^{\text{out}}}$ for the compliance test; no polynomial commitment to P is formed.

T5. Choose the output values and recipient keys so that $\sum_j v_j^{\text{out}} = \sum_i v_i^{\text{in}}$. For each j , sample a fresh seed $\rho_j^{\text{out}} \xleftarrow{\$} \mathbb{F}$ and fresh commitment randomness $r_j^{\text{out}} \xleftarrow{\$} \mathcal{R}$, set

$$\mathbf{c}_j^{\text{out}} = (v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \rho_j^{\text{out}}, \text{bv}^{\text{cur}}, \mathcal{G}_j^{\text{out}}),$$

and compute $\text{cm}_j^{\text{out}} \leftarrow \text{Com}(r_j^{\text{out}}; \mathbf{c}_j^{\text{out}})$.

T6. Sample a fresh token seed $\rho_{\text{ct}}^{\text{out}} \xleftarrow{\$} \mathbb{F}$ and fresh commitment randomness $r_{\text{ct}}^{\text{out}} \xleftarrow{\$} \mathcal{R}$, set

$$\mathbf{ct}_u^{\text{out}} = (\text{idk}_u, \text{pk}_u, \text{tc}_u + m, \rho_{\text{ct}}^{\text{out}}),$$

and compute $\text{cm}_{\text{ctu}}^{\text{out}} \leftarrow \text{Com}(r_{\text{ct}}^{\text{out}}; \mathbf{ct}_u^{\text{out}})$.

T7. Choose $0 \leq \text{bv}^{\text{in}} \leq \min_i \text{bv}_i^{\text{in}}$, as permitted by (R7), and recompute $Q := Q_{\text{bv}^{\text{in}} \rightarrow \text{bv}^{\text{cur}}}$ from the public blacklist. Take $Q \equiv 1$ when $\text{bv}^{\text{in}} = \text{bv}^{\text{cur}}$; no polynomial commitment to Q is formed.

T8. Compute $\text{uid}_u \leftarrow \text{PRF}(\text{idk}_u; 0)$ and locate the leaves β_u^-, β_u^+ at consecutive positions $\gamma, \gamma+1$ of $\mathcal{T}_{\text{user}}$ with $\beta_u^- < \text{uid}_u < \beta_u^+$, together with their authentication paths to rt_{user} . *Abort* if no such pair exists: the user identifier is then already a leaf and u is blacklisted.

T9. Sample $\mu_0, \mu_1, s_0, s_1 \xleftarrow{\$} \mathbb{F}$ and set $M := \mu_0 + \mu_1 X$, $\tilde{P} := P + MQ$, $S := s_0 + s_1 X$. Run extended Euclid on $(\tilde{P}, Q)$ for the reduced pair (U_0, W_0) , taking $(U_0, W_0) = (0, 1)$ when $Q \equiv 1$, and *abort* if no such pair exists. Randomize by setting $U := U_0 + SQ$ and $W := W_0 - S\tilde{P}$, which preserves $U\tilde{P} + WQ = 1$. All three polynomials have degree at most the KZG bound d .

T10. With fresh r_{mask} , compute $\text{cm}_{\text{mask}} \leftarrow \text{Com}(r_{\text{mask}}; \mu_0, \mu_1)$. Set $C_U \leftarrow \text{KZG.Com}(U)$ and $C_W \leftarrow \text{KZG.Com}(W)$, assemble ctx as in (11), and derive $z \leftarrow H(\text{ctx})$ and $q \leftarrow Q(z)$. If $q = 0$, *restart* T9 with fresh $\mu_0, \mu_1, s_0, s_1, r_{\text{mask}}$. The output graph is bound before the challenge through $\text{cm}_1^{\text{out}} \in \text{ctx}$, and C_U, C_W bind the Bezout witnesses before z is sampled.

T11. Publish $p' \leftarrow \tilde{P}(z)$ and $u' \leftarrow U(z)$ with the KZG openings π_U, π_W of C_U, C_W at z . These are verified outside the NIZK, and there are no openings for input coins, output coins, or Q .

T12. Run $\text{NIZK.Prove}(\text{crs}, x_{\text{NIZK}}, w_{\text{NIZK}}) \rightarrow \pi$ on the statement x_{NIZK} of Section 6.3, discharging (R1)–(R10), with witness

$$\begin{aligned} w_{\text{NIZK}} = & (\text{sk}_u, \text{idk}_u, \text{pk}_u, \text{tc}_u, \rho_{\text{ct}}^{\text{in}}, r_{\text{ct}}^{\text{in}}, \rho_{\text{ct}}^{\text{out}}, r_{\text{ct}}^{\text{out}}, \{\eta_j^{\text{out}}\}_j, \eta^{\text{pub}}, \\ & \{(v_i^{\text{in}}, \text{pk}_u, \rho_i^{\text{in}}, \text{bv}_i^{\text{in}}, \mathcal{G}_i^{\text{in}}, r_i^{\text{in}}, \text{path}_i)\}_i, \\ & \{(v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \rho_j^{\text{out}}, \text{bv}_j^{\text{cur}}, \mathcal{G}_j^{\text{out}}, r_j^{\text{out}}, V_j^{\text{out}})\}_j, \\ & \text{path}_{\text{ctu}}^{\text{in}}, \beta_u^-, \beta_u^+, \gamma, \text{path}_u^-, \text{path}_u^+, \mu_0, \mu_1, r_{\text{mask}}). \end{aligned}$$

B.2 Transaction payload

The spender publishes

$$\begin{aligned} \text{tx} = & (\{\text{sn}_i^{\text{in}}\}_i, \text{sn}_{\text{ctu}}^{\text{in}}, \{\text{cm}_j^{\text{out}}\}_j, \text{cm}_{\text{ctu}}^{\text{out}}, \\ & \text{bv}^{\text{in}}, \text{bv}^{\text{cur}}, b_{\text{now}}, \text{rt}, \text{rt}_{\text{user}}, \text{eik}^{\text{pub}}, \text{cm}_{\text{mask}}, C_U, C_W, \\ & p', u', \pi_U, \pi_W, \pi). \end{aligned}$$

Neither z nor q is transmitted: validators reconstruct ctx from the payload and derive both.

B.3 Validator verification

Validators accept tx iff all of the following succeed.

V1. Every $\text{sn} \in \{\text{sn}_1^{\text{in}}, \dots, \text{sn}_n^{\text{in}}, \text{sn}_{\text{ctu}}^{\text{in}}\}$ is absent from $\mathcal{T}_{\text{sn}}$, and these $n+1$ values are pairwise distinct.

V2. rt is an accepted historical root of $\mathcal{T}_{\text{cm}}$, b_{now} matches the current block counter, $(\text{bv}^{\text{cur}}, \text{rt}_{\text{user}})$ is exactly the latest published blacklist state, and $\text{bv}^{\text{in}} \leq \text{bv}^{\text{cur}}$.

V3. Recompute Q from the public blacklist, reconstruct ctx as in (11), derive $z \leftarrow H(\text{ctx})$, and evaluate $q \leftarrow Q(z)$ by multiplying the public root factors. Reject if $q = 0$.

V4. Set $w' \leftarrow (1 - u'p')/q$ and check

$$\text{KZG.Verify}_{\text{pp}}(C_U, z, u', \pi_U) = 1,$$

$$\text{KZG.Verify}_{\text{pp}}(C_W, z, w', \pi_W) = 1.$$

V5. Reconstruct x_{NIZK} from tx and the locally derived (z, q) . Then run

$$\text{NIZK.Verify}(\text{crs}, x_{\text{NIZK}}, \pi).$$

Here (R8) proves the sender's non-membership against the current user-blacklist root, (R9) opens the mask commitment, and (R10) ties p' to the direct evaluation of the first output graph. The circuit performs no KZG verification.

V6. On acceptance, insert the $n + 1$ nullifiers into $\mathcal{T}_{\text{sn}}$ and the $m + 1$ commitments into $\mathcal{T}_{\text{cm}}$.

Completeness and soundness. An honestly assembled tx satisfies V1–V5 by construction: the nullifiers are computed in T1 and are fresh and pairwise distinct because an honest spender selects distinct unspent coins and her unique unspent compliance token; V2 holds because the spender uses the latest accepted roots and the current block; the validator’s values (ctx, z, q) from V3 match those from T10; the openings of V4 verify by (10); and the proof of T12 satisfies $\mathcal{R}$. Soundness of the pipeline is Theorem 6 in Appendix C.3: V1–V5 together with NIZK simulation extractability, KZG knowledge soundness, and the Schwartz–Zippel argument at $z = H(\text{ctx})$ force every accepted transaction to respect the update rule of Section 6.1 and the coprimality condition (8).

C Formal Security Analysis

This appendix gives the formal threat model, security definitions, and proof sketches summarized in Section 8. We write $\text{Adv}_{\mathcal{A}}^{\bullet}(\lambda)$ for an advantage and $\text{negl}(\lambda)$ for a negligible function. The definitions and construction allow general transaction shapes (n, m) , but when not generalized to (n, m) , the proofs are written for the fixed shape $n = m = 2$, as in Zerocash, and the displayed bounds are instantiated at that shape. The generalization to arbitrary (n, m) is mainly there to increase the usability and flexibility of the construction for different transaction shapes.

C.1 Threat model and assumptions

We consider three PPT adversaries. $\mathcal{A}_{\text{pub}}$ sees the full ledger: the commitment tree $\mathcal{T}_{\text{cm}}$, the nullifier tree $\mathcal{T}_{\text{sn}}$, the versioned trace blacklists $\{B_{\text{idt}}^{(v)}\}_v$, the user tree $\mathcal{T}_{\text{user}}$, and the transactions $\{\text{tx}_i\}_i$. It may inject transactions on behalf of a corrupted user set $\mathcal{U}_{\text{corr}}$, but does not hold sk_{Reg} . $\mathcal{A}_{\text{hold}}$ has the same view plus the openings of a set $\mathcal{C}_{\text{hold}}$ of coins it controls, including their embedded provenance graphs. $\mathcal{A}_{\text{reg}}$ holds sk_{Reg} and the identity registry, follows the protocol honestly, and may receive disclosure openings at a user’s request.

Standard assumptions. We assume ledger indistinguishability and balance of base Zerocash [5]; simulation extractability and zero knowledge of NIZK, with respective advantages $\epsilon_{\text{NIZK}}^{\text{SE}}$ and $\epsilon_{\text{NIZK}}^{\text{ZK}}$; KZG knowledge soundness and binding under q -DLOG in the algebraic group model, with respective advantages $\epsilon_{\text{KZG}}^{\text{KS}}$ and $\epsilon_{\text{KZG}}^{\text{bind}}$; computational hiding and binding of Com, with respective advantages $\epsilon_{\text{Com}}^{\text{hide}}$ and $\epsilon_{\text{Com}}^{\text{bind}}$; IND-CPA security of (Enc, Dec) , with advantage $\epsilon_{\text{Enc}}^{\text{CPA}}$; and pseudorandomness of PRF, with advantage ϵ_{PRF} . We further assume collision resistance of PRF, with advantage $\epsilon_{\text{PRF}}^{\text{CR}}$: for independently sampled distinct keys k and k' and any inputs x and x' , even when the keys are exposed, $\Pr[\text{PRF}(k; x) = \text{PRF}(k'; x')] \leq \epsilon_{\text{PRF}}^{\text{CR}}$. This is the analogue of the collision-freeness Zerocash requires of its nullifier PRF and is separate from pseudorandomness. We work in the random-oracle model for the Fiat–Shamir hash H , with q_H bounding the adversary’s queries to H .

Parameters and payload. We write d for the KZG degree bound, chosen as in Section 6.2 so that it bounds the degrees of $\tilde{P}$, U , W and $Q_{\text{bv}^{\text{in}} \rightarrow \text{bv}^{\text{cur}}}$ for every admissible transaction. Identity trace

blacklist entries are generated from registered identity keys and counters, independently of the KZG trapdoor τ ; Remark 2 shows why this independence is needed. Three features of the payload of Appendix B are load-bearing below: $z = H(\text{ctx})$ and $q = Q(z)$ are rederived by validators rather than transmitted and hence are functions of ctx ; coins carry provenance graphs and no polynomial commitments, so nothing in the system commits to P or to Q ; and the only KZG commitments are the per-transaction C_U, C_W , bound into ctx and opened exactly once, at z , under (V4), outside the circuit.

Compliance-token synchronization. We assume the regulator’s per-transaction counter update is sequenced atomically with transaction acceptance, so that, at any compliance check, $\text{tc}_u^* = \text{tc}_u$, where tc_u is the user’s pre-transaction counter. This is consistent with the compliance-token mechanism, which forces tc_u to be monotonically incremented with each accepted transaction.

C.2 Balance

THEOREM 5 (BALANCE). *The construction satisfies balance with losses additive in $\epsilon_{\text{NIZK}}^{\text{SE}}$ and $\epsilon_{\text{Com}}^{\text{bind}}$.*

PROOF SKETCH. Constraint (R1) reproduces the Zerocash base-layer obligations identically on the extended commitment input

$$(v, \text{pk}, \rho, \text{bv}, \mathcal{G}),$$

and (V1) additionally forces all input-coin and compliance-token nullifiers to be pairwise distinct, so no coin is counted twice. So in particular for the 2-input transactions, no input coin is counted twice. Commitment binding therefore applies unchanged, and balance reduces losslessly to the Zerocash guarantee [5]. $\square$

C.3 Compliance soundness

DEFINITION 1 (COMPLIANCE SOUNDNESS).

The experiment $\text{Compsound}^{\mathcal{A}}(\lambda)$ proceeds as follows. The challenger runs $\text{Setup}(1^\lambda)$, generating the CRS, the KZG parameters pp and the regulator keys, registers an honest user set $\mathcal{U}_{\text{hon}}$ via the KYC interface, and hands $\mathcal{A}$ the public parameters. $\mathcal{A}$ may corrupt users outside $\mathcal{U}_{\text{hon}}$, learning their (sk, idk) , and has oracle access to honest-user transactions, to blacklisting requests on any registered user, and to the random oracle H . It finally outputs tx^* , and wins iff validators accept tx^* and either

- (a) *some output coin has $\text{Traces}(\mathcal{G}_j^{\text{out}}) \cap \text{DeltaBlack}(\text{bv}^{\text{in}}, \text{bv}^{\text{cur}}) \neq \emptyset$, or*
- (b) *some $\mathcal{G}_j^{\text{out}}$ differs from $\text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\}$, or*
- (c) *the sender’s identifier is in the current user blacklist $\mathcal{T}_{\text{user}}$.*

For clause (b), the challenger records, for every coin it creates, the opening and the nullifier that will identify it when spent; it resolves the input nullifiers of tx^* against those records and defines $\mathcal{G}^{\text{in}}$ as the multiset union of the corresponding embedded graphs. The graph in the winning condition is thus fixed by challenger-maintained ledger state.

THEOREM 6. *For every PPT algebraic $\mathcal{A}$,*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{Compsound}}(\lambda) &\leq \epsilon_{\text{NIZK}}^{\text{SE}} + 2\epsilon_{\text{KZG}}^{\text{KS}} + 2\epsilon_{\text{KZG}}^{\text{bind}} \\ &\quad + 5\epsilon_{\text{Com}}^{\text{bind}} + 2|B_{\text{idt}}| \epsilon_{\text{PRF}}^{\text{CR}} + \mathcal{O}\left(\frac{q_H \cdot d}{|\mathbb{F}|}\right), \end{aligned}$$

where d is the KZG degree bound, $|B_{\text{idt}}|$ is the size of the current identity trace blacklist, and q_H is the adversary's query bound on H .

PROOF SKETCH. (1) Clause (b) and (c) are enforced structurally. Simulation extractability applied to π recovers a witness for $\mathcal{R}$: the input- and output-coin openings with their Merkle paths (R1), the compliance-token openings carrying idk_u and tc_u (R2), the adjacent-leaf witnesses (R8), and the mask opening $(\mu_0, \mu_1, r_{\text{mask}})$ (R9). Commitment binding fixes the two spent commitments, the two output commitments and cm_{mask} , for a loss of $5\epsilon_{\text{Com}}^{\text{bind}}$. Constraint (R6) then forces $\mathcal{G}_j^{\text{out}} = \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\}$ as an equality of structured multisets, with (R3)–(R5) pinning the fresh vertex and (R7) the version fields. Constraint (R8) enforces that the sender's identifier is not in the current user blacklist $\mathcal{T}_{\text{user}}$.

(2) $\tilde{P}$ is determined before z . Write $P := P_{\mathcal{G}_1^{\text{out}}}$, $M := \mu_0 + \mu_1 X$, $Q := Q_{\text{bv}^{\text{in}} \rightarrow \text{bv}^{\text{cur}}}$ and $\tilde{P} := P + MQ$. Since cm_1^{out} and cm_{mask} are inputs of ctx , binding makes P and M functions of ctx , and Q is a deterministic function of $(\text{bv}^{\text{in}}, \text{bv}^{\text{cur}}) \in \text{ctx}$ and of the public blacklist state. Constraints (R9) and (R10) force $p' = y_p + (\mu_0 + \mu_1 z)q$ with $y_p = \prod_{V \in \mathcal{G}_1^{\text{out}}} (z - V.\text{idt})$ computed on the multiset bound inside cm_1^{out} , that is, $p' = \tilde{P}(z)$.

(3) *Extraction and Schwartz–Zippel.* $C_U, C_W \in \text{ctx}$ are produced before z is sampled. Since $\mathcal{A}$ is algebraic, it outputs them with representation vectors over the SRS elements it has seen, from which the KZG extractor reads $\tilde{U}, \tilde{W} \in \mathbb{F}[X]$ of degree at most d with $C_U = \text{KZG.Com}(\tilde{U})$, $C_W = \text{KZG.Com}(\tilde{W})$. The two openings accepted at (V4) force $\tilde{U}(z) = u'$ and $\tilde{W}(z) = w' = (1 - u'p')/q$ up to evaluation binding, with $q \neq 0$ by the rejection rule (V3). The validator's derivation of w' makes $u'p' + w'q = 1$ hold by construction, so

$$\tilde{U}(z)\tilde{P}(z) + \tilde{W}(z)Q(z) = 1$$

at $z = H(\text{ctx})$, where all four polynomials are determined by ctx . Modeling H as a random oracle and union-bounding over the q_H queries, Schwartz–Zippel then allows to write the polynomial identity $\tilde{U}\tilde{P} + \tilde{W}Q = 1$ except with probability $O(q_H d / |\mathbb{F}|)$. By Bezout's identity, $\gcd(\tilde{P}, Q) = 1$, and $\tilde{P} \equiv P \pmod{Q}$ gives $\gcd(P_{\mathcal{G}_1^{\text{out}}}, Q) = 1$.

(4) *From the first output to every output.* By step (1), $P_{\mathcal{G}_j^{\text{out}}} = (X - \text{idt}_j^{\text{out}}) \cdot P_{\text{Update}(\mathcal{G}^{\text{in}})}$ for every j , so the shared trunk inherits coprimality with Q and only the fresh traces remain to be excluded. Acceptance means that (R8) proves the non-membership of $\text{uid}_u = \text{PRF}(\text{idk}_u; 0)$ under the current root rt_{user} , so u is not blacklisted. Every entry of the current B_{idt} , and a fortiori of the delta, is of the form $\text{PRF}(\text{idk}_{u'}; j)$ for some blacklisted user u' . Membership of $\text{idt}_j^{\text{out}}$ in B_{idt} would therefore exhibit a collision between independently sampled distinct PRF keys, which collision resistance rules out even when both keys are known, with loss at most $2|B_{\text{idt}}|\epsilon_{\text{PRF}}^{\text{CR}}$ after union-bounding over both fresh traces and all blacklist entries. Clause (a) thus fails for every output. The losses sum to the stated bound. $\square$

C.4 Disclosure integrity

DEFINITION 2 (DISCLOSURE INTEGRITY).

In the experiment $\text{Disclnt}^{\mathcal{A}}(\lambda)$, setup is as in $\text{Compsound}^{\mathcal{A}}$. The adversary outputs $\text{cm} \in \mathcal{T}_{\text{cm}}$ and an opening

$$\text{op} = (v, \text{pk}, \rho, \text{bv}, \mathcal{G}^{\dagger}; r).$$

It wins iff both

$$\text{cm} = \text{Com}(r; v, \text{pk}, \rho, \text{bv}, \mathcal{G}^{\dagger}),$$

$$\mathcal{G}^{\dagger} \neq \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V^{\text{out}}\}$$

hold for the $(\mathcal{G}^{\text{in}}, V^{\text{out}})$ prescribed by the on-chain transaction that produced cm .

THEOREM 7. For every PPT $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{Disclnt}}(\lambda) \leq \epsilon_{\text{NIZK}}^{\text{SE}} + \epsilon_{\text{Com}}^{\text{bind}}$. Moreover, every source handle whose child and intended parent both remain in the disclosed graph $\mathcal{G}^{\dagger}$ is reconstructed faithfully, and each handle matches at most one active parent trace except with probability at most $|\text{src}| \cdot |\text{Traces}(\mathcal{G}^{\dagger})| \cdot \epsilon_{\text{PRF}}$.

PROOF SKETCH. Simulation extractability applied to the proof of the transaction that produced cm recovers a witness for $\mathcal{R}$, including the output graph $\mathcal{G}_j^{\text{out}}$ stored inside cm . Since cm binds $(v, \text{pk}, \rho, \text{bv}, \mathcal{G}_j^{\text{out}})$, a disclosed $\mathcal{G}^{\dagger} \neq \mathcal{G}_j^{\text{out}}$ is a second opening of cm and contradicts binding. Constraint (R6) forces $\mathcal{G}_j^{\text{out}} = \text{Update}(\mathcal{G}^{\text{in}}) \uplus \{V_j^{\text{out}}\}$ as a structured multiset, that is, including hop counts, block numbers, source handles and per-vertex ciphertexts, and not only identity traces. Combining the two gives the claim.

For edge reconstruction, every handle attached to a vertex with trace idt_j has the form $\sigma = \text{PRF}(\text{idk}_u; \text{idt}_\ell \parallel \text{idt}_j)$ by (4), where idk_u is the emitter key the regulator recovers from eik . If both endpoints survive the active-window filter, the edge is recovered; if the parent was filtered while the child survived, the handle may have no active match and records an edge crossing the active-window boundary. By PRF pseudorandomness, a different active trace also satisfies the equation with probability at most ϵ_{PRF} per candidate pair, giving the stated bound across the graph. $\square$

C.5 Transaction indistinguishability

Before defining the leakage functions, we isolate the property that makes the published Bezout values harmless. It holds statistically once the mask commitment has been hidden, and remains valid whether or not the adversary receives sk_{Reg} and the identity registry.

LEMMA 8 (PROVENANCE INDEPENDENCE). The published compliance values

$$(C_U, C_W, p', u', \pi_U, \pi_W)$$

are simulatable without knowledge of P .

PROOF. First consider an execution in which $z = H(\text{ctx})$ is a fresh independent field element, and assume that $Q(\tau) \neq 0$, $z \neq \tau$, and $Q(z) \neq 0$. A previous query on ctx is treated as a bad event and accounted for below.

The proof uses one elementary fact: if A is a uniform linear polynomial and $x \neq y$, then $(A(x), A(y))$ is uniform on $\mathbb{F}^2$. Here $U = U_0 + SQ$ and $\tilde{P} = P + MQ$, where S and M are independent uniform linear polynomials. Multiplication by the nonzero values $Q(\tau)$ and $Q(z)$ preserves uniformity. Therefore

$$(\mu, \psi, u', p') := (U(\tau), \tilde{P}(\tau), U(z), \tilde{P}(z))$$

is uniform on $\mathbb{F}^4$. Evaluating the Bezout identity at τ gives

$$\mu\psi + W(\tau)Q(\tau) = 1, \quad (13)$$

and hence

$$(C_U, C_W, p', u') = ([\mu]_1, [(1 - \mu\psi)/Q(\tau)]_1, p', u').$$

This distribution does not depend on P . For every $\mu \neq 0$, the uniform value ψ makes C_W uniform given C_U . Only $\mu = 0$, which occurs with probability $1/|\mathbb{F}|$, forces $C_W = [1/Q(\tau)]_1$. Thus this tuple is less than $1/|\mathbb{F}|$ away from uniform.

A simulator can now sample C_U, C_W uniformly and insert them into ctx before deriving z . It then samples p', u' uniformly, sets $w' = (1 - u'p')/Q(z)$, and uses τ to compute the accepting KZG proofs π_U and π_W . The simulation uses no information about P . The events $Q(\tau) = 0, z = \tau, Q(z) = 0$, or a previous query on ctx have total probability $O((q_H + d)/|\mathbb{F}|)$: the probability of $Q(\tau) = 0$ or $Q(z) = 0$ is $O(d/|\mathbb{F}|)$ and the probability of a previous query on ctx is $O(q_H/|\mathbb{F}|)$. The proof holds for every admissible T7 choice and does not use sk_{Reg} or the registry, so they may be given as auxiliary input. Hence the values (C_U, C_W, p', u') are simulatable without knowledge of P . $\square$

Remark 1 (Mask and witness randomizers necessity). Both randomizers are needed for the statistical independence in Lemma 8. The mask M hides P , while S hides the reduced witness U_0 . The regulator, knowing the initial state of the system, could try to make guesses on P at each transfer. Without such randomizers, the regulator could test each candidate P against the observed commitments and openings, potentially compromising the privacy of the underlying provenance information as explained below.

Instead of a degree-1 polynomial, a constant S is insufficient: for each candidate P , the value u' determines $S = (u' - U_0(z))/Q(z)$, allowing a check against C_U . A constant M makes p' uniform but leaves the relation

$$\tilde{P}(\tau)Q(z) - p'Q(\tau) = P(\tau)Q(z) - P(z)Q(\tau),$$

so the joint distribution of the published values can still depend on P . When $\tau \neq z$, a polynomial of degree at most one with uniform coefficients has independent uniform evaluations at these two points. Sampling M and S independently this way therefore gives the independence used in the lemma, provided $Q(\tau)Q(z) \neq 0$, which is ensured by the protocol's rejection rules.

Remark 2 (Trapdoor destruction is needed for unlinkability). Nothing in the protocol constrains identity trace blacklist entries to be well-formed PRF outputs. A regulator who learned τ could publish the entry $\beta = \tau$, forcing $Q(\tau) = 0$; then $\mu = U_0(\tau)$ is a deterministic function of the candidate provenance polynomial and C_U becomes a direct candidate oracle. Destruction of τ after setup is thus load-bearing for privacy, and not only for binding.

DEFINITION 3 (PUBLIC LEAKAGE). For a transaction with n inputs and m outputs, $\text{Leak}_{\text{pub}}(\text{tx}) := (n, m, \text{bv}^{\text{in}}, \text{bv}^{\text{cur}}, b_{\text{now}})$. The field bv^{in} is chosen by the spender in T7, and when every user adopts the convention $\text{bv}^{\text{in}} = 0$ it is constant and carries no information, at the cost of proving coprimality against the full blacklist. The ciphertext eik^{pub} and the compliance values $(\text{cm}_{\text{mask}}, C_U, C_W, p', u', \pi_U, \pi_W)$ are excluded from Leak_{pub} and allowed to differ between challenge transactions: the former is handled by IND-CPA of Enc, and hybrid (H2)

below shows via Lemma 8 that the latter are simulatable without private input.

DEFINITION 4 (HOLDER-SIDE LEAKAGE). Let $\mathcal{A}_{\text{hold}}$ hold $C_{\text{hold}} = \{\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(C)}\}$ with embedded graphs $\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(C)}$, across all corrupted users. Let $\text{idt}_1, \dots, \text{idt}_N$ be the distinct identity traces occurring in $\bigcup_i \text{Traces}(\mathcal{G}^{(i)})$ in canonical order, and let b_v be the creation block recorded by every vertex carrying idt_v . For $i \in [C]$ and $v \in [N]$, let

$$\mathbf{H}_{i,v} := \{V.h : V \in \mathcal{G}^{(i)}, V.\text{idt} = \text{idt}_v\}$$

be the multiset of hop counts at which idt_v occurs in $\mathcal{G}^{(i)}$, empty when $\text{idt}_v \notin \text{Traces}(\mathcal{G}^{(i)})$. The holder-side leakage is $\text{Leak}_{\text{hold}}(C_{\text{hold}}) := (\{b_v\}_v, \{\mathbf{H}_{i,v}\}_{i,v})$, recomputed over the current state of C_{hold} at every point of the experiment.

Remark 3 (Closure of $\text{Leak}_{\text{hold}}$). Every remaining component of a coin opening visible to $\mathcal{A}_{\text{hold}}$ is uniform or determined by $\text{Leak}_{\text{hold}}$, which is what hybrids (H3)–(H4) require: traces and source handles are uniform after (H4), per-vertex ciphertexts are encryptions of zero after (H3), and each handle list has exactly n entries.

DEFINITION 5 (TIND). The game $\text{TIND}^{\mathcal{A}}(\lambda)$ is defined for $\mathcal{A} \in \{\mathcal{A}_{\text{pub}}, \mathcal{A}_{\text{hold}}\}$; in particular, $\mathcal{A}$ does not hold sk_{Reg} .

- (1) The challenger C samples $b \xleftarrow{\$} \{0, 1\}$ and initializes two ledgers L_0, L_1 under the same public parameters. It maintains paired references for corresponding honest and corrupted coins in the two branch states. The adversary is given the public parameters and read access only to L_b .
- (2) $\mathcal{A}$ chooses a corruption set $\mathcal{U}_{\text{corr}}$ (whose secret keys it receives) and an initial holder set C_{hold} of coins it owns. Corruption is static: $\mathcal{U}_{\text{corr}}$ is fixed at this step and $\mathcal{A}$ may not corrupt further users during the experiment. The holder set C_{hold} nevertheless grows during the experiment, since challenge transactions may address outputs to corrupted users. Openings and subsequent holder-state updates are returned only for the selected branch. Paired references let $\mathcal{A}$ address the corresponding object in either branch without learning the unselected ledger.
- (3) The adversary submits pairs of transaction instructions $(\text{tx}_0, \text{tx}_1)$ from honest senders using the paired coin references. Each instruction specifies the inputs, recipients, values, and inherited blacklist version. The challenger samples the honest sender's randomness and constructs the transaction in each ledger. tx_i is valid in the current state of L_i and the pair is admissible in the following sense.
 - (A1) **Public consistency.** The pair satisfies the base Zerocash public-consistency requirements, including the same transaction shape and any intentionally public balance data, and $\text{Leak}_{\text{pub}}(\text{tx}_0) = \text{Leak}_{\text{pub}}(\text{tx}_1)$.
 - (A2) **Agreement on adversary-held outputs.** For every output index j whose recipient key pk_j^{out} belongs to a user in $\mathcal{U}_{\text{corr}}$, the two branches agree on $(v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \text{bv}_j^{\text{out}})$. The seed ρ_j^{out} and the commitment randomness r_j^{out} are not required to agree: they are sampled uniformly and independently in both branches.
 - (A3) **Structural agreement.** The branch holder states satisfy

$$\text{Leak}_{\text{hold}}(C^{(0)}) = \text{Leak}_{\text{hold}}(C^{(1)}),$$

where $C^{(i)}$ is the holder state in branch i after executing tx_i .

Branch-specific commitments, nullifiers, ciphertexts, identity traces, source handles, and proofs may differ. The challenger executes both transactions, appending the transaction resulting from tx_i only to L_i , and exposes only the new L_b view and its authorized holder openings.

- (4) $\mathcal{A}$ may also submit ordinary transactions for its own coins. The challenger executes the corresponding transaction in both ledgers if they are admissible. In either case only the L_b result is exposed.
- (5) $\mathcal{A}$ outputs b' and wins iff $b' = b$.

THEOREM 9 (PUBLIC TRANSACTION INDISTINGUISHABILITY). For every PPT $\mathcal{A}_{\text{pub}}$, that is, every adversary with $C_{\text{hold}} = \emptyset$ throughout the experiment, condition (A3) is vacuous and

$$\begin{aligned} \text{Adv}_{\mathcal{A}_{\text{pub}}}^{\text{TIND}}(\lambda) &\leq |Q_{\text{ch}}| \cdot \left(\epsilon_{\text{NIZK}}^{\text{zk}} + (m+2) \epsilon_{\text{Com}}^{\text{hide}} + (m+1) \epsilon_{\text{Enc}}^{\text{cpa}} \right. \\ &\quad \left. + (n+m+nm+1) \epsilon_{\text{PRF}} + O((q_H + d)/|\mathbb{F}|) \right) \\ &\quad + \epsilon_{\text{PRF}} + \text{Adv}_{\mathcal{B}}^{\text{Zerocash-LI}}(\lambda). \end{aligned}$$

where $|Q_{\text{ch}}|$ is the number of challenge pairs and the bound is instantiated at $n = m = 2$ and $\text{Adv}_{\mathcal{B}}^{\text{Zerocash-LI}}(\lambda)$ is the advantage of the reduction against the Zerocash ledger indistinguishability game. Equivalently, a public observer's view is Zerocash ledger indistinguishability augmented by Leak_{pub} alone, and by nothing else.

THEOREM 10 (HOLDER TRANSACTION INDISTINGUISHABILITY). For every PPT $\mathcal{A}_{\text{hold}}$, the same bound holds, with $(m+2) \epsilon_{\text{Com}}^{\text{hide}}$ replaced by $(m_{\text{hon}} + 2) \epsilon_{\text{Com}}^{\text{hide}}$, where m_{hon} is the number of challenge outputs addressed to honest users.

PROOF SKETCH. The two statements share one hybrid sequence, applied to each challenge transaction. They differ only in whether any challenge output is addressed to a corrupted user, which affects the commitment-hiding count in (H2) and nothing else. The reduction runs the KZG setup itself and keeps τ , which it uses only to simulate opening proofs and never gives to $\mathcal{A}$.

(H1) Simulate the NIZK, at a loss of $\epsilon_{\text{NIZK}}^{\text{zk}}$ per challenge transaction, using zero knowledge. The proof then carries no information about coin and compliance token openings, embedded graphs, Merkle paths, the identity key, or the mask opening.

(H2) Hide the commitments the adversary never opens, and simulate the published compliance values. Writing m_{hon} for the number of challenge outputs addressed to honest users, replace cm_{mask} , those m_{hon} output commitments and the updated compliance token commitment by commitments to unrelated messages under fresh valid randomness, in $m_{\text{hon}} + 2$ subhybrids of total loss $(m_{\text{hon}} + 2) \epsilon_{\text{Com}}^{\text{hide}}$. The compliance token belongs to the honest sender and is never opened to $\mathcal{A}$. Commitments to corrupted-user outputs are left unchanged, exactly as disclosed commitments are in Theorem 11, and their openings are identically distributed once the remaining hybrids are applied: the tuple $(v_j^{\text{out}}, \text{pk}_j^{\text{out}}, \text{bv}_j^{\text{out}})$ agrees by (A2), ρ_j^{out} and r_j^{out} are uniform and independent per branch, the embedded ciphertexts become encryptions of zero in (H3), traces and handles become uniform in (H4), and the rest of $\mathcal{G}_j^{\text{out}}$ agrees by (A3).

The mask subhybrid must precede the invocation of Lemma 8, which then makes the remaining compliance values indistinguishable to a distribution using no private input and respecting the

commit-then-hash order: sample $C_U, C_W \xleftarrow{\$} \mathbb{G}_1$, assemble ctx and derive z, q , sample $p', u' \xleftarrow{\$} \mathbb{F}$, set $w' = (1 - u'p')/q$, and compute $\pi_U = (C_U g_1^{-u'})^{1/(\tau-z)}$ and $\pi_W = (C_W g_1^{-w'})^{1/(\tau-z)}$. The simulator therefore needs τ but not the discrete logarithm of either sampled commitment, at a loss of $O((q_H + d)/|\mathbb{F}|)$ per challenge transaction.

(H3) Replace each regulator-key ciphertext by an encryption of zero. This covers the per-vertex eik of fresh output vertices, observable to a corrupted holder through $\text{Leak}_{\text{hold}}$, and the public eik^{pub} , for a loss of $3\epsilon_{\text{Enc}}^{\text{cpa}}$ per challenge transaction.

(H4) Replace each PRF output keyed by an honest user's secret material by a uniform value. The per-transaction PRF values are two identity traces, at most four source handles, and the three nullifiers of the input coins and the compliance token, hence a loss of $9\epsilon_{\text{PRF}}$. The public-key derivation $\text{PRF}(\text{sk}_u; 0)$ is switched once across the game, outside the $|Q_{\text{ch}}|$ multiplier, for the separate loss ϵ_{PRF} . The user identifier uid_u is never published and contributes no term.

After (H1)–(H4), every per-transaction field outside $\text{Leak}_{\text{pub}} \cup \text{Leak}_{\text{hold}}$ is uniform or a deterministic function of public values, so (A1)–(A3) make the residual views of tx_0 and tx_1 identical, and the remaining gap reduces to base Zerocash ledger indistinguishability. A trace candidate $\text{idt}^\dagger$ in $\mathcal{A}$'s auxiliary input gives no membership oracle on third-party graphs: by (H2) the transaction exposes no evaluation of the provenance polynomial or of any Bezout witness depending on $\mathcal{A}$'s private input. $\square$

C.6 Regulator transaction indistinguishability

The preceding game excludes the regulator's secret key. We now give the adversary that key and require indistinguishability only beyond what the regulator is intended to learn.

DEFINITION 6 (RegIND). RegIND is the two-ledger game of Definition 5 with two changes. The adversary is $\mathcal{A}_{\text{reg}}$ and receives sk_{reg} and the identity registry at setup. In addition to (A1)–(A3), every challenge pair must satisfy

$$\begin{aligned} \text{Leak}_{\text{reg}}(\text{tx}_0) &= \text{Leak}_{\text{reg}}(\text{tx}_1), \\ \text{Leak}_{\text{reg}}(\text{tx}) &:= (\text{idk}_u, \{\text{idt}_j^{\text{out}}\}_{j=1}^m), \end{aligned}$$

where idk_u is what the regulator obtains by decrypting eik^{pub} and the output traces are what it derives from that key and its synchronized counter. User-authorized disclosure openings are explicit leakage: disclosure queries must be consistent between branches, and the openings are given to the adversary in either branch.

THEOREM 11 (REGULATOR TRANSACTION INDISTINGUISHABILITY). For every PPT $\mathcal{A}_{\text{reg}}$,

$$\begin{aligned} \text{Adv}_{\mathcal{A}_{\text{reg}}}^{\text{RegIND}}(\lambda) &\leq |Q_{\text{ch}}| \cdot \left(\epsilon_{\text{NIZK}}^{\text{zk}} + 3 \epsilon_{\text{Com}}^{\text{hide}} + 3 \epsilon_{\text{PRF}} \right. \\ &\quad \left. + O((q_H + d)/|\mathbb{F}|) \right) + \text{Adv}_{\mathcal{B}}^{\text{Zerocash-LI}}(\lambda), \end{aligned}$$

Thus, apart from user-authorized disclosures, the regulator can group transactions by sender and derive their fresh output traces, but cannot distinguish challenge transactions agreeing on this leakage.

PROOF SKETCH. Run the hybrids (H1)–(H4) from the proof of Theorem 9, with $(\text{sk}_{\text{reg}}, \text{registry})$ as auxiliary input to every reduction, and with three changes.

First, (H2) hides only three commitments: cm_{mask} , whose sub-hybrid must again precede the invocation of Lemma 8, and the two output-coin commitments, for a total of $3\epsilon_{\text{Com}}^{\text{hide}}$. A commitment opened through an authorized disclosure is already explicit leakage and must agree across branches, so it is left unchanged rather than switched. The compliance-token commitment needs no switch here: Leak_{reg} fixes the sender and the output traces, and hence the registry key, synchronized counter, and public key agree between admissible branches, while the fresh token seed and randomness are identically distributed in both.

Second, (H3) is dropped entirely. Decrypting the public and embedded regulator ciphertexts yields the same admissible sender key in both branches, and undisclosed embedded ciphertexts are already shielded by the output-commitment subhybrids.

Third, (H4) switches only the three nullifiers, at a loss of $3\epsilon_{\text{PRF}}$. The identity traces are *not* switched, since the regulator knows idk_u and Leak_{reg} requires the traces to agree, and no public-key-derivation switch is needed because the sender agrees between admissible branches.

After these hybrids, every compliance-layer field not contained in Leak_{reg} or an authorized disclosure is uniform, simulated, or a deterministic function of public state, and the remaining gap is the base-payment view, which reduces to Zerocash ledger indistinguishability. Summing over $|Q_{\text{ch}}|$ queries gives the claim. $\square$

C.7 Holder identity-hiding

We next isolate what a non-regulator holder can infer from the structural leakage of Definition 4.

COROLLARY 12 (HOLDER IDENTITY-HIDING). *Let $u^\star \neq u^\dagger$ be honest users, let V be a provenance vertex in $\mathcal{A}_{\text{hold}}$'s view, and let $E^\star, E^\dagger$ be the two executions that agree everywhere except that V is contributed by $u^\star$ in the first and by $u^\dagger$ in the second. If $(E^\star, E^\dagger)$ is admissible in the sense of (A1)–(A3), then for every PPT $\mathcal{A}_{\text{hold}}$ there is a PPT $\mathcal{B}$ with*

$$|\Pr[\mathcal{A}_{\text{hold}} \rightarrow u^\star \mid E^\star] - \Pr[\mathcal{A}_{\text{hold}} \rightarrow u^\star \mid E^\dagger]| \leq \text{Adv}_{\mathcal{B}}^{\text{TIND}}(\lambda).$$

PROOF SKETCH. $\mathcal{B}$ relays $\mathcal{A}_{\text{hold}}$'s setup and corruption choices to the TIND challenger and submits $(E^\star, E^\dagger)$ as a challenge pair, admissible by hypothesis. Every other transaction is identical in both branches and forwarded unchanged. $\mathcal{B}$ relays the L_b view and authorized openings, and outputs 0 if $\mathcal{A}_{\text{hold}}$ answers $u^\star$ and 1 otherwise. The simulation is perfect, so $\mathcal{B}$'s advantage equals the attribution bias of $\mathcal{A}_{\text{hold}}$. $\square$

Remark 4 (The hypothesis is not removable). Without (A3), the statement is false, and not for cryptographic reasons. If $u^\star$ and $u^\dagger$ transacted at different blocks or at different lineage depths, the two executions differ in $\text{Leak}_{\text{hold}}$ itself and $\mathcal{A}_{\text{hold}}$ separates them by inspection. Identity-hiding is therefore a claim about emitters that are structurally interchangeable in the adversary's view. It is separately silent about emitters whom the adversary can identify out of band, a situation that lies outside both privacy games and is discussed in Section 8.4 rather than formalized here.