Cross Ledger Transaction Consistency for Financial Auditing

Vlasis Koutsos[1], Xiangan Tian[1], Dimitrios Papadopoulos[1], and Dimitris Chatzopoulos[2]

[1] Hong Kong University of Science and Technology
{vkoutsos,xtianae,dipapado}@cse.ust.hk
[2] University College Dublin
dimitris.chatzopoulos@ucd.ie

Abstract. Auditing throughout a fiscal year is integral to organizations with transactional activity. Organizations transact with each other and record the details for all their economical activities so that a regulatory committee can verify the lawfulness and legitimacy of their activity. However, it is computationally infeasible for the committee to perform all necessary checks for each organization. To overcome this, auditors assist in this process: organizations give access to all their internal data to their auditors, who then produce reports regarding the consistency of the organization's data, alerting the committee to any inconsistencies. Despite this, numerous issues that result in fines annually revolve around such inconsistencies in bookkeeping *across organizations*. Notably, committees wishing to verify the correctness of auditor-provided reports need to redo all their calculations; a process which is computationally proportional to the number of organizations. In fact, it becomes prohibitive when considering real-world settings with thousands of organizations. In this work, we propose two protocols, CLOSC and CLOLC, whose goals are to enable auditors and a committee to verify the consistency of transactions across different ledgers. Both protocols ensure that for every transaction recorded in an organization's ledger, there exists a dual one in the ledger of another organization while safeguarding against other potential attacks. Importantly, we minimize the information leakage to auditors and other organizations and guarantee three crucial security and privacy properties that we propose: *(i)* transaction amount privacy, *(ii)* organization-auditor unlinkability, and *(iii)* transacting organizations unlinkability. At the core of our protocols lies a two-tier ledger architecture alongside a suite of cryptographic tools. To demonstrate the practicality and scalability of our designs, we provide extensive performance evaluation for both CLOSC and CLOLC. Our numbers are promising, i.e., all computation and verification times lie in the range of seconds, even for millions of transactions, while the on-chain storage costs for an auditing epoch are encouraging i.e. in the range of GB for millions of transactions and thousands of organizations.

1   Introduction

Bookkeeping is an indispensable part of *organizations* (e.g., businesses, municipalities, banks). One of the main reasons organizations maintain ledgers with their transactions (to/from other organizations) is to convince *committees* (e.g., the Public Company Accounting Oversight Board[1] in the United States of America or the Financial Reporting Council[2] in the United Kingdom) about the integrity and lawfulness of their operations. They periodically produce statements about the integrity and correctness of their finances, signed by an *auditor*. The goal of auditors is to ensure that there are no mistakes or inconsistencies in the organization-reported numbers [HN96]. To that end, they sign/generate a report on the organization-provided data. Figure 1 showcases a (simplified) flowchart model of organizations, auditors, and the committee during a financial epoch. Organizations transact with each other, keep respective records, and disclose them to their auditors at the appropriate time. The auditors, after examining the provided data, generate and sign a report attesting to the judicious activity of their client-organization, and send it over to a committee who verifies its content.

Although auditors have access to *all* organization-reported data and paperwork of their clients, checking for consistency is prohibitive in terms of human resources and time constraints [Ear15, HSJ+23]. This is mainly because organizations under audit may record hundreds of transactions daily. To address this issue, auditors have developed probabilistic *processes* that check for consistency between the received data and the actual paperwork, but not for the entirety of the data. This, in turn, opens up the audit to additional risk. However, it is currently the best tool professionals use to ensure the audit's feasibility [ABAAA23].

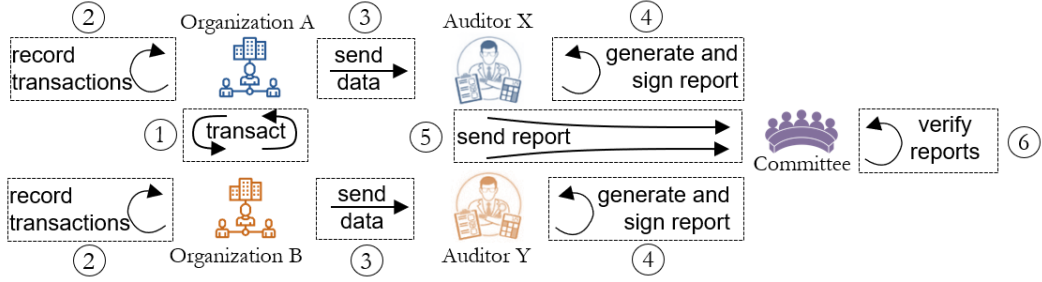[1] https://pcaobus.org   [2] https://frc.org.uk

Fig. 1: Entity setting, interactions, and flowchart for current financial auditing. Organization A is audited by Auditor X and B by Y respectively. Auditors generate reports on data received from their client-organization that the Committee can later verify.

Established auditing process. Organizations record their transactions in a ledger throughout the fiscal year, meaning that a *ledger* is a list of financial transactions an organization holds over time. At a later point, the auditing period begins, during which auditors sample a percentage of the total reported transactions and request to examine the corresponding paperwork from their client-organizations. Upon conducting all relevant checks (e.g., validity of signatures, and consistency of amounts and timestamps) the auditor includes their findings regarding the auditing output in a report, which they later sign and make public. After the auditing period, a committee can select to verify the consistency between the auditor-generated reports and the recorded data of the organizations. For consistency verifiability purposes, *all transactions have to be kept in the ledgers of both transactional parties*. Especially for organization-to-organization (O2O) transactions, *each of them* needs to register a transaction that is the dual equivalent of the other.

1.1 The problem we focus on

There are numerous cases annually of organizations misreporting transactions. For example, they fabricate and report sham transactions, which in turn leads to auditing scandals involving fraud and fines in the range of millions of USD[3] [Nah16a, Nah16b]. In fact, this derives from a crucial limitation of the existing auditing ecosystem: auditors cannot check if a recorded transaction in their client's ledger has a dual counterpart in another organization's ledger [Hol23]. E.g., an organization may procure illicit funds and fabricate transaction records, for which no other organization would have dual transactions to. Despite its importance, to the best of our knowledge, no process exists to check the duality of transactions between two ledgers where the organizations that maintain them are examined by separate auditors. Hence, we pose the following question:

Can we ensure that an auditor, who does not have access to internal data of any organization except of her client-organization, can verify the duality of all O2O transactions of her client?

Limitations of applying existing cryptographic approaches. Before going further, we examine two ideas utilizing existing techniques to solve this problem at a high level. First, the auditor and the two transacting parties could engage in a multi-party computation protocol at the time of the auditing. Such an attempt requires the auditor to interact with all organizations its client has transactions with, introducing a linear communication overhead proportional to the number of organizations and auditors. Moreover, let's consider having just two organizations, o_i and o_j, who initiate a 2-party computation protocol and agree on a common identifier for a mutual transaction. Upon deal completion, o_j will provide o_i with a digest $dgs_{j,z}$ (e.g., a cryptographic accumulator) and a proof that $dgs_{j,z}$ "contains" the o_j-part of the trade, and vice versa, proving the existence of the transactions in question. However, in reality, organizations do not trust each other, and auditors do not assume organizations to behave honestly. In fact, interesting questions arise in such scenarios:

1. What happens if o_i or o_j do not follow the protocol?
2. What happens if o_i computes a digest, sends it over to o_j with convincing proof, but later on includes the transaction in its ledger using different data?
3. What happens if o_i appends an incorrect proof with its data on their internal ledger?

---

[3] https://corporatefinanceinstitute.com/resources/accounting/top-accounting-scandals/

2

Since organizations cannot access the ledger of their trading counterparts, auditors cannot distinguish even between trivial scenarios e.g., identify which party made mistakes or uploaded inconsistent data. Requiring auditor collaboration to discover the truth behind inconsistencies is far from a realistic assumption — both from a performance and a real-world perspective.

Another approach could be the following: Consider two organizations transacting with each other and recording this transaction on their ledgers. Each of them now can produce and communicate to the other a zero-knowledge proof (ZKP) about the inclusion of the transaction record in their respective ledgers. However, this approach works only when assuming that both organizations are honestly maintaining their ledgers. E.g., a client might generate and provide a ZKP about an O2O transaction to its counterpart, but later alter its ledger state, before the audit begins. The ZKP would still verify, as it was honestly generated at the time, but vitally the ledger alteration would be undetectable. Thus, a notion of ledger-immutability is necessary, on top of such a technique.

There exists an additional limitation in the current auditing ecosystem: committees wishing to verify process outputs need to re-perform all operations themselves. Specifically, a committee that attempts to verify all process outputs needs to expend the entirety of the collective effort from all auditors. Verifying all processes on all ledgers is rendered impractical in this case since the verification time is linearly proportional to the number of organizations and auditors. Instead, committees perform checks on a number of reports and thus trust implicitly the remaining ones to be generated honestly. Various systems utilize techniques such as verifiable computation, secure hardware, or tailored ZKPs to enable auditing parties to verify function output results in sublinear time, and in Section 2 we investigate them in more detail. However, in the scope of the *cross-ledger transaction consistency for financial auditing* that we examine no such solution exists to date. Therefore, we adjust and pose a newer version of our previous question:

Can we ensure that an auditor, without access to internal data of any organization except of her client-organization, (1) can verify the duality of all O2O transactions of her client and (2) produce a result that a committee can verify without having access to any internal organization data, efficiently?

State-of-the-art. There exist prior works that indirectly provide a solution to part (1) of our problem, however, they operate in a different model. Specifically, zkLedger [NVV18] and Miniledger [CB21] consider a scenario where *all* organizations maintain a single ledger in a distributed manner that includes *all* transactions in a hiding manner. Nevertheless, their model does not correspond to the real-world alternative, where bookkeeping is being done individually by each organization. E.g., a small enterprise that logs a thousand transactions annually should not need to record any data from all the remaining transactions of other organizations. Another relevant work to our problem revolves around cross-chain bridges[4]. Their core functionality is enabling proof generation of an event that occurred on one chain to be verified on another. This indeed fits into our problem setting, but most current bridges[5][6][7] either suffer from poor performance or rely on central entities. We refer the reader to Section 2 for a more elaborate comparison between prior works and our solution.

Our results. First, we introduce and formulate the problem of cross-ledger transaction consistency for financial auditing, including the system and threat models, as well as crucial security goals. We then propose two protocols, `CLOSC` (Cross Ledger cOnsistency with Smart Contracts) and `CLOLC` (Cross Ledger cOnsistency with Linear Combinations), implement them upon a two-tier ledger/blockchain-based architecture, and provide extensive evaluation results regarding their performance. Additionally, we formally define three privacy and security properties, namely *transaction amount privacy*, *organization-auditor unlinkability*, and *transacting organizations unlinkability*, and prove that both our protocols satisfy them.

`CLOSC` utilizes smart contracts for storing transaction-related data and proofs from organizations. For each transaction, both organizations deploy a smart contract and fill in their "half" regarding the consistency-checking method, to both smart contracts. In this way, both auditors have all the information needed to verify the consistency between the reported transactions. `CLOLC` relies on linear combinations with organizations now maintaining a separate list of transactions for each of their transaction counterparts. The consistency checking is performed on an "O2O-pair" basis, where each auditor verifies consistency for each organization their client transacts with individually. To enable this, the committee assigns and distributes to auditors weights for each individual O2O transaction for every transacting pair. Then, for each transacting organization, the auditor calculates the linear

---

combination of the amounts with the corresponding weights and exchanges the result with the counterpart's auditor. The committee in both `CLOSC` and `CLOLC` essentially performs two types of checks: *(i)* consistency between the reported data from an organization and its auditor, and *(ii)* reported data from the two auditors. This has a dual purpose: First, when all verifications succeed, this signifies that consistency exists across all organization ledgers, and second, when a verification is unsuccessful, the committee needs only to further investigate the particular O2O transacting organization-pair. Importantly, all checks above are lightweight and do not impose prohibitive overheads for the committee.

We test our system on AWS machines and implement our architecture over Hyperledger Fabric. To execute an auditing epoch in `CLOSC` for 1024 organizations, with each of them recording $1M$ transactions requires on average ≈ 43 mins per organization, ≈ 18mins per auditor, and ≈ 4secs for the committee, whereas for the same organizations and total transactions in `CLOLC` it takes on average ≈ 30mins per organization, ≈ 39mins per auditor, and ≈ 4mins for the committee. We provide an extensive evaluation of both our proposed solutions in Section 7 and we include a comparison analysis in terms of protocol computation and storage needs complexity with prior works that have "similar-enough" auditability goals.

Paper organization. In Section 2 we expand on prior works relevant to our problem and in Section 3 we introduce the background for our system and protocol design. Following, we concretely formulate the problem we are focusing on in this work in Section 4. Then, we analyze our system architecture and provide the details of our two protocols (`CLOSC` and `CLOLC`) in Section 5, and provide the security analysis of our solutions, including the definitions of our three newly proposed security properties, in Section 6. In Section 7 we present the implementation details of our protocols and demonstrate their performance. Last, we provide a discussion on the limitations and potential future directions in Section 8, and conclude our work in Section 9.

2   Related Work

The combination of privacy-enhancing and blockchain technologies has been gaining interest, especially for "traditional" financial applications [BCDF23]. A core property of blockchains is immutability and, as a result, multiple researchers have tried to enable/construct blockchain-assisted auditing. Generally, there seems to be a consensus amongst researchers and industry professionals as to the anticipation that blockchains are a disrupting force in the auditing ecosystem and that their role will get increasingly important [AČE⁺20, RV18, Koz18, DV17, DDM20, AN17, CMM⁺22], potentially even shifting the auditing process from backward to forward-looking [BGMS18]. However, as pointed out in [HSJ⁺23] the vast majority of blockchain-related works do not look into how to utilize blockchains and smart contracts to address the challenges revolving around financial auditing, with a small set of notable exceptions. The authors of [DV17] propose a triple-booking system where a copy of all records is kept on a blockchain (on top of an existing double-booking protocol). The inclusion of smart contracts in the design of such systems was proposed in [Koz18, RV18]. However, in both [Koz18, RV18] the access rights/patterns are not clearly outlined and questions arise about how potential leakage can be used for malicious purposes.

Single-ledger approaches. To enforce that during a transaction no entity expends more than the total amount of assets they hold, the authors of zkLedger [NVV18] introduced tailored *proof of assets*, specifically in a banking setting, similar to our O2O scenario. Importantly, this work uses zero-knowledge proofs to enable confidential transactions while *(i)* allowing for regulatory compliance and auditability and *(ii)* guaranteeing the condition above without revealing to the other system's participants anything about transaction amounts. However, zkLedger has not been implemented on any blockchain platform and the proposed protocol does seem not scale well with the number of protocol participating entities. This is mainly due to all participants needing to maintain the same version of a *single ledger* in a distributed manner with storage complexity $\mathcal{O}(nm)$, where n are the total banks and m are the total transactions in the system. Additionally, zkLedger cannot support multiple transactions happening in parallel. This derives from the fact that in order for the ledger to accept a transaction, it needs to verify the correctness of all appended proofs to it, rendering the cases where multiple transactions are being submitted concurrently impossible to handle; there needs to be an ordering protocol. The authors of Miniledger [CB21] overcome the scalability issues by introducing a pruning technique for the ledger in question. However, Miniledger suffers from other shortcomings such as large transaction creation time ($\approx 5s$ for a single transaction in a setting with 100 banks) and

requires both a synchronization and a transaction-ordering protocol, both of which are unrealistic in real-world scenarios where organizations transact asynchronously with each other daily and in the thousands totally. More recently, ACA [LHNH23] has been proposed with focus to anonymity, confidentiality, and auditability of transactions. The authors rely on a conventional blockchain layer for recording the transactions which are smaller than zkLedger and Miniledger, however the size of general transactions ($\approx 3.8KB$) and the verification times ($\approx 1.4sec$) are prohibitive for settings as the ones we consider. In Section 7 we demonstrate the end-to-end of an entire auditing epoch in both CLOSC and CLOLC takes minutes even for millions of transactions, regardless of the number of organizations participating in our system.

Communication between different blockchains. An emerging research area revolves around *cross-chain bridges*, a technique that increases token utility by facilitating cross-chain liquidity between distinct blockchains. Specifically, they enable users to transform their tokens from one blockchain to another, usually by burning or locking existing tokens from the original-chain, and minting or unlocking "new" ones to the target-chain[4]. Essentially bridges utilize messaging protocols that in theory can be used to support arbitrary messages across different chains, usually via smart contracts[5,6,7]. Recently, zkBridge [XZC+22] has been proposed, a protocol that betters the performance of existing bridges while achieving higher security standards. The authors utilize zk-SNARKs for generating proofs for relaying block headers. However, this approach relies on the construction of smart contracts, which is not ideal due to storage and computational costs required. In Section 7 we showcase the difference in efficiency and scalability between our two protocols. CLOSC relies on smart contracts, whereas CLOLC on linear combinations, making clear the trade-offs between them.

Other non-blockchain-based systems with auditing capabilities. Recently, a line of works explores the area of authenticated data structures. Transparency logs are a prominent example when considering auditing. In these works, the goal of an auditor is to examine digests published by untrusted servers to avoid server equivocation (e.g., Merkle[2] [HHK+21] and CONIKS [MBB+15]). Other works revolve around ensuring token liability when transacting across a system. Specifically, the authors of [JC21, RMY+23] approach this problem by requiring entities to publish attestations on their total liabilities e.g., on public bulletin boards. Their goal is to safeguard against data-leaking attacks while allowing auditors to verify the validity of statements regarding liabilities. While these works examine auditability, the view on auditing is through a different lens. First, our system design is more complex, auditors examine organizations and then a committee verifies the auditor-generated results. Additionally, our protocols propose an auditing process that revolves around detecting individual transaction (in)consistency, a considerably more challenging task than, for example, proving the sum of an organization's assets.

3 Preliminaries

General Notation. Let $\mathbb{E}$ be an elliptic curve defined over a large prime field $\mathbb{F}_p$ with $G, H \in \mathbb{E}$ as publicly known generators. We denote by $x \leftarrow\!\!\$\ A$ the random sampling of the element x from the domain A, and the set $\{1, \cdots, n\}$ by $[n]$. We denote by λ a security parameter and by $\mathsf{negl}[\lambda]$ a negligible function in λ.

Commitment Schemes [KP23]. A commitment scheme consists of a pair of PPT algorithms (Com.Setup, Com.Commit). The Com.Setup algorithm generates public parameters pp for the scheme, for security parameter λ: $\mathsf{pp} \leftarrow \mathsf{Com.Setup}(1^\lambda)$. The commitment algorithm defines a function $\mathcal{M}_{\mathsf{pp}} \times \mathsf{R}_{\mathsf{pp}} \rightarrow \mathcal{C}_{\mathsf{pp}}$, for message space $\mathcal{M}_{\mathsf{pp}}$, for randomness space R_{pp}, and for commitment space $\mathcal{C}_{\mathsf{pp}}$, determined by pp. It takes as input a message x and randomness r and outputs $c \leftarrow \mathsf{Com.Commit}_{\mathsf{pp}}(\mathsf{x}; r)$. Specifically, a Pedersen commitment of x with randomness r is in the form of $\mathsf{cm}(x, r) = x \cdot G + r \cdot H$. Pedersen commitments are *additively homomorphic*, i.e., $\mathsf{cm}(x_1, r_1) + \mathsf{cm}(x_2, r_2) = \mathsf{cm}(x_1 + x_2, r_1 + r_2)$, *computationally binding* (after committing it is not feasible to "change one's mind"), and *perfectly hiding* (they reveal nothing about the committed data). A computationally binding and perfectly hiding commitment scheme must satisfy the following properties:

- **Computationally Binding:** It is not easy to find two strings x_0 and x_1 that map to the same commitment. More formally, if $\mathsf{cm}^0 \leftarrow \mathsf{Com.Commit}(x_0; r_0)$:

$$\Pr[\mathsf{Adv}(\mathsf{cm}^0) \rightarrow (x_1, r_1) : \mathsf{cm}^0 = \mathsf{Com.Commit}(x_1; r_1)] \leq \mathsf{negl}[\lambda].$$

- **Perfectly Hiding:** It is not easy to identify which value was used in the generation of a commitment cm←Com.Commit($\cdot, \cdot$). Formally, $\forall$ x_0, x_1 (of the same length), for all *non-uniform* PPT adversaries Adv, we have that:

$$|\Pr[\mathsf{Adv}(\mathsf{Com.Commit}(x_0; r_0)) = 1] - \Pr[\mathsf{Adv}(\mathsf{Com.Commit}(x_1; r_1)) = 1]| = 0.$$

Hash Function [Dam89]. A cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is pre-image resistant if the probability of reversing the hash output to obtain the underlying pre-image is negligible: $\Pr[\mathsf{Adv}(y) \to x | y = \mathcal{H}(x)] \le \mathsf{negl}[\lambda]$.

Public-Key Encryption (PKE) Scheme. A PKE scheme $\mathcal{E}$ consists of a tuple of algorithms PKE.KeyGen,PKE.Enc,PKE.Dec. Specifically the elliptic curve El-Gamal encryption scheme [Kob87] is as follows:

- KeyGen(λ) $\to (sk, pk)$. Given the security parameter λ, KeyGen samples a secret key $sk \leftarrow\!\!\$ \{0,1\}^\lambda$, computes the public key $pk = sk \cdot G$. It outputs the key-pair (sk, pk).
- Enc($pk, x; r$) $\to ct_x$. To encrypt a value x, the algorithm takes input a randomness r and outputs the curve point $(r \cdot G, P_x + \cdot r(sk \cdot G))$. Here, P_x is a publicly-known mapping of a value x to a curve point in $\mathbb{E}$.
- Dec(ct_x, sk) $\to x$. To decrypt x from $E(pk, x; r)$, the algorithms computes $x := P_x + \cdot r(sk \cdot G) - r \cdot sk \cdot G$.

Regarding security, we say a PKE scheme is IND-CPA secure whenever an adversary Adv plays an indistinguishability game with a challenger $\mathcal{C}$ where the former has encryption oracle access and at some point gives a left-or-right challenge (x^0, x^1) to $\mathcal{C}$, who depending on a bit $b \leftarrow\!\!\$ \{0,1\}$ which it had picked during setup returns $ct^b \leftarrow \mathsf{PKE.Enc}(x^b)$. Finally, Adv outputs a guess b' on b and wins if $\Pr[b = b'] \ge \frac{1}{2} + \mathsf{negl}[\lambda]$.

Merkle Tree [Mer87]. A Merkle tree (MT) is a binary tree whose leaf nodes can store any information and each parent node being calculated as the hash of its children. The time complexity and the space complexity to find a data entry in a MT with n entries, given its path, is $O(\log n)$. MTs support *membership* proofs $\pi_m \leftarrow \mathsf{ProofExists}(root_{MT}, x)$, proving the inclusion of a leaf node x to a tree whose root is denoted by $root_{MT}$ and its root-to-leaf-path by $path_{leaf}$. $path_{leaf}$ comprises the siblings of the nodes while traversing from the leaf to $root_{MT}$. Additionally, there exists a leaf-inclusion verification algorithm $0/1 \leftarrow \mathsf{VerifyExists}(root_{MT}, x, \pi_m)$ that enables a verifier to check whether a value x resides in the merkle tree MT, given a proof π_m and the root $root_{MT}$.

Blockchain & Smart contracts. A blockchain is a peer-distributed ledger made secure through cryptography and incentives. Peers using consensus mechanisms agree upon which information to store in blocks. Blockchain technology has found other uses apart from cryptocurrency applications, especially via using smart contracts [Woo14]. A smart contract is a computer program that can be run in an on-chain manner, has internal states and its own on-chain storage. Uploading data on a smart contract method requires time to be verified and agreed upon by the blockchain peers. Contrary, reading data from the contract is almost instantaneous and does not require any form of consensus.

4 Problem Formulation

Below, we formalize the problem we solve in this work, including the system and threat model revolving around it, and provide corresponding security goals.

4.1 System model

Our model includes three entity types, namely organizations, auditors, and a committee. Organizations transact with each other and are responsible to maintain a copy of each and every of their bipartite O2O transactions in a private ledger, to which only their auditor and the committee are privy to (if needed). Auditors examine the transaction records of their clients and are responsible to extract *results* about their individual economical activity and share them with the committee. The committee can access organization records and is responsible to verify auditor-provided results.

4.2 Threat model

First, we assume each organization to be audited by a single auditor, which is in line with current practices[8]. Notably, even in cases where the same auditing entity e.g., one of the Big Four (KPMG, EY, PwC, Delloite) audits different organizations, this does not translate to each individual auditor having access to all organization data of each of the company's client. Additionally, we assume no inter-organization or inter-auditor collusions. Such cases are outside our design rational since they are impossible to detect macroscopically as described in our system model. E.g., consider a scenario where two organizations transact with each other but do not log this transaction in either of their ledgers. To identify such misreported transactions, their auditors or/and the committee need to perform on-site auditing and compare tangible assets (e.g., monetary or other asset reserves with the reported ones). While this on-site part is integral in the auditing process as a whole, *we solely focus on the consistency across all organization-reported data*. Other than that, we consider all entities in our system to be malicious, except for the committee, which is trusted. Specifically, on top of misreporting their transaction activity while trying to avoid detection, organizations wish to extract information as to the client-relation between other organizations and auditors, infer amounts of any organization transactions apart from the ones they are privy to, and check whether cross-ledger transaction consistency holds for other organizations. Adversarial auditors have the same objectives. Such deviant behaviors/attack goals are not only realistic but can also lead to tangible rewards e.g., through insider trading [AKST18].

4.3 Security goals and rationale

Based on the malicious behavior above we determine the security goals that our solutions need to achieve for cross-ledger transaction consistency. First, our solutions need to provide a notion of soundness, in the sense that no organization or auditor should be able to misreport data or results and avoid detection[9], in order to prevent fraudulent asset in/deflation e.g., through the inclusion of sham transactions in their ledger or via generating fake reports [7, 21, 29]. Then, *transaction amount privacy* ensures that only an organization and its auditor should have access to raw transaction data of the former. Financial data is considered to be sensitive and having access unrightfully to an organization's data (e.g., by another organization) may lead to market manipulation via insider trading [14], as mentioned above. Additionally, *transacting organizations unlinkability* guarantees that only the two transacting organizations should have knowledge of the fact that there exists transaction activity between them. Similarly, *organization-auditor unlinkability* ensures that no other system entity can infer any organization-auditor bipartite relation, except the ones they are part of.

Although to the best of our knowledge we are the first to consider the transacting organizations and organization-auditor unlinkability properties, there exist prior works that have built their systems and architectures in such a way that implicitly achieves comparable notions of security. zkLedger [NVV18] and Miniledger [CB21] are perfect examples of this: The respective ledger atop which both these systems are based on essentially satisfies our transacting organizations security property; this can be derived from transaction amount privacy that the hiding property of the commitments provide in the system. As for the auditor-organization unlinkability, existing industry standards already safeguard the confidentiality of the relation between an auditor and its client. Currently, this type of information might be disclosed after the auditor performs all necessary consistency examinations and produces its report, usually after the fiscal year concludes [P716].

Last, only auditors and the committee should be able to verify transaction (in)consistency across ledgers. In the auditors' case crucially, only for organizations their client is transacting with. Being able to infer statistical financial data of organizations may untimely lead to investing behaviors that can affect global markets. Additionally, current techniques require auditors to have access to raw transactional (and proprietary) data of their clients to carry out respective auditing processes, thus organizations carefully pick who they give such data access to. However, giving unrestricted auditing capabilities to other entities apart from the auditors could in fact contribute positively in the financial auditing ecosystem, rendering the financial activities of organizations even more transparent. We estimate that achieving such a notion of "public auditability" needs to be coupled with a strong

---

[8] `https://www.law.cornell.edu/uscode/text/31/9105` [9] Such a property is fairly common in systems like ours and this is why we do not introduce it as a separate property. They are usually achieved through an arbiter, which is the committee in our case.

notion of anonymity as the one presented in [KDP+24]. This is not trivial to achieve, especially considering all the other challenges financial auditing has regarding accountability and traceability of the auditing process, and we leave this as future work.

5 Our Solution

Our system design is epoch-based and includes all three entities (i.e., organizations, auditors, and a committee). We propose two protocols that exist atop a two-tier ledger architecture. In our solutions, we construct this architecture over blockchains, taking advantage of their immutability to avoid attacks like the one presented in Section 1. Each organization maintains a *"local" ledger*, to which we refer to as *localchain*—LocalChain for storing information related to its O2O transactions (txOrg). Additionally, organizations maintain plaintext records of their financial activity offline (e.g., in a database). The committee maintains two *"global" ledgers* in the form of blockchains, to which we refer to as *globalchains*: one "for organizations", namely OrgChain, where organizations upload localchain-related data (dataOrg), and another "for auditors", namely AudChain, who upload report-related data (dataAud). Organizations have access to their off-chain ledger, their LocalChain, and the OrgChain, while auditors have access only to AudChain and their client's LocalChain. Recall that the auditor and the committee may access off-chain records if they ask explicitly to examine them to investigate further potential fraudulent behavior. At a high level, an organization stores hiding versions of its transactions on its LocalChain and aggregated transaction data on OrgChain, while auditors store result-related hiding data on AudChain.

5.1 Design rationale

This architecture enables ① auditors to check the consistency of their clients reported transactional amounts across other ledgers and ② the committee to verify *(i)* the consistency between the data uploaded between an auditor and its client-organization and *(ii)* the consistency between the data uploaded between two auditors whose clients have transacted during the epoch. Figure 2 depicts the architectural model our protocols operate in, the phases with their respective operations, and the transitional triggers across phases. Contrary to prior works based on a single-ledger approach (e.g., [NVV18, CB21, LHNH23]), our two-tier ledger architecture allows organizations to store data only pertaining to their own transactional activity; reducing considerably the individual storage costs.

We refer to our different ledger tiers (local and global) as chains, since *(i)* we need a notion of ledger immutability in our system design and *(ii)* we implement them later as blockchains. However, we stress that these ledgers are not explicitly blockchains and our system is ledger-agnostic at its core. In fact, any immutable ledger may be used in our design. By combining the ledger immutability of our two-tier architecture with cryptographic components, we can guarantee the three security properties mentioned above. More specifically, in both our solutions we employ hiding techniques for dataOrg to ensure transaction amount privacy and ensure that no entity except for the Committee can associate the economic activity of any organization to another. Last, by uploading dataAud in a hiding manner, no other entity except for the committee can infer relations between organizations and auditors.

5.2 Phases

Each epoch is comprised of four phases, namely *(i)* Initialization ($\mathcal{IN}$), *(ii)* Transaction recording ($\mathcal{TR}$), *(iii)* Consistency examination ($\mathcal{CE}$), and *(iv)* Result verification ($\mathcal{RV}$). In the initialization phase the committee performs the necessary setup operations and distributes information to the other entities accordingly. During transaction recording, organizations record data about their O2O transactions to their LocalChain, and the OrgChain. During the consistency examination phase auditors first perform computations on their clients data and extract results which they upload on the AudChain. Afterwards, they compare their results against those from the respective auditors of their clients' transacting counterparts, which are already uploaded on AudChain. Finally, the committee during result verification collects data from OrgChain and AudChain, and conducts checks as to the consistency of the reported/uploaded data. Upon identifying any inconsistencies, the committee may investigate further the ledgers of the suspicious organizations and/or the results produced by their auditors, and potentially assign penalties.
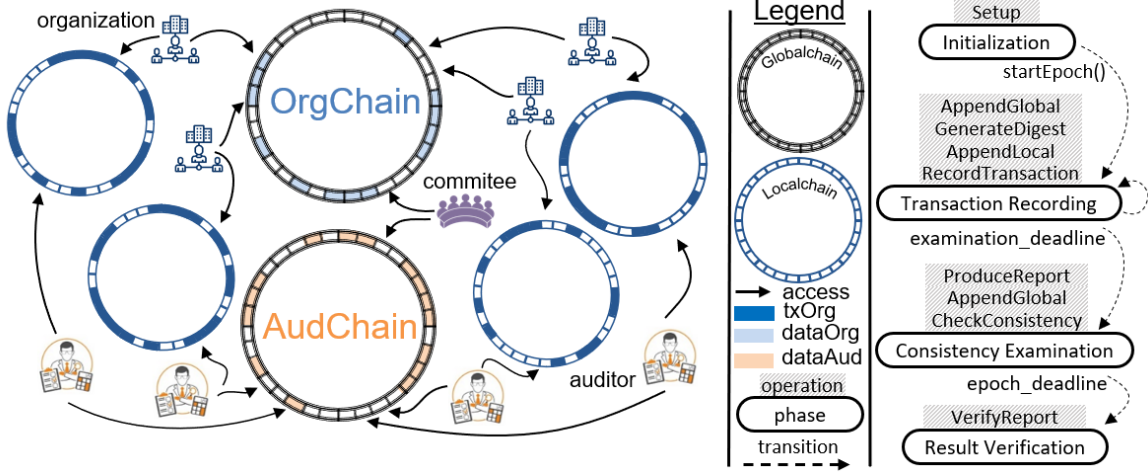
Fig. 2: Our architectural two-tier ledger design and access rights of the entities involved, alongside the protocol phases with respective operations and triggers.

5.3 Protocol preliminaries

Let $[n]$ be the index set of the organizations, m the maximum transactions that can be recorded in an epoch, by a single organization, and $[l]$ the index set of the Auditors. Therefore, let $\mathcal{O}_n = \{o_i\}_{i \in [n]}$ denote the set of organizations, $\mathcal{A}_l = \{a_z\}_{z \in [l]}$ the set of auditors, and Com the committee. Let $\mathscr{L}_i$ denote the respective LocalChain of o_i, $\mathscr{L}_o$ the OrgChain, and $\mathscr{L}_a$ the AudChain. Let Op $= \langle$Setup, RecordTx, AppendLocal, GenerateDigest, CheckConsistency, ProduceReport, AppendGlobal, VerifyReport$\rangle$ be the list of allowed operations. Then, the tuple $\langle \mathcal{O}_i, \mathcal{A}_l, \text{Com}, \{\mathscr{L}_i\}_{i \in [n]}, \mathscr{L}_o, \mathscr{L}_a, \text{Op} \rangle$ can describe fully both our protocols. Below, we explain the operations at a high level and provide in Subsections 5.4 and 5.5 our proposed protocols' specifics. In the detailed description of our protocols, for readability purposes we break down all operations to specific steps.

- Setup($\mathcal{O}_n, \mathcal{A}_l, t$): Com generates the public parameters pp, blinding identifiers $\text{Lid}^{o,b}$ (for organizations) and $\text{Lid}^{a,b}$ (for auditors), and other protocol-specific parameters psp for epoch t.
- RecordTx(Sender,Receiver,Value): o_i invokes this operation to record a transaction dataLocal of Value from a Sender or to a Receiver, where $\{$Sender, Receiver$\} \in \mathcal{O}_n$ and Sender $\neq$ Receiver.
- AppendLocal(tx,$\mathscr{L}_i$): o_i invokes this operation to append a hiding version tx^h of a transaction tx to its ledger $\mathscr{L}_i$.
- GenerateDigest(t,$\mathscr{L}_i$): o_i invokes this operation to generate a "digest" dataOrg of the state of its ledger $\mathscr{L}_i$ for epoch t.
- CheckConsistency($\mathscr{L}_i,\mathscr{L}_a$): a_z checks whether forall transactions $\in \mathscr{L}_i$ there exists a transaction in the ledger of another $o_j \in \mathcal{O}_n$ and produces a Result.
- ProduceReport(t,$\mathscr{L}_i$,dataOrg$_i$): a_i invokes this operation to produce a "report" dataAud about the consistency between the data on its client's ledger $\mathscr{L}_i$, dataLocal$_i$ with dataOrg$_i$ for epoch t.
- AppendGlobal(data*,$\mathscr{L}^\star$): o_i invokes this operation to append data*=dataOrg to $\mathscr{L}^\star = \mathscr{L}_o$ or a_z to append result-related data data*=dataAud to $\mathscr{L}^\star = \mathscr{L}_a$.
- VerifyResult($\mathscr{L}_o,\mathscr{L}_a,(\text{id}_{\alpha_z},\text{dataAud})$): Com invokes this operation to verify the consistency of dataAud (generated from a_z) with $\mathscr{L}_o$ and $\mathscr{L}_a$.

5.4 CLOSC (Cross Ledger cOnsistency with Smart Contracts)

Our first protocol utilizes smart contracts as the name indicates. In more details, in addition to an organization *(i)* maintaining a copy of all its transactions offline in a local ledger/database and *(ii)* computing and uploading a hiding copy of each such transaction to its LocalChain, now it needs to *(iii)* maintain a tree structure MT^t whose leaves correspond to hiding transactions, *(iv)* upload the root of the tree $root_{\text{MT}^t}$ to OrgChain, and *(v)* upload a smart contract on its LocalChain for every transaction. Below we explain both at a high level and explicitly the details of CLOSC.
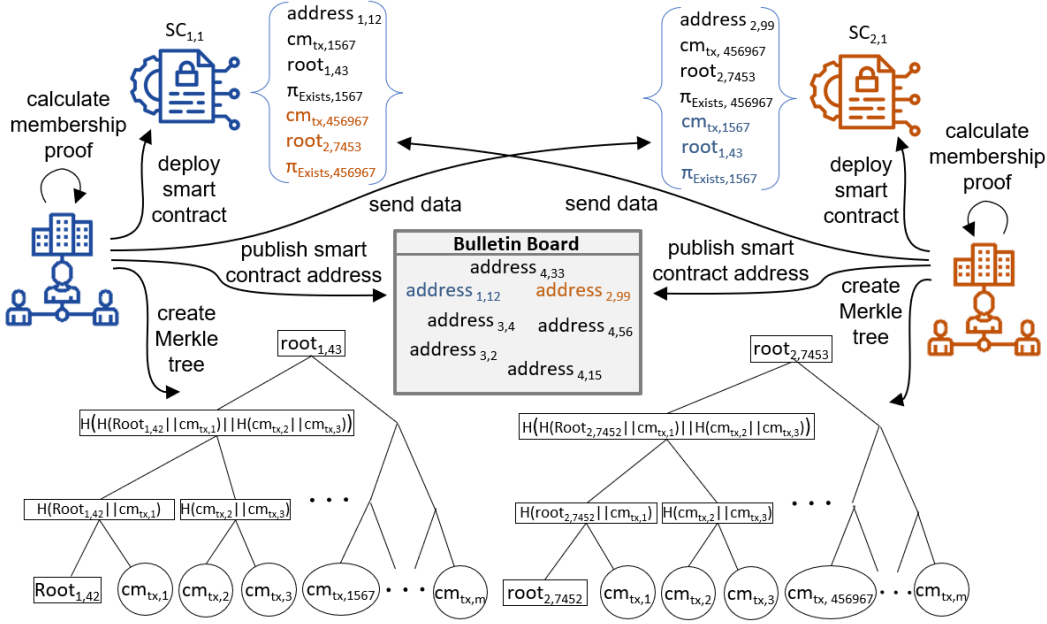
Fig. 3: CLOSC core components and interactions. Each organization stores transactions in a Merkle Tree, then deploys a smart contract where it adds all transaction-related data of a transaction stored as a commitment at the leaf level of the Merkle tree, and uploads the smart contract address to a bulletin board. The other transacting organization performs equivalent steps and uploads their own transaction-related records also to the latter's smart contract.

Let $\mathsf{tx}_{\mathsf{S},\mathsf{R}} = \langle \mathsf{Sender}, \mathsf{Receiver}, \mathsf{Amount}, \mathsf{nonce}, \mathsf{timestamp} \rangle$ describe a transaction tuple, where nonce is an O2O-specific transaction unique random identifier. Then, hiding commitments of the form $\mathsf{cm}_{\mathsf{nonce}}^{\mathsf{timestamp}} = g^{\mathsf{Amount}} \cdot h^{\mathcal{H}(\mathsf{timestamp},\mathsf{nonce})}$, are stored at the MT^t leaf-level, with the sole exception of the utmost left leaf that is equaly to the merkle tree root of the previous epoch. The non-leaf nodes of the tree are calculated as follows: $\mathsf{node}_i = \mathcal{H}(node_{lc}||node_{rc})$; essentially the hash of the concatenation of the node's left and right children (denoted by lc and rc respectively) also depicted in Figure 3. Importantly, for incoming transactions, we consider Amount to be positive, and negative otherwise. Now recall that an auditor should, ideally, be able to verify that: For every organization-related transaction $\mathsf{tx}_{i,j}$ included in its client's ledger $\mathscr{L}_i$ there exists a dual transaction in $\mathscr{L}_j$, crucially, without having access to $\mathscr{L}_j$. We enable auditors to verify this via the following:

When o_j transacts with o_i the latter deploys a smart contract on $\mathscr{L}_i$ and whitelists o_j to be able to submit data to it, and o_j operates similarly. These smart contracts will store the committed transactions alongside the proofs that the financial transactions corresponding to the deal have been included to the respective trees whose roots are uploaded to $\mathscr{L}_o$.

With this design o_i can deploy the contract, communicate its "address" to o_j, upload its data and proof there, and await for its counterpart to upload the other half. The smart contract will store the two instances of $\{\mathsf{cm}_{tx}, root_{\mathsf{MT}^t}, \pi_{\mathsf{Exists}}\}$, with the latter proving that cm_{tx} is in the tree whose root is $root_{\mathsf{MT}^t}$. The only potential problem here is that o_j may never submit its half of the deal and claim not knowing the address. To solve this "plausible deniability" problem, we employ a bulletin board, maintained by the committee. In fact, organizations publish the addresses of their smart contracts there to ensure that any organization can use them. Notably, only the committee can erase information stored on the bulletin board at the end of each epoch to keep storage use low. At the same time this construction ensures that no organization can deny knowing where to upload their data.

Now, only the duality has yet to be shown. For this we require o_i to commit to the amount $x_{i,j,k}$ (for transaction with nonce k) in the form of $g^{x_{i,j,k}}$. In CLOSC, $\mathsf{cm}_{tx} = \mathsf{cm}_{j,i,k}^{\mathsf{timestamp}} = g^{x_{i,j,k}} \cdot h^{\mathcal{H}(\mathsf{timestamp},\mathsf{j},\mathsf{k})}$. To ensure cross-ledger transaction consistency, both auditors have to verify the 2 uploaded proofs stored inside each of their client's smart contracts. We implicitly assume that the two transacting organizations have agreed on a common and unique (timestamp,nonce) pair ahead of time. Last, to assist the committee with the result verification, the auditors utilize a MergeProofs algorithm that

Protocol $(i,j,z \in [n]; k \in [m], epoch = t, timestamp = \mathsf{tstp})$	
$\mathcal{IN}.1)$ Generate $r_{a_{z,i}} \leftarrow\!\!\$\ \mathbb{Z}_p$	*Committee*
$\mathcal{IN}.2)$ Distribute $\mathsf{ID}_A = \{g^{r_{a_{z,i}}}\}_z$	
$\mathcal{TR}.1)$ Store $\mathsf{Ltx}_i = \{\mathsf{Sender}, \mathsf{Receiver}, x_{i,j,k}, k, \mathsf{tstp}\}_{j,k}$ off-chain	
$\mathcal{TR}.2)\ \mathsf{cm}_{j,k}^{\mathsf{tstp}} = g^{x_{i,j,k}} \cdot h^{\mathcal{H}(\mathsf{tstp},j,k)}$	
$\mathcal{TR}.3)$ Upload $\mathsf{Lcm}_i = \{\mathsf{cm}_{j,k}^{\mathsf{tstp}}\}_{j,k}$ to $\mathsf{LocalChain}$	
$\mathcal{TR}.4)$ Create MT_i^t, with $root_{\mathsf{MT}_i^t}$	*Organization*
$\mathcal{TR}.5)$ Upload $root_{\mathsf{MT}_i^t}$ to $\mathsf{OrgChain}$	
$\mathcal{TR}.6)\ \pi_{j,k}^m \leftarrow \mathsf{ProveExists}(x_{i,j,k}, root_{\mathsf{MT}_i^t})$	
$\mathcal{TR}.7)$ Deploy $\mathsf{SC}_{i,j,k}$ on $\mathsf{LocalChain}$	(1$^{\text{st}}$part)
- -	
$\mathcal{TR}.8)$ Store $\{\mathsf{cm}_{j,k}^{\mathsf{tstp}}, root_{\mathsf{MT}_i^t}, \pi_{j,k}^m\}$ in $\mathsf{SC}_{i,j,k}$ and $\mathsf{SC}_{j,i,k}$	(2$^{\text{nd}}$part)
$\mathcal{CE}.1)\ b_{i,j,k} \leftarrow \mathsf{VerifyExists}(\mathsf{cm}_{i,j,k}^{\mathsf{tstp}}, root_{\mathsf{MT}_i^t}, \pi_{i,j,k}^m), \forall \mathsf{SC}_{i,j,k}$	
$\mathcal{CE}.2)\ b_{j,i,k} \leftarrow \mathsf{VerifyExists}(\mathsf{cm}_{j,i,k}^{\mathsf{tstp}}, root_{\mathsf{MT}_i^t}, \pi_{j,i,k}^m)$	
$\mathcal{CE}.3)\ B = \prod_{(j,k)=(1,1)}^{(n,m)} b_{i,j,k} \cdot b_{j,i,k}$, if $B = 0$ alert Com	
$\mathcal{CE}.4)$ Check $\prod_{k=1}^m \mathsf{cm}_{i,j,k}^{\mathsf{tstp}} \cdot \mathsf{cm}_{j,i,k}^{\mathsf{tstp}} \cdot h^{-\mathcal{H}(\mathsf{tstp},j,k)-\mathcal{H}(\mathsf{tstp},i,k)} \overset{?}{=} 1$	
$\mathcal{CE}.5)\ \mathsf{cm}_{i,j}^t = g^{r_{a_{z,i}}} \cdot \prod_{k=1}^m \mathsf{cm}_{i,j,k}^{\mathsf{tstp}}$	*Auditor*
$\mathcal{CE}.6)\ \pi_{i,j}^{m'} \leftarrow \mathsf{MergeProofs}(\{\pi_{i,j,k}^m\}_k), \{\mathcal{H}(\pi_{i,j}^{m'})\}_j$	
$\mathcal{CE}.7)$ Upload $\{\mathsf{cm}_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}$ to $\mathsf{AudChain}$	
$\mathcal{CE}.8)$ Forward $\{\mathsf{Lcm}_{i,j}\}_k = \{\mathsf{cm}_{i,j,k}^{tstp}\}_k$ and $\{\pi_{i,j}^{m'}\}_j$ to Com	
$\mathcal{RV}.1)\ \{\mathsf{cm}_{i,j}^{tstp}\}_k = \mathsf{Lcm}_{i,j}$	*Committee*
$\mathcal{RV}.2)\ b'_{i,j} \leftarrow \mathsf{VerifyExists}(\mathsf{Lcm}_{i,j}, root_{\mathsf{MT}_i^t}, \pi_{i,j}^{m'})$	
$\mathcal{RV}.3)$ Check $B' = \prod_{(i,j)=(1,1)}^{(n,n)} b'_{i,j} \cdot b_{j,i} \overset{?}{=} 0$	
$\mathcal{RV}.4)$ Check $\prod_{i=1}^n \prod_{j=1}^n \mathsf{cm}_{i,j}^t \cdot \mathsf{cm}_{j,i}^t \overset{?}{=} g^{r_{a_{z,i}}} \cdot g^{r_{a_{z,j}}}$	
$\mathsf{LocalChain}$: $\mathsf{Lcm} = \{\mathsf{Lcm}_i\}_{j,k}$, $\mathsf{Lsc} = \{\mathsf{SC}_{i,j,k}\}_{j,k}$ $\mathsf{OrgChain}$: $\{root_{\mathsf{MT}_i^t}\}_i$ $\mathsf{AudChain}$: $\{\mathsf{cm}_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}_{i,j}$	

Fig. 4: CLOSC phase and on-chain storage analysis. Regarding operations, Setup:$\mathcal{IN}$, RecordTx:$\mathcal{TR}.1, 2$, AppendLocal:$\mathcal{TR}.3, 7, 8$, GenerateDigest:$\mathcal{TR}.4, 6$, CheckConsistency:$\mathcal{CE}.1$-6, 8, AppendGlobal:$\mathcal{TR}.5, \mathcal{CE}.7$, VerifyResult:$\mathcal{RV}.1$-4.

combines the individual proofs received from their clients into a single one, that they later on pass to the committee. Figure 4 showcases the protocol details including the on-chain storage, where i, j refer to organizations, k to transaction nonces, and z to auditors.

5.5 CLOLC (Cross Ledger cOnsistency with Linear Combinations)

This construction lies on linear combinations. Specifically, we aim to exploit the fact that it is infeasible to generate two set of values that, combined with a vector of secret-random "weights", results in the same weighted sum evaluation. To this end, the committee during initialization samples random values for each potential transaction to be made within the epoch and shares them to the respective auditors. Contrary to CLOSC, organizations now do not need to maintain a tree structure or deploy smart contracts. Instead, to record their transaction, they need to upload to their LocalChain lists of commitments for each transaction they make during the epoch and the (homomorphic) product of all commitments per transacting organization on the OrgChain. Importantly, each organization "masks" this product by multiplying it with a hash of its own unique identifier, assisting in guaranteeing that no other OrgChain participants can identify transacting organization pairs.

The consistency examination phase is comprised of two parts in CLOLC. First, the auditors calculate a weighted sum of the transaction amounts (at the exponent) with the committee-generated values (during the initialization phase) through homomorphic multiplications. Afterwards, each auditor uploads these products to the AudChain alongside specific identifiers, from where the respective auditor of the transacting organization can retrieve all matching products, using the corresponding common identifiers. Auditors can then use these product pairs to verify the consistency between the reported data from their client organizations and their respective {organization, auditor} pair. Upon identifying an inconsistency, the auditor can approach the committee, who can then investigate fur-

Protocol $(i, j, z \in [n]; k \in [m])$	
$\mathcal{IN}.1)$ $\{r_{i,j,k}\}_{i,j,k} \xleftarrow{\$} \mathbb{Z}_p,\ r_{i,j,k} = r_{j,i,k}$	*Committee*
$\mathcal{IN}.2)$ $\{id_i\}_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_p,\ \{id_z\}_{z\in[l]} \xleftarrow{\$} \mathbb{Z}_p$	
$\mathcal{IN}.3)$ $sk_{\mathsf{com}}, \{sk_i\}_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_p,\ pk_{\mathsf{com}} = g^{sk_{\mathsf{com}}}, pk_i = g^{sk_i}$	
$\mathcal{IN}.4)$ Publish $pk_{\mathsf{com}}, \{pk_i\}_i$	
$\mathcal{IN}.5)$ Forward $\{r_{i,j,k}\}_{i,j,k}, \{sk_i\}_i$ to Auditor z	
$\mathcal{TR}.1)$ $\mathsf{com}(x_{i,j,k}) = g^{x_{i,j,k}} \cdot h^{\rho_{i,j,k}},\ \{\rho_{i,j,k}\}_{i,j,k} \xleftarrow{\$} \mathbb{Z}_p$	*Organization*
$\mathcal{TR}.2)$ $A_{i,j} = \prod_{k=1}^{m} \mathsf{com}(x_{i,j,k}) = g^{\sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k}}$	
$\mathcal{TR}.3)$ Upload $\mathsf{Ltx}_{i,j} = \{\mathsf{com}(x_{i,j,k})\}_k$ to $\mathtt{LocalChain}$	
$\mathcal{TR}.4)$ Upload $\{\mathcal{H}(id_i) \cdot A_{i,j}\}_j$ to $\mathtt{OrgChain}$	
$\mathcal{CE}.1)$ Read from $\mathtt{LocalChain}$ $\mathsf{Ltx}_{i,j}$	
$\mathcal{CE}.2)$ $\{\mathsf{res}_{i,j}\}_j = \left\{ \prod_{k=1}^{m} \left(\mathsf{com}(x_{i,j,k})^{r_{i,j,k}}\right) \right\}_j = \left\{ g^{\sum_{k=1}^{m} x_{i,j,k} \cdot r_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k}} \right\}_j$	
$\mathcal{CE}.3)$ $B_{i,j} = h^{r_{i,j,f}},\ r_{i,j,f} = -\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k}$	
$\mathcal{CE}.4)$ Read from $\mathtt{OrgChain}$ $\{Y_{i,j} = \mathcal{H}(id_i) \cdot A_{i,j}\}_j$	
$\mathcal{CE}.5)$ $\{A_{i,j} = Y_{i,j} \cdot \mathcal{H}(id_i)^{-1}\}_j$ and $C_{i,j} = \mathsf{res}_{i,j}^{-1} \cdot A_{i,j}$	*Auditor*
$\mathcal{CE}.6)$ $\{D_{i,j}\}_j = \{g^{-\sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^{m} \rho_{i,j,k}}\}_j$	(1^{st}part)
$\mathcal{CE}.7)$ $ct(\mathsf{res}_{i,j}) \leftarrow \mathsf{Enc}(pk_j, \mathsf{res}_{i,j}),\ ct(B_{i,j}) \leftarrow \mathsf{Enc}(pk_j, h^{r_{i,j,f}})$	
$\mathcal{CE}.8)$ $ct(C_{i,j}) \leftarrow \mathsf{Enc}(pk_{\mathsf{com}}, \mathsf{res}_{i,j} \cdot A_{i,j}),\ ct(\mathcal{H}(id_z) \cdot D_{i,j}) \leftarrow \mathsf{Enc}(pk_{\mathsf{com}}, \mathcal{H}(id_z) \cdot D_{i,j})$	
$\mathcal{CE}.9)$ Upload to $\mathtt{AudChain}$: $\mathsf{Lres}_i = \left\{ \mathcal{H}(id_i \| \sum_{k=1}^{m} r_{i,j,k}), ct(\mathsf{res}_{i,j}), ct(B_{i,j}), ct(C_{i,j}), ct(\mathcal{H}(id_z) \cdot D_{i,j}) \right\}_j$	
- -	
$\mathcal{CE}.10)$ $\mathcal{H}(id_i \| \sum_{k=1}^{m} r_{i,j,k}), \forall j \in [n]$	(2^{nd}part)
$\mathcal{CE}.11)$ Retrieve and decrypt from $\mathtt{AudChain}$ $\{\mathsf{res}'_{i,j}, B'_{i,j}\}_j$	
$\mathcal{CE}.12)$ Check $\mathsf{res}'_{i,j} \cdot B'_{i,j}, \cdot \mathsf{res}_{i,j} \cdot B_{i,j} \stackrel{?}{=} 1, \forall j \in [n]$	
$\mathcal{RV}.1)$ Read $\{\mathcal{H}(id_i) \cdot A_{i,j}\}_j$ from $\mathtt{OrgChain}$	*Committee*
$\mathcal{RV}.2)$ Read from $\mathtt{AudChain}$ and decrypt $\left\{ ct(B_{i,j}), ct(C_{i,j}), ct(\mathcal{H}(id_z) \cdot D_{i,j}) \right\}_{i,j}$	
$\mathcal{RV}.3)$ Check $\forall (i,z)$: $\mathcal{H}(id_i) \cdot A_{i,j} \cdot B_{i,j} \cdot \mathcal{H}(id_z) \cdot D_{i,j} \stackrel{?}{=} \mathcal{H}(id_i) \cdot \mathcal{H}(id_z)$	
$\mathcal{RV}.4)$ Check $\forall (z,z')$: $\mathcal{H}(id_z) \cdot D_{i,j} \cdot C_{i,j} \cdot \mathcal{H}(id_{z'}) \cdot D'_{i,j} \cdot C'_{i,j} \stackrel{?}{=} \mathcal{H}(id_z) \cdot \mathcal{H}(id_{z'})$	

$\mathtt{Localchain}$: $\mathsf{Ltx}_{i,j} = \left\{\mathsf{com}(x_{i,j,k})\right\}_k = \{g^{x_{i,j,k}} \cdot h^{\rho_{i,j,k}}\}_{i,j,k}$

$\mathtt{OrgChain}$: $\left\{ \left\{ \mathcal{H}(id_i) \cdot g^{\sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k}} \right\}_j \right\}_i$

$\mathtt{AudChain}$: $\mathsf{Lres}_{i,j} = \left\{ \left\{ \mathcal{H}(id_i \cdot \sum_{k=1}^{m} r_{i,j,k}), ct_{pk_j}(\mathsf{res}_{i,j}), ct_{pk_j}(h^{r_{i,j,f}}) \right\}_j \right\}_i$

$\qquad \left\{ \mathcal{H}(id_z) \cdot h^{\sum_{k=1}^{m} -\rho_{i,j,k} \cdot r_{i,j,k}}, g^{-\sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^{m} \rho_{i,j,k}} \right\}_{(i,j)}$

$\qquad \left\{ g^{\sum_{k=1}^{m} x_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^{m} \rho_{i,j,k}} \right\}_{(i,j)}$

$\qquad \left\{ g^{-\sum_{k=1}^{m} x_{i,j,k}} \cdot h^{\sum_{k=1}^{m} \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^{m} \rho_{i,j,k}} \right\}_{(i,j)}$

Fig. 5: $\mathtt{CLOLC}$ phase and on-chain storage analysis. X' denotes a retrieved/decrypted value X, generated by other auditors or organizations. Regarding operations, Setup:$\mathcal{IN}$, RecordTx:$\mathcal{TR}.1$, AppendLocal:$\mathcal{TR}.3$, GenerateDigest:$\mathcal{TR}.2$, CheckConsistency:$\mathcal{CE}.1\text{-}7, 9, 10$, AppendGlobal:$\mathcal{TR}.4, \mathcal{CE}.8$, VerifyResult:$\mathcal{RV}.1\text{-}4$.

ther. Having the auditors upload their results in a hiding manner assists in guaranteeing that no other auditor can "mix and match" reports on $\mathtt{AudChain}$ to identify organization-auditor pairs.

The committee verifies results by accessing only $\mathtt{OrgChain}$ and $\mathtt{AudChain}$, meaning it requires no access to any of the underlying localchain commitments, unless it specifically asks for them, e.g., to cross-check any data regarding an auditor-reported potential inconsistency. We enable the committee to carry out result verification checks by ensuring that *(i)* the data organizations upload to $\mathtt{OrgChain}$ are consistent with the data their auditor uploads to $\mathtt{AudChain}$ and *(ii)* the data auditors upload to $\mathtt{AudChain}$ are consistent with each other. Importantly, we tie each auditor-generated commitment sum with the unique identifier of said auditor so as to avoid possible brute-force matching attacks, since these commitments ($\mathsf{cm}_{i,j}^t$) are uploaded on $\mathtt{AudChain}$. Importantly, the linear combination with random values is what on one side assists the auditors in their duties but at the same time hides the transactional amounts from other participants. The infeasibility of reversing these linear combinations resides at the heart of this protocol, which has been used previously in other application contexts as well e.g., for oblivious linear-function evaluation [BCGI18], or function secret sharing [BGI15]. Figure 5 showcases the $\mathtt{CLOLC}$ protocol phases in detail alongside which elements are stored on-chain.

5.6 Protocol Considerations

Since our system model includes multiple types of entities, we need to examine the robustness of our proposed approaches against a set of variables. While we present our solutions above assuming all organizations participate in our system honestly and in their entirety, there are multiple real-world scenarios where this is not the case and below we analyze such cases. Specifically, below we focus on ① the synchronization requirements of our protocols regarding their epochs and ② the relaxation of the percentage of participating organizations in our proposed solutions. Regarding malicious behaviors we provide the security analysis in, the following, Section 6.

① A valid concern in systems operating in epochs and involving numerous organizations and auditors revolves around the strictness of the respective deadlines and how are errors in synchronization handled. We note that our two protocols have different behaviors regarding epoch deadlines and synchronization. In CLOSC, for a pair of auditors to be able to check the consistency of records across the reported data from their two client-organizations, the only requirement is that both Merkle trees need to be finalized and their corresponding roots uploaded on the OrgChain. This is in line with current practices where organizations need to submit their reports by the end of the fiscal year in their respective environment they operate in. In cases of differences in these deadlines the checks can be carried out at the latest deadline available, however, we argue that this is not a considerable caveat, especially considering the time-efficiency of the consistency examination and result verification phases in CLOSC. Contrary, in CLOLC the only requirement revolves around a mutually agreed-upon deadline between each two transacting organizations. For example, the auditor of organization A may be able to execute the consistency examination for organization B at time t_{AB}, and for organization C at time t_{AC}, with $t_{AB} \neq t_{AC}$. The same restrictions apply for the committee as well; however, both auditors now need to have performed their examinations before a specific deadline t'.

② Another concern of large scale systems, like the one we consider, has to do with tolerating entities dropping in and out of the protocols across epochs. Regarding the functionality of the auditing process when a subset of all system-wide organizations does not participate, specifically in our proposed solutions, we make an important observation. First, neither CLOSC nor CLOLC have an "all or nothing" requirement, meaning that the crucial phases of consistency examination and result verification can essentially be executed correctly regardless of how many organizations choose to record their transactions in our proposed manners. The only obvious relaxation in such cases, however, is that transaction consistency and result verification can be conducted only amongst the organizations that opt to participate in all phases of our protocols. This indicates that auditors and the committee can utilize a hybrid approach: First, they can utilize both our protocols for all organization-transacting pairs participating in them. Furthermore, they can use traditional methods that, however, currently do not provide our proposed security properties (which are outlined below).

6 Security Analysis

We aim to design security properties that safeguard our protocols against the various adversarial behaviours of the entities involved in our system design. To this end we introduce three security game-based definitions, as described at a high level in 4.3 and provide proofs about how our constructions satisfy them under various security assumptions regarding the underlying cryptographic components of our designs. We remind the reader here that, depending on the property, we consider organizations and auditors to assume the role of the adversary, who may also choose to corrupt a set of organizations and auditors. All our properties have game-based definitions and our first one revolves around *transaction amount privacy*. The adversary in this game should not be able to distinguish between two transaction vectors between two organizations that are not corrupted. Next, we present *organization-auditor unlinkability*. The adversary here selects two distinct non-corrupted organizations and auditors and the challenger "matches" them arbitrarily. The adversary should not be able to distinguish which organization is paired with which auditor. Last, we describe the notion of *transacting organizations unlinkability*. Similarly to the previous property, the adversary selects four distinct organizations and the challenger pairs them arbitrarily. The goal of the adversary is to identify these pairs. Notably, in all above properties the unlinkability or the privacy hold regarding external parties. Recall that the committee may have plaintext access upon submitting such a request, and each auditor has access to all localchain data of its client. To the best of our knowledge, we are the first to define formally system-wide security properties for financial auditing processes.

Definition 1 (Transaction Amount Privacy). *Let $\mathcal{G}_{Priv}$ be the following game between PPT adversary $\mathcal{A}$ and challenger $\mathcal{C}$:*

- Initialization*: $\mathcal{A}$ specifies parameters n, l, λ. The challenger $\mathcal{C}$ runs the initialization phase to register n organizations and l auditors, such that the organization set for $\mathcal{G}_{Priv}$ for epoch t is $\mathcal{O}_t = \{o_1, \cdots, o_n\}$, the auditor set is $\mathcal{A}_t = \{a_1, \cdots, a_l\}$, and samples a bit $b \leftarrow\!\!\$\ \{0,1\}$.*
- Corruption queries*: When Adv issues such a query, it specifies a set of organizations $\mathcal{O}_c \subseteq \mathcal{O}_t$ and a set of auditors $\mathcal{A}_c \subseteq \mathcal{A}_t$, and $\mathcal{C}$ $\forall o_i \in \mathcal{O}_c$ and $\forall a_i \in \mathcal{A}_c$ provides all respective private information to Adv.*
- Challenge*: Adv picks two organizations $o_{i_1}, o_{i_2} \notin \mathcal{O}_c$, the total number of transactions z for the epoch between them, and provides $\mathcal{C}$ with two transaction vectors $(\mathbf{tx^0}, \mathbf{tx^1})$, where $\mathbf{tx}^0 = \{x^0_{i_1,i_2,z}\}_{z \in [k]}$ and $\mathbf{tx}^1 = \{x^1_{i_1,i_2,z}\}_{z \in [k]}$, and $\sum x^0_{i_1,i_2,k} = \sum x^1_{i_1,i_2,k}$. $\mathcal{C}$ executes the epoch using the transaction vector $\mathbf{tx}^b$ for the (o_{i_1}, o_{i_2}) pair and $\forall o_i \notin \mathcal{O}_c + \{o_{i_1}, o_{i_2}\}$, $\forall a_i \notin \mathcal{A}_c$, it samples random amounts and computes all necessary encryptions, commitments, and hashes using the information for all participating organizations. Then, $\mathcal{C}$ and forwards all information to be uploaded on `OrgChain` and `AudChain` to Adv.*
- Finalization*: Adv sends $b' \in \{0,1\}$ to $\mathcal{C}$.*

Adv *wins in the game if $b' = b$ and we remark that a naive adversary, by sampling randomly b' has probability of winning equal to $\frac{1}{2}$. We denote the advantage that Adv has of winning as $Adv^{Priv}(\mathsf{Adv})$ and we say a protocol Π has transaction amount privacy if for any PPT Adv, $Adv^{Priv}(\mathsf{Adv}) = |Pr[b' = 1 | b = 1] - Pr[b' = 1 | b = 0]| \leq \mathsf{negl}[\lambda]$.*

Definition 2 (Organization-Auditor Unlinkability). *Let $\mathcal{G}_{OAU}$ be the next game between PPT adversary $\mathcal{A}$ and challenger $\mathcal{C}$:*

- Initialization*: $\mathcal{A}$ specifies parameters n, l, λ. The challenger $\mathcal{C}$ samples a bit $b \leftarrow\!\!\$\ \{0,1\}$ and runs the initialization phase to register n organizations and l auditors, such that the organization set for $\mathcal{G}_{OAU}$ for epoch t is $\mathcal{O}_t = \{o_1, \cdots, o_n\}$ and the auditor set is $\mathcal{A}_t = \{a_1, \cdots, a_l\}$.*
- Corruption queries*: When Adv issues such a query, it specifies a set of organizations $\mathcal{O}_c \subseteq \mathcal{O}_t$ and a set of auditors $\mathcal{A}_c \subseteq \mathcal{A}_t$, and $\mathcal{C}$ $\forall o_i \in \mathcal{O}_c$ and $\forall a_i \in \mathcal{A}_c$ provides all respective private information to Adv.*
- Challenge: Adv *selects two organizations $o_{i_0}, o_{i_1} \notin \mathcal{O}_c$, and two auditors $a_0, a_1 \notin \mathcal{A}_c$. $\mathcal{C}$ "matches" o_i with $a_{|b-i|}$, $i \in \{0,1\}$. The challenger $\forall o_i \notin \mathcal{O}_c$ samples random transactions for the in-between them pairs and computes all necessary information. Last, $\mathcal{C}$ forwards all data to be uploaded on the `Orgchain` and `Audchain` to Adv.*
- Finalization*: Adv sends $b' \in \{0,1\}$ to $\mathcal{C}$.*

Adv *wins in the game if $b' = b$ and we remark that a naive adversary, by sampling randomly b' has probability of winning equal to $\frac{1}{2}$. We denote the advantage that Adv has of winning as $Adv^{OAU}(\mathsf{Adv})$ and we say a protocol Π has organization-auditor unlinkability if for any PPT adversary Adv, $Adv^{OAU}(\mathsf{Adv}) = |Pr[b' = 1 | b = 1] - Pr[b' = 1 | b = 0]| \leq \mathsf{negl}[\lambda]$.*

Definition 3 (Transacting Organizations Unlinkability). *Let $\mathcal{G}_{OOU}$ be the next game between PPT adversary $\mathcal{A}$ and challenger $\mathcal{C}$:*

- Initialization*: $\mathcal{A}$ specifies parameters n, l, λ. The challenger $\mathcal{C}$ runs the initialization phase to register n organizations and l auditors, such that the organization set for $\mathcal{G}_{OOU}$ for epoch t is $\mathcal{O}_t = \{o_1, \cdots, o_n\}$ and the auditor set is $\mathcal{A}_t = \{a_1, \cdots, a_l\}$, and samples two bits $b, b^\star \leftarrow\!\!\$\ \{0,1\}$.*
- Corruption queries*: When Adv issues such a query, it specifies a set of organizations $\mathcal{O}_c \subseteq \mathcal{O}_t$ and a set of auditors $\mathcal{A}_c \subseteq \mathcal{A}_t$, and $\mathcal{C}$ $\forall o_i \in \mathcal{O}_c$ and $\forall a_i \in \mathcal{A}_c$ provides all respective private information to Adv.*
- Challenge*: Adv specifies four organizations $o_{i_1}, o_{i_2}, o_{i_3}, o_{i_4} \notin \mathcal{O}_c$. The challenger $\forall o_i \notin \mathcal{O}_c + \{o_{i_1}, o_{i_2}, o_{i_3}, o_{i_4}\}$ samples random transactions for the in-between them pairs and computes all necessary information. It does the same for $(o^\dagger, o^{nc})$ pairs, where $o^\dagger \in \{o_{i_1}, o_{i_2}, o_{i_3}, o_{i_4}\}$ and $o^{nc} \in \mathcal{O}_t - \mathcal{O}_c - \{o_{i_1}, o_{i_2}, o_{i_3}, o_{i_4}\}$. Now if $b = 0$, $\mathcal{C}$ samples and records a list of transaction for the pair (o_{i_1}, o_{i_2}). Otherwise, if $b^\star = 0$ $\mathcal{C}$, samples and records a list of transaction for the pair (o_{i_1}, o_{i_3}), otherwise for the pair (o_{i_1}, o_{i_4}). Last, $\mathcal{C}$ forwards all data to be uploaded on `Orgchain` and `Audchain` to Adv.*
- Finalization*: Adv sends two bits $b', b^{\star'} \in \{0,1\}$ to $\mathcal{C}$.*

14

Adv *wins in the game if* $b' = b$ *and* $b^{\star'} = b^{\star}$ *and we remark that a naive adversary, by sampling randomly* b' *has probability of winning equal to* $\frac{1}{4}$. *We denote the advantage that* Adv *has of winning as* $Adv^{OOU}(\mathsf{Adv})$ *and we say a protocol* Π *has transacting organizations unlinkability if for any PPT adversary* Adv, $Adv^{OOU}(\mathsf{Adv}) = |Pr[b' = 1 \wedge b^{\star'} = 1 | b = 1 \wedge b^{\star} = 1] - Pr[b' = 1 \wedge b^{\star'} = 1 | b = 0 \wedge b^{\star'} = 0]| \leq \mathsf{negl}[\lambda]$.

<u>Remark:</u> We will rely on the hiding property of our proposed commitment scheme to prove that `CLOSC` satisfies all three properties above and following we derive that `cm` is a perfectly hiding commitment scheme in the random oracle model. In fact, assuming that the output distribution of $\mathcal{H}$ is random, which we achieve by modeling $\mathcal{H}$ as a random oracle, `cm` is perfectly hiding as it degenerates to standard Pedersen commitments.

Theorem 1 *Assuming that* `cm` *is a perfectly hiding commitment scheme in the random oracle model (ROM),* `CLOSC` *satisfies Transaction Amount Privacy as per Definition 1.*

Proof. The adversary in $\mathcal{G}_{priv}$ gains information either via corruption or challenge queries. Specifically in `CLOSC`, all b-related information the adversary sees is: *(i)* the Merkle tree root, *(ii)* the list of transaction amount commitments, and *(iii)* the membership proofs; all of which are `cm` commitments or hash outputs. We prove that `CLOSC` satisfies transaction amount privacy through contraposition. Now, we consider an adversary Adv_{priv} that can break the privacy game $\mathcal{G}_{priv}$ and we build an adversary Adv' that breaks the hiding property of our commitment scheme by distinguishing between two commitment vectors with non-negligible probability. Adv' performs the initialization phase of `CLOSC` and when Adv_{priv} issues corruption queries Adv' responds by forwarding all respective private information to Adv_{priv}. Now, when Adv_{priv} provides a challenge query by submitting transaction vectors $\{\mathbf{tx}^0, \mathbf{tx}^1\}$ to Adv', the latter forwards individually the commitments to its challenger, who responds with transaction commitments cm_z^b. Upon collecting all commitments, Adv' forms the committed transaction vector $\mathbf{cm}^b$, generates the Merkle tree with root $root_{MT^t}$, for epoch t, and the respective membership proofs $\{\pi_z^m\}_{z \in [k]}$, and forwards all this information to Adv_{priv}. Assuming Adv_{priv} has non-negligible advantage in winning in $\mathcal{G}_{priv}$, Adv' has equal non-negligible advantage in winning in the commitment hiding game, which contradicts our initial assumption.

Theorem 2 *Assuming* `com` *generates and distributes* ID_A *correctly,* `CLOSC` *satisfies Organization-Auditor Unlinkability as per Definition 2.*

Proof. We observe that the sole element uploaded in either `OrgChain` or `Audchain` that can reveal the client-relation between organizations and auditors hides inside $cm_{i,j}^t$, and specifically in the identifier $g^{r_{\alpha_{z,i}}}$. Now, since the random identifiers ID_A are sampled from scratch every epoch, even if the list of underlying transactions commitments is compromised, no organization or auditor can extract any of the two challenge organization-auditor relation. Importantly, this holds even in case the adversary has corrupted all organizations the challenge organizations are transacting with.

Theorem 3 *Assuming that* `cm` *is a perfectly hiding commitment,* `CLOSC` *satisfies Transacting Organizations Unlinkability as per Definition 3.*

Proof. The "link" between transacting organizations exists in the underlying transacting amounts that are inside the commitments and additionally in the total sum which is committed inside $cm_{i,j}^t$. Similarly to the proof of Theorem 1 no PPT adversary can win in $\mathcal{G}_{OOU}$ unless it can break the hiding property of `cm`.

Theorem 4 *Assuming at* `cm` *is a perfectly hiding commitment scheme and the encryption scheme is IND-CPA secure,* `CLOLC` *satisfies Transaction Amount Privacy as per Definition 1 in the ROM.*

Proof. We prove this property through a series of hybrid indistinguishability games. We start from $\mathcal{G}_{priv}^0$ signifying the privacy game when the challenger bit $b = 0$ and we progress through a series of hybrid games to $\mathcal{G}_{priv}^1$, when $b = 1$. We introduce a hybrid game $\mathcal{H}_{priv}^{k^{\star}}$, where we change the list of committed transaction amount plaintexts from $\{x_1^0, \cdots, x_{k^{\star}-1}^0, x_{k^{\star}}^1, \cdots, x_k^1\}$ to $\{x_1^0, \cdots, x_{k^{\star}}^0, x_{k^{\star}+1}^1, \cdots, x_k^1\}$ and we change the ciphertext generation accordingly. This indistinguishability transformation reduces to the IND-CPA security property of the underlying encryption scheme and via these two transformations we get that $\mathcal{H}_{priv}^{k^{\star}} \approx \mathcal{H}_{priv}^{k^{\star}+1}$, and by making subsequent changes from $\mathcal{G}_{priv}^0 = \mathcal{H}_{priv}^0 \approx \mathcal{H}_{priv}^1 \approx \cdots \approx \mathcal{H}_{priv}^n = \mathcal{G}_{priv}^1$.

Theorem 5 *Assuming that $\mathcal{H}$ is pre-image resistant and the underlying encryption scheme is IND-CPA secure, `CLOLC` satisfies Organization-Auditor Unlinkability as per Definition 2 in the ROM.*

Proof. We observe that all information uploaded on `AudChain` is either ciphertexts or the output of a hash function, modeled as a random oracle. Additionally, on `OrgChain` products of commitments with hash outputs are stored. All this data cannot be used to reveal the client-relation between organizations and auditors unless *(i)* the pre-image resistance of the hash or *(ii)* the security of the underlying scheme is compromised.

Theorem 6 *Assuming that $\mathcal{H}$ is pre-image resistant and cm is a perfectly hiding commitment scheme, `CLOLC` satisfies Transacting Organizations Unlinkability as per Definition 3 in the ROM.*

Proof. We observe that the link between transacting organizations potentially exists in the total sum which is committed inside $A_{i,j}$. Similarly to the proof of Theorem 5 no PPT adversary can win in $\mathcal{G}_{OOU}$ unless it can break first the pre-image resistance property of $\mathcal{H}$ or the hiding property of cm.

7 Implementation & Experimental Evaluation

We develop working prototypes of both our cross-ledger consistency checking protocols and measure their performance against different choice of parameters regarding the number of organizations, auditors, and transactions during an epoch. We implement our two-tier blockchain architecture over Hyperledger Fabric[10] and all off-chain components using Golang 1.20.5. We use the stable 2.5.1 HLFabric version with the Raft consensus algorithm with three orderers for all blockchains. All off-chain computations in `CLOSC` and `CLOLC` are implemented over the Edwards25519 Elliptic Curve (EC), and specifically we utilize the EC Library of[11]. Below we report on the performance of each individual component of our prototype and we have made both the implementation and the experimental evaluation results available online[12]. All off-chain components evaluation and blockchain experiments were conducted on an 8-core AWS EC2 instance (m5n.2xlarge) running Ubuntu 20.04, with 32GB RAM and 50GB storage. For `LocalChain` experiments, we consider having 2 peer nodes (for the organization and the auditor) in the network whereas for `OrgChain` and `AudChain` we utilized 8 peer nodes. All nodes, including peers and orderers, are tested as Docker containers within the same network, on a single EC2 instance. We conduct our experiments for a varying number of organizations and our numbers are taken as a mean of 10 runs with standard deviation of $< 5\%$. Last, we assume that each organization has a unique auditor, which is in fact the worst-case scenario from the implementation perspective. We use a publicly available MT library[13] and to further improve the our protocols' performance we configure our chaincodes to accept batch transaction submitting. Specifically, we bundle $5K$ transactions together and upload them via a single invoke request.
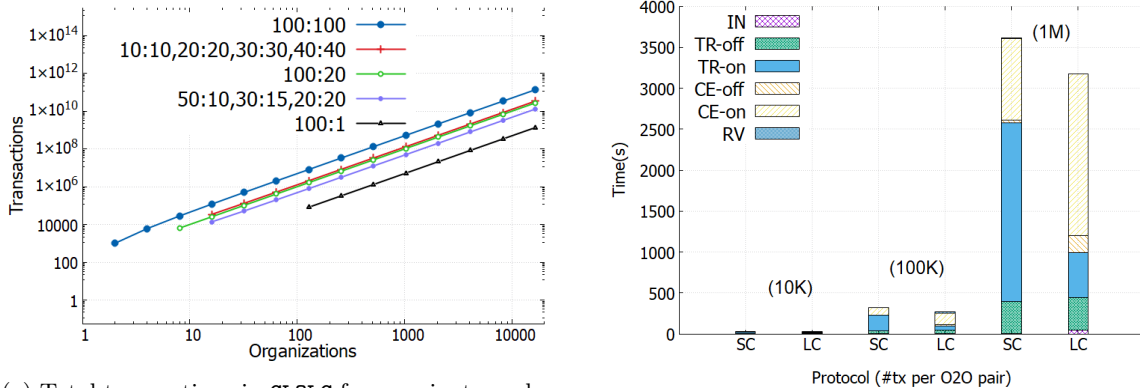
7.1 Performance evaluation of `CLOSC`

We measure all time and space requirements of our proposed solution. Recall that this is a smart contract based approach, where each transaction is stored doubly in two smart contracts in addition to a Merkle tree. Below we analyze the performance of each protocol phase. The initialization takes $\approx 1s$ for 1024 organizations, scaling linearly with their number. During transaction recording, computing 1M transaction commitments takes $\approx 7\text{mins}$ while uploading them to the respective `LocalChain` takes an extra $\approx 8\text{mins}$. The resulting on-chain storage for 1M transaction commitments is 432MB and reading all this data requires $\approx 1\text{min}$. Generating a MT with depth 20 (for $\approx 1\text{M}$ commitments), whose leaves are the commitments above, along with all individual membership proofs takes $\approx 2s$. Last, deploying $\mathsf{SC}_{i,j,k}$ and uploading the triple $\{\mathsf{cm}_{j,k}^{\mathsf{timestamp}}, root_{\mathsf{MT}_i^t}, \pi_{j,k}^m\}$ on it takes $\approx 64s$, and for 1M transactions the total on-chain storage overhead is 515GB per `LocalChain`. Auditors or the committee wishing to read all this data can do so in $\approx 16s$. As for consistency examination, verifying 1M membership proofs and computing $\mathsf{cm}_{i,j}^t$ and $\pi_{i,j}^{m'}$ takes $\approx 28s$, while computing the merged proof takes $\approx 2.1s$. Uploading 256 $\{\mathsf{cm}_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}$ tuples on `Audchain` takes $\approx 2.3s$ and requires 556KB. Finally, the committee can perform the result verification phase for 1M transactions in $\approx 4s$, for a Merkle tree of depth 20.

---

[10] https://www.hyperledger.org/use/fabric   [11] https://github.com/dedis/kyber
[12] https://github.com/ac83ae/auti   [13] https://github.com/txaty/go-merkletree

(a) Total transactions in `CLOLC` for a variant number of organizations and settings ($X : Y$ signifies that $X\%$ of the total organizations are transacting with the $Y\%$ of the rest, with 1024 O2O txs per pair).

(b) E2E time across all different phases for `CLOSC`(SC) and `CLOLC`(LC), for different number of transactions (10K, 100K, and 1M).

Fig. 6: Scalability behavior of `CLOSC` and `CLOLC` in terms of total transactions and time depending on different number of transacting pairs and the amount of their bipartite pairwise transactions.

7.2 Performance evaluation of `CLOLC`

Following, we present the respective time and space requirements of our second solution. Recall that this approach is based on checking the consistency per transacting organization pairs and we make the "worst-case assumption" that all organizations will transact with all others. In such a case we report that for 256 organizations the initialization phase takes $\approx 154s$, for 1024 maximum transactions per organization pair. We observe that in `CLOLC` each organization's computational expenses grow linearly with the number of its total transactions and thus, for example, a case where 2 organizations sign 100 transactions with each other is equivalent to a case where 4 organizations sign 25 transactions with each other, from a system design perspective. Notably, this holds especially during the initialization phase, where the committee needs to generate a unique identifier for each different transaction the system can support. Therefore, we pick several different settings where, instead of assuming solely that all organizations transact with each other, we explore other, more realistic cases. Figure 6a depicts the total number of transactions per number of organizations in different settings `CLOLC` can support per epoch, with 1024 O2O transactions per organization pair. Before going any further, we remark here that we do not provide a similar analysis for `CLOSC`, since its cost of $\mathcal{IN}$ is not affected by such metrics.

At a high level, we chose the following 5 representative settings: *(i)* each organization transacts with all others, *(ii)* organizations are operating in fully connected transaction pair islands, *(iii)* each organization transacts with the same (smaller) portion of the others, *(iv)* fewer organizations transact with more of the rest, while most organizations transact with a small portion of the total organizations, and *(v)* each organization transact with the same (small) portion of the rest. Importantly, we observe that the initialization costs scale linearly with the number of maximum transactions, which results to the following: Having 256 organizations, each conducting 1024 transactions with all other organizations ($\approx 33,5M$ txs in total) is approximately equivalent to having a setting with 100K organizations, where on average each organization transacts with 20 others and sign a total of 33 txs over the epoch. This is a rather realistic scenario, especially when epochs have short duration e.g., a week or a month. Next, transaction recording has both off-chain and on-chain parts. Generating $1M$ committed transactions take $\approx 402s$, while accumulating them in $A_{i,j}$ takes $\approx 1s$. Uploading this list of committed transactions on `LocalChain` takes $\approx 544s$ and requires 767MB, while fetching it takes $\approx 28s$. As for `OrgChain`, things are similar. Uploading $1M$ accumulated values takes $\approx 835s$ and 436MB, while reading all these values takes $\approx 36s$. During consistency examination, generating combined all $\{res_{i,j}, B_{i,j}, C_{i,j}, D_{i,j}\}$ takes ≈ 690ms and encrypting ≈ 1.4ms, for 1024 underlying transactions. Uploading $1M$ such encrypted tuples takes ≈ 39mins and 2GB, while retrieving, decrypting and checking the consistency takes $\approx 0.5s$ per transacting O2O pair. Last, for result verification, `Com` conducts both checks in $\approx 2s$.

17

Protocol	$\mathcal{IN}$	$\mathcal{TR}$	$\mathcal{CE}$	$\mathcal{RV}$
CLOSC	52ms	103.2s + _299.8s_	8.7s + _212.2s_	3.6s
CLOLC	153.6s	105.5s + _134.9s_	12.8s + _7.6s_	12.3s + _8.2s_

(a) Computational times per phase per organization and epoch.

Protocol	LocalChain	OrgChain	AudChain
CLOSC	704MB	543KB	42.6MB
CLOLC	201MB	28.9MB	134MB

(b) Blockchain storage needs per organization and epoch.

Table 1: Computational times (off-chain + _on-chain_) and blockchain storage needs of CLOSC and CLOLC per epoch for 256 organizations, 1024 max transactions per epoch and organization pair, for a total of $33,423,360$ max total transactions per epoch.

Protocol	$\mathcal{IN}$	$\mathcal{TR}$	$\mathcal{CE}$	$\mathcal{RV}$	Storage
ZkLedger [NVV18]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(n^2m)$
Miniledger [CB21]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(n^2m)^\star$
ACA [LHNH23]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(nm)$
CLOSC	$\mathcal{O}(n^2)$	$\mathcal{O}(nm + m log m)$	$\mathcal{O}(nm)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2m)$
CLOLC	$\mathcal{O}(n^2m)$	$\mathcal{O}(nm)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 + nm)$

Table 2: Protocol phase complexity and ledger storage asymptotics of CLOSC and CLOLC against other works with similar potential capabilities across comparable phases of our protocols. n and m respectively denote the total number of organizations and transactions per auditing epoch.
$\star$: Miniledger has a pruning technique to regularly minimize the size of the ledger.

7.3 Comparing the two protocols

Tables 1a & 1b, show that overall there is no clear "better" solution and we observe a trade-off in our designs as for the required time, communication, and on-chain storage. CLOSC has a faster initialization phase, requiring also less communication. Contrary, during transaction recording CLOLC outperforms significantly CLOSC, as the latter requires a separate smart contract per recorded transaction. The performance is reversed once again during consistency examination. As CLOLC splits this phase into two parts, it takes more time to produce the auditor results including all auxiliary elements the committee needs for the next phase, however, in CLOSC the auditor forwards the vector of transactions to the committee. Last, during result verification both our protocols are very efficient allowing our designs to scale regardless of the number of organizations, auditors, or total transactions. In Figure 6b we depict the performance of CLOSC and CLOLC across all phases, for different amount of supported transactions per epoch. The performance of both our protocols is linearly affected by the number of total transactions per auditing epoch. This can offer a point of flexibility depending on the number of organizations and the frequency of their transaction rates and their auditors' consistency examination responsibilities. The time required is calculated as the necessary time for one of each entities to execute all operations during each phase (e.g., how long it take for an organization to perform the transaction recording phase). Last, the storage required for both our solutions is not prohibitive, considering also that after the auditing epoch ends there is no need to keep all the uploaded data in any data structure as a whole. For historicity, maintaining a small digest (e.g., a hash output) of the ledgers might suffice to enable post-hoc verification.

7.4 Comparing with other works

To the best of our knowledge there exists no other implemented system that concretely supports auditing protocols for cross-ledger transactions consistency. Nevertheless, there exist approaches that

could serve similar purposes if adjusted appropriately. E.g., the authors of [NVV18, CB21, LHNH23] designed systems close to our design in some aspects regarding auditability/verifiability but they are based on different setups, including solely users and auditors. We provide a comparative analysis in terms of computation and storage asymptotics between our protocols and these works for a single epoch in Table 2.

Notably, CLOSC and CLOLC have higher initialization ($\mathcal{IN}$) complexity because more parameters need to be generated in order to speedup the consistency examination ($\mathcal{CE}$) and result verification ($\mathcal{RV}$) phases of our protocols. Regarding transaction recording ($\mathcal{TR}$) and considering that [NVV18, CB21,LHNH23] employ a single-ledger approach, in ACA the transacting entity posts a single transaction pertaining solely to the transfer of its funds, while in zkLedger and Miniledger it also computes and uploads "dummy" transactions for the rest. In CLOSC the dominant term corresponds to the creation of the smart contracts, following by the Merkle tree creation time, whereas in CLOLC organizations create a linear combination of their records per transaction pair. During auditing, all proposed solutions parse through the ledger transaction records, except for CLOLC, where organizations have reported a digest of their entire records, rendering the consistency examination faster. Last, prior works do not consider an $\mathcal{RV}$ phase. In both CLOSC and CLOLC the committee performs only multiplications and hashing operations during this protocol phase, introducing a negligible overhead in the total required computational time as shown in Table 1a.

Regarding storage requirements, the single-ledger approaches record each and every transaction in the ledger, and all participants need to store the whole ledger. To combat growing ledger-storing costs, Miniledger employs a pruning technique, rendering the storage required ultimately less than in zkLedger and ACA. However, this pruning is not instantaneous and needs to be verified by all other participants. An important diversification in our designs is the following: Even though at a system level the storage is similar in terms of complexity to prior approaches, each organization only records their own activity, leading to significantly less storage requirements individually as shown in Table 1b. Additionally, after $\mathcal{RV}$ concludes only a small digest might be needed to remain from each ledger for historicity purposes, as mentioned above. Overall, the costs in CLOSC and CLOLC are comparable or lower than the other works whilst not only guaranteeing our three proposed security properties but also operating in a more relaxed security setting.

8 Discussion

On protocol generalization and limitations. Cross-ledger transaction consistency is an important check auditors can use to detect fraudulent behavior of their client-organizations e.g., reported sham transactions or inflated assets. We believe that our protocols can be further augmented to facilitate execution and verification of other popular financial auditing processes e.g., 4-way matching[14]. A possible direction to enable such a process could be for every organization to utilize the same specific SNARK construction that would take as input the four transaction commitments that satisfy the 4-way matching relation and output a proof about their consistency. Based on current practices we expect such an addition to behave similarly (from a computation perspective) to our proposed solutions. Last, we believe that incorporating atomic cross-chain exchange techniques like the ones in [NRVN22] on top of our auditability protocols may result in a system with even further capabilities. However, this remains challenging as the entity setting is different and additional processes will need to be designed. As for limitations, in CLOSC the committee currently needs to receive all commitments to be able to verify the auditor-generated results and in CLOLC it needs to generate a randomness per transaction per epoch during the initialization; recall that this is essentially a trade-off that allows auditors to perform the consistency examination phase more efficiently. Even though these limitations are not prohibitive in terms of performance as demonstrated in our previous evaluation, we leave improving and uplifting our protocols from these restrictions as future work.

On the inclusion of blockchains in our designs. Arguably, our architecture does not need to rely on blockchains, however, a two-level generic ledger approach is not sufficient either. Blockchains are immutable, rendering attack scenarios such as the one outlined in Section 1 impossible. Furthermore, we believe that building our system atop a blockchain with smart contract capabilities architecture will enable the inclusion of more auditing processes in the future. However, space and monetary costs are the main concerns when deploying blockchain systems. In our case, after epochs are concluded, the

---

[14] https://www.dataserv.com/blog/difference-between-2-3-4-way/

relevant data stored in all blockchains can be deleted, since it will not be used in any of the following epochs and no need on-chain asset transferring is executed either. Last, based on the above discussion and since the problem we are focusing on requires different entities to have access to different data, we observe that a permissioned network fits best our design. Nevertheless,we identify that designing our system atop a permissionless architecture while maintaining our three proposed security properties is challenging. Opening the system to arbitrary participation may lead to novel confidentiality and privacy attacks, and we leave this as a future research direction.

9   Conclusion

In this work we proposed CLOSC and CLOLC, the first two protocols attempting to tackle our newly defined problem of cross-ledger transaction consistency in the context of financial auditing. Both are built atop a two-tier blockchain architecture and CLOSC utilizes smart contracts while CLOLC linear combinations to achieve the consistency examination and verification. Moreover, we proved that both our protocols satisfy three crucial security and privacy properties. Finally, both our protocols scale well with the number of organizations, auditors, and transactions per epoch. We demonstrated this via extensive experimentation that showed both our solutions to be practical, deployable in real-world settings including hundreds of organizations and auditors, and millions of total transactions per auditing epoch.

Acknowledgments

References

ABAAA23. G. K. Amoako, J. Bawuah, E. Asafo-Adjei, and C. Ayimbire. Internal audit functions and sustainability audits: Insights from manufacturing firms. *Cogent Business & Management*, 10(1):2192313, 2023.

AČE⁺20. S. Allen, S. Čapkun, I. Eyal, G. Fanti, B. A. Ford, J. Grimmelmann, A. Juels, K. Kostiainen, S. Meiklejohn, A. Miller, et al. Design choices for central bank digital currency: Policy and technical considerations. Technical report, National Bureau of Economic Research, 2020.

AKST18. S. Arif, J. Kepler, J. Schroeder, and D. Taylor. Audit process, private information, and insider trading. *SSRN Electronic Journal. doi*, 10, 2018.

AN17. D. Appelbaum and R. Nehmer. Designing and auditing accounting systems based on blockchain and distributed ledger principles. *Feliciano School of Business*, pages 1–19, 2017.

BCDF23. C. Baum, J. H. Chiang, B. David, and T. K. Frederiksen. Sok: Privacy-enhancing technologies in finance. In *5th Conference on Advances in Financial Technologies, AFT 2023, October 23-25, 2023, Princeton, NJ, USA, LIPIcs* 282, pages 12:1–12:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

BCGI18. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018.

BGI15. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In *Advances in Cryptology - EURO-CRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II, Lecture Notes in Computer Science* 9057, pages 337–367. Springer, 2015.

BGMS18. N. Brender, M. Gauthier, J.-H. Morin, and A. Salihi. The potential impact of blockchain technology on audit practice. 2018.

CB21. P. Chatzigiannis and F. Baldimtsi. Miniledger: Compact-sized anonymous and auditable distributed payments. In *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part I, Lecture Notes in Computer Science* 12972, pages 407–429. Springer, 2021.

CMM⁺22. F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, and F. Tiezzi. Engineering trustable and auditable choreography-based systems using blockchain. *ACM Trans. Manag. Inf. Syst.*, 13(3):31:1–31:53, 2022.

Dam89.   I. B. Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

DDM20.   S. Demirkan, I. Demirkan, and A. McKee. Blockchain technology in the future of business cyber security and accounting. *Journal of Management Analytics*, 7(2):189–208, 2020.

DV17.   J. Dai and M. A. Vasarhelyi. Toward blockchain-based accounting and assurance. *J. Inf. Syst.*, 31(3):5–21, 2017.

Ear15.   C. E. Earley. Data analytics in auditing: Opportunities and challenges. *Business horizons*, 58(5):493–500, 2015.

HHK⁺21.   Y. Hu, K. Hooshmand, H. Kalidhindi, S. J. Yang, and R. A. Popa. Merkle^2: A low-latency transparency log system. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 285–303. IEEE, 2021.

HN96.   K. Hackenbrack and M. W. Nelson. Auditors' incentives and their application of financial accounting standards. *Accounting Review*, pages 43–59, 1996.

Hol23.   Holly Doyle, Simon Passfield, Guildhall Chambers. SHAMS: WHEN IS A TRANSACTION NOT A TRANSACTION?, 2023. Online; accessed 27 April 2023.

HSJ⁺23.   H. Han, R. K. Shiwakoti, R. Jarvis, C. Mordi, and D. Botchie. Accounting and auditing with blockchain technology and artificial intelligence: A literature review. *Int. J. Account. Inf. Syst.*, 48:100598, 2023.

JC21.   Y. Ji and K. Chalkias. Generalized proof of liabilities. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3465–3486. ACM, 2021.

KDP⁺24.   V. Koutsos, S. Damle, D. Papadopoulos, D. Chatzopoulos, and S. Gujar. Avecq: Anonymous verifiable crowdsourcing with worker qualities. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2024.

Kob87.   N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

Koz18.   S. Kozlowski. An audit ecosystem to support blockchain-based accounting and assurance. In *Continuous Auditing: Theory and Application*, pages 299–313. Emerald Publishing Limited, 2018.

KP23.   V. Koutsos and D. Papadopoulos. Publicly auditable functional encryption. In *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, *Lecture Notes in Computer Science* 13906, pages 396–425. Springer, 2023.

LHNH23.   C. Lin, X. Huang, J. Ning, and D. He. ACA: anonymous, confidential and auditable transaction systems for blockchain. *IEEE Trans. Dependable Secur. Comput.*, 20(6):4536–4550, 2023.

MBB⁺15.   M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman. CONIKS: bringing key transparency to end users. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 383–398. USENIX Association, 2015.

Mer87.   R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, *Lecture Notes in Computer Science* 293, pages 369–378. Springer, 1987.

Nah16a.   M. A. Naheem. Internal audit function and aml compliance: the globalisation of the internal audit function. *Journal of Money Laundering Control*, 19(4):459–469, 2016.

Nah16b.   M. A. Naheem. Money laundering: A primer for banking staff. *International Journal of Disclosure and Governance*, 13(2):135–156, 2016.

NRVN22.   K. Narayanam, V. Ramakrishna, D. Vinayagamurthy, and S. Nishad. Atomic cross-chain exchanges of shared assets. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*, pages 148–160. ACM, 2022.

NVV18.   N. Narula, W. Vasquez, and M. Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, pages 65–80. USENIX Association, 2018.

P716.   P7. Auditing disclosures in financial statements, 2016. Online; accessed 21 May 2024.

RMY⁺23.   D. Reijsbergen, A. Maw, Z. Yang, T. T. A. Dinh, and J. Zhou. TAP: transparent and privacy-preserving data services. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 6489–6506. USENIX Association, 2023.

RV18.   A. M. Rozario and M. A. Vasarhelyi. Auditing with smart contracts. *International Journal of Digital Accounting Research*, 18, 2018.

Woo14.   G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

XZC⁺22.   T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song. zkbridge: Trustless cross-chain bridges made practical. In *CCS 2022*, pages 3003–3017. ACM, 2022.