Efficient Transaction Traceability for Auditable Privacy-Preserving Ledgers

Elli Androulaki, Angelo De Caro, Kaoutar Elkhiyaoui, Rebekah Mercer, and Elina van Kempen

IBM Research Europe – Zurich

Abstract. Privacy-preserving distributed ledgers enable transaction processing systems in which users can submit transactions without revealing their identities or transaction details. Regulated and institutional settings impose additional requirements: authorized parties must be able to efficiently trace transactions to their originators without compromising overall system privacy. Existing approaches suffer from important limitations, including restricted parallel transaction formation, high computational overhead, and overly broad auditor access to user secrets.
We present a framework for efficient tracing that eliminates concurrency issues while limiting auditor access. We formalize our security requirements via an ideal functionality and propose a black-box construction based on pseudorandom functions and anonymous credentials, with two concrete instantiations: one using hash-based PRFs and zk-SNARKs, and another using algebraic PRFs and Sigma protocols. Our experimental evaluation demonstrates practicality: per-transaction cost stays within a small factor of a deployed anonymous credential system that offers no tracing at all, while the cost of an audit is independent of the size of the ledger.

1 Introduction

Distributed ledger technologies (DLTs) are increasingly deployed in settings where financial transparency and regulatory compliance must coexist with strong user privacy guarantees. While early permissionless systems such as Bitcoin provide only limited pseudonymity through one-time public keys [26], transaction graphs and account balances remain globally visible and become linkable over time [6]. Ethereum and similar platforms offer comparable pseudonymous models. Even when user identities are not explicitly revealed, transaction metadata and graph structure can enable de-anonymization in practice. Privacy-focused cryptocurrencies such as Zcash aim to provide stronger transaction anonymity by hiding senders, receivers, and amounts within a large anonymity set [5,23]. However, these systems are designed primarily for open, permissionless environments and do not address the requirements of regulated financial systems.

As DLTs are adopted for regulated assets, central bank digital currencies (CBDCs), and institutional settlement systems, the requirements of accountability and non-repudiation become increasingly prominent [22,21]. In these settings,

users are typically enrolled by a registration authority and issued credentials attesting to their eligibility. They must be able to transact privately, without revealing their identities, transaction contents, or activity patterns, while still enabling regulatory oversight under well-defined conditions [1,2,3,29,25]. This gives rise to a fundamental tension: systems must ensure strong anonymity against unauthorized parties, while simultaneously supporting auditing mechanisms that satisfy legal and compliance requirements.

One important oversight mechanism in such systems is transaction tracing. Given a specific user and appropriate authorization, an auditor should be able to identify all transactions associated with that user, without learning information about unrelated users. In other words, a tracing authorization for one user should not expose the activity of others.

Existing approaches to tracing in privacy-preserving ledgers exhibit significant limitations. One common approach embeds encrypted identifiers in each transaction, requiring ledger-wide processing to trace a single user and offering limited support for efficient selective tracing [7,16]. Others rely on sequential counter-based tags, which enable recomputation of user identifiers but impose strict ordering constraints, thereby limiting parallel transaction generation.

Solution Overview. We propose a tracing mechanism tailored to regulated privacy-preserving ledger systems that avoids encrypted user identifiers and sequential counter-based tag derivation. Instead of maintaining an incrementing sequence number, a user derives transaction tags using a pseudorandom function (PRF) evaluated on independently chosen indices from a bounded range $[0,max)$. Then, they prove in zero knowledge that each tag is correctly computed with respect to their secret key and that the chosen index lies within the permitted range.

Transactions are organized into epochs, providing a natural expiration point for auditing permissions. At the start of each epoch, every user derives a tracing key that is used exclusively within that epoch. As a result, tracing permissions tied to a given tracing key are confined to a well-defined time window and are automatically revoked when it elapses.

Concrete Instantiations and Evaluation. We instantiate our framework in two concrete ways suitable for deployment in real-world systems. The first leverages hash-based PRFs and zk-SNARKs, benefiting from fast proof verification at the cost of requiring a trusted setup. The second employs algebraic PRFs over bilinear groups, combined with sigma protocols, yielding less efficient verification but offering the advantage of a transparent setup.

We implement both instantiations and evaluate their performance, measuring proof generation time, verification time, and communication overhead. Our results demonstrate that the proposed tracing mechanisms are practical for high-throughput privacy-preserving transaction systems and readily compatible with modern DLT-based applications. The algebraic instantiation is described in Section 5 and the hash-based instantiation in Appendix E. Both are evaluated in Section 5.2.

Contributions. We summarize our contributions as follows:

- We formalize, using a simulation-based definition, the concept of privacy-preserving ledger systems with tracing capabilities, capturing the requirements for user privacy and authorized traceability.
- We present a black-box construction, based on anonymous credentials, pseudorandom functions, and zero-knowledge proofs, that enables epoch-bound tracing without sequential counters or encrypted identifiers, and prove its security in the static model.
- We describe two concrete instantiations compatible with widely deployed zero-knowledge proof systems.
- We implement and benchmark both constructions, demonstrating their practicality for real-world privacy-preserving ledger deployments.

1.1 Related Work

Prior work on regulatory oversight in privacy-preserving payment systems typically supports either *anonymity revocation* or *tracing*. Anonymity revocation allows an auditor to identify the user responsible for a specific transaction, whereas tracing enables the auditor to recover all transactions associated with a given user. While tracing can be emulated by applying anonymity revocation to each transaction individually, this approach incurs computational cost linear in the number of transactions and exposes information about non-targeted users.

Anonymity revocation. Many systems attach a verifiable encryption or otherwise hidden user identifier to each transaction [14,15,7,16]. An auditor holding the corresponding secret key can selectively recover the sender's identity for a given transaction. However, tracing a user's activity requires scanning and decrypting all transactions on the ledger, incurring overhead linear in the ledger size and unnecessarily exposing unrelated activity. To sum up, these approaches enable identity recovery, but they do not support efficient targeted tracing.

Tracing via sequential tags. Other approaches derive per-transaction tags from a user's tracing secret combined with a monotonically increasing counter, and use zero-knowledge proofs to demonstrate that these tags are correctly formed and bound to the transaction's origin [24,31]. This mechanism ensures that users cannot later repudiate their transactions. Similar techniques can also be applied to trace asset lineage rather than user activity [19]. In a nutshell, given the tracing secret of a target user (or asset), an auditor can identify the corresponding transactions by recomputing the tags and matching them against those recorded on the ledger. However, the reliance on sequential counters introduces inherent limitations: a user's transactions must be generated sequentially, which hinders concurrent transaction creation and degrades performance, particularly in high-throughput settings.

Auditable privacy systems. Other systems provide auditability through interactive protocols or global consistency checks. ZKLedger [27] enables auditors to verify transaction correctness but requires interaction and incurs overhead linear in the number of participants. Privacy-focused cryptocurrencies such as Monero, Zcash, and Quisquis achieve strong anonymity guarantees but do not directly support efficient tracing [28,5,18,11]. Retrofitting tracing into such systems typically requires additional interaction, user cooperation, or linear-time analysis [30]. Platypus and ZKAT [33,2] incorporate regulatory policy enforcement (e.g., balance limits), but do not provide mechanisms for efficient per-user tracing across arbitrary transactions.

Our contribution. We introduce a tracing mechanism that eliminates the need for encrypted user identifiers and sequential counters. Our approach supports efficient tracing while enabling users to generate and submit multiple transactions in parallel without restrictions. Additionally, the design gives users greater control through epoch-bound tracing permissions. These permissions allow auditors to identify all transactions linked to a specific user within a predefined time window. Once the epoch expires, the permissions become invalid, and new ones must be issued.

2 Privacy-preserving Traceable Ledgers

A privacy-preserving and traceable ledger system enables users to submit *confidential* transactions without revealing their identities, while *selectively authorizing auditors* to *efficiently* trace their activity. A transaction is confidential in the sense that its payload is concealed, for e.g., via encryption or hiding commitments. Efficient tracing means that the computational overhead incurred by an auditor is independent of the total number of transactions stored on the ledger. Crucially, tracing permissions must be user-controlled and revocable: in the same way that users grant tracing rights, they must be able to withdraw them, which our construction supports at each epoch boundary. Once an auditor's tracing authorization for a given user is revoked, the auditor must no longer be able to trace that user's future transactions.

An epoch-based privacy-preserving traceable ledger addresses revocation by partitioning time into epochs and allowing users to grant auditors tracing permissions on a per-epoch basis. When an epoch expires, the corresponding authorization automatically lapses, thereby ensuring that previously authorized auditors cannot trace a user's transactions in subsequent epochs. In this paper, we focus on epoch-based privacy-preserving traceable ledgers.

2.1 System Participants

A privacy-preserving traceable ledger system involves the following participants:

- **Ledger:** A publicly accessible, append-only data structure that stores all submitted transactions. Its sole role is to ensure data availability and immutability for the purposes of transaction validation. Namely, we assume that there are transaction validators that on input of the ledger decide which transactions are valid and mark them as such.
- **Registration Authority (RA):** An honest-but-curious authority responsible for user registration. The RA authenticates users and issues the identity credentials required to submit transactions and enable selective traceability.
- **Users:** Potentially malicious entities that generate and submit confidential transactions to the ledger. Users remain anonymous with respect to the ledger, while retaining fine-grained control over which auditors are authorized to trace their transactions and during which epochs.
- **Auditors:** Potentially malicious entities that may be granted permissions to trace the transactions in the ledger of specific users in specific epochs.

2.2 Security Requirements

The goal of any coalition involving the honest-but-curious registration authority, malicious users, and malicious auditors is twofold: (1) enable unauthorized auditors to de-anonymize the transactions of honest users; and (2) allow malicious users to conceal transactions from honest auditors who are properly authorized. To counter such attacks, a privacy-preserving traceable ledger must satisfy the following security properties:

- **Anonymity.** Only auditors holding valid tracing permissions for specific user and epoch can de-anonymize that user's transactions during that epoch.
- **Non-repudiation.** All transactions submitted by a user within a given epoch can be correctly traced back to that user by any auditor possessing valid tracing permissions for that user and epoch.
- **Permission Expiration.** Tracing permissions are automatically revoked at the end of each epoch. As a result, auditors do not retain tracing capabilities beyond the epochs for which they were explicitly authorized.

2.3 Security formalization

We formalize the security of privacy-preserving traceable ledgers using the ideal-real world paradigm.

Ideal World. We define ideal functionality $\mathcal{F}$ that is secure by construction (cf. Figure 1): it receives the inputs of the participants through interfaces and produces the corresponding correct outputs. The interfaces of $\mathcal{F}$ are as follows:

- Register. User **U** invokes this interface to register. If the RA approves and **U** has not previously invoked Register, then $\mathcal{F}$ marks **U** as registered.
- Submit. Registered user **U** invokes Submit to submit up to max transactions during the current epoch. Each successful invocation results in $\mathcal{F}$ recording the transaction along with the identity of **U** and the current epoch.

$\mathcal{F}$ maintains three data structures $\mathbb{R}$, $\mathbb{T}$ and $\mathbb{P}$ to record user activity. $\mathbb{R}$ tracks user registration status, $\mathbb{T}$ user transactions, and $\mathbb{P}$ tracks user permissions given to auditors.
We consider a static corruption model, where simulator $\mathcal{S}$ corrupts users and auditors of its choice before any call to $\mathcal{F}$ is made. Let $\mathbb{C}$ denote the set of corrupt participants.

Register. On input Register from user **U**, send (Register, **U**) to **RA** and wait for confirmation. Upon confirmation, do:

1. If $\mathbb{R}[\mathbf{U}] \neq \perp$, abort. *A user can register at most once.*
2. Send (Register, **U**) to $\mathcal{S}$ and wait for the go-ahead. On receiving the go-ahead, continue.
3. $\mathbb{R}[\mathbf{U}] \leftarrow$ true.
4. Return (RegisterEnd) to **U** and $\mathcal{S}$.

Submit. On input $(\mathsf{Submit}, \mathsf{tx}, e)$ from **U**, do:

1. If $\mathbb{R}[\mathbf{U}] = \perp$, stop. *Only registered users can submit transactions that will be accepted.*
2. current $\leftarrow \mathcal{C}$.
3. If $e \neq$ current, stop.
4. If $|\mathbb{T}[(\mathbf{U}, \mathsf{current})]| + 1 > max$, stop. *Users can submit at most max transactions per epoch.*
5. If $\mathbf{U} \in \mathbb{C}$, then origin $\leftarrow \mathbf{U}$; else origin $\leftarrow \varnothing$.
6. Send (Submit, tx, e, origin) to $\mathcal{S}$, and wait for the go-ahead. *If the user is corrupt, then $\mathcal{S}$ recognizes the origin of the transaction.*
7. $\mathbb{T}[(\mathbf{U}, \mathsf{current})] \leftarrow \mathbb{T}[(\mathbf{U}, \mathsf{current})] \cup \{\mathsf{tx}\}$.
8. Return SubmitEnd to **U** and $\mathcal{S}$.

Grant. On input (Grant, **A**, e) from **U**, send (Grant, **U**, e) to auditor **A** and wait for the confirmation. Upon confirmation, do:

1. If $\mathbb{R}[\mathbf{U}] = \perp$, stop.
2. Send (Grant, **A**, **U**, e) to $\mathcal{S}$ and wait for the go-ahead. Upon receipt of the go-ahead, do:
   (a) $\mathbb{P}[(\mathbf{A}, \mathbf{U}, e)] \leftarrow$ true.
   (b) Return GrantEnd to **U**, **A**, and $\mathcal{S}$.

Trace. On input (Trace, **U**, e) from auditor **A**, do:

1. If $\mathbb{P}[(\mathbf{A}, \mathbf{U}, e)] \neq$ true, stop. *Only authorized auditors are able to trace the transactions of the users.*
2. Else $T \leftarrow \mathbb{T}[(\mathbf{U}, e)]$. *The auditor traces all the transactions submitted by the user in the epoch (non-repudiation).*
3. If $\mathbf{A} \in \mathbb{C}$, then $T^* = T$; else $T^* = \varnothing$.
4. Return (TraceEnd, T) to **A** and (TraceEnd, T^*)to $\mathcal{S}$. *If the auditor is corrupt, $\mathcal{S}$ is also able to trace the user's transactions.*

Fig. 1. Ideal functionality $\mathcal{F}$ for epoch-based privacy-preserving traceable ledgers.

- Grant. An honest user **U** invokes Grant to authorize an auditor **A** to trace **U**'s transactions for a specific epoch e. $\mathcal{F}$, accordingly, records that **A** is authorized to trace **U**'s transaction for epoch e. Once e elapses, **A** loses their authorization.
- Trace. Auditor **A** who has been granted permission for some user **U** and epoch e invokes Trace to obtain the list of **U**'s transactions during epoch e.

To determine the current epoch, $\mathcal{F}$ is additionally parameterized by a clock $\mathcal{C}$, which, when invoked, returns the current epoch.

Real World. We consider a concrete protocol Π. We say that Π realizes $\mathcal{F}$ if there exists a simulator $\mathcal{S}$ such that, for any adversary $\mathcal{A}$, the view generated by $\mathcal{S}$ while interacting only with $\mathcal{F}$ is computationally indistinguishable from the view of $\mathcal{A}$ in a real execution of Π.

3 Building Blocks

Commitments A commitment scheme is defined by the following algorithms.

(1) $\mathsf{ComKeygen}(1^\kappa) \to CK$: on input security parameter κ, ComKeygen returns a public commitment key CK.
(2) $\mathsf{Commit}(CK, m) \to (C, \rho)$: on input commitment key CK and message m, Commit outputs a commitment C and an (optional) secret opening hint ρ. When the randomness is fixed explicitly we write $C = \mathsf{Commit}(CK, m, \rho)$.
(3) $\mathsf{Open}(CK, m, \rho, C) \to 0/1$: on input commitment key CK, message m, hint ρ, and commitment C, Open outputs 1 if C is a commitment to m under ρ, and 0 otherwise.

A commitment scheme is *secure* if it satisfies the properties of correctness and binding. A commitment scheme can additionally be hiding.

Pseudorandom Functions (PRF) A family of *deterministic* functions $\mathbf{F}(k, \cdot) : \{0,1\}^l \to \{0,1\}^\kappa$, indexed by seed $k \in \{0,1\}^\kappa$, is pseudorandom if it is **(1) efficiently computable:** given $(k, x) \in \{0,1\}^\kappa \times \{0,1\}^l$, $\mathbf{F}(k, x)$ can be computed in polynomial time; and **(2) pseudorandom:** on input x provided by a distinguisher, it is computationally infeasible for any probabilistic polynomial time (PPT) distinguisher to tell the output of $\mathbf{F}(k, x)$ apart from that of a uniformly random function, even when given access to a polynomial-number of outputs $y_i = \mathbf{F}(k, x_i)$ for inputs x_i of their choosing.

Digital Signatures (DS) A digital signature scheme involves algorithms:

(1) $\mathsf{DSKeygen}(1^\kappa) \to (SK, PK)$: on input security parameter κ, DSKeygen outputs a pair of secret key SK and public key PK.
(2) $\mathsf{DSSign}(SK, m) \to \sigma$: on input of secret key SK and message $m \in \{0,1\}^*$,DSSign outputs signature σ.

(3) $\mathsf{DSVerify}(PK, m, \sigma) \rightarrow 0/1$: on input public key PK, message m, and signature σ, $\mathsf{DSVerify}$ outputs 1 to indicate that the signature is valid; otherwise, it outputs 0.

A DS scheme is **existentially unforgeable** if for any PPT adversary that is given access to signatures σ_i : $\mathsf{DSVerify}(PK, m_i, \sigma_i) \rightarrow 1$ for some public key PK and messages m_i of their choosing, they cannot output a pair $(m, \sigma) : m \notin m_i \wedge \mathsf{DSVerify}(PK, m, \sigma) \rightarrow 1$.

Non-interactive Arguments of Knowledge Let $\mathcal{R}$ be a binary relation. We say that pair $(\mathbb{x}, \mathbb{w})$ satisfies $\mathcal{R}$ iff $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, and call $\mathbb{x}$ the instance, and $\mathbb{w}$ the witness.

A non-interactive argument of knowledge for $\mathcal{R}$ comprises three algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. $\mathcal{G}(\kappa, \mathcal{R}) \rightarrow (\tau, \mathbb{c})$ takes security parameter κ and a description of $\mathcal{R}$ as input, and outputs public parameters $\mathbb{c}$ and a (possibly empty) trapdoor τ. On input $(\mathbb{c}, \mathbb{x}, \mathbb{w})$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, PPT prover $\mathcal{P}$ outputs a proof π. Verifier $\mathcal{V}(\mathbb{c}, \mathbb{x}, \pi) \rightarrow \{0, 1\}$ outputs 1 to signal that π is valid relative to $(\mathbb{c}, \mathbb{x})$ and we say that π is accepting, and 0 otherwise.

An argument of knowledge is **complete** if for all pairs $(\mathbb{x}, \mathbb{w})$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, $\mathcal{V}(\mathbb{c}, \mathbb{x}, \mathcal{P}(\mathbb{c}, \mathbb{x}, \mathbb{w})) = 1$ except with a negligible probability. It is **knowledge-sound** if it is computationally infeasible for a malicious prover to produce an accepting proof π for $(\mathbb{c}, \mathbb{x})$ without knowledge of a witness $\mathbb{w}$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$; more formally, there exists an extractor which, given trapdoor τ, is able to recover $\mathbb{w}$ from any accepting π. It is **zero-knowledge** if there exists a simulator which, given $(\tau, \mathbb{c}, \mathbb{x})$, is able to produce a proof that is computationally indistinguishable from one produced by an honest prover $\mathcal{P}$.

Signatures of Knowledge (SoKs). A signature of knowledge (SoK) allows a party to sign a message while proving that they hold witness $\mathbb{w}$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$. A SoK is defined by the following algorithms:

(1) $\mathsf{SoKGen}(\mathcal{R}) \rightarrow \mathbb{c}$: on input relation $\mathcal{R}$, SoKGen outputs public parameters $\mathbb{c}$.
(2) $\mathsf{SoKSign}(\mathbb{c}, \mathbb{x}, \mathbb{w}, m) \rightarrow \sigma$: on input public parameters $\mathbb{c}$, instance $\mathbb{x}$, witness $\mathbb{w}$ such that $\mathcal{R}(\mathbb{w}, \mathbb{x}) = 1$, and message m, $\mathsf{SoKSign}$ outputs a SoK σ.
(3) $\mathsf{SoKVerify}(\mathbb{c}, \mathbb{x}, m, \sigma) \rightarrow \{0, 1\}$: on input public parameters $\mathbb{c}$, instance $\mathbb{x}$, message m, and signature σ, $\mathsf{SoKVerify}$ outputs 1 if it deems σ a valid SoK on m relative to pair $(\mathbb{c}, \mathbb{x})$; otherwise, it outputs 0.

To formalize the security properties, three additional algorithms are introduced.

- $\mathsf{SimGen}(\mathcal{R}) \rightarrow (\hat{\mathbb{c}}, \hat{\tau})$: on input a relation $\mathcal{R}$, output simulated public parameters $\hat{\mathbb{c}}$ and trapdoor $\hat{\tau}$.
- $\mathsf{SimSign}(\hat{\mathbb{c}}, \mathbb{x}, \hat{\tau}, m) \rightarrow \hat{\sigma}$: on input simulated public parameters $\hat{\mathbb{c}}$, instance $\mathbb{x}$, trapdoor $\hat{\tau}$, and message m, output a simulated SoK $\hat{\sigma}$.
- $\mathsf{Extract}(\hat{\mathbb{c}}, \mathbb{x}, \hat{\tau}, \hat{\sigma}, m) \rightarrow \hat{\mathbb{w}}$: on input simulated public parameters $\hat{\mathbb{c}}$, instance $\mathbb{x}$, trapdoor $\hat{\tau}$, simulated SoK $\hat{\sigma}$, and message m, $\mathsf{Extract}$ outputs witness $\hat{\mathbb{w}}$.

We say that $\mathsf{SoKGen}, \mathsf{SoKSign}, \mathsf{SoKVerify}$ is a SoK for relation $\mathcal{R}$ if it satisfies the following properties.

Correctness. $\forall\ (\mathbb{w}, \mathbb{x})\ :\ \mathcal{R}(\mathbb{w}, \mathbb{x}) = 1$ and $\forall m$:

$$\Pr\left[\mathsf{SoKVerify}(\mathbb{c}, \mathbb{x}, \pi) = 1 \;\middle|\; \begin{array}{l} \mathbb{c} \xleftarrow{\$} \mathsf{SoKGen}(\mathcal{R}), \\ \pi \xleftarrow{\$} \mathsf{SoKSign}(\mathbb{c}, \mathbb{x}, \mathbb{w}, m) \end{array}\right] = 1 - \varepsilon(\kappa).$$

Simulatability. Let $\mathcal{O}$ be an oracle that receives triple $(\mathbb{x}, \mathbb{w}, m)$, verifies whether $\mathcal{R}(\mathbb{w}, \mathbb{x}) = 1$, and if so responds with $\hat{\sigma} \leftarrow \mathsf{SimSign}(\hat{\mathbb{c}}, \mathbb{x}, \hat{\tau}, m)$. We say that simulatability is achieved, if and only if for all PPT adversaries $\mathcal{A}$ and auxiliary inputs ζ:

$$\begin{aligned} \Big|\Pr\Big[b = 1 \;\Big|\; (\hat{\mathbb{c}}, \hat{\tau}) \leftarrow \mathsf{SimGen}(\mathcal{R}),\ b \leftarrow \mathcal{A}^{\mathcal{O}(\hat{\mathbb{c}}, \tau, \cdot, \cdot, \cdot)}(\zeta, \hat{\mathbb{c}})\Big] \\ - \Pr\Big[b = 1 \;\Big|\; \mathbb{c} \leftarrow \mathsf{SoKGen}(\mathcal{R}),\ b \xleftarrow{\$} \mathcal{A}^{\mathsf{SoKSign}(\mathbb{c}, \cdot, \cdot, \cdot)}(\zeta, \mathbb{c})\Big]\Big| \leq \varepsilon(\kappa). \end{aligned}$$

Simulation extractability. Let Q denote the set of queries $(\mathbb{x}, \mathbb{w}, m)$ submitted by an adversary $\mathcal{A}$ and successfully answered by oracle $\mathcal{O}$. We say that simulation extractability is met, if and only if for all PPT adversaries $\mathcal{A}$ and all auxiliary inputs ζ:

$$\Pr\left[\begin{array}{c} \mathcal{R}(\mathbb{x}, \hat{\mathbb{w}}) = 1 \\ \vee\ (\mathbb{x}, \star, m) \in Q \\ \vee\ \mathsf{SoKVerify}(\hat{\mathbb{c}}, \mathbb{x}, \hat{\sigma}) = 0 \end{array} \;\middle|\; \begin{array}{l} (\hat{\mathbb{c}}, \hat{\tau}) \leftarrow \mathsf{SimGen}(\mathcal{R}), \\ (\mathbb{x}, m, \hat{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}(\hat{\mathbb{c}}, \hat{\tau}, \cdot, \cdot, \cdot)}(\zeta, \hat{\mathbb{c}}), \\ \hat{\mathbb{w}} \leftarrow \mathsf{Extract}(\hat{\mathbb{c}}, \mathbb{x}, \hat{\tau}, \hat{\sigma}, m) \end{array}\right] = 1 - \varepsilon(\kappa).$$

4 Black-box Construction

Let $\mathcal{L}$ denote the ledger that processes user transactions. Let $\mathbf{G}$ and $\mathbf{G}'$ be two one-way functions, and let $\mathbf{F}_{\mathsf{E}}$ and $\mathbf{F}_{\mathsf{T}}$ be two PRF families: $\mathbf{F}_{\mathsf{E}}$ is keyed by a tracing key tk and maps an epoch identifier to an epoch tracing key ek, and $\mathbf{F}_{\mathsf{T}}$ is keyed by an epoch tracing key ek and maps a counter in $[0, max)$ to a transaction tag. $\mathbf{F}_{\mathsf{T}}$ is additionally parameterized by the public tracing key $T = \mathbf{G}'(tk)$, which the registration authority certifies at registration; these are public parameters and not key material. We require pseudorandomness of both families to hold against a distinguisher that holds T, so nothing that T carries need be kept secret. Let max be the upper-bound of the number of transactions a user can submit during any given epoch e. Epochs are defined with respect to the ledger's block height. Specifically, an epoch e is the interval spanning from block $\lambda \times e$ to block $\lambda \times (e+1) - 1$, where λ denotes the epoch duration in terms of number of blocks.

At a high level, the protocol operates as follows. Each user holds exactly two long-term secrets: a signing key sk and a tracing key tk. The two are cryptographically bound to each other by a single credential issued by the registration authority (**RA**) at registration time. No further long-term key material is certified. For each epoch e, the user derives one epoch tracing key $ek \leftarrow \mathbf{F}_{\mathsf{E}}(tk, e)$ from their tracing key and the epoch identifier. We treat $\mathbf{F}_{\mathsf{E}}$ and $\mathbf{F}_{\mathsf{T}}$ abstractly here, so their key, parameter and output spaces are fixed by the instantiation. In

Section 5.1, for example, tk is a pair of field elements and T carries a vector of max group elements alongside the image of tk, while ek is a single group element. All transactions submitted during epoch e are tagged using a PRF evaluated under ek and an index in the range $[0, max)$. Users accompany each transaction with zero-knowledge (ZK) proofs and SoKs attesting to the correctness of the tag generation, which can be efficiently verified by transaction validators. For audit purposes, the epoch tracing key ek is disclosed to the appropriate auditors. This enables auditors to locally compute all valid transaction tags corresponding to a given user and epoch. Finally, when the system transitions to a new epoch, users derive a fresh epoch tracing key. Tracing capabilities granted for prior epochs automatically expire.

Without loss of generality, we assume that system participants establish secure channels to communicate with each other.

Setup. During setup, **RA** calls $(SK, PK) \leftarrow \mathsf{DSKeygen}(1^\kappa)$ and communicates PK to the ledger operator. The ledger operator calls $CK \leftarrow \mathsf{ComKeygen}(1^\kappa)$ and produces four sets of public parameters:

- $\mathbb{c}_0 \leftarrow \mathsf{SoKGen}(\mathcal{R}_0)$ for relation $\mathcal{R}_0$ defined for instance and witness:

$$\mathbb{x}_0 = (\mathbf{G}, \mathbf{G}', S, T), \quad \mathbb{w}_0 = (sk, tk), \tag{1}$$

  such that $\mathcal{R}_0(\mathbb{x}_0, \mathbb{w}_0) = 1$ if and only if:

$$S = \mathbf{G}(sk) \quad \wedge \quad T = \mathbf{G}'(tk). \tag{2}$$

- $(\tau_1, \mathbb{c}_1) \leftarrow \mathcal{G}(\mathcal{R}_1)$ for relation $\mathcal{R}_1$ defined for instance and witness:

$$\mathbb{x}_1 = (\mathbf{F}_\mathsf{E}, \mathbf{G}', T, e, ek), \quad \mathbb{w}_1 = tk, \tag{3}$$

  such that $\mathcal{R}_1(\mathbb{x}_1, \mathbb{w}_1) = 1$ if and only if:

$$T = \mathbf{G}'(tk) \quad \wedge \quad ek = \mathbf{F}_\mathsf{E}(tk, e). \tag{4}$$

- $\mathbb{c}_2 \leftarrow \mathsf{SoKGen}(\mathcal{R}_2)$ for relation $\mathcal{R}_2$ defined for instance and witness:

$$\mathbb{x}_2 = (\mathbf{G}, \mathbf{G}', PK, CK, C), \quad \mathbb{w}_2 = (\sigma, sk, tk, \rho), \tag{5}$$

  such that $\mathcal{R}_2(\mathbb{x}_2, \mathbb{w}_2) = 1$ if and only if:

$$\mathsf{DSVerify}(PK, \mathbf{G}(sk) \| \mathbf{G}'(tk), \sigma) \to 1 \ \wedge \ \mathsf{Open}(CK, tk, \rho, C) \to 1. \tag{6}$$

- $(\tau_3, \mathbb{c}_3) \leftarrow \mathcal{G}(\mathcal{R}_3)$ for relation $\mathcal{R}_3$ defined for instance and witness:

$$\mathbb{x}_3 = (\mathbf{F}_\mathsf{E}, \mathbf{F}_\mathsf{T}, CK, C, e, max, \mathsf{tag}), \quad \mathbb{w}_3 = (tk, \rho, ek, \gamma), \tag{7}$$

  such that $\mathcal{R}_3(\mathbb{x}_3, \mathbb{w}_3) = 1$ if and only if:

$$\begin{aligned} &\mathsf{Open}(CK, tk, \rho, C) \to 1 \quad \wedge \quad ek = \mathbf{F}_\mathsf{E}(tk, e) \\ \wedge \quad &\mathsf{tag} = \mathbf{F}_\mathsf{T}(\mathbf{G}'(tk), ek, \gamma) \quad \wedge \quad 0 \leq \gamma < max. \end{aligned} \tag{8}$$

The ledger operator concludes the setup by recording in the ledger entry:

$$\mathcal{L}[\mathbb{c}] \leftarrow (\mathbf{G}, \mathbf{G}', \mathbf{F}_{\mathsf{E}}, \mathbf{F}_{\mathsf{T}}, PK, CK, \mathbb{c}_0, \mathbb{c}_1, \mathbb{c}_2, \mathbb{c}_3, max).$$

These parameters reflect the SoK that will be used during initial user registration, the argument that will be used to prove the correctness of an epoch tracing key to an auditor, the SoK each user employs to authorize transactions, and the argument that will be leveraged to prove the correctness of transaction tags.

Register. The registration authority **RA** first checks if user **U** has never registered before. Then, **U** randomly selects a signing key sk and a tracing key tk, computes $S = \mathbf{G}(sk)$ and $T = \mathbf{G}'(tk)$, and then engages in an interactive protocol with registration authority **RA**. During this interaction, **U** receives a nonce $\mathfrak{n}$, computes and sends back $\psi \leftarrow \mathsf{SoKSign}(\mathbb{c}_0, \mathbb{x}_0, \mathbb{w}_0, \mathfrak{n})$, for $(\mathbb{x}_0, \mathbb{w}_0)$ as defined in Equations 1 and 2. If **RA** accepts ψ, then it calls $\mathsf{DSSign}(SK, S \| T)$, records that **U** registered with pair (S, T), and sends the resulting signature σ to **U**. From this point onward, we assume that **U** uses $\mathbf{G}(sk)$ as their public key in the system, in particular, vis-a-vis auditors.

Grant. Before the start of some epoch e, auditors issue requests to get tracing authorization for user **U**. In response, **U** computes the epoch's tracing key $ek = \mathbf{F}_{\mathsf{E}}(tk, e)$, and generates a zero-knowledge proof $\pi \leftarrow \mathcal{P}(\mathbb{c}_1, \mathbb{x}_1, \mathbb{w}_1)$, where $\mathbb{x}_1$ and $\mathbb{w}_1$ are defined in Equations 3 and 4. This can be reused for all authorization requests for epoch e. An authorization request identifies the auditor **A** and contains pair $(e, \mathfrak{n})$ whereby $\mathfrak{n}$ is a random nonce. **U** grants **A** the tracing authorization by sending tuple $(S, T, \sigma, ek, \pi, \psi)$, where $\psi \leftarrow \mathsf{SoKSign}(\mathbb{c}_0, \mathbb{x}_0, \mathbb{w}_0, \pi \| \mathfrak{n})$ and $(\mathbb{x}_0, \mathbb{w}_0)$ are defined as in Equations 1 and 2. If $1 \leftarrow \mathsf{DSVerify}(PK, S \| T, \sigma)$, and π and ψ are valid, then **A** accepts tracing key ek.

Submit. Let e denote the current epoch and ek the corresponding epoch tracing key. **U** first prepares the *confidential* payload of their transaction denoted $\mathfrak{p}$, selects a fresh counter $0 \leq \gamma < max$, and computes $\mathsf{tag}_\gamma = \mathbf{F}_{\mathsf{T}}(T, ek, \gamma)$. Next, **U** computes commitment $(C, \rho) \leftarrow \mathsf{Commit}(CK, tk)$ and generates $\pi \leftarrow \mathcal{P}(\mathbb{c}_3, \mathbb{x}_3, \mathbb{w}_3)$, for $(\mathbb{x}_3, \mathbb{w}_3)$ are defined as in Equations 7 and 8. **U** produces then $\psi \leftarrow \mathsf{SoKSign}(\mathbb{c}_2, \mathbb{x}_2, \mathbb{w}_2, m)$, where $m = \mathfrak{p} \| \mathsf{tag}_\gamma \| C \| \pi$ and $(\mathbb{x}_2, \mathbb{w}_2)$ are defined as in Equations 5 and 6. Finally, **U** submits transaction $\mathsf{tx} = (\mathfrak{p}, \mathsf{tag}_\gamma, C, \pi, \psi)$ to the ledger for validation.

Once the transaction is appended to the ledger, the transaction validators check if **(1)** tag_γ has never accompanied a previous transaction that was marked valid, **(2)** π is a valid ZK proof for relation $\mathcal{R}_3$ and current epoch e, ψ is a valid SoK on message $m = \mathfrak{p} \| \mathsf{tag}_\gamma \| C \| \pi$ relative to relation $\mathcal{R}_2$. If all the checks succeed, the transaction validators mark the transaction as valid by adding entry $\mathcal{L}[\mathsf{tag}_\gamma] \leftarrow \mathfrak{p}$.

Trace. Given T and key ek for epoch e, auditor **A** traces the transactions that **U** submitted during epoch e as follows. $\forall\ 0 \leq \gamma < max$: **A** computes $\mathsf{tag}_\gamma = \mathbf{F}_{\mathsf{T}}(T, ek, \gamma)$ and retrieves entry $\mathcal{L}[\mathsf{tag}_\gamma]$ if not empty.

Increment Epoch. When the ledger reaches block height $(\mathsf{current}+1)\times\lambda$, the system participants move to epoch $\mathsf{current}+1$.

Theorem 1. *Assuming existential unforgeability of* DS*, pseudo-randomness of* $\mathbf{F}_{\mathsf{E}}$ *and* $\mathbf{F}_{\mathsf{T}}$*, one-wayness of* $\mathbf{G}$ *and* $\mathbf{G}'$*, and security of ZK arguments of knowledge and* SoK*, the protocol described above securely realizes* $\mathcal{F}$*.*

The proof of this theorem is given in Appendix F.

5 Instantiation

We describe a concrete instantiation of the black-box construction in pairing-based groups using algebraic PRFs and tailored Sigma protocols, yielding a transparent setup. An alternative construction that uses hash-based PRFs and zk-SNARKs, with faster verification but a trusted setup, is given in Appendix E.

5.1 Algebraic PRF-based Instantiation

System Setup. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of large prime order p admitting an efficient bilinear pairing $e:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$. We require a Type-3 pairing, with no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, as the analysis assumes DDH in $\mathbb{G}_1$ and in $\mathbb{G}_T$. Let P_0, P_1, P_2, P_3, Q be random generators of $\mathbb{G}_1$. Let H and D be two independent random generators of $\mathbb{G}_2$.

The user holds the two long-term keys sk and tk of Section 4, and derives one epoch key ek per epoch. The tag PRF used below is an algebraic PRF whose evaluation needs both a group element and an exponent. The tracing key is therefore a pair of field elements, $tk=(tk',tk'')\in\mathbb{Z}_p^2$. Here tk' is the component the epoch PRF $\mathbf{F}_{\mathsf{E}}$ raises to, and tk'' is the component the tag PRF $\mathbf{F}_{\mathsf{T}}$ inverts against. Both components are drawn once at registration and certified together in the same credential, and tk'' is unchanged across epochs.

Let $max=2^\ell$ bound the number of transactions a user may submit per epoch, and let $\mathcal{H}_{\mathsf{E}}:\{0,1\}^*\to\mathbb{G}_1$ be a cryptographic hash function. One-way function $\mathbf{G}$ is instantiated as $\mathbf{G}(sk)=P_1^{sk}$. The public tracing key carries, alongside the image of tk, the vector $\mathbb{W}$ that a tag evaluation inverts against:

$$T=\mathbf{G}'(tk)=\big(P_2^{tk'}\cdot P_3^{tk''},\ \mathbb{W}\big),\qquad \mathbb{W}=\big(W_\gamma\big)_{\gamma\in[0,max)},\qquad W_\gamma=D^{1/(tk''+\gamma)}.$$

We write $T=P_2^{tk'}\cdot P_3^{tk''}$ for the first component. The two PRFs are then defined as follows, with $\mathbb{W}$ entering $\mathbf{F}_{\mathsf{T}}$ as a public parameter rather than as key material:

$$ek=\mathbf{F}_{\mathsf{E}}(tk,e)=\mathcal{H}_{\mathsf{E}}(e)^{tk'}\in\mathbb{G}_1,\qquad \mathbf{F}_{\mathsf{T}}(T,ek,\gamma)=e\big(ek,\ W_\gamma\big)\in\mathbb{G}_T.$$

This is the split the construction of Section 4 asks for. $\mathbf{F}_{\mathsf{E}}$ is keyed by tk', has range $\mathbb{G}_1$, and is a pseudorandom function under DDH in $\mathbb{G}_1$ with $\mathcal{H}_{\mathsf{E}}$ modeled as a random oracle (Appendix D.1); every component of its output varies with the epoch. $\mathbf{F}_{\mathsf{T}}$ is keyed by ek, which is exactly the range of $\mathbf{F}_{\mathsf{E}}$, and is a pseudorandom

function on $[0, max)$ under the q-mDBDHI assumption, for a distinguisher that holds $\mathbb{W}$ (Appendix D.2). Placing $\mathbb{W}$ in T is what keeps $\mathbf{F}_{\mathsf{E}}$ pseudorandom: were the vector part of the epoch key, $\mathbf{F}_{\mathsf{E}}$ would repeat that component in every epoch and could not be a PRF at all. It costs nothing to do so, as T is already certified by the registration authority and already carried by the Grant message.

An auditor authorized for epoch e therefore receives the single group element ek, and evaluates tags with the $\mathbb{W}$ that came with T. $\mathbb{W}$ does not vary with the epoch, so it is sent only the first time a given auditor is authorized; we treat it as known to the adversary throughout the analysis.

Commit is instantiated using a Pedersen commitment with key $CK = (P_2, P_3, Q)$, giving $(P_2^{tk'} \cdot P_3^{tk''} \cdot Q^{\rho}, \rho) \leftarrow \mathsf{Commit}(CK, tk)$.

The RA signature scheme is instantiated using BBS+ [9]. The RA holds BBS+ keys $(SK, PK) \leftarrow \mathsf{BBSKeyGen}(1^{\kappa}, 3)$ to sign the pair (sk, tk), which is carried as the triple of field elements $(sk, tk', tk'') \in \mathbb{Z}_p^3$, where $SK = z \in \mathbb{Z}_p$ and $PK = (P_0, P_1, P_2, P_3, Q, H, Z) \in \mathbb{G}_1^5 \times \mathbb{G}_2^2$ with $Z = H^z$ (see Appendix B for more details).

User Registration. The user must obtain a credential binding sk and tk without revealing them to the RA in the clear. This is achieved via signing over committed inputs. To do this, the user sends $(S, T) \in \mathbb{G}_1 \times \mathbb{G}_1$ and produces a SoK that attests that $\exists(sk, tk', tk'') : S = P_1^{sk} \wedge T = P_2^{tk'} \cdot P_3^{tk''}$. The RA verifies the resulting transcript and issues a BBS+ signature σ. Concretely, $\mathcal{R}_0$ is defined for $\mathbb{x}_0 = (P_1, P_2, P_3, S, T)$, $\mathbb{w}_0 = (sk, tk', tk'')$, $\mathcal{R}_0(\mathbb{x}_0, \mathbb{w}_0) = 1$ **iff** $S = P_1^{sk} \wedge T = P_2^{tk'} \cdot P_3^{tk''}$, and $\mathbb{c}_0 \leftarrow \mathsf{SoKGen}(\mathcal{R}_0)$ is set to hash function $\mathcal{H}_p : \{0,1\}^* \rightarrow \mathbb{Z}_p$. On receiving a nonce $\mathfrak{n}$ from the RA, user $\mathbf{U}$ samples $sk, tk', tk'' \xleftarrow{\$} \mathbb{Z}_p$, and sends $(S = P_1^{sk}, T = P_2^{tk'} \cdot P_3^{tk''}, \psi)$ where $\psi \leftarrow \mathsf{SoKSign}(\mathbb{c}_0, \mathbb{x}_0, \mathbb{w}_0, \mathfrak{n})$ corresponds to $(A, B, \eta, \upsilon_1, \upsilon_2)$ with $(A, B) = (P_1^{\alpha}, P_2^{\beta_1} \cdot P_3^{\beta_2})$, $\eta = \alpha + c \cdot sk$, $\upsilon_1 = \beta_1 + c \cdot tk'$, and $\upsilon_2 = \beta_2 + c \cdot tk''$, where $c = \mathcal{H}_p(\mathbb{x}_0, A, B, \mathfrak{n})$. Given ψ, the RA computes $c = \mathcal{H}_p(\mathbb{x}_0, A, B, \mathfrak{n})$, and checks if $A \cdot S^c = P_1^{\eta} \wedge B \cdot T^c = P_2^{\upsilon_1} \cdot P_3^{\upsilon_2}$. If so, it returns $\sigma = (e, r, E = (P_0 \cdot S \cdot T \cdot Q^r)^{1/(z+e)})$; otherwise, it rejects. $S = P_1^{sk}$ serves as the user's public key.

Epoch Key Computation. $\mathbf{U}$ computes the epoch key $ek = \mathbf{F}_{\mathsf{E}}(tk, e) = \mathcal{H}_{\mathsf{E}}(e)^{tk'}$ and sends it to the auditor with T. Recall that the exponent component tk'' of the tracing key is never disclosed. What the auditor receives instead is the vector $\mathbb{W}$ carried by T, which is what a tag evaluation needs in order to keep tk'' secret (Appendix D.2). In the random oracle model, $\mathbf{F}_{\mathsf{E}}$ is a PRF under the decisional Diffie-Hellman (DDH) assumption, as shown in Appendix D.1; this is why $\mathbb{W}$ belongs to T and not to ek.

The elements $\{W_\gamma\}$ do not depend on e. As part of T they are therefore sent once per auditor, at that auditor's first grant, and reused for every later epoch. Every subsequent grant to the same auditor is constant-size. The epoch key ek is a single element of $\mathbb{G}_1$, and its value in one epoch is independent of its value in any other.

Relation $\mathcal{R}_1$ is defined for **(1)** $\mathbb{x}_1 = (\mathcal{H}_{\mathsf{E}}, P_2, P_3, D, T, e, ek, \{W_\gamma\}_{\gamma\in[0,max)})$, **(2)** $\mathbb{w}_1 = (tk', tk'')$, and **(3)** $(\varnothing, \mathbb{c}_1) \leftarrow \mathcal{G}(\mathcal{R}_1)$ with $\mathbb{c}_1$ set to hash function $\mathcal{H}_p$. It proves both components of T well-formed against one tracing key, alongside the epoch key. We then have $\mathcal{R}_1(\mathbb{x}_1, \mathbb{w}_1) = 1$ **if and only if** the following holds:

$$T = P_2^{tk'} \cdot P_3^{tk''} \quad \wedge \quad ek = \mathcal{H}_{\mathsf{E}}(e)^{tk'}$$
$$\wedge \quad \forall\, \gamma \in [0, max) :\ W_\gamma^{tk''} = D \cdot W_\gamma^{-\gamma}.$$

The last equality is simply $W_\gamma = D^{1/(tk''+\gamma)}$ rearranged, so that it becomes an equality of discrete logarithms against the committed tk''. As tk'' is hidden, the auditor does not recompute $\mathbb{W}$ itself; we prove the opening of T and the well-formedness of $\mathbb{W}$ against the same tk''.

We prove the *max* equalities in batch. Both parties derive $t_\gamma = \mathcal{H}_p(\{W_\gamma\}, \gamma)$ and set the following:

$$\widehat{W} = \prod_{\gamma=0}^{max-1} W_\gamma^{t_\gamma}, \qquad \widehat{V} = \prod_{\gamma=0}^{max-1} \left(D \cdot W_\gamma^{-\gamma}\right)^{t_\gamma}.$$

The prover then shows $\widehat{W}^{tk''} = \widehat{V}$ in place of the *max* separate equalities. By the small-exponents argument [4], if any one element is malformed then the batched equality holds with probability at most $q_{\mathcal{H}_p}/p$ over the random oracle, where $q_{\mathcal{H}_p}$ bounds the prover's hash queries.

On receiving nonce $\mathfrak{n}$ from auditor **A**, **U** samples $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ and computes proof $\pi = (A, B, K, \eta, \zeta)$ as follows:

$$A = P_2^{\alpha} \cdot P_3^{\beta} \in \mathbb{G}_1, \qquad B = \mathcal{H}_{\mathsf{E}}(e)^{\alpha} \in \mathbb{G}_1, \qquad K = \widehat{W}^{\beta} \in \mathbb{G}_2,$$
$$c = \mathcal{H}_p(\mathbb{x}_1, A, B, K, \mathfrak{n}), \qquad \eta = \alpha + c \cdot tk', \qquad \zeta = \beta + c \cdot tk''.$$

U also produces SoK $\psi \leftarrow \mathsf{SoKSign}(\mathbb{c}_0, \mathbb{x}_0, \mathbb{w}_0, \pi \| \mathfrak{n})$ for relation $\mathcal{R}_0$, and responds with $(S, T, \sigma, ek, \pi, \psi)$. Auditor **A** parses σ as (e, r, E) and recomputes $\{t_\gamma\}$, $\widehat{W}$, $\widehat{V}$ and c. It accepts if ψ is valid, $e(P_0 \cdot S \cdot T \cdot Q^r, H) = e(E, Z \cdot H^e)$, and the following hold:

$$P_2^{\eta} \cdot P_3^{\zeta} = A \cdot T^c, \qquad \mathcal{H}_{\mathsf{E}}(e)^{\eta} = B \cdot ek^c, \qquad \widehat{W}^{\zeta} = K \cdot \widehat{V}^c.$$

Since $\widehat{W}$ and $\widehat{V}$ depend only on $\mathbb{W}$, the third equation need only be checked the first time a given auditor is authorized. Subsequent grants to that auditor drop K and $\mathbb{W}$, and leave the two $\mathbb{G}_1$ equations against the same certified T; formally these are two relations, $\mathcal{R}_1$ and its restriction $\mathcal{R}_1'$ to the first two conjuncts, with $\mathbb{x}_1$ shortened accordingly. The binding of T ties the two proofs to one pair (tk', tk'').

Verifiable Tag Computation. **U** first computes the Pedersen commitment $C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^{\rho}$ and the tag $\mathsf{tag}_\gamma = \mathbf{F}_{\mathsf{T}}(T, ek, \gamma) = e(ek, W_\gamma)$, where the W_γ are the group elements carried by T.

Security of $\mathbf{F}_{\mathsf{T}}$ rests on the q-mDBDHI assumption, as shown in Appendix D.2. Note that tk'' is a hidden witness of $\mathcal{R}_3$ below, so the auditor never needs it in order to check a transaction.

Relation $\mathcal{R}_3$ is defined by **(1)** $\mathbb{x}_3 = (\mathcal{H}_{\mathsf{E}}, D, P_2, P_3, Q, C, max, e, \mathsf{tag}_\gamma)$, **(2)** $\mathbb{w}_3 = (tk', tk'', \rho, ek, \gamma)$, and **(3)** $(\varnothing, \mathbb{c}_3) \leftarrow \mathcal{G}(\mathcal{R}_3)$ with $\mathbb{c}_3$ describing a hash function $\mathcal{H}_p : \{0,1\}^* \to \mathbb{Z}_p$. We then have $\mathcal{R}_3(\mathbb{x}_3, \mathbb{w}_3) = 1$ **if and only if** the following holds:

$$C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^\rho \wedge ek = \mathcal{H}_{\mathsf{E}}(e)^{tk'} \wedge \mathsf{tag}_\gamma = e\Big(ek,\, D^{1/(tk''+\gamma)}\Big) \wedge 0 \le \gamma < max.$$

Proving $\mathcal{R}_3$ directly would require an argument over a secret group element (ek) paired against a secret-exponent term, which would be expensive in practice. Instead the user randomly samples $s \leftarrow \mathbb{Z}_p$, and with it computes the following auxiliary values:

$$U = \mathcal{H}_{\mathsf{E}}(e)^s, \qquad \mathit{\Pi} = e\big(ek^s,\, D\big), \qquad X = \mathsf{tag}_\gamma^s.$$

The randomized epoch key ek^s is itself a witness and is never published; only its image $\mathit{\Pi} \in \mathbb{G}_T$ under the pairing with D is. This matters, and we return to it after the protocol. The user also samples $r_\gamma \leftarrow \mathbb{Z}_p$ and forms a Pedersen commitment $C_\gamma = P_3^\gamma \cdot Q^{r_\gamma}$ to the transaction index. This commitment is what carries γ between the Sigma protocol and the range argument below, so that the two speak about the same value.

We know that $\mathit{\Pi} = e(ek, D)^s$, and if tag_γ is correctly formed then $\mathsf{tag}_\gamma^{tk''+\gamma} = e(ek, D)$. Putting the two together gives the following:

$$\mathit{\Pi} = X^{\,tk''+\gamma}.$$

The user then proves the equivalent relation $\widehat{\mathcal{R}}_3$, defined for **(1)** $\widehat{\mathbb{x}}_3 = (\mathbb{x}_3, U, \mathit{\Pi}, X, C_\gamma)$, **(2)** $\widehat{\mathbb{w}}_3 = (\mathbb{w}_3, s, r_\gamma)$, and **(3)** $\widehat{\mathbb{c}}_3 = \mathbb{c}_3$, which together verify the following:

$$\begin{gathered} C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^\rho \quad \wedge \quad U = \mathcal{H}_{\mathsf{E}}(e)^s \quad \wedge \quad \mathit{\Pi} = e(U, D)^{tk'} \\ \wedge \quad X = \mathsf{tag}_\gamma^s \quad \wedge \quad \mathit{\Pi} = X^{tk''} \cdot X^\gamma \\ \wedge \quad C_\gamma = P_3^\gamma \cdot Q^{r_\gamma} \quad \wedge \quad 0 \le \gamma < max. \end{gathered}$$

We prove this in two parts. The first is a Sigma protocol proving the following equalities for witness $(tk', tk'', \rho, \gamma, s, r_\gamma)$:

$$\begin{gathered} C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^\rho \quad \wedge \quad U = \mathcal{H}_{\mathsf{E}}(e)^s \quad \wedge \quad \mathit{\Pi} = e(U, D)^{tk'} \\ \wedge \quad X = \mathsf{tag}_\gamma^s \quad \wedge \quad \mathit{\Pi} = X^{tk''} \cdot X^\gamma \\ \wedge \quad C_\gamma = P_3^\gamma \cdot Q^{r_\gamma}. \end{gathered}$$

U samples $\alpha, \beta, \kappa, \nu, \mu, \xi \xleftarrow{\$} \mathbb{Z}_p$, and computes and sends the following:

$$\begin{gathered} A = P_2^\alpha \cdot P_3^\beta \cdot Q^\kappa, \quad B = \mathcal{H}_{\mathsf{E}}(e)^\nu, \quad M = e(U, D)^\alpha, \\ K = \mathsf{tag}_\gamma^\nu, \quad F = X^\beta \cdot X^\mu, \quad A_\gamma = P_3^\mu \cdot Q^\xi. \end{gathered}$$

With challenge $c = \mathcal{H}_p(\widehat{\mathbb{x}}_3, A, B, M, K, F, A_\gamma)$, the user computes and sends the following values:

$$\theta_1 = \alpha + c \cdot tk', \quad \theta_2 = \beta + c \cdot tk'', \quad \theta_h = \kappa + c \cdot \rho,$$
$$\theta_s = \nu + c \cdot s, \quad \theta_\gamma = \mu + c \cdot \gamma, \quad \theta_r = \xi + c \cdot r_\gamma.$$

The verifier computes challenge c, and rejects unless $U \neq 1_{\mathbb{G}_1}$ and $\Pi \neq 1_{\mathbb{G}_T}$. It then accepts if and only if the following hold:

$$P_2^{\theta_1} \cdot P_3^{\theta_2} \cdot Q^{\theta_h} = A \cdot C^c, \quad \mathcal{H}_{\mathsf{E}}(e)^{\theta_s} = B \cdot U^c, \quad e(U, D)^{\theta_1} = M \cdot \Pi^c,$$
$$\mathsf{tag}_\gamma^{\theta_s} = K \cdot X^c, \quad X^{\theta_2} \cdot X^{\theta_\gamma} = F \cdot \Pi^c,$$
$$P_3^{\theta_\gamma} \cdot Q^{\theta_r} = A_\gamma \cdot C_\gamma^c.$$

The second part proves $0 \leq \gamma < max$. This is done following the standard range proof techniques [12], run on the commitment C_γ that the Sigma protocol above already opens. Since θ_γ is the response for the value committed in C_γ, this shows both that γ is in the required range, and consistency of γ in each relation. Equivalence of $\mathcal{R}_3$ and $\widehat{\mathcal{R}}_3$, and zero knowledge of the whole argument, are proven in Appendix H.

Transaction Authorization. Each transaction carries a signature of knowledge that attests it has originated from a registered user, following relation $\mathcal{R}_2$ defined in Equations 5 and 6. Since user **U** is registered using BBS+ signature $\sigma = (e, r, E = (P_0 \cdot P_1^{sk} \cdot P_2^{tk'} \cdot P_3^{tk''} \cdot Q^r)^{1/(z+e)})$, $\mathcal{R}_2$ is defined for $\mathbb{x}_2 = (PK, C) = (P_0, P_1, P_2, P_3, Q, H, Z, C)$, $\mathbb{w}_2 = (sk, tk', tk'', \rho, e, r, E)$, $\mathcal{R}_2(\mathbb{x}_2, \mathbb{w}_2) = 1$ **if and only if** the following holds:

$$C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^\rho \ \wedge \ e(E, H^e \cdot Z) = e(P_0 \cdot P_1^{sk} \cdot P_2^{tk'} \cdot P_3^{tk''} \cdot Q^r, H).$$

$\mathbb{c}_2 \leftarrow \mathsf{SoKGen}(\mathcal{R}_2)$ is set to hash function $\mathcal{H}_p : \{0,1\}^* \to \mathbb{Z}_p$.

Concretely, the SoK for $\mathcal{R}_2$ on message m can be formed by transforming $\mathcal{R}_2$ into an equivalent relation $\widehat{\mathcal{R}}_2$ that does not require the generation of proofs over bilinear equations, following the technique of Camenisch, Drijvers, and Lehmann [13] for proving knowledge of a BBS+ signature, specialized to $n = 3$ messages and $\mathcal{D} = \varnothing$ disclosed (full derivation in Appendix C). **U** samples fresh $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and sets $E' = E^{r_1}$, $r_3 = 1/r_1$, $d = (P_0 \cdot P_1^{sk} \cdot P_2^{tk'} \cdot P_3^{tk''} \cdot Q^r)^{r_1} \cdot Q^{-r_2}$, $\bar{E} = E'^{-e} \cdot d \cdot Q^{r_2}$ (so $\bar{E} = E'^z$), and $r'' = r - r_2 r_3 - \rho$.

$\widehat{\mathcal{R}}_2$ is then given as follows:

$$\widehat{\mathbb{x}}_2 = (PK, C, E', \bar{E}, d), \qquad \widehat{\mathbb{w}}_2 = (sk, tk', tk'', \rho, e, r_2, r_3, r'')$$
$$C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^\rho \ \wedge \ e(E', Z) = e(\bar{E}, H) \ \wedge \ \bar{E}/d = E'^{-e} \cdot Q^{r_2}$$
$$\wedge \ (P_0 \cdot C)^{-1} = P_1^{sk} \cdot d^{-r_3} \cdot Q^{r''}.$$

$\widehat{\mathcal{R}}_2$ is proven via a standard Sigma protocol. **U** samples $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \lambda_1, \lambda_2, \lambda_3 \xleftarrow{\$} \mathbb{Z}_p$ and computes:

$$A = P_2^{\alpha_1} \cdot P_3^{\alpha_2} \cdot Q^{\alpha_3}, \qquad B = E'^{-\beta_1} \cdot Q^{\beta_2}, \qquad K = P_1^{\lambda_1} \cdot d^{-\lambda_2} \cdot Q^{\lambda_3},$$

with challenge $c = \mathcal{H}_p(\widehat{\mathbb{x}}_2, A, B, K, m)$ and responses

$$\phi_1 = \alpha_1 + c \cdot tk', \quad \phi_2 = \alpha_2 + c \cdot tk'', \quad \phi_3 = \alpha_3 + c \cdot \rho, \quad \phi_4 = \beta_1 + c \cdot e,$$
$$\phi_5 = \beta_2 + c \cdot r_2, \quad \phi_6 = \lambda_1 + c \cdot sk, \quad \phi_7 = \lambda_2 + c \cdot r_3, \quad \phi_8 = \lambda_3 + c \cdot r''.$$

The user outputs SoK $\psi = (E', \bar{E}, d, A, B, K, \boldsymbol{\phi})$. A verifier checks $e(E', Z) = e(\bar{E}, H)$, recomputes c, and accepts iff:

$$P_2^{\phi_1} \cdot P_3^{\phi_2} \cdot Q^{\phi_3} = A \cdot C^c, \qquad E'^{-\phi_4} \cdot Q^{\phi_5} = B \cdot (\bar{E}/d)^c,$$
$$P_1^{\phi_6} \cdot d^{-\phi_7} \cdot Q^{\phi_8} = K \cdot \left((P_0 \cdot C)^{-1}\right)^c.$$

Security Analysis

Theorem 2. *The instantiation securely realizes $\mathcal{F}$, in the random oracle model, under the Decisional Diffie-Hellman assumption in $\mathbb{G}_1$ (for $\mathbf{F}_{\mathsf{E}}$), the q-mDBDHI assumption (for $\mathbf{F}_{\mathsf{T}}$), the Decisional Diffie-Hellman assumption in $\mathbb{G}_T$ (for the zero knowledge of the tag argument), the discrete logarithm assumption in $\mathbb{G}_1$ (for the range argument), and the q-Strong Diffie-Hellman assumption (for BBS+).*

The proof is given in Appendix G.

5.2 Experimental Evaluation

Experimental Setup All experiments were conducted on an Apple M2 laptop with 16GB RAM, running four performance and four efficiency cores. Measurements were collected using Go's built-in benchmarking framework, averaged over the number of iterations taken to reach 1 second of computation, and summarised across ten repetitions with `benchstat`. All reported figures carry a variation of at most 4% across repetitions. Every operation is single-threaded except where noted. As a scale for the figures that follow, on this machine one exponentiation in $\mathbb{G}_1$ costs 62.6us over BN254 and 113.8us over BLS12-381, and one pairing evaluation costs 0.956ms and 1.141ms respectively.

Scope of the measurements The prototype implements all four relations $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, in both instantiations, and every reported figure is a complete cost for the operation named. In the SNARK instantiation the range check on the transaction index is compiled into the $\mathcal{R}_3$ circuit as a bit decomposition, and is therefore already inside the $\mathcal{R}_3$ rows of Table 1. In the algebraic instantiation it is a separate Bulletproofs argument over C_γ, reported on its own row of Table 2; the per-transaction cost of $\mathcal{R}_3$ is the sum of the Sigma protocol and that argument; the total rows in Table 2 are measured as a single end-to-end benchmark rather than summed, and so differ slightly from the sum of the two component rows. The SNARK implementation uses the `gnark` backend, which includes optimized multi-pairing routines and assembly-level curve arithmetic.

Table 1. Performance of the SNARK-based PRF instantiation. Times in ms.

Operation	Frequency	BN254	BLS12-381
$\mathcal{R}_1$ setup	once per deployment	75.2	146.2
$\mathcal{R}_1$ proof	per epoch	10.2	16.6
$\mathcal{R}_1$ verification	per epoch	1.67	2.90
$\mathcal{R}_1$ proof size (bytes)	per epoch	128	192
$\mathcal{R}_1$ constraints	–	661	667
$\mathcal{R}_1$ prover RAM (MB)	per epoch	14	18
$\mathcal{R}_1$ verifier RAM (MB)	per epoch	18	26
$\mathcal{R}_3$ setup	once per deployment	161.8	256.0
$\mathcal{R}_3$ proof	per transaction	20.7	32.3
$\mathcal{R}_3$ verification	per transaction	2.36	3.03
$\mathcal{R}_3$ proof size (bytes)	per transaction	128	192
$\mathcal{R}_3$ constraints	–	1330	1342
$\mathcal{R}_3$ prover RAM (MB)	per transaction	22	18
$\mathcal{R}_3$ verifier RAM (MB)	per transaction	22	26

Hash-based instantiation. For the hash-based instantiation, correctness of the PRF is proven using the Groth16 zk-SNARK system [20]. Each proof consists of three group elements. $\mathcal{R}_1$ proves the derivation of the epoch key and $\mathcal{R}_3$ additionally proves tag correctness and that the transaction index lies in range, so the two are compiled and benchmarked as separate circuits. Verification requires 3 pairing evaluations and 1 multi-exponentiation over the public inputs. Benchmark results are summarized in Table 1.

The $\mathcal{R}_3$ circuit is two MiMC permutations and an ℓ-bit decomposition of the transaction index, which is what separates its constraint count from that of $\mathcal{R}_1$. At $\ell = 8$ the range check is eight constraints against the 661 and 667 a single permutation costs over BN254 and BLS12-381, so it is free relative to the hashing and the SNARK timings do not move measurably with ℓ.

Algebraic instantiation. We next benchmark the algebraic instantiation, in which the constraints are enforced via pairing-based Sigma protocols.

For the proof of knowledge of the openings of S and T, $\mathcal{R}_0$, the prover costs are 1 exponentiation and 1 two-base multi-exponentiation in $\mathbb{G}_1$, and 3 scalar responses. The verifier costs are 3 exponentiations and 1 two-base multi-exponentiation in $\mathbb{G}_1$. The same proof is produced at registration, on the registration authority's nonce, and at each grant, on $\pi \| n$. As σ is sent in the clear at a grant, the auditor additionally verifies it directly, at the cost of 2 pairing evaluations.

For the proof of knowledge of a BBS+ signature, $\mathcal{R}_2$, the prover costs are 3 scalar multiplications to randomize the signature, 3 multi-exponentiations in $\mathbb{G}_1$ (of sizes three, two and three), and 8 scalar responses. The verifier costs are 2 pairing evaluations and 3 multi-exponentiations in $\mathbb{G}_1$.

For the epoch key proof, $\mathcal{R}_1$, the per-epoch part costs the prover 2 exponentiations in $\mathbb{G}_1$ and 2 scalar responses, and the verifier 4 exponentiations in $\mathbb{G}_1$. A user's *first* grant additionally carries the group elements $\{W_\gamma\}_{\gamma\in[0,max)}$ and the batched proof that they open to the certified tk'': max exponentiations in $\mathbb{G}_2$ to form them, and two max-term multi-exponentiations in $\mathbb{G}_2$ for $\widehat{W}$ and $\widehat{V}$ on each side. This is $max \cdot |\mathbb{G}_2|$ bytes of communication, 24KB at $\ell = 8$ over BLS12-381. The vector is a public parameter carrying no secret, and does not vary with the epoch, so how it reaches an auditor is a deployment choice: it may be sent at that auditor's first grant, or published once alongside T and fetched by any auditor. Either way it is formed and proven once per user, and every grant after the first is constant-size.

For the transaction tag proof, $\mathcal{R}_3$, the Sigma protocol costs the prover 7 exponentiations in $\mathbb{G}_1$, 5 exponentiations in $\mathbb{G}_T$ and 1 pairing evaluation, and the verifier 9, 5 and 1 of the same. Opening the commitment C_γ to the range argument accounts for 2 of those $\mathbb{G}_1$ exponentiations on each side, and one $\mathbb{G}_1$ element of the proof. Only one pairing is needed because $\mathit{\Pi} = e(ek^s, D) = e(U, D)^{tk'}$, so the randomized epoch key is never formed and the prover exponentiates $e(U, D)$ instead of pairing a second time. Exponentiation in $\mathbb{G}_T$ is the dominant cost here, at 0.83ms over BN254 and 1.00ms over BLS12-381, close to a pairing; the five of them account for roughly three quarters of the figures in Table 2.

The range argument is a Bulletproof over C_γ, sharing the bases P_3 and Q with that commitment so that the two arguments speak about one γ. Its inner product argument halves its vectors each round, so it runs over a vector of length ℓ directly whenever ℓ is a power of two, which is why we take $\ell = 8$. For other ℓ the vectors must be padded to $2^{\lceil \log_2 \ell \rceil}$, and the cost is then that of the padded length rather than of ℓ; $\ell = 10$ and $\ell = 16$ cost the same. The proof is $2\log_2 \ell + 4$ elements of $\mathbb{G}_1$ and 5 scalars, which is 10 group elements and 5 scalars at $\ell = 8$, or 640 bytes over BLS12-381.

The prover's dominant group operations are the folding of the two generator vectors, $2(\ell - 1)$ exponentiations in $\mathbb{G}_1$, together with a further $\ell - 1$ to form the shifted generators and multi-exponentiations of total length $O(\ell)$ across the $\log_2 \ell$ rounds. The verifier folds the shift into its exponents and checks the argument as a single multi-exponentiation of $2\ell + 2\log_2 \ell + 4$ terms, which is 26 terms at $\ell = 8$.

There is no proof needed from the auditor itself. The auditor recomputes $\mathsf{tag}_\gamma = \mathbf{F}_\mathsf{T}(T, ek, \gamma) = e(ek, W_\gamma)$ for every $\gamma \in [0, max)$ and reads the ledger at each resulting tag, for a total of max pairing evaluations and max lookups. These evaluations are mutually independent, so the work parallelises directly; we report both a single-threaded figure and one across all eight cores.

We run the benchmarks with $\ell = 8$, so a user may submit up to $max = 256$ transactions in any one epoch. A user reaching that cap waits for the next epoch boundary; since an epoch spans λ blocks and λ is a deployment parameter, the cap is set by choosing λ against the throughput a single user is expected to need. The choice of ℓ trades this cap against three costs that all grow with max: the size of $\{W_\gamma\}$, the work an auditor does to trace one epoch, and the range

Table 2. Microbenchmarks for the algebraic PRF instantiation, $\ell = 8$. Times in ms.

Operation	Frequency	BN254	BLS12-381
$\mathcal{R}_0$ proof	registration, per grant	0.26	0.42
$\mathcal{R}_0$ verification	registration, per grant	0.37	0.64
$\mathcal{R}_1$ proof	per epoch	0.29	0.55
$\mathcal{R}_1$ verification	per epoch	0.43	0.79
$\mathcal{R}_2$ proof	per transaction	1.46	2.39
$\mathcal{R}_2$ verification	per transaction	2.25	3.07
$\mathcal{R}_3$ Sigma proof	per transaction	5.56	6.93
$\mathcal{R}_3$ Sigma verification	per transaction	5.44	7.01
Range argument proof	per transaction	5.13	8.11
Range argument verification	per transaction	0.80	1.22
$\mathcal{R}_3$ total, proof	per transaction	10.59	15.07
$\mathcal{R}_3$ total, verification	per transaction	6.33	8.32
$\{W_\gamma\}$ generation	once per user	61.81	96.27
$\{W_\gamma\}$ batched proof	once per user	6.53	10.53
$\{W_\gamma\}$ batched verify	once per user	6.90	11.19
Trace, single-threaded	per audit	265.1	313.6
Trace, eight cores	per audit	47.95	57.89

argument. Table 2 reports benchmark results, grouped by how often each cost is incurred.

Per-epoch and per-transaction costs are not additive, and we report them separately rather than as a single total. The cost a validator pays for each transaction is the sum of the $\mathcal{R}_2$ row and the $\mathcal{R}_3$ total row.

Comparison with an anonymous credential system. To place these figures in context we benchmark Hyperledger Idemix on the same machine and the same curve. Idemix is a deployed anonymous credential system; its Aries variant uses BBS+ over BLS12-381. We measure it in two configurations: a standard signature, which offers no oversight mechanism at all, and one carrying pseudonymous commitments to the enrollment id and revocation handle, which is Idemix's auditability feature. A fresh pseudonym is derived per signature in both cases, as unlinkability requires. Results are given in Table 3, which reports the full per-transaction cost of each system.

Both of our rows are the sum of $\mathcal{R}_2$ and $\mathcal{R}_3$, and both include the range check on the transaction index. The SNARK row combines the Groth16 figures of Table 1 with the measured $\mathcal{R}_2$ Sigma protocol; the hash-based instantiation certifies (sk, tk) rather than (sk, tk', tk''), so its $\mathcal{R}_2$ is marginally cheaper than the figure used here.

Neither instantiation is uniformly cheaper than Idemix, and neither needs to be: Idemix offers no efficient tracing at all, and the configuration it does offer answers a different and weaker question. The SNARK instantiation verifies

Table 3. Per-transaction cost against Hyperledger Idemix, over BLS12-381.

System	Prove (ms)	Verify (ms)
Idemix, no oversight mechanism	7.74	5.60
Idemix, with audit pseudonyms	9.11	7.22
This work, algebraic	17.5	11.4
This work, SNARK	34.7	6.10

faster than Idemix does *with* its audit pseudonyms enabled, at the cost of a substantially slower prover. The algebraic instantiation proves in about 1.9 times and verifies in about 1.6 times Idemix's cost with audit pseudonyms enabled, and requires no trusted setup anywhere. Since a transaction is generated once and verified by every validator, verification cost is what governs throughput, and it is on verification that both instantiations remain within a small factor of a deployed credential system that provides no tracing.

The cost of tracing. Idemix's audit mechanism answers a different question from ours: given a candidate enrollment id and a signature, it decides whether the two match. Idemix has no notion of an epoch, so recovering the transactions a single user submitted over some period requires one such test per ledger entry in that period. We measure that test at 0.42ms. Tracing a user over one epoch in our scheme is instead a fixed *max* pairing evaluations, measured at 0.31s, and does not depend on how many transactions the ledger holds. Both figures are single-threaded, and both scans parallelise in the same way, as every test and every pairing is independent of the others. The two approaches cost the same at roughly 750 ledger entries per epoch. At 10^6 entries, the Idemix approach requires around 7 minutes against our 0.31s, and the ratio between the two grows linearly with the number of entries. This is the asymptotic separation of Section 1 made concrete.

Overall, our results show that both instantiations achieve practical performance. The SNARK-based construction provides fast verification and compact proofs, while the algebraic construction achieves competitive verification without requiring trusted setup. These results support the feasibility of deploying our scheme in real-world systems.

6 Conclusion

We introduced a construction for efficient tracing in DLT-based transaction systems, enabling authorized auditors to identify all transactions associated with a target user within a given epoch without compromising the anonymity of others. Unlike prior work, auditor complexity does not grow with the ledger size, and users may generate parallel transactions without risking validation failures. We presented two instantiations: a SNARK-based construction offering compact proofs and verification in about six milliseconds at the cost of a trusted

setup, and an algebraic construction verifying in about eleven milliseconds with a transparent setup. In both, an audit costs a fixed number of pairing evaluations rather than a scan: tracing a user over one epoch takes 0.31 seconds however many transactions the ledger holds, against roughly seven minutes for an anonymity-revocation approach at a million entries. These results demonstrate that regulatory compliance can be integrated into privacy-preserving transaction systems without sacrificing efficiency or usability.

References

1. Sarah Allen, Srdjan Capkun, Ittay Eyal, Giulia Fanti, Bryan A Ford, James Grimmelmann, Ari Juels, Kari Kostiainen, Sarah Meiklejohn, Andrew Miller, et al. Design choices for central bank digital currency: Policy and technical considerations. Technical report, National Bureau of Economic Research, 2020.
2. Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, page 255–267, New York, NY, USA, 2020. Association for Computing Machinery.
3. Bank for International Settlements. Central bank digital currencies: System design and privacy/regulatory trade-offs. BIS Papers OTHP 88, Bank for International Settlements, 2022.
4. Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250, Berlin, Heidelberg, 1998. Springer.
5. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014.
6. Ferenc Béres, István A Seres, András A Benczúr, and Mikerah Quintyne-Collins. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *2021 IEEE international conference on decentralized applications and infrastructures (DAPPS)*, pages 69–78, 2021.
7. Dmytro Bogatov, Angelo De Caro, Kaoutar Elkhiyaoui, and Björn Tackmann. Anonymous transactions with revocation and auditing in hyperledger fabric. In *Cryptology and Network Security: 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings*, page 435–459, Berlin, Heidelberg, 2021. Springer-Verlag.
8. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, Heidelberg, 2004. Springer.
9. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 41–55, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
10. Nicholas Brandt, Dennis Hofheinz, Julia Kastner, and Akin Ünal. The price of verifiability: Lower bounds for verifiable random functions. In Eike Kiltz and Vinod

Vaikuntanathan, editors, *Theory of Cryptography*, pages 747–776, Cham, 2022. Springer Nature Switzerland.
11. Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020 , Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, page 423–443, Berlin, Heidelberg, 2020. Springer-Verlag.
12. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018.
13. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In Michael Franz and Panos Papadimitratos, editors, *Trust and Trustworthy Computing - 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings*, volume 9824 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2016.
14. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2006.
15. Jan Camenisch and Anja Lehmann. Privacy-preserving user-auditable pseudonym systems. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 269–284. IEEE, 2017.
16. Aisling Connolly, Jérôme Deschamps, Pascal Lafourcade, and Octavio Perez-Kempner. Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, volume 13774 of *Lecture Notes in Computer Science*, pages 249–271. Springer, 2022.
17. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431, Berlin, Heidelberg, 2005. Springer.
18. Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, 2019.
19. Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In Jens Grossklags and Bart Preneel, editors, *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2016.
20. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *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*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

21. BIS Innovation Hub. Project mbridge: Connecting economies through cbdc. *https://www.bis.org/publ/othp59.htm*, 2022.
22. Etienne Jinze. First look: China's central bank digital currency. *https://research.binance.com/analysis/china-cbdc*, 2019.
23. George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 463–477, 2018.
24. Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1739–1752, 2022.
25. Kieran Murphy, Sun Tao, Yong Sarah Zhou, Natsuki Tsuda, Victor Budau Nicolas Zhang, Frankosiligi Solomon, Kathleen Kao, Morana Vucinic, and Kristina Miggiani. Central bank digital currency data use and privacy protection. Imf fintech note 2024/004, international monetary fund, washington, dc, International Monetary Fund, 2024.
26. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008.
27. Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In Sujata Banerjee and Srinivasan Seshan, editors, *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, pages 65–80. USENIX Association, 2018.
28. Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptol. ePrint Arch.*, page 1098, 2015.
29. OECD. Central bank digital currencies (CBDCs) and democratic values. Technical report, OECD Business and Finance Policy Papers, OECD Publishing, Paris, 2023.
30. George Papadoulis, Danai Balla, Panagiotis Grontas, and Aris Pagourtzis. AQQUA: augmenting quisquis with auditability. In Marc Fischlin and Veelasha Moonsamy, editors, *Applied Cryptography and Network Security - 23rd International Conference, ACNS 2025, Munich, Germany, June 23-26, 2025, Proceedings, Part II*, volume 15826 of *Lecture Notes in Computer Science*, pages 283–312. Springer, 2025.
31. Amirreza Sarencheh, Aggelos Kiayias, and Markulf Kohlweiss. Parscoin: A privacy-preserving, auditable, and regulation-friendly stablecoin. In Markulf Kohlweiss, Roberto Di Pietro, and Alastair R. Beresford, editors, *Cryptology and Network Security - 23rd International Conference, CANS 2024, Cambridge, UK, September 24-27, 2024, Proceedings, Part I*, volume 14905 of *Lecture Notes in Computer Science*, pages 289–313. Springer, 2024.
32. Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious prf, with applications. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 674–705, Cham, 2022. Springer International Publishing.
33. Karl Wüst, Kari Kostiainen, Noah Delius, and Srdjan Capkun. Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 2947–2960. ACM, 2022.

A Assumptions

Decisional Diffie-Hellman (DDH). The DDH assumption is said to hold in $\mathbb{G}_1$ if no PPT adversary can distinguish (P^x, P^y, P^{xy}) from (P^x, P^y, P^z) for uniformly random $x, y, z \in \mathbb{Z}_p$.

q-Strong Diffie-Hellman (q-SDH). The q-SDH assumption is said to hold if no PPT adversary, given $(P, P^x, P^{x^2}, \ldots, P^{x^q}, H, H^x)$ for a uniformly random $x \in \mathbb{Z}_p$, can output a pair $(c, P^{1/(x+c)})$ for any $c \in \mathbb{Z}_p \setminus \{-x\}$ with non-negligible probability.

q-Decisional Bilinear Diffie-Hellman Inversion with published elements (q-mDBDHI). Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a Type-3 bilinear group of prime order p with generators $P \in \mathbb{G}_1$ and $D \in \mathbb{G}_2$. The q-mDBDHI assumption is said to hold if no PPT adversary, given $(P, D, \{D^{1/(x+\gamma)}\}_{\gamma \in [0,q)})$ for uniformly random $x \in \mathbb{Z}_p$ and $A \in \mathbb{G}_1$, can distinguish $\{e(A, D^{1/(x+\gamma)})\}_{\gamma \in [0,q)}$ from $\{\Delta_\gamma\}_{\gamma \in [0,q)}$ for random $\Delta_\gamma \in \mathbb{G}_T$ with non-negligible advantage. Neither A nor x is given to the adversary. We use it with $q = max$.

This is the decisional variant of the q-BDHI assumption [8]. In our variant, the base elements are shared and the $\mathbb{G}_1$ element is withheld.

Similar q-type assumptions and variants are the norm for constructions of this type. A similar partially oblivious PRF based on the Dodis-Yampolskiy PRF with a secret group element is analyzed under a new strong Diffie-Hellman inversion assumption of its own [32]. Their construction differs from ours as they wish to avoid pairings. For the general case, it has been shown that pairing based constructions of this type cannot have both short proofs and a reduction to a simple assumption [10]. That work also references the Dodis-Yampolskiy construction specifically.

B BBS+ Signatures

The BBS+ signature scheme [9] allows a party to sign a vector of messages of a predefined length n. The scheme is defined by three algorithms as follows:

- $\mathsf{BBSKeyGen}(1^\kappa, n) \to (SK, PK)$: On input security parameter κ and size n of the vectors of messages to be signed, $\mathsf{BBSKeyGen}$ first samples generators $P_0, P_1, \ldots, P_n, Q \xleftarrow{\$} \mathbb{G}_1$ and $H \xleftarrow{\$} \mathbb{G}_2$, followed by sampling $z \xleftarrow{\$} \mathbb{Z}_p$. Next, $\mathsf{BBSKeyGen}$ computes $Z = H^z$, and outputs secret key $SK = z$ and public key $PK = (P_0, P_1, \ldots P_n, Q, H, Z)$.
- $\mathsf{BBSSign}(SK, \boldsymbol{m}) \to \sigma$: On input secret key $SK = z$ and vector of n messages $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, $\mathsf{BBSSign}$ randomly samples $e, r \xleftarrow{\$} \mathbb{Z}_p$ such that $e + z \neq 0$, computes $E = (P_0 \cdot Q^r \prod_{i=1}^n P_i^{m_i})^{\frac{1}{z+e}}$, and finally, outputs signature $\sigma = (E, e, r)$.

– $\mathsf{BBSVerify}(PK, \boldsymbol{m}, \sigma) \to 0/1$: On input public key PK, vector of n messages $\boldsymbol{m}$, and signature $\sigma = (E, e, r)$, $\mathsf{BBSVerify}$ outputs 1 if $e(E, H^e Z) = e(P_0 \cdot Q^r \prod_{i=1}^{n} P_i^{m_i}, H)$, indicating, hence, that the signature is valid. Otherwise, $\mathsf{BBSVerify}$ outputs 0, signifying that the signature as invalid.

BBS+ signatures are existentially unforgeable under the q-Strong Diffie-Hellman (q-SDH) assumption.

C Proof of Knowledge of a BBS+ Signature

The proof of knowledge of a BBS+ signature that $\widehat{\mathcal{R}}_2$ is built from is given as follows [13]. We keep the notation of Appendix B throughout.

Let $P_0, P_1, \ldots, P_n, Q$ be generators of $\mathbb{G}_1$, and let H and $Z = H^z$ be generators of $\mathbb{G}_2$ for issuer secret key $z \in \mathbb{Z}_p$. A BBS+ signature on a vector of messages $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$ is the triple $\sigma = (E, e, r)$, where E is defined as follows:

$$E = \left(P_0 \cdot Q^r \prod_{i=1}^{n} P_i^{m_i}\right)^{1/(z+e)}.$$

We write b for the base of this expression, as follows:

$$b = P_0 \cdot Q^r \prod_{i=1}^{n} P_i^{m_i}.$$

Let $\mathcal{D} \subseteq [1, n]$ be the set of disclosed message indices, and let $\bar{\mathcal{D}} = [1, n] \setminus \mathcal{D}$ be the undisclosed ones. The prover holds σ together with the full message vector, and proceeds as follows.

The prover samples $r_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and $r_2 \xleftarrow{\$} \mathbb{Z}_p$, and constructs E' and d as follows:

$$E' \leftarrow E^{r_1}, \qquad r_3 \leftarrow 1/r_1, \qquad d \leftarrow b^{r_1} \cdot Q^{-r_2}, \qquad r' \leftarrow r - r_2 \cdot r_3.$$

It then sets $\bar{E} \leftarrow E'^{-e} \cdot b^{r_1}$, and proves knowledge of $(\{m_i\}_{i \in \bar{\mathcal{D}}}, e, r_2, r_3, r')$ such that the following holds:

$$\frac{\bar{E}}{d} = E'^{-e} \cdot Q^{r_2}$$
$$\wedge \quad P_0 \prod_{i \in \mathcal{D}} P_i^{m_i} = d^{r_3} \cdot Q^{-r'} \cdot \prod_{i \in \bar{\mathcal{D}}} P_i^{-m_i}.$$

The proof is then $(E', \bar{E}, d, \pi)$.

Verification. The verifier checks $E' \neq 1_{\mathbb{G}_1}$, checks $e(E', Z) = e(\bar{E}, H)$, and checks that π verifies.

D PRF security arguments

D.1 PRF Security of $\mathbf{F}_\mathsf{E}$

Lemma 1. *Under the DDH assumption in $\mathbb{G}_1$ and modeling $\mathcal{H}_\mathsf{E}$ as a random oracle, the function $\mathbf{F}_\mathsf{E}(tk', e) = \mathcal{H}_\mathsf{E}(e)^{tk'}$ is a secure pseudorandom function.*

We show that no polynomial-time adversary can distinguish $\mathbf{F}_\mathsf{E}(\cdot,\cdot)$ from a random function with non-negligible advantage.

Let $\mathcal{A}$ be an adversary with oracle access to either $\mathbf{F}_\mathsf{E}(tk', \cdot)$ or a random function $R : \{0,1\}^* \to \mathbb{G}_1$. Let $\mathcal{A}$ distinguish $\mathbf{F}_\mathsf{E}(tk', \cdot)$ from a random function with advantage $\varepsilon > 0$. The advantage is defined as $\varepsilon = |Pr[\mathcal{A} = 1|\mathbf{F}_\mathsf{E}] - Pr[\mathcal{A} = 1|R]|$. We use $\mathcal{A}$ to construct a simulator $\mathcal{S}$ that breaks DDH in $\mathbb{G}_1$.

$\mathcal{S}$ receives a DDH challenge (P, P^a, P^b, Z) where $Z = P^{ab}$ (DDH) or $Z \leftarrow \mathbb{G}_1$ (random). We use the distinguisher that can tell apart values output by $\mathbf{F}_\mathsf{E}$ and random values to build an adversary who can distinguish DDH tuples from random challenges. It does the following:

- Implicitly, the secret key is $tk'^* = b$.
- For queries e_i to $\mathcal{H}_\mathsf{E}(e_i)$, draw fresh $r_i, t_i \leftarrow \mathbb{Z}_p$ and set $\mathcal{H}_\mathsf{E}(e_i) = (P^a)^{r_i} \cdot P^{t_i} = P^{ar_i + t_i}$.
- For queries e_i to $\mathbf{F}_\mathsf{E}(tk'^*, \cdot)$, set $\mathbf{F}_\mathsf{E}(tk'^*, e_i) = Z^{r_i} \cdot (P^b)^{t_i}$.

The pair (r_i, t_i) is stored on the first query on e_i and reused afterwards, so both oracles are consistent across repeated queries.

If $Z = P^{ab}$, then $\mathbf{F}_\mathsf{E}(tk'^*, e_i) = P^{abr_i + bt_i} = \mathcal{H}_\mathsf{E}(e_i)^b$, so $\mathcal{S}$ has reproduced $\mathbf{F}_\mathsf{E}$ exactly, with key b.

If $Z = P^z$ for uniform z, then the answer pair for e_i is $\left(P^{ar_i + t_i}, P^{zr_i + bt_i}\right)$. The exponents are the image of the uniform vector (r_i, t_i) under the matrix $\left(\begin{smallmatrix} a & 1 \\ z & b \end{smallmatrix}\right)$, which is invertible whenever $ab \neq z$. Each pair is therefore uniform in $\mathbb{G}_1^2$, and, as the (r_i, t_i) are drawn independently, the pairs are mutually independent. The view of the distinguisher is then that of a random function, and we lose only the event $ab = z$, of probability $1/p$.

Re-randomizing with r_i alone would not suffice: it would give answers $P^{zr_i} = \mathcal{H}_\mathsf{E}(e_i)^{z/a}$, which is again $\mathbf{F}_\mathsf{E}$, under key z/a rather than b, and the two cases would be identically distributed. The term P^{t_i} is what breaks that correlation.

The distinguisher may hold T throughout. Its first component $T = P_2^{tk'} \cdot P_3^{tk''}$ is a perfectly hiding Pedersen commitment for $tk'' \xleftarrow{\$} \mathbb{Z}_p$, so $\mathcal{S}$ samples it uniformly, and its second component $\mathbb{W}$ is a function of tk'' alone, which $\mathcal{S}$ draws itself. Neither touches the challenge. As $\mathcal{S}$ also plays the registration authority it chooses the remaining BBS+ generators and the issuing key, and can produce every other public value of the protocol.

An adversary that distinguishes $\mathbf{F}_\mathsf{E}$ from random can then distinguish between the case where Z is a valid DDH tuple (and the responses are outputs of $\mathbf{F}_\mathsf{E}$) and the case where Z is random (and responses are uniformly random).

We then have the following advantage:

$$Adv[\mathcal{A}_\mathbf{F}(\mathbf{F}_\mathsf{E})] \leq Adv[\mathcal{A}_\mathsf{DDH}(\mathcal{S})] + 1/p.$$

D.2 PRF Security of $\mathbf{F_T}$

We use the Dodis-Yampolskiy PRF [17] as a black box, keyed with tk''. Its key must then stay secret, in order for PRF security to hold. We therefore give the auditor its outputs rather than its key, that is, the group elements defined as follows:

$$W_\gamma = D^{1/(tk''+\gamma)} \in \mathbb{G}_2, \qquad \gamma \in [0, max).$$

We set $\mathbb{W} = \{W_\gamma\}_{\gamma\in[0,max)}$ and $\mathbf{F_T}(T, ek, \gamma) = e(ek, W_\gamma)$. Tracing is unaffected, as an auditor holding ek and T computes every tag directly.

We cannot release the scalar tk'' to an auditor. Writing $Y = e(ek, D)$, the tags of an epoch are $\mathsf{tag}_\gamma = Y^{1/(tk''+\gamma)}$; an auditor holding tk'' recovers $Y = \mathsf{tag}_\gamma^{tk''+\gamma}$ from any single tag and then computes every other tag of that user for that epoch. Since tk'' is the same in every epoch, one grant would let that auditor trace the user in all later epochs, rather than being epoch bound as we desire. Giving the auditor the group elements $\{W_\gamma\}$ instead removes the need for the scalar, and tk'' stays secret.

Those elements belong in T and not in ek. They do not depend on the epoch, so an $\mathbf{F_E}$ that emitted them would repeat a component of its output in every epoch and could not be a pseudorandom function at all, whatever the hardness assumption. As public parameters of $\mathbf{F_T}$ they cost nothing: T is already certified at registration and already carried by the Grant message, and $\mathbf{F_E}$ stays pseudorandom in their presence because they are a function of tk'' while $\mathbf{F_E}$ is keyed by tk'. The analysis below accordingly treats $\mathbb{W}$ as known to the distinguisher.

We obtain pseudorandomness of the blinded outputs from the q-mDBDHI assumption of Appendix A.

Lemma 2. *Let ek be uniform in $\mathbb{G}_1$ and unknown to $\mathcal{A}$, and let the public parameters $\mathbb{W} = \{W_\gamma\}_{\gamma\in[0,max)}$ be given to $\mathcal{A}$. Under the q-mDBDHI assumption with $q = max$, the function $\mathbf{F_T}(T, ek, \gamma) = e(ek, W_\gamma)$ is a secure pseudorandom function on domain $[0, max)$.*

Let $\mathcal{A}$ distinguish the tag vector from random with advantage ε. We use $\mathcal{A}$ to build a simulator $\mathcal{S}$ that breaks q-mDBDHI. $\mathcal{S}$ receives a challenge, forwards $\{D^{1/(x+\gamma)}\}$ to $\mathcal{A}$ as $\{W_\gamma\}$ and the challenge vector as the tags, and echoes $\mathcal{A}$'s guess. Setting $ek = A$ and $tk'' = x$, the challenge is distributed exactly as the real tag vector when the challenge is real, and exactly as uniform tags otherwise. We then have the following advantage:

$$Adv[\mathcal{A}_{\mathbf{F}}(\mathbf{F_T})] = Adv[\mathcal{A}_{q\text{-mDBDHI}}(\mathcal{S})].$$

That ek is uniform in $\mathbb{G}_1$ in any epoch for which $\mathcal{A}$ holds no permission is Lemma 1, whose statement gives $\mathcal{A}$ the whole of T, so the two lemmas compose directly. We also note that the elements $\{W_\gamma\}$ do not compromise tk'' itself: recovering tk'' from $\{D^{1/(tk''+\gamma)}\}_{\gamma\in[0,max)}$ is the computational problem underlying q-mDBDHI above, and is implied by the hardness of the q-BDHI problem [8].

As the domain $[0, max)$ is of polynomial size, indistinguishability of the full tag vector from a uniform vector, which is what the assumption gives, is equivalent to pseudorandomness of $\mathbf{F}_{\mathsf{T}}$ in the sense of Section 3: a distinguisher making adaptive queries can be run on the vector directly.

E Instantiation using Hash-based PRFs

We use the BBS+ signature scheme for registration credentials; the Pedersen commitment scheme for commitments; hash-based PRFs for epoch keys and transaction tags; and zk-SNARKs to instantiate the NIZK proof systems $\mathcal{G}(\mathcal{R}_1)$ and $\mathcal{G}(\mathcal{R}_3)$. The SoK systems $\mathsf{SoKGen}(\mathcal{R}_0)$ and $\mathsf{SoKGen}(\mathcal{R}_2)$ are instantiated using Fiat-Shamir proofs of knowledge.

We use independent hash functions $\mathcal{H}_{\mathsf{FS}}$ for Fiat-Shamir challenges, $\mathcal{H}_{\mathsf{E}}$ for epoch key derivation, and $\mathcal{H}_{\mathsf{T}}$ for tags. We instantiate the components as follows:

(1) The one-way functions $\mathbf{G}$ and $\mathbf{G}'$ are instantiated as $\mathbf{G}(sk) = P^{sk}$ and $\mathbf{G}'(tk) = Q^{tk}$.
(2) The PRF families $\mathbf{F}_{\mathsf{E}}$ and $\mathbf{F}_{\mathsf{T}}$ are instantiated using independent hash-based PRFs $\mathcal{H}_{\mathsf{E}}$ and $\mathcal{H}_{\mathsf{T}}$: $\mathbf{F}_{\mathsf{E}}(tk, e) = \mathcal{H}_{\mathsf{E}}(tk, e)$, $\mathbf{F}_{\mathsf{T}}(ek, \gamma) = \mathcal{H}_{\mathsf{T}}(ek, \gamma)$.
(3) The signature scheme is instantiated using the BBS+ signature scheme.
(4) Commitments are instantiated using the Pedersen commitment scheme.
(5) The proof systems are instantiated using zk-SNARKs.
(6) The SoK systems are instantiated using proofs of knowledge, made non-interaction using the Fiat-Shamir heuristic in the random oracle model, using hash function $\mathcal{H}_{\mathsf{FS}}$.

This instantiation is benchmarked in Section 5.2.

F Security argument for Theorem 1

We prove security of Theorem 1 via real/ideal world indistinguishability.

Let $\mathcal{A}$ be a polynomial-time adversary interacting with our protocol, denoted hereafter Π. In the following, we construct a simulator $\mathcal{S}$ which, based on its interactions with ideal functionality $\mathcal{F}$, simulates an execution of Π to $\mathcal{A}$, and then prove that $\mathcal{A}$ cannot distinguish between a real execution of Π and a simulated one.

More specifically, we define a sequence of games, starting from a real-world execution of Π and ending with ideal functionality $\mathcal{F}$, and demonstrate that each game is (computationally) indistinguishable from the previous one.

Throughout the argument we use the key material of the black-box construction as it is defined in Section 4. Each user holds the two long-term keys (sk, tk) certified by the registration authority, together with an epoch tracing key $ek = \mathbf{F}_{\mathsf{E}}(tk, e)$ per epoch. We treat $\mathbf{F}_{\mathsf{E}}$ and $\mathbf{F}_{\mathsf{T}}$ as abstract PRF families in the sense of Section 3, with the range of $\mathbf{F}_{\mathsf{E}}$ equal to the key space of $\mathbf{F}_{\mathsf{T}}$, and we use nothing about them beyond pseudorandomness. The public parameters

T of $\mathbf{F}_{\mathsf{T}}$ are handed to $\mathcal{A}$ throughout, as both properties are required to hold in their presence.

In the following, $\mathcal{S}$ maintains the following tables, which are analogous to $\mathbb{R}$, $\mathbb{P}$, and $\mathbb{T}$ maintained by $\mathcal{F}$: **(1)** $\widehat{\mathbb{R}}[\mathbf{U}] = \langle sk, tk, \sigma \rangle$, logging registered users; **(2)** $\widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)] = ek$, logging tracing permissions and epoch keys; **(3)** $\widehat{\mathbb{T}}[(\mathbf{U}, e)]$, logging submitted transactions.

Game 0. This is the real execution of protocol Π.

Game 1. This is an execution of Π except that $\mathcal{S}$ simulates honest participants. Namely, $\mathcal{S}$ receives the inputs and generates the outputs of the ledger, registration authority, honest users, and honest auditors by following the instructions of Π. By construction, Game 1 is indistinguishable from Game 0.

Game 2. This game is identical to Game 1, except that $\mathcal{S}$ now, instead of executing Π during user registration, simulates the output of the registration authority solely based on the output $\mathcal{F}$ upon Register calls. For each Register request by some honest user $\mathbf{U}$, $\mathcal{S}$ selects a random key pair (sk, tk) and simulates an execution of Π, and stores $\widehat{\mathbb{R}}[\mathbf{U}] = \langle sk, tk, \sigma \rangle$.

Upon receiving a *valid* registration request from a corrupt user $\mathbf{U}$, $\mathcal{S}$ issues a Register request to $\mathcal{F}$ on behalf of the user and returns the corresponding output. If $\mathcal{F}$ successfully registers the user, then $\mathcal{S}$ uses the extractability property of SoK to extract the pair (sk, tk) and records $\widehat{\mathbb{R}}[\mathbf{U}] = \langle sk, tk, \sigma \rangle$, where σ is computed following Π's instructions and sent to $\mathbf{U}$.

This game is indistinguishable from Game 1 unless $\mathcal{F}$'s output upon one of the Register calls diverges from that of an execution of the registration functionality of Π. This only occurs if the registration authority, in the real world, accepts (rejects resp.) a registration request that $\mathcal{F}$ rejects (accepts resp.). Note that thanks to the correctness of Π, the event that $\mathcal{F}$ accepts while Π rejects will never occur. Similarly, the event that $\mathcal{F}$ rejects while Π accepts will never occur; in fact, this event can only take place if a corrupt user is able to trick the registration authority, in the real world, to register them twice. However, since the registration authority keeps track of registration requests, this will never succeed.

Game 3. This is identical to Game 2, except that $\mathcal{S}$ simulates the responses of honest users to tracing authorization requests, based only on the output of $\mathcal{F}$ upon Grant calls.

Upon a Grant call to functionality $\mathcal{F}$, for epoch e, honest user $\mathbf{U}$, and *honest auditor* $\mathbf{A}$, $\mathcal{S}$ updates $\widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)] \leftarrow \epsilon$ and simulates the transcript between $\mathbf{U}$ and $\mathbf{A}$ using randomly generated information. Given that information between $\mathbf{U}$ and $\mathbf{A}$ is exchanged using a secure channel, this is indistinguishable from an execution of Π, in the real world.

Upon an authorization request, in the real world, from a corrupt auditor $\mathbf{A}$ intended for honest user $\mathbf{U}$, $\mathcal{S}$ forwards the request to $\mathbf{U}$ in the ideal world. $\mathbf{U}$, accordingly, calls Grant, if $\mathcal{F}$ accepts, then $\mathcal{S}$ retrieves tuple $\langle sk, tk, \sigma \rangle \leftarrow \widehat{\mathbb{R}}[\mathbf{U}]$, simulates $\mathbf{U}$'s response following Π's instructions, and sets $\widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)] \leftarrow \mathbf{F}_{\mathsf{E}}(tk, e)$.

It is easy to infer that the simulated response is indistinguishable from an actual response generated by $\mathbf{U}$ during an execution of Π. Therefore, Game 3 is indistinguishable from Game 2.

Game 4. This is identical to Game 3, except that upon an authorization request from a corrupt auditor for some honest user $\mathbf{U}$ and epoch e, $\mathcal{S}$ samples an epoch key $ek^\star$ uniformly at random from the range of $\mathbf{F}_{\mathsf{E}}$, instead of computing $\mathbf{F}_{\mathsf{E}}(tk, e)$, sets $\widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)] \leftarrow ek^\star$, and then simulates the ZK proof for relation $\mathcal{R}_1$. Since tk is never used again in the simulation of $\mathbf{U}$'s epoch keys, the epoch keys that $\mathcal{S}$ hands out to corrupt auditors are now uniform and mutually independent across epochs.

Furthermore, when simulating an honest user submitting a transaction tx to the ledger, $\mathcal{S}$ produces the transaction following the description of Π, except that since now $ek^\star$ is sampled randomly, it simulates the ZK proof for relation $\mathcal{R}_3$. Now, thanks to the pseudorandomness of $\mathbf{F}_{\mathsf{E}}$, the zero-knowledge property of ZK proofs, and simulatability of SoKs, we conclude that this game is indistinguishable from Game 3.

Game 5. This is identical to Game 4, except that $\mathcal{S}$ now also simulates the output of transaction submission based on the output of $\mathcal{F}$ in response to Submit requests from honest users.

For each call by an honest user to Submit interface during epoch e, $\mathcal{S}$ checks if there is a corrupt auditor with a tracing permission that can trace the transaction back to its origin. This is achieved by issuing Trace queries on behalf of corrupt auditors for all the honest users that granted them tracing permissions, until there is a match. More specifically, $\mathcal{S}$ executes the following steps:

1. For each pair of honest user and corrupt auditor $\langle \mathbf{U}, \mathbf{A} \rangle$ with $\widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)] \neq \perp$, call Trace and receive $\mathbb{T}[(\mathbf{U}, e)]$.
2. If $\mathbb{T}[(\mathbf{U}, e)] \neq \widehat{\mathbb{T}}[(\mathbf{U}, e)]$, let $TX = \mathbb{T}[(\mathbf{U}, e)] \setminus \widehat{\mathbb{T}}[(\mathbf{U}, e)]$, update $\widehat{\mathbb{T}}[(\mathbf{U}, e)] \leftarrow \widehat{\mathbb{T}}[(\mathbf{U}, e)] \cup TX$, and output pair $\langle \mathbf{U}, TX \rangle$.

Note that since steps 1 and 2 are executed every time an honest user calls Submit with transaction tx, we have, by construction, for any honest user $\mathbf{U}$ who granted a tracing permission for epoch e to a corrupt auditor $\mathbb{T}[(\mathbf{U}, e)] = \widehat{\mathbb{T}}[(\mathbf{U}, e)]$ or $\mathbb{T}[(\mathbf{U}, e)] = \widehat{\mathbb{T}}[(\mathbf{U}, e)] \cup \mathsf{tx}$. Furthermore, we have either $\mathbb{T}[(\mathbf{U}, e)] = \widehat{\mathbb{T}}[(\mathbf{U}, e)]$ for all such users, or there exists one user $\mathbf{U}^\star$ such that $\mathbb{T}[(\mathbf{U}^\star, e)] = \widehat{\mathbb{T}}[(\mathbf{U}^\star, e)] \cup \mathsf{tx}$ and that's the user who called Submit. If $\mathcal{S}$ encounters the first scenario, then this implies that the honest user who called Submit did not grant tracing permissions to any of the corrupt auditors. If it encounters the second, then $\mathbf{U}^\star$ is the user who called Submit, and at least one corrupt auditor is authorized to trace that user's transactions for epoch e. Accordingly, we consider the following two cases.

Case 1. No corrupt auditor has a tracing permission for the honest user for the epoch. $\mathcal{S}$ sets the confidential payload to some arbitrary ciphertext and instead of computing the commitment as $(C, \rho) \leftarrow \mathsf{Commit}(CK, tk)$ and the tag as $\mathsf{tag}_\gamma = \mathbf{F}_{\mathsf{T}}(ek, \gamma)$, it samples uniformly at random commitment $C^\star$ and tag $\mathsf{tag}^\star_\gamma$ and simulates the corresponding SoK and ZK proof. Since, after Game 4, the epoch

keys released to corrupt auditors are uniform and independent across epochs, the keys released for preceding and succeeding epochs carry no information about the epoch key in use here, which $\mathcal{A}$ therefore does not hold. As a result, we can use the hiding property of Commit, the pseudo-randomness of $\mathbf{F}_{\mathsf{T}}$ under a key $\mathcal{A}$ does not hold, which holds even though $\mathcal{A}$ holds T, and the simulatability of SoK, and zero-knowledge of ZK proofs to conclude that aforementioned modifications are indistinguishable from submitting transactions during an execution of Π.

Case 2. There is a corrupt auditor with valid tracing permission for the honest user. $\mathcal{S}$ in this case is able to get pair $\langle \mathbf{U}, TX \rangle$ where $\mathbf{U}$ identifies the honest user and TX consists of the transaction tx that $\mathbf{U}$ sent with their Submit query. Given this information, $\mathcal{S}$ retrieves tuple $\langle sk, tk, \sigma \rangle \leftarrow \widehat{\mathbb{R}}[\mathbf{U}]$ and the random epoch key $ek^{\star} \leftarrow \widehat{\mathbb{P}}[(\mathbf{A}, \mathbf{U}, e)]$. $\mathcal{S}$ then, in the real world, sets the confidential payload $\mathfrak{p}$ to tx, and simulates the rest of the transaction similarly to Game 4.

Game 6. This is identical to Game 5, except that $\mathcal{S}$ now simulates the output of transaction submission in response to transactions from corrupt users, based on the output of $\mathcal{F}$'s Submit interface.

When a corrupt user $\mathbf{U}$ submits a transaction with a valid tag (i.e., a tag with valid ZK proof and SoK and which does not appear in the ledger), $\mathcal{S}$ uses the extractability property of SoKs and the knowledge soundness of ZK proofs to extract $\mathbf{U}$'s witness. Furthermore, thanks to the existential unforgeability of the digital signature used during registration, $\mathcal{S}$ will be able to identify $\mathbf{U}$ using table $\widehat{\mathbb{R}}$. Next, $\mathcal{S}$ sends a Submit request on behalf of $\mathbf{U}$ to $\mathcal{F}$ and returns the corresponding output. If $\mathcal{F}$ accepts the transaction, then $\mathcal{S}$ records it in $\widehat{\mathbb{T}}[(\mathbf{U}, e)]$.

This game is indistinguishable from Game 5 unless $\mathcal{F}$ rejects a transaction that Π deems valid. We recall that $\mathcal{F}$ rejects a transaction in one of two cases:

1. $\mathbf{U}$ *is not registered:* this cannot be the case given that each entry in $\widehat{\mathbb{R}}$ corresponds to a successful Register call to $\mathcal{F}$.
2. $\mathbf{U}$ *already submitted max transactions in the current epoch:* Given the knowledge soundness of ZK proofs and determinism and injectivity of $\mathbf{F}_{\mathsf{T}}$ on $[0, max)$, Π will not accept a transaction from $\mathbf{U}$ in this case as the tag will already appear in the ledger.

Game 7. This is identical to Game 6, except that $\mathcal{S}$ aborts the game if the output of the Trace interface differs from the output of the tracing functionality of protocol Π.

Note that $\mathcal{S}$ aborts if one of these two events occurs.

- ABORT 1. A coalition of corrupt auditors and users are able to trace a transaction submitted by an honest user during an epoch, even though the user did not grant permissions to any corrupt auditor for that epoch.
- ABORT 2. An honest auditor fails to trace a transaction of a corrupt user that was submitted during an epoch, although they have been granted the authorization.

We now show that the probability of these events is negligible.

ABORT 1. Since the transactions of honest users during an epoch for which the honest users did not grant tracing permissions to any corrupt auditor are generated at random, this event does not occur.

ABORT 2. We recall that a transaction tx from a corrupt user is only recorded in $\widehat{\mathbb{T}}$ and forwarded to $\mathcal{F}$, if it carries a valid tag tag_γ; that is, the SoK for relation $\mathcal{R}_2$ and the ZK proof for relation $\mathcal{R}_3$ are both valid. Given the extractability property of SoKs and knowledge soundness of ZK proofs, we can safely conclude that $\mathsf{tag}_\gamma = \mathbf{F}_\mathsf{T}(ek, \gamma) \ \wedge \ ek = \mathbf{F}_\mathsf{E}(tk, e)$ with (sk, tk) verifying Equations 6 and 8. Therefore, an honest auditor with the right permission will fail to trace tx only if they have accepted a bogus epoch key $\widehat{ek} \neq \mathbf{F}_\mathsf{E}(tk, e)$. However, given the knowledge soundness of the ZK argument for $\mathcal{R}_1$, the extractability of the SoK for $\mathcal{R}_0$, and the existential unforgeability of digital signatures, this event occurs only with a negligible probability.

G Security Argument for Theorem 2

Unforgeability of registration depends on the unforgeability of the signature scheme used by the registration authority. We instantiate this with BBS+ signatures. Therefore, unforgeability of registration depends on the q-SDH assumption.

Transaction unforgeability corresponds to existential unforgeability of the digital signature scheme, and soundness of the NIZKs and SoKs. Soundness of the tag argument, including the equivalence of $\mathcal{R}_3$ and $\widehat{\mathcal{R}}_3$, is proven in Appendix H; soundness of the range argument on C_γ rests on the discrete logarithm assumption in $\mathbb{G}_1$.

Privacy of transactions depends on the pseudorandomness of $\mathbf{F}_\mathsf{E}$ and $\mathbf{F}_\mathsf{T}$, the zero knowledge property of ZK proofs, and simulatability of SoKs. Our NIZKs and SoKs are simulatable in the random oracle model.

The two PRFs. It remains to check that the instantiation supplies $\mathbf{F}_\mathsf{E}$ and $\mathbf{F}_\mathsf{T}$ as the construction of Section 4 asks: two pseudorandom functions, the range of the first being the key space of the second, both secure in the presence of T.

$\mathbf{F}_\mathsf{E}(tk, e) = \mathcal{H}_\mathsf{E}(e)^{tk'}$ has key space $\mathbb{Z}_p$ and range $\mathbb{G}_1$, and is a PRF under DDH in $\mathbb{G}_1$ by Lemma 1. Every component of its output varies with the epoch, which is what placing $\mathbb{W}$ in T rather than in ek buys: an epoch key is a single group element, and two epochs share no material. The lemma is stated and proven with T in the distinguisher's hands, which it can be, as T is perfectly hiding and $\mathbb{W}$ is a function of tk'' alone while $\mathbf{F}_\mathsf{E}$ is keyed by tk'.

$\mathbf{F}_\mathsf{T}(T, ek, \gamma) = e(ek, W_\gamma)$ is keyed by ek, which is exactly the range of $\mathbf{F}_\mathsf{E}$, and is a PRF on $[0, max)$ under the q-mDBDHI assumption with $q = max$ by Lemma 2. Its public parameters are $\mathbb{W}$, and the lemma gives them to the distinguisher.

Game 4 of Appendix F replaces the epoch key handed to a corrupt auditor by a uniform element of $\mathbb{G}_1$, which is Lemma 1; Game 5 then argues about the tags of an epoch whose key the adversary does not hold, which is Lemma 2. An

auditor authorized in earlier epochs can recompute those epochs' tags from its own view, so releasing them adds nothing, and $\mathbb{W}$ is public in both games by construction. Zero knowledge of the transaction argument additionally requires DDH in $\mathbb{G}_T$, for the reason given in Appendix H.

H $\mathcal{R}_3$ and $\widehat{\mathcal{R}}_3$ equivalence proofs

Recall from Section 5.1 that $\mathcal{R}_3(\mathbb{x}_3, \mathbb{w}_3)$ has $\mathbb{x}_3 = (\mathcal{H}_{\mathsf{E}}, D, P_2, P_3, Q, C, max, e, \mathsf{tag}_\gamma)$ and $\mathbb{w}_3 = (tk', tk'', \rho, ek, \gamma)$. We then have $\mathcal{R}_3(\mathbb{x}_3, \mathbb{w}_3) = 1$ **if and only if** the following holds:

$$C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^{\rho} \wedge ek = \mathcal{H}_{\mathsf{E}}(e)^{tk'} \wedge \mathsf{tag}_\gamma = e\Big(ek,\, D^{1/(tk''+\gamma)}\Big) \wedge 0 \le \gamma < max.$$

Proving the tag equality directly would require an argument over a secret $\mathbb{G}_1$ element paired against a term whose $\mathbb{G}_2$ exponent is also secret. The prover therefore samples $s, r_\gamma \xleftarrow{\$} \mathbb{Z}_p$ and publishes the following randomized values, the last of which is a Pedersen commitment to the transaction index:

$$U = \mathcal{H}_{\mathsf{E}}(e)^s, \quad \Pi = e(ek^s, D), \quad X = \mathsf{tag}_\gamma^s, \quad C_\gamma = P_3^\gamma \cdot Q^{r_\gamma}.$$

We define $\widehat{\mathcal{R}}_3(\widehat{\mathbb{x}}_3, \widehat{\mathbb{w}}_3)$ with $\widehat{\mathbb{x}}_3 = (\mathbb{x}_3, U, \Pi, X, C_\gamma)$ and $\widehat{\mathbb{w}}_3 = (\mathbb{w}_3, s, r_\gamma)$. We then have $\widehat{\mathcal{R}}_3(\widehat{\mathbb{x}}_3, \widehat{\mathbb{w}}_3) = 1$ **if and only if** the following holds:

$$\begin{aligned} C = P_2^{tk'} \cdot P_3^{tk''} \cdot Q^{\rho} \quad \wedge \quad U = \mathcal{H}_{\mathsf{E}}(e)^s \quad \wedge \quad \Pi = e(U, D)^{tk'} \\ \wedge \quad X = \mathsf{tag}_\gamma^s \quad \wedge \quad \Pi = X^{tk''} \cdot X^{\gamma} \\ \wedge \quad C_\gamma = P_3^\gamma \cdot Q^{r_\gamma} \quad \wedge \quad 0 \le \gamma < max. \end{aligned}$$

Every equality here is now an equality of discrete logarithms in $\mathbb{G}_1$ or $\mathbb{G}_T$ with public bases. The single pairing Π is computed by the verifier from public data.

For simplicity, we argue equivalence considering the interactive variant of the zero-knowledge proofs - the Sigma protocol without the Fiat-Shamir heuristic.

Correctness. Let $\mathbb{w}_3$ satisfy $\mathcal{R}_3$ and sample $s \xleftarrow{\$} \mathbb{Z}_p$. We set $U = \mathcal{H}_{\mathsf{E}}(e)^s$, $V = ek^s$ and $X = \mathsf{tag}_\gamma^s$. The equalities for C, for U, for X, for C_γ and for the range of γ then hold by construction, and $V = (\mathcal{H}_{\mathsf{E}}(e)^{tk'})^s = U^{tk'}$ gives the one for V. For the pairing equality, $\mathsf{tag}_\gamma = e(ek, D^{1/(tk''+\gamma)})$ gives $\mathsf{tag}_\gamma^{tk''+\gamma} = e(ek, D)$. We then have the following:

$$X^{tk''} \cdot X^{\gamma} = \big(\mathsf{tag}_\gamma^{tk''+\gamma}\big)^s = e(ek, D)^s = e(ek^s, D) = \Pi.$$

Soundness. The Sigma protocol of Section 5.1 extracts $(tk', tk'', \rho, s, \gamma)$ from two accepting transcripts that share a first message. The epoch key is not extracted directly. We instead fix $ek := \mathcal{H}_{\mathsf{E}}(e)^{tk'}$, which gives the second equality of $\mathcal{R}_3$ by definition, and the first equality is the first verification equation. What

remains is the tag equality, together with showing that the extracted γ is the value the range argument speaks about.

From $U = \mathcal{H}_{\mathsf{E}}(e)^s$ and $\Pi = e(U, D)^{tk'}$ we get $V = \mathcal{H}_{\mathsf{E}}(e)^{s \cdot tk'} = ek^s$. With $X = \mathsf{tag}_\gamma^s$, the pairing equality then reads as follows:

$$e(ek^s, D) = \left(\mathsf{tag}_\gamma^s\right)^{tk''+\gamma}, \qquad \text{that is} \qquad e(ek, D)^s = \left(\mathsf{tag}_\gamma^{tk''+\gamma}\right)^s.$$

The verifier rejects unless $U \neq 1_{\mathbb{G}_1}$, which forces $s \neq 0$, as $\mathcal{H}_{\mathsf{E}}(e)$ is a generator of $\mathbb{G}_1$ with overwhelming probability over the random oracle. Cancelling s then leaves the following:

$$e(ek, D) = \mathsf{tag}_\gamma^{tk''+\gamma}. \tag{9}$$

The verifier also rejects unless $\Pi \neq 1_{\mathbb{G}_T}$. As $V = ek^s$ with $s \neq 0$, this forces $ek \neq 1_{\mathbb{G}_1}$, and non-degeneracy of the pairing then gives $e(ek, D) \neq 1_{\mathbb{G}_T}$. By Equation 9 the right hand side is then not the identity either, so we have $tk'' + \gamma \neq 0 \bmod p$ and we can invert it as follows:

$$\mathsf{tag}_\gamma = e(ek, D)^{1/(tk''+\gamma)} = e\Big(ek,\ D^{1/(tk''+\gamma)}\Big).$$

This is the tag equality of $\mathcal{R}_3$. Both non-degeneracy checks are needed here. If $s = 0$ then U, V and X all collapse to identities and the pairing equality says nothing. If $tk'' = -\gamma$ then a prover holding $tk' = 0$ satisfies Equation 9 with any tag at all.

Finally, we look at the fifth verification equation, $X^{\theta_2} \cdot X^{\theta_\gamma} = F \cdot \Pi^c$. As $X^{\theta_2} \cdot X^{\theta_\gamma} = X^{\theta_2+\theta_\gamma}$, this equation on its own binds only the sum $tk''+\gamma$, which is all that Equation 9 needs. We separate the two values elsewhere. The response θ_2 is shared with the first verification equation, where the binding property of the Pedersen commitment C pins tk'' to the value certified for the user. Given tk'', the sum then pins γ. The response θ_γ is in turn shared with the sixth verification equation, $P_3^{\theta_\gamma} \cdot Q^{\theta_r} = A_\gamma \cdot C_\gamma^c$, which opens C_γ to the extracted γ. As the range argument is run on that same C_γ, the value shown to lie in $[0, max)$ is the same γ that appears here. A prover who shifts tk'' to $tk'' + \delta$ and γ to $\gamma - \delta$ still satisfies the fifth equation, but breaks the first.

Zero knowledge. We construct a simulator $\mathcal{S}$ that, given only $\mathbb{x}_3$ and without $\mathbb{w}_3$, produces a transcript (U, Π, X, π) with $\pi = (A, B, M, K, F, c, \theta_1, \theta_2, \theta_h, \theta_s, \theta_\gamma)$ that is computationally indistinguishable from an honestly generated one.

Simulating (U, Π, X). $\mathcal{S}$ samples $s \xleftarrow{\$} \mathbb{Z}_p$ and sets $U = \mathcal{H}_{\mathsf{E}}(e)^s$ and $X = \mathsf{tag}_\gamma^s$. It can compute both of these, as $\mathcal{H}_{\mathsf{E}}(e)$ and tag_γ are public. It then samples $\Pi \xleftarrow{\$} \mathbb{G}_T$ uniformly, as it cannot compute $e(ek^s, D)$. The only deviation from the real distribution is therefore Π.

We state the claim against a distinguisher that does not hold ek for the epoch in question; against an authorized auditor the transaction is traceable by design, so no hiding is claimed there. Such a distinguisher may, however, hold $\mathbb{W}$, which

is public and does not expire, and this is what the argument has to survive. Writing the two values out in terms of the public U gives

$$X = e(U, W_\gamma)^{tk'}, \qquad \Pi = e(U, D)^{tk'},$$

so (X, Π) is a same-exponent pair over the two $\mathbb{G}_T$ bases $e(U, W_\gamma)$ and $e(U, D)$, both of which a holder of $\mathbb{W}$ can compute. Replacing Π by a uniform element is therefore exactly a DDH instance in $\mathbb{G}_T$. Across several transactions of the same user the exponent tk' is shared, so the hop is a hybrid over transactions whose steps are DDH instances in $\mathbb{G}_T$; by the random self-reducibility of DDH the loss is independent of the number of transactions. We also note that X needs no separate assumption, as it is a function of public data and of the simulator's own s.

We publish Π and not ek^s, because with $V = ek^s$, bilinearity would give $e(V, W_\gamma) = e(ek, W_\gamma)^s = X$, so a holder of $\mathbb{W}$ could test $X = e(V, W_\gamma)$ for each $\gamma \in [0, max)$ and identify the transaction outright, with no assumption and without ever holding ek. As $\mathbb{W}$ does not vary with the epoch, that test would be available to an auditor long after its permission lapsed. Publishing Π means this unwanted check is no longer possible.

The range argument for $0 \leq \gamma < max$ is simulated by the zero-knowledge simulator of the underlying protocol [12] on C_γ with an arbitrary committed value. The hiding property of C_γ, which holds as r_γ is uniform and Q is independent of P_3, means that no distinguisher can tell which value was used.

Therefore, under the DDH assumption in $\mathbb{G}_T$ and the zero-knowledge property of the range argument, (U, Π, X, π) is zero-knowledge against any party that does not hold the epoch key.

The non-interactive variant. *Soundness.* Instead of rewinding the prover, we rely on the forking lemma to extract $\widehat{\mathfrak{w}}_3$. The argument for soundness then follows exactly as before.

Zero-knowledge. We argue zero-knowledge using the same argument as above, except that the challenge c is a hash, modeled as a programmable random oracle. $\mathcal{S}$ samples c, computes $(A, B, M, K, F, A_\gamma)$ exactly as above, and programs $\mathcal{H}_p(\widehat{\mathfrak{x}}_3, A, B, M, K, F, A_\gamma) = c$.