

Private, Auditable, and Distributed Ledger for Financial Institutes

Shaltiel Eloul*, Yash Satsangi*, Yeoh Wei Zhu*, Omar Amer, Georgios Papadopoulos, and Marco Pistoia

Global-Technology Applied-Research, JP Morgan Chase, London UK

Abstract. Distributed ledger technology offers several advantages for banking and finance industry, including efficient transaction processing and cross-party transaction reconciliation. The key challenges for adoption of this technology in financial institutes are (a) the building of a privacy-preserving ledger, (b) supporting auditing and regulatory requirements, and (c) flexibility to adapt to complex use-cases with multiple digital assets and actors. This paper proposes a framework[†]. for a private, audit-able, and distributed ledger (PADL) that adapts easily to fundamental use-cases within financial institutes PADL employs widely-used cryptography schemes combined with zero-knowledge proofs to propose a transaction scheme for a ‘table’ like ledger. It enables fast confidential peer-to-peer multi-asset transactions, and transaction graph anonymity, in a no-trust setup, but with customized privacy. We prove that integrity and anonymity of PADL is secured against a strong threat model. Furthermore, we showcase three fundamental real-life use-cases, namely, an assets exchange ledger, a settlement ledger, and a bond market ledger. Based on these use-cases we show that PADL supports smooth-lined inter-assets auditing while preserving privacy of the participants. For example, we show how a bank can be audited for its liquidity or credit risk without violation of privacy of itself or any other party, or how can PADL ensures honest coupon rate payment in bond market without sharing investors values. Finally, our evaluation shows PADL’s advantage in performance against previous relevant schemes.

Keywords: Distributed Ledger, Privacy, De-Fi, Zero-knowledge Proofs, Smart Contract, Blockchain, Multi-assets Ledger

1 Introduction

Blockchain and distributed ledgers technology has opened-up a new form of financial interaction, to transact with no requirement of mediator bodies. The impact is clearly evident in the continuous growth of public blockchain domains such as Bitcoin [31], Ethereum [39] and other ledgers based on smart contract platforms, notably HyperLedger [3]. For financial industry, which is originally

* These authors contributed equally to this work

† [PADL-source-code](#)

centralized, adapting distributed ledger technology potentially offers several advantages, such as reducing efforts involved in settlement and post-trades processing, streamlining laborious auditing processes, shifting to cloud environments, and enabling nearly instant approval of transactions. However, the current public blockchain ecosystem does not necessarily match the needs of financial institutions that must comply with government regulations and laws. A specific example is that of privacy: public blockchains requires that all participants have access to the transaction for its validation. This is unacceptable (or even illegal) to majority of institutions that require their clients' personal information or trading strategy be kept private and safe.

In recent years the focus on 'private' blockchains has increased to address concerns on privacy in public blockchain, which are considered only pseudo-anonymous. Privacy may be enhanced on blockchain by encrypting or anonymizing certain information, for example, the transaction's values, participants addresses, type of assets, or transaction graph. Inevitably though, encryption of information adds to the complexity of transaction validation and auditing. While encryption may help with enabling private transaction, it makes it nearly impossible for financial institutions to meet trading requirements or answer auditor's queries without revealing sensitive information.

Zero-knowledge proofs (ZKP) introduced as interactive [18] or non-interactive protocols (NIZK) [8] between a *prover* and *verifier* to convince the truth of a statement without revealing any further information. This technology can be used in private ledger to validate integrity of transactions and answer auditor queries without revealing sensitive information. ZKP systems have seen rapid advancements, particularly in terms of efficiency [19], succinctness, and general applicability. These developments build upon foundational frameworks such as zk-SNARKs [6] and extend to transparent systems that do not require a trusted setup, such as zk-STARKs [5], Ligerio [7,2], and Bulletproofs [9]. One of the primary advantages of modern proof systems is their ability to produce short proofs and facilitate the batching verification of range proofs, which demonstrate that a hidden value lies within a specified interval. Evidently, these frameworks can be combined with public blockchains to enhance the privacy of these blockchains with ZKP [35,38]. However current efforts focused on public blockchains fail to address institutions or enterprise blockchain applications that have a different set of auditing and privacy requirements or constraints that are not the focus of public crypto-assets. In practice, it is also recognized that adapting to new cryptography tools may require far more time in a regulated environment where security vulnerability cannot be compromised. Finally, financial industry use-cases involve multiple actors and multiple assets or asset-type in the trade or transaction. Existing public blockchain hardly address the issue of maintaining or exchanging multiple assets, not to mention, inter-asset auditing or customizing privacy. Hence, a private distributed ledger for financial institutions must be considerate of the following requirements:

Auditing. Auditing inevitably requires disclosing information to assess and limit the financial and security risks institutes possess. Auditing in a private

ledger can be divided into two types. First is privacy preserving auditing which does not require violating privacy of other participants (if they are not directly involved). In this optimal private auditing, a *prover* shares a ZKP to convince an auditor with the information encrypted to ensure; traceability, transaction integrity, or regulating trading requirements. This privacy preserving auditing would open-up in future the capability to audit financial institutes without laborious efforts or book disclosure, for example, maintaining level of capital at risk in a confidential multi-assets ledgers as we show here. Additionally, some regulations require opening all values. Hence ‘full’ auditing in essence, letting only a specified party, the capability to open an encrypted value. The requirement here is that this type of auditing should be limited to the specified party. We later discuss the case of a ‘settlement bank’, where tracing values can be done by a settlement bank for ‘full’ auditing.

Private Multi-Asset Ledger. Public blockchain ledgers, in their original implementation use anonymous accounts, but values, transactions graphs, balances, and assets are public. By private ledger, we consider this information fully or partially to be hidden with the combination of encryption and zero-knowledge proofs to create private blockchains. Note, that anonymity includes also the hiding of the transaction graph and their assets. Transaction graph can be an extremely resourceful information [1,44]. From trading perspective, even a small history of transaction can be sufficient to reveal its strategy or position, however, from auditing perspective, it is still required to maintain traceability, either in privacy preserving manner, with using ZKP, or with decryption of information by an auditor. Furthermore, majority of existing financial use-cases involve multiple assets and multiple actors. For example, existing trading applications support multiple equities, exchanges, loans or other complex financial instruments. Hence, confidential multi-asset and audit-able privacy between assets is a necessity in order to handle practical use-cases within financial institutions.

No-trust Setup. No trusted setup is highly desired in the context of external auditing using ZKP. Some distributed ledgers may offer auditing capabilities with the help of a trusted entity such as third-party auditors. This indeed does not provide data protection. Moreover, trusted setup or shared setup, using original zk-SNARK systems for example, would require the auditor to trust the setup. Hence, regulatory auditing would require transparent proofs, i.e. without verification or proving keys. Even if the auditor holds or trusts the holders of the keys, it makes auditing an issue when inter-banking applications may have several and different auditing bodies, where sharing the keys with the bodies, compromises the privacy of the data. The trusted setup, in the auditing context, is hence highly undesirable to most financial institutions as a pre-requirement.

In this paper, we propose PADL which follows a ‘table’ ledger scheme and combines encrypted commitments with audit-tokens and NIZK to support private peer-to-peer multi-asset transaction. It supports privacy preserving auditing of inter-asset balances to prove rates and liquidity risks of financial institutions with no trusted setup, and supports full auditing for provable traceability of transactions. We show that with PADL framework, it is straight-forward to

build financial market ledger such as bond markets, confidential asset swap, settlement bank with customized privacy. The transaction scheme enables asynchronous proof generations amongst the participants, and provides a general transaction purpose for various types of multi-asset transactions. Furthermore, PADL is an open-source package, compatible with smart-contract to showcase its integration to various blockchain integration. Finally, our performance evaluation shows that PADL, is not only more flexible, but outperforms other private ‘table’ ledgers without sacrificing privacy or auditing capabilities in any form. To summarize, our contributions are as follows:

- **PADL framework.** We propose a complete and industry-ready distributed ledger for private and anonymous multi-asset transactions while supporting privacy-preserving and full auditing.
- **New privacy-preserving audits notions.** We identified several core privacy preserving auditing proofs that enables operational risk control and are crucial for financial institution operations.
- **Evaluations.** We provide a comprehensive theoretical and empirical evaluation of PADL. We evaluate the security and privacy properties, and show its practicality via performance evaluation.
- **Usecase Showcase.** We showcase practical use-cases for financial institution ledgers that are enabled with PADL, thereby demonstrating the capability of PADL in constructing complex transaction scheme for financial applications.

2 Related Work

Our work is closest to zkLedger, that is also a table based private distributed ledger that combines NIZK based on Σ -protocols [12] with Pedersen commitment to propose a private and audit-able transaction scheme. However, zkLedger is limited to single non-private asset and thus it cannot support complex multi-asset applications that are common within finance industry. PADL’s flexible transaction scheme supports confidential multi-asset transactions with inter-asset auditing, making it possible to use for different use-cases such as asset exchange, trading loan ledger, settlement ledger, bond market, or secondary markets. Not only this, as shown in our evaluation, PADL transaction scheme scales better with number of assets and participants compared to zkLedger’s transaction scheme.

Other private ledgers that offer privacy and auditing include Solidus [10], PReDi [26] Miniledger [11], Fabzk [24], Azeroth [22], etc. Some of these ledger do not necessarily permit auditing while maintaining privacy [16], or that their auditing is possible only if the ledger is open to public or to a trusted third party, or a fixed auditor [4]. Furthermore, these ledgers are focussed and restricted to single asset transactions and auditing. Similarly, Platypus shows an ‘e-cash’ scheme for meeting regulations under privacy [40], but with the trust of a central bank. This can be a significant limitation of privacy, since auditor in such cases might be able to access private information related to banks and entities that are

not the target of the audit or have different regulatory requirements. Miniledger [11], simply presents a different implementation of the zkLedger whilst losing some capabilities in order to simplify it, but does not offer additional privacy value. PEReDi [26] does not support multi-asset swaps and/or offline transactions. Many privacy preserving protocols based on membership proof, such as on Monero [33] and ZCash [35] are intended for efficiency in much larger number of participants than in the case of enterprise blockchain. Therefore, they are concerned mostly with single asset/currency and (pseudo) anonymity without auditing or without multi-assets capability. However recent works investigate and offer Monero protocols with tractability [28]. Here, we show that in a ‘table’ data structure, it is straight-forward to deal with privacy preserving or full auditing, and to introduce auditing between confidential assets. Similarly, ZCash which is based on succinct schemes, is also being studied for the propose of regulation enforcement protocols [42,27,42], or for the ability to introduce multi-assets for ZCash [14,41]. These ledgers and others based on zk-SNARK [14,23,41,17,40], however do not focus on enterprise blockchains use-cases where trust setup as with original zk-SNARK implementations is not practical for auditing context. Finally, there are also applications of privacy preserving ledgers that are in this work context, for example, [25,30] for dynamic pricing and securities market, respectively. We extend auditing capabilities, and add to these set of applications by applying PADL to private atomic swap exchanges and bond markets in a single framework and general transaction scheme.

The rest of the paper is structured as follows. Sec. 3 describes the PADL design and the transaction scheme informally. This is immediately followed by the application use-cases in Sec. 4. Sec. 5 describes formal treatment of our PADL followed by its security and performance analysis in Sec. 6. For supplementary, we add appendix providing detailed examples of transactions (B), and further security analysis with complementary proofs (C,D).

3 PADL - Ledger Design

This section describes PADL design and the proposed private transaction scheme informally. We postpone the formal definition of PADL transaction scheme to Sec. 5 but for clarity of this section, introduce essential components; Let \mathbb{G} be a cyclic group and let g, h be two random generators of \mathbb{G} . Then in the setup of a ledger, each participant generates its public key, pk which is related to its secret key sk via the following relation: $pk = h^{sk}$.

The PADL structure is an ‘append-only’ transaction ledger, where each transaction is added to a table in a synchronized manner. Figure 1 illustrates the different actors that can interact with the distributed ledger (a), the information stored in the encrypted ledger table (b) and high-level description of the transaction scheme (c). Asynchronous transactions is also possible in various ways, but out of scope at this stage. A transaction is sent to the host of the ledger by the ‘sender’. The sender knows and hides all participants’ values for a specific transaction. A multi-asset transaction, consisting of one or more assets, repre-

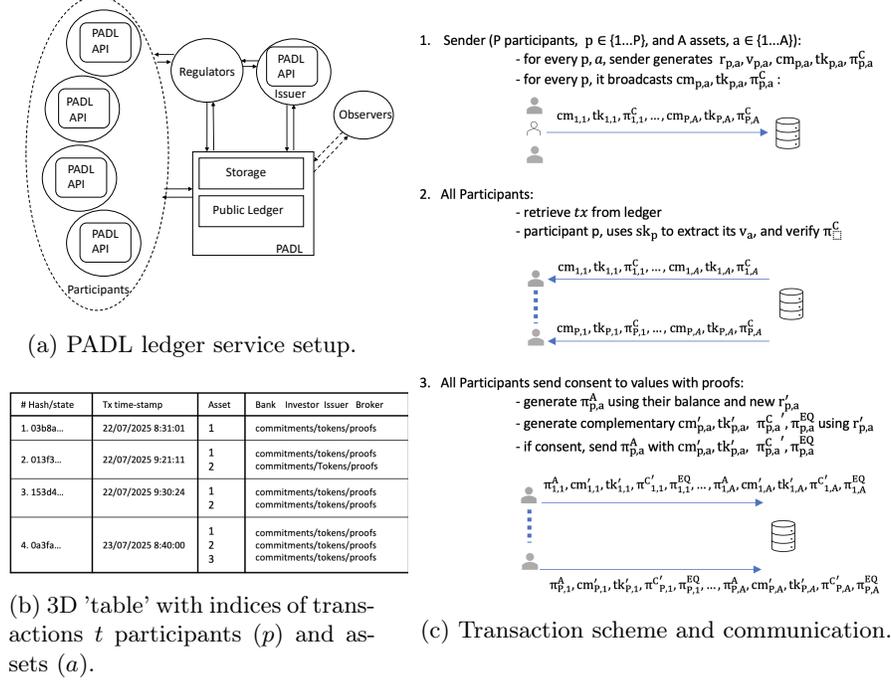


Fig. 1: PADL high-level design of transaction and a 'table' ledger.

sents one row that may have several sub-rows depending on the number of assets in the transaction. Each transaction can possess multiple confidential assets, so the assets and their amounts in the transactions are encrypted using Pedersen commitment for each participant to hide their values. The commitments are accompanied with tokens and the proofs required to validate the integrity of the transaction scheme. Thus, for each transaction t , each participant p , and each asset a , we associate a cell of a 3-dimensional and dynamical array, $\text{Cell}_{t,p,a}$, where t, p, a are the indices of the transaction, participant, and asset, respectively:

$$\text{Cell}_{t,p,a} = \{cm_{t,p,a}, tk_{t,p,a}, cm'_{t,p,a}, tk'_{t,p,a}, \pi_{t,p,a}^A, \pi_{t,p,a}^C, \pi_{t,p,a}^{C'}, \pi_{t,p,a}^{EQ}\}$$

In the above $\text{Cell}_{t,p,a}$, $cm_{t,p,a}$ is Pedersen commitment, defined by $cm_{t,p,a} = g^{v_{t,p,a}} h^{r_{t,p,a}}$. Here $v_{t,p,a}$ is the value participant wishes to hide and $r_{t,p,a}$ is a random blinding factor. The commitment is associated with a blinding factor as a token, $tk_{t,p,a} = (pk_p)^{r_{t,p,a}}$. In order to prove the binding of the blinding factor in the token to the commitment, a proof of 'consistency' is always accompanied here ($\pi_{t,p,a}^C$). To make sure that no asset is created or destroyed in a transaction, it is required that the product of commitments across participants for each asset is equal to 1, where $\forall a, t, \prod_p cm_{t,p,a} = 1$. This ensures that the sum of blinding factors and the sum of values are equal to zero, $\sum_p v_{t,p,a} = 0$

and $\forall a, t, \sum_p r_{t,p,a} = 0$. Table 1 provides the algorithms for PADL construction, and the generation of commits and blinding factors is detailed in the algorithm PACT.Spend.

Additionally, a proof of asset is required to validate that no participant overspends, $\pi_{t,p,a}^A$. In order for each participant to provide a proof of asset, we introduce a complementary commit $\text{cm}'_{t,p,a}$, a complementary token $\text{tk}'_{t,p,a}$, and proofs. The complimentary commit is a commitment to the same value but with a different blinding factor r' and the token is the public key raised to r' . The complimentary commits are required since the participants do not know the blinding factor in the original commitment, hence they re-commit their values. Finally, each participant starts with an initial cell for each asset that can be minted by the participants themselves and/or verified by other parties (Def. 3).

PADL Transaction. The transaction scheme is designed for general purpose, confidential multi-assets transactions (Fig. 1c). A formal definition of the transaction scheme (Def. 3) is provided later at Sec. 5. Before the broadcasting step, a sender generates commitment-token pairs and proofs of consistency, for every asset and participant. It also hashes the commitment-token pairs $H(\text{cm}, \text{tk})$ to provide a unique identification of the transaction information. The transaction is then broadcast, without the proof of assets, allowing negative values for other participants, e.g. a broker, or in ‘atomic’ asset exchange. Subsequently, it also does not reveal who is the sender (traditionally known as spender), i.e. the sender is not the only participant who can commit to negative value (sends asset). After broadcasting the transaction, each participant, can extract the values committed, $\text{extract}(\text{cm}_{t,p,a}, \text{tk}_{t,p,a}, \text{sk}_p)$ using their secret keys by calculating $\text{cm}_{t,p,a}^{\text{sk}_p} \cdot \text{tk}_{t,p,a}^{-1} = g^{\text{sk}_p v_{t,p,a}}$ and use brute-force to solve the $\text{dlog}_{g^{\text{sk}_p}} g^{\text{sk}_p v_{t,p,a}} = v_{t,p,a}$. Only the holder of the secret key can extract and be convinced with its value. This extraction is possible in practical cases where value $v_{t,p,a}$ belongs to a finite small set, and within a maximum range.

Next, the participants fill their proof of assets. Since $r_{t,p,a}$ is not known for the participant, participants provide also a complementary commit ($\text{cm}'_{t,p,a}$), complementary token ($\text{tk}'_{t,p,a}$) and complementary consistency proof ($\pi_{t,p,a}^C$) for a new private $r'_{t,p,a}$. In order to verify that the same v is used, the participant also generates an equivalence proof (π^{EQ}). This proof is also sufficient to verify the identity, as it requires knowledge of its secret key, this proof is detailed in Sec. 5. The participant then adds a proof of asset π^A for proving the sufficient balance in asset a with the product of previous commits, $\prod_t \text{cm}'_{t,p,a}$.

This scheme has several advantages. First, the proving system is simpler than previously suggested [32,11], since it does not require generating the conditions of using a disjunctive proof.

Second and more importantly, it enables a single exchange with multi-assets, by committing minus values for other participants in a transaction. This offers practical flexibility when applying PADL to different use-cases where a trade-off between privacy, performance and future commitments/contracts is needed. Third, it hides the sender from the broadcaster or from the ledger. Fourth,

generating the proof of asset is computationally expensive and likely the limiting step, and in this scheme, the participants generate the proof for themselves, asynchronously.

Extension to ‘injective tx’. PADL transaction scheme requires other participants besides the sender to interact with the ledger to fill up the proof of assets. However, this is not necessarily a bottleneck since on any encrypted ledger, participants would need to consent to an encrypted value, even if it is a positive value, e.g. in loan assets, or where assets accrued interest. With that, our scheme can be easily changed to the simple case of an ‘injective’ transaction where consent is not required. In such case, the sender would provide range proofs to all transaction values to be positive, and proof of asset for itself. Along with another disjunctive proofs for everyone to conceal the sender [32]. Note that in any case the validation of proofs is required.

Validation and extension to dropping parties. In order to validate transactions, each participant can verify the ZKP proofs at its own local node or machine, and also validate the hash declared in the transaction to ensure the commits and tokens corresponds to the number of participants and assets cells seen. An API can approve or reject the transaction. Alternatively, verification of the proofs for all broadcast cells can be obtained by a centralized service or with smart contract (which is also implemented in the PADL code). In case that some proof of assets are not valid, or not filled due to dropping parties, the sender can exclude these participants ($O \subset P$) and the transaction can be still validated. The exclusion is simply done by an extra step. The sender(p_s), regenerates its own cell with a modified blinding factor, r_{p_s} , that is the sum of the excluded participants and itself, $r_{t,p_s,a} = r_{t,b,a}^{orig} + \sum_{o \in O} r_{t,o,a}$. That makes the proof of balance valid again. In fact, the sender is able to remove any participant as long as the commit value is 0. Otherwise, the proof of balance would fail. The trade-off here, is anonymity of the transaction graph. The less participants involved in the transaction, the more information on the transaction graph can be extracted. The security of anonymity and integrity of the transaction scheme is defined in Sec. 6 and formally analyzed in Appendix C.

4 Use-Cases

We discuss several fundamental use cases for transactions and trading applications where PADL can provide privacy with audit-ability. Other uses cases can be built on top of the ledger components here. Concrete examples of transaction generation are also shown in Appendix B.

4.1 Simple exchange ledger

We describe how PADL can be seamlessly used for exchange of confidential assets atomically. Assuming that two participants would like to exchange assets, with

PADL it can be done in a single transaction. To do so, participant A creates a transaction consisting of commitments and tokens committing to transferring x amount of asset a to participant B. In the same transaction, participant A also writes a commitment on behalf of participant B, committing participant B to y amount of b . Then, participant A can broadcast this transaction, participant B can decide to either accept or reject this transaction. To accept the transaction, participant B simply writes its Cell's proofs, which is then finalized and verified by participant A and the transaction is appended to the ledger. Note that by adding their proof of assets, they consent to the exchange. Next, the transaction is appended if validation passes. In the validation, the ZKPs are verified and the immutability of original broadcast transaction is checked by checking the hash of the final transaction. Finally, if a participants provides a 'wrong' or 'bad' cell in the transaction, it simply results in rejection of the transaction.

4.2 Settlement Trusted Bank ledger

Currently, bank payment systems are required a trusted party in order to be audit-able and follows 'Know-Your-Customer', Anti-Money Laundering, specific rules, or keys recovery criteria. In such cases, the ledger contains a party that acts as a settlement bank, and would have access to the balance of some participants ('full' auditing). In a distributed solution, we wish to avoid key management by the trusted parties, meaning parties would not share their private keys with any party. This scenario can be treated in PADL, by an additional token, tk^I ; signing $r_{t,p,a}$ by the public key of the trusted party. For many cases, the issuer can be defined as the 'auditor' or 'trusted' party, where its participant index is $p = I$. The transaction cell structure is altered to facilitate this requirement:

$$\text{Cell}_{t,p,a} = \{ \text{cm}_{t,p,a}, \text{tk}_{t,p,a}, \text{cm}'_{t,p,a}, \text{tk}'_{t,p,a}, \pi_{t,p,a}^A, \pi_{t,p,a}^C, \pi_{t,p,a}^{C'}, \pi_{t,p,a}^{\text{EQ}}, \text{tk}_{t,p,a}^I, \pi_{t,p,a}^{C'} \}$$

Note that we also need to add consistency proof for the new added token. However, if in the committee consensus, the issuer is also designated as the approving entity (no need for consensus), the Cell structure can be reduced to:

$$\text{Cell}_{t,p,a} = \{ \text{cm}_{t,p,a}, \text{tk}_{t,p,a}, \pi_{t,p,a}^C, \text{tk}_{t,p,a}^I, \pi_{t,p,a}^{C'} \}$$

The issuer can also check for any asset balance, by extracting the balance for each asset itself, and proof of asset can be avoided. This accelerates largely the ZKP system for a very typical and realistic use-case. In addition since, the token is designated to a single cell, it is flexible to define many audit tokens for a list of parties (pks) and for any cell in a transaction.

4.3 Bond Market ledger

In this subsection we describe the application of PADL to a third and a more complex use case of trading bonds. The application of blockchain in the debt capital markets with arrival of 'smart bonds' is highly anticipated. However, in

many cases, the bond issuers or investors might be reluctant to disclose their positions. We show how PADL is applied to make private transaction in the bond markets. For this case, we are mainly interested in exchange of two types of assets, bonds and a USD-backed digital token. The bond issuer would like to issue bonds and borrow USD-coin in return. The investors wish to lend USD-coin to buy bonds, however, the USD-coin for the investors are handled by another party, i.e. custodians. Also, the bonds are handled by brokers with fees, thereby interacting with the bond issuer and the investors. The privacy requirements in this scenario are as follows:

- Only the broker knows the details of bond deals.
- Custodian knows only the amount of money released to an investor.
- The bond issuer does not need to know the individual investors' contribution.
- The investors does not know about other investors.
- Coupon (or interest rate) payment by the issuer can be issued without the issuer learning about the distribution of the payment amongst the investors.

PADL allows the investors and bond issuer to make these transaction while maintaining privacy as described below (in brief). The entire transaction flow is described in Appendix.

We start by assuming all the participants: issuers, custodians, broker and the bond issuer are on PADL, and a ledger consists of two assets, a bond and a USD tokens. The ledger is initialized such that bonds are minted by the bond issuer and the USD coin asset by the custodians. The custodian can then issue USD token to the investors as requested by the investors, by creating a transaction on PADL. In order to exchange assets (buy bond for USD coin), the broker creates a transaction on behalf of both the investor and bond issuer. This transaction consists of commitments that are written in every participant's columns. The bond issuer can open the commitments in its column to verify that it is committing to x number of bonds in exchange for y number of USD coins. Similarly, the investor can verify that it is committing to USD token in exchange for receiving bonds. Furthermore, the broker creates transactions for the bond issuer to pay the coupons to the investors. Again, the only value the bond issuer can verify is that it pays the correct coupon rate for the money it received for the bonds, however, it cannot decrypt the payments that each of the participants on PADL receive. Each investor however can verify that it receives the right amount of coupon payment as promised. Finally once the bond matures, the broker in a single transaction (see example in Appendix B), can return the USD coins to the investors while taking its commission away from all participants, and the bond issuer can destroy the bond asset.

The use-cases above shows the applicability of PADL to develop various private and audit-able markets. We next discuss the cryptography and methods which define the transaction scheme, auditing, and ledger construction followed by security analysis.

5 Methods

5.1 Notations and Background

We denote by $y \leftarrow \mathcal{A}(x)$ the execution of algorithm \mathcal{A} on input x and produces y . By $\mathcal{A}^{\mathcal{O}}$, we denote an algorithm \mathcal{A} that has access to oracle \mathcal{O} . We use $r \leftarrow \$ S$ to denote that r is sampled uniformly at random from a set S . Let \mathbb{G} be p prime-ordered additive subgroup of the elliptic curve $E(\mathbb{Z}_q)$ generated by some generator g and some prime q . For simplicity, we use multiplicative notation for group \mathbb{G} throughout the paper. By $\text{cm}_{t,p,a}$, we denote the participant p (with public key pk_p) account's commitment coin in the t -th transaction row for a specific asset a . Similarly, by $\text{vt}_{t,p,a}$, we denote the corresponding commitment coin value. We exploit the notation for $\text{vt}_{\text{tx},p,a}$ to mean the same for a specific transaction tx . We often omit a if the context is clear.

Discrete Log (DLOG) Assumption: Given g, g^x where $g \in \mathbb{G}$ and $x \in \mathbb{Z}_P$, no PPT adversary can output x with non-negligible probability. This paper relies on the intractability of solving the discrete logarithm problem.

Commitment Scheme. A commitment scheme consists of two steps: first the sender commits to a value as a commitment; later the sender may choose to reveal the committed value. A commitment scheme should satisfy the hiding and binding properties. Additionally, a commitment scheme has additive homomorphic property, if given two commitments, we have $\text{Com}(\text{ctk}, m_0, r_0) \cdot \text{Com}(\text{ctk}, m_1, r_1) = \text{Com}(\text{ctk}, m_0 + m_1, r_0 + r_1)$. We briefly recall notions for the Pedersen commitment scheme [34] while hiding and binding definitions [20] are recalled in the Appendix. ctk is omitted for the rest of the paper since it is clear from the context.

Definition 1 (Pedersen Commitment). *Pedersen (homomorphic) commitment scheme (HCOM) consists of the following set of algorithms:*

- $\text{CKeyGen}(1^\lambda)$: on input a security parameter 1^λ , this algorithm computes $\text{ctk} := (g, h) \leftarrow \$ \mathbb{G}$ where g, h are generators and outputs ctk .
- $\text{Com}(\text{ctk}, m, r)$: on input a commitment key ctk , a message m and a randomness r , this algorithm parses $(g, h) := \text{ctk}$, and outputs $g^m h^r$ as the commitment cm .

Zero-Knowledge Proof. A non-interactive zero-knowledge proof (NIZK) [21] allows a prover to prove to a verifier of some statement in zero-knowledge. NIZK is defined as follows:

Definition 2 (Non-interactive Zero-Knowledge Argument System). *Let \mathcal{R} be an NP-relation and $\mathcal{L}_{\mathcal{R}}$ be the language defined by \mathcal{R} . A non-interactive zero-knowledge argument system for $\mathcal{L}_{\mathcal{R}}$ consists of the following algorithms:*

$\text{ZKSetup}_{\mathcal{L}_{\mathcal{R}}}(\lambda)$: Takes as input a security parameter λ , and outputs a common reference string crs .

$\text{ZKProve}_{\mathcal{L}_{\mathcal{R}}}(\text{crs}, \text{stmt}, \text{wit})$: Takes as input a common reference string crs , a statement stmt and a witness wit , and outputs either a proof π_{ZKP} or \perp .

$\text{ZKVerify}_{\mathcal{L}_{\mathcal{R}}}(\text{crs}, \text{stmt}, \pi_{\text{ZKP}})$: Takes as input a common reference string crs , a statement stmt , and a proof π_{ZKP} , and outputs either a 0 or 1.

5.2 Privacy Preserving Multi-Asset Transaction with Audit

In this section, we formally introduce the notion of a confidential transaction with privacy-preserving audit scheme (PACT) which is later used to show PADL construction. PACT is based on the syntax from (ring) confidential transaction (Ring-CT) [37,43] but additionally introduces notions that capture the requirements for a variety of privacy-preserving audits. Modifications are also made to the syntax of confidential transaction scheme, allowing the scheme to better capture the functionality of table-based multi-asset transaction rather than graph-based mono-asset transaction. The security and privacy of the transaction scheme are detailed in Appendix C.

PACT scheme. In general, a PACT scheme consists of algorithms that are used to make confidential transactions. It is assumed that the distributed ledger is properly maintained and updated at all time*. Recipient accounts (accounts where the commitment coin is positive) and spending accounts (accounts where the commitment coin is negative) are implicitly captured by the transaction amount list \mathcal{V} . Note that the transaction considered here is a multi-asset transaction. We additionally introduce PACT.Endorse to capture the transaction flexibility of the proposed PADL scheme, which permits policy-based spending. Policy enforcement is flexible and can include rules such as always accepting positive deposits, only accepting negative amounts if the transaction is initiated by the account owner, only accepting negative amounts if it is an authorized scheduled direct debit, and other possible variations. In practice, the default policy, Pol_G is configured to accept positive amounts and only accept negative amounts if the transaction is initiated by the same account.

Definition 3 (PACT). A privacy preserving-enabled confidential transaction scheme PACT is a tuple of algorithms,

$\text{PACT} = (\text{PACT.Setup}, \text{PACT.KeyGen}, \text{PACT.Mint}, \text{PACT.Spend}, \text{PACT.Verify})$:

- $\text{PACT.Setup}(1^\lambda)$: on input a security parameter 1^λ , this algorithm outputs a public parameter pp . All algorithms defined will implicitly receive pp as part of their inputs.
- $\text{PACT.KeyGen}(1^\lambda)$: on input a security parameter 1^λ , this algorithm outputs an account secret key sk and an account public key pk .

* A consensus protocol is used to manage the state of the distributed ledger. However, the exact details are considered outside the scope of this work.

- $\text{PACT.Mint}(v)$: on input a transaction amount v , this algorithm outputs a commitment cm and a commitment blinding factor r .
- $\text{PACT.Endorse}(v, \text{Pol}, \text{sk}, \text{aux})$: on input a transaction amount v , a transaction policy Pol , an account secret key sk , and an auxiliary information aux , this algorithm outputs an endorsement data En .
- $\text{PACT.Spend}(\mathcal{V}, \mathcal{T}, \mathcal{PK})$: on input a transaction amount list \mathcal{V} , a history transactions list \mathcal{T} , and a master account public key list \mathcal{PK} , this algorithm outputs a new transaction tx , and a validity proof π .
- $\text{PACT.Verify}(\text{tx}, \pi, \mathcal{T}, \mathcal{PK})$: on input a new transaction tx , a validity proof π , a history transactions list \mathcal{T} , and an account public key list \mathcal{PK} , this algorithm outputs a verification bit $b \in \{0, 1\}$.

In the context of auditing distributed ledgers, it is often required that the auditor can open the transactions and trace their graph to make conclusions about the financial health of the organization. In Provisions [13], more privacy-preserving auditing is considered whereby solvency is proved without revealing the entire transactions history. However, existing works do not consider a couple of fundamental audit information that is used in financial auditing landscape. We identified the following sets of privacy-preserving auditing concepts that would be of interest to the financial auditing of participants in the distributed ledger. PACT should support the following set of privacy-preserving audit capabilities:

1. **Basic Asset Balance.** Without revealing the transaction history, the verifier should be convinced of the asset balance of the ledger participant.
2. **Liquidity.** Without revealing the transaction history and an asset balance, the verifier should be convinced of the liquidity or credit of the asset.
3. **Inter-transactions Rate.** Without revealing the transaction history, the verifier should be convinced of the inter-transaction rate between two subset of transactions.

5.3 Final PADL construction

In this section, we lay out the cryptography setup and elements involved in the transaction scheme. As discussed in the preceding sections, PACT.Setup is executed to obtain g, h which are assumed to be public in the ledger. Each participant will generate their keypair sk_p, pk_p using PACT.KeyGen where pk_p will now be the participant's identity and unique account address. Transactions with proofs are generated using PACT.Spend while the transaction value list \mathcal{V} used could be prepared by any party. During the generation of a transaction, the sender contacts the broadcaster using BCast.GetProofs (in PADL, broadcast is a functionality provided by the ledger) to obtains endorsements from all parties. Meanwhile, the transaction amount is committed using PACT.Mint . Lastly, PACT.Verify is used to verify the transaction with proofs generated by PACT.Spend with respect to the latest ledger transaction state and participant list. Each participant maintains its internal state stat_p which records $\text{sk}_p, \text{pk}_p, \text{Pol}_p$. The construction is summarised in Table 1 and all the elements of PADL's

$\text{Cell}_{t,p,a}$ are listed as follows (indices t , p and a are omitted whenever appropriate for fluency).

Commitment commits to the value in a transaction as defined in Def. 1: $\text{cm} := \text{Com}(v, r)$.

Token hides the blinding factor, r with a public-key, $\text{tk}(r, \text{pk}) = \text{pk}^r$. The token has two distinctive roles: to facilitate verification of proof and to enable extractability of the committed value without knowledge of r , as long as the value belongs to a small set. For example, participant, p , can take a commitment $\text{cm}(v, r)$ and token $\text{tk}(r, \text{pk})$, and use its secret key to compute $\frac{\text{cm}}{\text{tk}}^{\text{sk}}$ to obtain $(g^{\text{sk}})^v$. The extractability is also used in assigning an auditor or a trusted party, as described in the settlement bank use-case. Note that the combined system (cm, tk) makes the scheme computationally hiding because adversary could solve the Discrete Log Problem for tk to obtain r and be able to open the cm and proof of consistency ensured that the correct r is used to construct the token.

Proof of balance maintains the conservation of all assets in a transaction tx , ensuring that the sum of each asset value in the transaction equals zero. Proof of balance is a proof for the language $\mathcal{L}_{\mathcal{R}_{\text{BA}}} := \{(\text{cm}_{\text{tx},1}, \text{cm}_{\text{tx},2}, \dots, \text{cm}_{\text{tx},|\mathcal{PK}|}) : \sum_{p=1}^{|\mathcal{PK}|} v_{\text{tx},p} = 0\}$. In PADL, proof of balance is checked by multiplying the commitments together and checked for equality with 1. By the homomorphic property of Pedersen commitment and assuming r_i is the additive sharing of zero, we have $\prod_{p=1}^{|\mathcal{PK}|} \text{cm}_{\text{tx},p} = g^{v_{\text{tx},1} + v_{\text{tx},2} + \dots + v_{\text{tx},|\mathcal{PK}|}} h^{r_{\text{tx},1} + r_{\text{tx},2} + \dots + r_{\text{tx},|\mathcal{PK}|}} = g^0 h^0 = 1$. The proof of balance is unforgeable due to the hardness of discrete log.

Proof of consistency (π^{C}) ensures that the randomness used in the commitment and token are consistent. The proof of consistency is a proof for the language $\mathcal{L}_{\mathcal{R}_{\text{C}}} := \{(\text{cm}, \text{tk}) : \exists v, r \text{ s.t. } \text{cm} = g^v h^r \wedge \text{tk} = \text{pk}^r\}$. The proof of consistency is a ZK- Σ -protocol where the *prover* chooses random $u_1, u_2 \in \mathbb{Z}_P$, and sets $t_1 \leftarrow g^{u_1} h^{u_2}$, $t_2 \leftarrow \text{pk}^{u_2}$, and generates a challenge with a hash function $c = H(h, g, t_1, t_2, \text{cm}, \text{tk}) \in \mathbb{Z}_P$. The *prover* calculates $s_1 \leftarrow u_1 + cv$, $s_2 \leftarrow u_2 + cr$. It then outputs the transcript (t_1, t_2, s_1, s_2) . The *verifier* verifies: $\{t_1 \cdot \text{cm}^c = g^{s_1} h^{s_2} \wedge t_2 \cdot \text{tk}^c = h^{s_2} \wedge c = H(h, g, t_1, t_2, \text{cm}, \text{tk})\}$.

Proof of equivalence (π^{EQ}) is used to show that two commitments have the same commitment value. The proof of equivalence is a proof for the language $\mathcal{L}_{\mathcal{R}_{\text{EQ}}} := \{(\text{cm}, \text{cm}') : \exists v, r, r' \text{ s.t. } \text{cm} = g^v h^r \wedge \text{cm}' = g^v h^{r'}\}$. Given a commitment-token pair (cm, tk) and their consistency-verified complementary counterpart (cm', tk') , we can compute $a := \frac{\text{tk}}{\text{tk}'} = h^{(r-r')^{\text{sk}}}$ and $b := \frac{\text{cm}}{\text{cm}'} = h^{r-r'}$, assuming $v = v'$. If $v \neq v'$, the prover would need to compute discrete log $x = d\log_{(g^{v-v'} b)} b^{\text{sk}}$ where $b = h^{r-r'}$ as the discrete log between g and h is not known. Therefore, it is sufficient to check for the knowledge of secret key between the base b and group element a to check for the commitment coin value equivalency. In PADL, Schnorr protocol [36] for knowledge of discrete log is used. Let $\frac{\text{tk}}{\text{tk}'} = a$ and $\frac{\text{cm}}{\text{cm}'} = b$. The *prover* chooses random $u \in \mathbb{Z}_P$, sets $t \leftarrow b^u$, and generates a challenge with a hash function $c = H(a, b, t) \in \mathbb{Z}_P$. The *prover* calculates $s \leftarrow u + c \cdot \text{sk} \in \mathbb{Z}_P$. It then outputs the transcript (c, t, s) . The *verifier*

verifies: $\{t \cdot a^c = b^s \wedge c = H(a, b, t)\}$.

Proof of asset (π^A) asserts that after summing the commitment coin values of an asset for a particular account p , it holds that the balance of the account is positive. Positive for PADL is defined to be within some range 2^N . Proof of asset is a proof for the language $\mathcal{L}_{\mathcal{R}_A} := \{(\mathbf{cm}_{1,p}, \mathbf{cm}_{2,p}, \dots, \mathbf{cm}_{|\mathcal{T}|+1,p}, 2^N) : \sum_{i=1}^{|\mathcal{T}|+1} v_i < 2^N \text{ where } g^{v_i} h^{r_i} = \mathbf{cm}_i\}$. Proof of asset for some participant p is instantiated by first asserting that a complementary commitment \mathbf{cm}' has the same coin value as the aggregated commitment coin via proof of equivalence whereby the aggregated commitment coin is obtained by multiplying all the commitment coins $\mathbf{cm}_\Pi := \prod_{t=0}^{|\mathcal{T}|+1} \mathbf{cm}_{t,p}^*$. Then, a NIZK range proof for Pedersen commitment such as Bulletproofs [9] is used to assert that the asset balance contained in the complementary commitment \mathbf{cm}' is positive.

Note that π^C is instantiated directly from the generalized special honest-verifier proof of knowledge (ZKPoK) Σ -protocol for the preimage of a group homomorphism by Maurer [29]. The aforementioned theorem and the instantiation are shown in the Appendix C. Additionally, interactive proofs are transformed into non-interactive zero-knowledge proofs (NIZK) using the Fiat-Shamir transform [15] in the random oracle model and the transaction identifier is a part of the statement for all proofs.

5.4 Privacy Preserving Auditing of Confidential Assets

Previous works, such as zkLedger [32], demonstrated the use of the homomorphic property of Pedersen commitments and audit tokens for auditing known assets. In the context of confidential assets, we can generate privacy-preserving audits that can be used by external auditors, or being used directly as conditions for transactions without sharing the sum of values. For example, it can be used for liquidity and credit risk, or interest rate return payments. We show the correctness for proofs of basic auditing of an asset balance, asset liquidity and inter-transactions rate proof without revealing transaction history, thereby achieving PADL privacy. The auditing ZKP can simply extend the transaction cell structure or being queried separately on the ledger.

Basic Asset Balance. Auditing any asset balance is done by proving the equivalence between the product of commitments signed by a secret key and the product of tokens as follows. Participant p (*prover*), shares with an auditor (*verifier*) the claimed balance $v_{p,a}$ for asset a . The *verifier* calculates product of commitments and product of tokens (for a particular asset and participant), $\prod_{t \in T_{xs}} \mathbf{cm}_{t,p,a} / g^{v_{p,a}} := c_1$ and $\prod_{t \in T_{xs}} \mathbf{tk}_{t,p,a} := c_2$, respectively.

The *prover* sends the zero knowledge proof [29] for the equivalence, $dlog_{c_1} c_2 = dlog_h \mathbf{pk}_p$, where both right and left terms open to \mathbf{sk} . The *verifier* verifies the

* Alternatively, by multiplying all the individual complementary commitment coins which is a re-committed commitment coin.

<p>PACT.Setup(1^λ) $(g, h) \leftarrow \text{CKeyGen}(1^\lambda)$ $\text{pp} := (g, h)$ return pp</p>	<p>PACT.KeyGen(1^λ) $(g, h) \leftarrow \text{pp}$ $\text{sk} \leftarrow \mathbb{Z}_P$ $\text{pk} := h^{\text{sk}}$ return sk, pk</p>	<p>PACT.Mint(v) $(g, h) \leftarrow \text{pp}$ $r \leftarrow \mathbb{Z}_P$ $\text{cm} := \text{Com}(v, r) = g^v h^r$ return cm, r</p>
<p>PACT.Spend($\mathcal{V}, \mathcal{T}, \mathcal{PK}$) $(g, h) \leftarrow \text{pp}$ $\text{tx} \leftarrow [], \pi \leftarrow []$ for each asset $\mathcal{V}_a := [v_1, \dots, v_{ \mathcal{PK} }]$ in \mathcal{V} : $\text{tx}_a \leftarrow [], \pi_a \leftarrow []$ $r_0 = 0$ for each participant p, its pk_p and v_p : if not last participant p : $\text{cm}, r := \text{PACT.Mint}(v_p)$ $r_0 += r$ else : $r := -r_0$ $\text{cm} := g^v h^r$ $\text{tk} := \text{pk}_p^r$ $\pi^C := \text{ZKProve}_{\mathcal{L}_{\mathcal{R}_C}}(\text{crs}, (\text{cm}, \text{tk}), (v_p, r))$ $\pi_{p,a} := (\pi^C)$ $\pi_a.append(\pi_{p,a})$ $\text{tx}_{p,a} := (\text{cm}, \text{tk})$ $\text{tx}_a.append(\text{tx}_{p,a})$ $\text{tx}.append(\text{tx}_a), \pi.append(\pi_a)$ $\text{tx} := (H(\text{tx}), \text{tx})$ En $\leftarrow \text{BCast.GetProofs}(\text{tx})$ for each asset a and each participant p : parse $(\text{cm}'_{p,a}, \text{tk}'_{p,a}, \pi_{p,a}^{A*}, \pi_{p,a}^{C'}, \pi_{p,a}^{\text{EQ}})$ from En update $\pi_{p,a} := (\pi_{p,a}^{A*}, \pi_{p,a}^C, \pi_{p,a}^{C'}, \pi_{p,a}^{\text{EQ}})$ in π update $\text{tx}_{p,a} := (\text{cm}_{p,a}, \text{cm}'_{p,a}, \text{tk}_{p,a}, \text{tk}'_{p,a})$ in tx return (tx, π)</p>	<p>PACT.Verify($\text{tx}, \pi, \mathcal{T}, \mathcal{PK}$) $(g, h) \leftarrow \text{pp}$ $b \leftarrow 1$ for each asset a in tx: $b_{\text{BA}} := (\prod_{p=1}^{ \mathcal{PK} } \text{cm}_{p,a} \stackrel{?}{=} 1)$ for each party p in tx_a : $(\pi^{A*}, \pi^C, \pi^{C'}, \pi^{\text{EQ}}) := \pi_{p,a}$ $(\text{cm}, \text{tk}, \text{cm}', \text{tk}') := \text{tx}_{p,a}$ $b_C := \text{ZKVerify}_{\mathcal{L}_{\mathcal{R}_C}}(\text{crs}, (\text{cm}, \text{tk}), \pi^C)$ $b_{C'} := \text{ZKVerify}_{\mathcal{L}_{\mathcal{R}_C}}(\text{crs}, (\text{cm}', \text{tk}'), \pi^{C'})$ $b_{\text{EQ}} := \text{ZKVerify}_{\mathcal{L}_{\mathcal{R}_{\text{EQ}}}}(\text{crs}, (\text{cm}/\text{cm}', \text{tk}/\text{tk}'), \pi^{\text{EQ}})$ $\text{cm}_\Pi := (\prod_{i=1}^{ \mathcal{T} } \text{cm}_{i,p}) \cdot \text{cm}$ $\text{cm}'_\Pi, \pi^A := \pi^{A*}$ $b_A := \text{ZKVerify}_{\mathcal{L}_{\mathcal{R}_A}}(\text{crs}, (\text{cm}_\Pi, \text{cm}'_\Pi, 2^N), \pi^A)$ $b_{p,a} := b_{\text{BA}} \wedge b_C \wedge b_{C'} \wedge b_{\text{EQ}} \wedge b_A$ $b := b \wedge b_{p,a}$ return b</p>	
<p>BCast.GetProofs(tx) $(\text{aux}, \text{tx}) := \text{tx}$ En = [] for each participant p and asset a : parse $\text{cm}_{p,a}, \text{tk}_{p,a}$ from tx En$_p, a \leftarrow \text{P.GetEndorsement}_p(\text{cm}_{p,a}, \text{tk}_{p,a}, \text{aux})$ En.append(En$_p, a$) return En</p>	<p>PACT.Endorse($v_p, \text{Pol}_p, \text{sk}_p, \text{aux}$) $\text{cm}', r' := \text{PACT.Mint}(v_p)$ $\text{tk}' := \text{pk}_p^{r'}$ $\pi^{C'} := \text{ZKProve}_{\mathcal{L}_{\mathcal{R}_C}}(\text{crs}, (\text{cm}', \text{tk}'), (v_p, r'))$ $\pi^{\text{EQ}} := \text{ZKProve}_{\mathcal{L}_{\mathcal{R}_{\text{EQ}}}}(\text{crs}, (\text{cm}/\text{cm}', \text{tk}/\text{tk}'), (\text{sk}_p))$ $\text{cm}_\Pi := (\prod_{i=1}^{ \mathcal{T} } \text{cm}_{i,p}) \cdot \text{cm}$ $v_\Sigma := (\sum_{i=1}^{ \mathcal{T} } v_{i,p}) + v_p$ $\text{cm}'_\Pi, r_\Pi := \text{PACT.Mint}(v_\Sigma)$ $\pi^A := \text{ZKProve}_{\mathcal{L}_{\mathcal{R}_A}}(\text{crs}, (\text{cm}_\Pi, \text{cm}'_\Pi, 2^N), (v_\Sigma, r_\Pi, \text{sk}_p))$ $\pi^{A*} := (\text{cm}'_\Pi, \pi^A)$ return $(\text{cm}', \text{tk}', \pi^{A*}, \pi^{C'}, \pi^{\text{EQ}})$ if $\text{Pol}(v_p, \text{aux}) = 1$ else return \perp</p>	
<p>P.GetEndorsement(cm, tk, aux) $(\text{sk}_p, \text{pk}_p, \text{Pol}_p) := \text{stat}_p$ $v_p := \text{extract}(\text{cm}, \text{tk}, \text{sk}_p)$ return $\text{PACT.Endorse}(v_p, \text{Pol}_p, \text{sk}_p, (\text{cm}, \text{pk}_p, \text{aux}))$</p>		

Table 1: The pseudocode of PADL for the algorithms in Definition 3. Without loss of generality, we assumed transaction value array in \mathcal{V} contains entries for all \mathcal{PK} , otherwise the array is zero-padded in appropriate missing entry. Algorithm BCast.GetProofs is executed by broadcaster, $\text{P.GetEndorsement}_p$ and PACT.Endorse_p are executed by p -participant.

proof, and also verifies c_1 and c_2 with the ledger. It is also easily inferred that using the basic auditing of the asset sum of transactions values, ratio, variance, and concentrations can be calculated as mentioned previously [32] for non-confidential assets. Next we show proofs that do not share the sum of values.

Asset Liquidity PADL can maintain a large amount of confidential assets. When used in a debit market scenario, these assets can be bonds or other type of loan asset tokens as well as currency tokens. The PADL transaction scheme allows a participant to be audited for credit or liquidity of an asset by generating a ‘liquidity’ proof. This can be done by calculating fractional ratios among different assets for multiple auditors and multiple participants without revealing any further information about the balance of the assets. The participant proves that given a Ledger, and using NIZK proof that the ratio of its investment in an asset to its total assets value is lower than a desired threshold, as follows:

For a given rational number, $f \in \mathbb{Q}$ and two integers, D, N such that $D/N = f$, participant, p proves for index asset, a^* its liquidity that:

$$\frac{\sum_{t \in Txs} v_{t,p,a^*}}{\sum_{a \in A} \sum_{t \in Txs} v_{t,p,a}} < f$$

The *prover* and *verifier* calculate from the ledger the two points,

$$\prod_{t \in Txs} cm'_{t,p,a^*} := c_1, \prod_{a \in A} \prod_{t \in Txs} cm'_{t,p,a} := c_2, \text{ and, } c_r = c_2^D / c_1^N.$$

The *prover* calculates: $v_r = D \sum_{a \in A} \sum_{t \in Txs} v_{t,p,a} - N \sum_{t \in Txs} v_{t,p,a^*}$.

The *prover* sends a range-proof, π^{AL} with the ratio commit, c_r , to prove that v_r lies in interval $[0..2^n - 1]$. *verifier* verifies the range-proof, π^{AL} and that $c_2^D / c_1^N = c_r$ using the ledger and (a^*, D, N) .

Inter-transactions Rate Proof In decentralized finance applications, transactions for multiple investors may consider fractional interest rates. These rates can be proved as part of the transaction verification process or for auditing purposes using Σ -protocol proofs between cells in the ledger. This approach can be used to conceal investors’ strategies while still being paid coupon payment in a verified manner (as discuss on Sec. 4, or to be used for concentration tractability in auditing, or trading in secondary markets. For a given rational fraction, *Rate*, participant p proves for asset a that:

$$\frac{\sum_{t \in txs_1 \subset T} v_{t,p,a} = \Sigma v_1}{\sum_{t \in txs_2 \subset T} v_{t,p,a} = \Sigma v_2} = Rate,$$

where $\sum_{t \in txs_1 \subset T} v_{t,p,a} = \Sigma v_1$ and $\sum_{t \in txs_2 \subset T} v_{t,p,a} = \Sigma v_2$ are the sum of values over two subsets of transactions. Here the *verifier* also provides two integers, D, N such that $D/N = Rate$. The *prover* and *verifier* calculate from the ledger:

$$\prod_{t \in txs_1} cm_{t,p,a} := c_1, \prod_{t \in txs_2} tk_{t,p,a} := \tau_1, \prod_{t \in txs_2} cm_{t,p,a} := c_2, \prod_{t \in txs_2} tk_{t,p,a} := \tau_2$$

and $c = c_1^N \cdot c_2^{-D}$, and $\tau = \tau_1^N \cdot \tau_2^{-D}$, and then *prover* provides the proof of knowledge of a solution (sk) that solves the equivalence DLP as follows:

For correctness, c, τ are shown to be:

$$c = g^{N\Sigma v_1 - D\Sigma v_2} h^{N\Sigma r_1 - D\Sigma r_2}, \quad \tau = \mathbf{pk}^{N\Sigma r_1 - D\Sigma r_2}$$

where $\Sigma r_1 = \sum_{t \in txs_1 \subset T} r_{t,p,a}$ and $\Sigma r_2 = \sum_{t \in txs_2 \subset T} r_{t,p,a}$. If $\sum v_1 / \sum v_2 = Rate$, then $c = h^{N\Sigma r_1 - D\Sigma r_2}$ and *prover* provides a proof for the equivalence of the DLP in F_P , $dlog_c \tau \equiv dlog_c c^{sk}$. Otherwise, $c^* = g^{v^*} h^{N\Sigma r_1 - D\Sigma r_2}$ and finding a solution for $dlog_{c^*} \tau$ is intractable.

5.5 Full Auditing and Customized Traceability

Our system provides a straightforward way to map auditing as required by regulators or parties. This is done by appending to a cell in a transaction, an additional tk, but with the auditing party’s public key. The ‘settlement’ bank use-case (Sec. 4) shows an example of such auditing, but this flexibility becomes even more powerful in a complex map of trust between parties with multiple assets. For example, in the debit market where loans contain multiple assets besides cash assets, and requires trusted parties as banks, custodians, and brokers.

6 Security and Privacy Evaluation

In this section, the proposed multi-asset transaction scheme PADL is analyzed in term of security, privacy and performance metrics. PADL should satisfy integrity preserving and anonymity preserving properties. A transaction scheme for a ledger-based system is considered integrity preserving if it maintains the state of the account balance properly. In PACT, it is required that the transaction scheme should never allows spending more than the account balance (double spending in the PADL context means spending more than the account balance), never allows asset creation in a transaction and never allows spending of balance from other accounts without owner endorsement. The aforementioned notions are captured in Definition 4. Integrity property is defined to preserve the invariant defined in Definition 4 and is formally defined in the Appendix. In contrast to Ring-CT in [37] where the recipient address is known, PACT scheme requires that the recipient and sender are anonymized because recipient address is fixed in PACT unlike stealth-address used in Monero transaction. In the context of PADL, the participant information, asset information and transaction value are anonymized among a anonymity set. The formal definitions for integrity and anonymity and their proofs can be found in Appendix C. Informally, integrity property preserves invariant. Asset overspending is prevented by the soundness of proof of asset and binding property of commitment scheme. Endorsement forgery is prevented by the discrete log hardness and zero-knowledge and soundness of proof. Transaction imbalance is prevented due to the hardness of computing discrete log of the commitment key h . Therefore, the system ensured that no asset is created and endorsements from account owners are required for the transaction to go through. In addition, PADL satisfies anonymity because the commitment hides the transaction value and the proofs generated are zero-knowledge. In addition, the tokens are just randomness. Therefore, the transaction scheme does not leak any information about the sender and recipient. In essence, PADL which is an instantiation of PACT scheme has the following security and privacy guarantee:

Definition 4 (Invariant). For the PACT transaction scheme to be integrity preserving, it is required that the following invariant hold:

- **I1 Asset balance:** For each party p and each asset a in a transaction tx , the spending (negative) amount v must be smaller than the current asset value $|v| \leq \sum v_{i,p,a}$ where i ranges over all transactions before the current transaction.
- **I2 Account Owner Endorsement:** For each account pk_p in a transaction tx , the endorser should possess the corresponding account secret sk_p .
- **I3 Transaction balance:** For a transaction with some transaction row index t and each asset a , it must be the case that the transaction value summed to zero, $\sum_{p=1}^{|\mathcal{PK}|} v_{t,p,a} = 0$.

Theorem 1. Let the underlying proofs system used be zero-knowledge and sound, the commitment scheme used has binding property and is homomorphic, and discrete log is hard, then PADL presented in Table 1 satisfied integrity defined in Definition 8 at Appendix C.

Theorem 2. Let the commitment scheme used has hiding property and underlying proof system used is zero-knowledge, then PADL presented in Table 1 satisfied anonymity defined in Definition 9 at Appendix C.

Proofs for both theorem above are provided in Appendix C.

7 Performance Analysis

We present PADL as a comprehensive package for constructing ledgers (link to code[†]), both as standalone systems and integrated with smart contracts. The implementation of PADL is designed to be modular, enabling the development of new financial use cases. It employs RUST for cryptographic protocols and primitives to achieve high performance and tested code, but the ledger construction is interfaced and packaged with Python to obtain flexibility and accessibility.

Ledger	Type (section)	Size (bytes)	Time/txn (sec)	Assets/Banks
Simple Exchange	Tx (Sec. 4.1)	4,704	0.34 \pm 0.01	2/2
Settlement Trusted Bank	Tx (Sec. 4.2)	3,726	0.21 \pm 0.001	1/3
Bond Market	Tx (Sec. 4.3)	16,464	0.41 \pm 0.03	2/7
proofs+commits	Cell (Sec. 3)	1,176	-	1/1
asset balance	proofs (Sec. 5.4)	98	-	-
inter-transactions	proofs (Sec. 5.4)	98	-	-
asset liquidity	proofs (Sec. 5.4)	688	-	-

Table 2: A table with the main experimental use-cases from Sec. 4 and individual auditing proofs. It includes the size of the cell, and the size and time of the transaction, and the number of assets/banks. We performed 10 runs for each use-case to get the average Time/txn and the \pm confidence intervals.

A performance analysis was conducted for the PADL implementation, alongside the zkLedger package and its default configuration. We summarize the transaction size and speed in PADL, the size of auditing proofs, and provide comparative metrics with zkLedger where feasible. From the results in [32], it is observed that the size of the range proof is 3.9KB, compared to 1.2KB for the entire commits and proofs in PADL (range proof for $2^{32} - 1$). Besides the utilization of Bulletproofs in PADL, another reason for the reduced size in PADL is related to the Disjunctive proof present in zkLedger.

Table 2 also illustrates the execution times for the examples discussed in Sec. 4. For the bond market example involving 2 assets and 7 participants, PADL requires approximately 0.41 seconds on average per transaction. Auditing proofs primarily involve homomorphic arithmetic on the set of transactions in the ledger and the construction of a Σ -protocol transcript proof. The results demonstrate the practicality and scalability of the solution within financial systems. Figure 2 depicts the performance characteristics of both systems under test. As observed in Fig. 2a, there is a linear increase in execution time (seconds, y-axis) relative to the number of assets (x-axis) for zkLedger when compared to PADL. Due to the asynchronous generation of proof of Asset, PADL exhibits better scalability with an increasing number of banks or assets, whereas zkLedger incurs a computationally costly overhead with significant increments in either parameter.

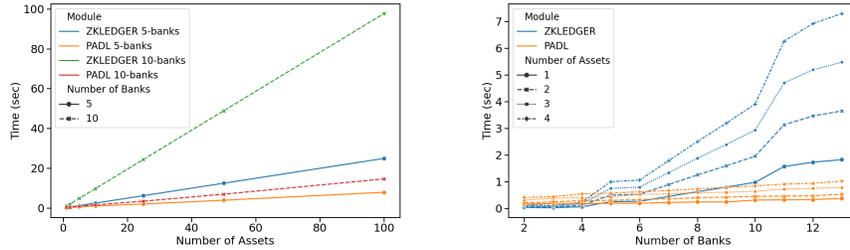


Fig. 2: PADL speed benchmark vs zkLedger algorithm. The y-axis in both figures shows the time (s) it takes for PADL and zkLedger to complete 4 transactions (proving and verification). In subplot **a** we show over different number of assets in the x-axis and in subplot **b** over different number of participants-banks. The type of lines and markers show different configurations for the benchmark. The average error (standard deviation in seconds)

8 Conclusions and Future Work

PADL proposes a combined system for a transaction scheme to hide values and balances of assets in distributed ledgers. The ledger focuses on developing financial institutes applications with confidential multi-assets. The proposed scheme with the security evaluation of integrity and anonymity of transaction, shows the general purpose usage of the transaction scheme with privacy preserving auditing and/or ‘full’ auditing. Performance evaluation of the ledger shows that

the PADL transaction scheme is scaled well with number of participants due to asynchronous proof generation and batch verification, as well as shorter transaction size when comparing to previous ‘table’ ledgers, such as zkLedger. This is also due to a simplified cryptography scheme used here with complementary commits and equivalence proof. Future direction to expand the framework is to include further set of auditing proofs related to membership and digital identity, which is required for; categories, sanction list membership, as well as additional proofs in the flow of trades of various complex financial markets.

9 Acknowledgment

We would like to acknowledge Onyx and Blockchain Engineer team at JPMorgan Chase for their support throughout this project. In particular, we thank Sudhir Upadhyay for his support and advice through-out the work. We also appreciate the assistance of Sai Valiveti, Joe Leung, and Imran Bashir, as well as the support provided by Suresh Shetty. Additionally, we thank Lianna Zhao for her help in the revision process.

10 Disclaimer

This paper was prepared for informational purposes by the Global Technology Applied Research Center of JPMorgan Chase & Co. This paper is not a product of the Research Department of JPMorgan Chase & Co. or its affiliates. Neither JPMorgan Chase & Co. nor any of its affiliates makes any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, without limitation, with respect to the completeness, accuracy, or reliability of the information contained herein and the potential legal, compliance, tax, or accounting effects thereof. This document is not intended as investment research or investment advice, or as a recommendation, offer, or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction.

References

1. I. Alarab, S. Prakoonwit, and M. I. Nacer. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th international conference on machine learning technologies*, pages 23–27, 2020.
2. S. Ames, C. Hazay, Y. Ishai, and M. Venkatasubramanian. Liger: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 acm sigsac conference on computer and communications security*, pages 2087–2104, 2017.
3. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

4. E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhyaoui, and B. Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. *Cryptology ePrint Archive*, 2019.
5. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, page 46, 2018.
6. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct Non-Interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 781–796, San Diego, CA, Aug. 2014. USENIX Association.
7. R. Bhadauria, Z. Fang, C. Hazay, M. Venkatasubramanian, T. Xie, and Y. Zhang. Ligero++: A new optimized sublinear iop. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2025–2038, 2020.
8. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. Association for Computing Machinery, New York, NY, USA, 2019.
9. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
10. E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi. Solidus: Confidential distributed ledger transactions via pvorm. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 701–717, 2017.
11. P. Chatzigiannis and F. Baldimtsi. Miniledger: compact-sized anonymous and auditable distributed payments. In *European Symposium on Research in Computer Security*, pages 407–429. Springer, 2021.
12. R. Cramer. Modular design of secure yet practical cryptographic protocols. *Ph. D.-thesis, CWI and U. of Amsterdam*, 2, 1996.
13. G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 720–731. ACM Press, Oct. 2015.
14. D. Ding, K. Li, L. Jia, Z. Li, J. Li, and Y. Sun. Privacy protection for blockchains with account and multi-asset model. *China Communications*, 16(6):69–79, jul 2020.
15. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO’86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, Berlin, Heidelberg, Aug. 1987.
16. Z. Gao, L. Xu, K. Kasichainula, L. Chen, B. Carbunar, and W. Shi. Private and atomic exchange of assets over zero knowledge based payment ledger. *arXiv preprint arXiv:1909.06535*, 2019.
17. C. Garman, M. Green, and I. Miers. Accountable Privacy for Decentralized Anonymous Payments. *Cryptology ePrint Archive*, 2016.
18. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal of Computing*, pages 186–208, 1989.
19. J. Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 341–358. Springer, 2010.

20. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer, Berlin, Heidelberg, Apr. 2015.
21. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, Berlin, Heidelberg, May / June 2006.
22. G. Jeong, N. Lee, J. Kim, and H. Oh. Azeroth: Auditable zero-knowledge transactions in smart contracts. *IEEE Access*, 2023.
23. W. Jia, T. Xie, and B. Wang. A privacy-preserving scheme with multi-level regulation compliance for blockchain. *Scientific Reports 2024 14:1*, 14(1):1–14, jan 2024.
24. H. Kang, T. Dai, N. Jean-Louis, S. Tao, and X. Gu. Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 543–555. IEEE, 2019.
25. H. A. Khattak, K. Tehreem, A. Almogren, Z. Ameer, I. U. Din, and M. Adnan. Dynamic pricing in industrial internet of things: Blockchain application for energy management in smart cities. *Journal of Information Security and Applications*, 55:102615, 2020.
26. A. Kiayias, M. Kohlweiss, and A. Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. Cryptology ePrint Archive, Paper 2022/974, 2022. <https://eprint.iacr.org/2022/974>.
27. Y. Li, W. Susilo, G. Yang, Y. Yu, X. Du, D. Liu, and N. Guizani. Toward privacy and regulation in blockchain-based cryptocurrencies. *IEEE Network*, 33(5):111–117, 2019.
28. Y. Li, G. Yang, W. Susilo, Y. Yu, M. H. Au, and D. Liu. Traceable monero: Anonymous cryptocurrency with enhanced accountability. *IEEE Transactions on Dependable and Secure Computing*, 18(2):679–691, 2019.
29. U. M. Maurer. Unifying zero-knowledge proofs of knowledge. In B. Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286. Springer, Berlin, Heidelberg, June 2009.
30. S. Meralli. Privacy-preserving analytics for the securitization market: a zero-knowledge distributed ledger technology application. *Financial Innovation*, 6(1):7, 2020.
31. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *SSRN Electronic Journal*, 2008.
32. 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 18)*, pages 65–80, 2018.
33. S. Noether, A. Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.
34. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Berlin, Heidelberg, Aug. 1992.
35. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

36. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology – CRYPTO’89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, New York, Aug. 1990.
37. S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen. RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In S. N. Foley, D. Gollmann, and E. Snekkenes, editors, *ESORICS 2017: 22nd European Symposium on Research in Computer Security, Part II*, volume 10493 of *Lecture Notes in Computer Science*, pages 456–474. Springer, Cham, Sept. 2017.
38. X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.
39. G. Wood. Ethereum: A secure decentralised generalised transaction ledger, 2019.
40. K. Wüst, K. Kostianen, N. Delius, and S. Capkun. Platypus: A Central Bank Digital Currency with Unlinkable Transactions and Privacy Preserving Regulation. *Cryptology ePrint Archive*, 1:2947–2960, nov 2021.
41. L. Xu, L. Chen, Z. Gao, K. Kasichainula, M. Fernandez, B. Carbutar, and W. Shi. PrivateEx: Privacy preserving exchange of crypto-assets on blockchain. *Proceedings of the ACM Symposium on Applied Computing*, 8(20):316–323, mar 2020.
42. L. Xu, Y. Zhang, and L. Zhu. Regulation-friendly privacy-preserving blockchain based on zk-snark. In *International Conference on Advanced Information Systems Engineering*, pages 167–177. Springer, 2023.
43. T. H. Yuen, S. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu. RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security. In J. Bonneau and N. Heninger, editors, *FC 2020: 24th International Conference on Financial Cryptography and Data Security*, volume 12059 of *Lecture Notes in Computer Science*, pages 464–483. Springer, Cham, Feb. 2020.
44. X. Zhi, Y. Satsangi, S. Moran, and S. Eloul. Ledgit: A service to diagnose illicit addresses on blockchain using multi-modal unsupervised learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM ’22, page 5069–5073, New York, NY, USA, 2022. Association for Computing Machinery.

Appendix

A Notations

\mathbb{G}	Group	π^A	Proof of asset
g, h	Generators	π^C	Proof of consistency
\mathbb{Z}_P	Scalar field	π^{EQ}	Proof of equivalence
p	Participant index	En	Endorsement data
t	Transaction index	\mathcal{PK}	Public key list
a	Asset index	\mathcal{V}	Transaction value list
sender	The participant who executes <code>PACT.Spend</code>		
T, \mathcal{T}	All transactions		
A, \mathcal{AS}	All assets		
P	All participants		
Cell	Entry in transaction row		
tx	Cells representing a transaction		
txs	Set of tx		
v	Transaction value		
r	Blinding factor / Randomness		
Pol	A predicate policy function		
cm	Pedersen Commitment		
cm'	Complementary Pedersen Commitment		
tk	Token		
tk'	Complementary Token		
π	Set of proofs for a tx		

B Use-Cases Transaction Examples

In this section we provide an example of construction of transaction commitments by the broadcaster and the setup of the ledger. We show examples of transaction for the experimental use cases discussed in Sec. 4.

B.1 Bond exchange and Coupon rate

We assume a simple use case of 2 investors that invest in bonds, with 2-year maturity, 10% yearly coupon rate, and a par value of 10\$ for each bond unit. In this exchange, there are several actors: custodian is the financial institution that safe keeps the asset of the investors (here USD) and can issue a digital coin/token that is pegged to fiat currency, USD. The bond issuer issues the bonds, and borrows USD. The broker is going to manage the deal, and the exchange between the investors and bond issuer.

Privacy assumptions:

- The broker knows the details of the bond deal: How many bonds unit each investor buys.
- The custodian has no other knowledge except the amount of money it issues to the investors.
- The bond issuer does not know the individual investors, just the sum of USD asset received from many (two) investors.
- The investors do not know about other investors values in the t .
- All participants (p) in the ledger know the time that an event of transaction (t) happened on ledger, but the content is encrypted.

In PADL we create a ledger that includes all participants. There is an initialisation where issuers are capable to mint assets to the ledger ($tx0$). Let us assume the bond issuer data: bond X , 2-years maturity, and 100 units with value 1,000\$, with yearly coupon rate of 10%. Asset name: $X10\$2y10\%$. The Custodian mint 3000\$, and the Bond issuer with mint 1000\$ and 300 X . The following transactions that form the ledger do not include all the information that the user will see in the PADL ledger. We have omitted items, such as proofs and complimentary data, and we have kept only the commits (cm) and the tokens (tk) for clarity.

2024 Jan: $tx0$, initialisation

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{3000\$} h^{r0a0}, pk^{r0a0}$	$g^{1000\$} h^{r0b0}, pk^{r0b0}$	$g^{0\$} h^{r0c0}, pk^{r0c0}$	$g^{0\$} h^{r0d0}, pk^{r0d0}$	$g^{0\$} h^{r0e0}, pk^{r0e0}$
1	$g^{0X} h^{r0a1}, pk^{r0a1}$	$g^{300X} h^{r0b1}, pk^{r0b1}$	$g^{0X} h^{r0c1}, pk^{r0c1}$	$g^{0X} h^{r0d1}, pk^{r0d1}$	$g^{0X} h^{r0e1}, pk^{r0e1}$

The notations above, for g, h, pk, r are explained in Sec. 3. The r (blending factors) is new for each t (transaction) and participant, as indicated by their indices.

$tx1$ and $tx2$ are USD-tokens with one custodian for both investors. The custodian moves money to investors, to mint tokens of stable USD as requested by the investors (2,000\$ to each investor).

2024 Jan: $tx1$

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{-2,000\$} h^{r1a}, pk^{r1a}$	$g^{0\$} h^{r1b}, pk^{r1b}$	$g^{0\$} h^{r1c}, pk^{r1c}$	$g^{2,000\$} h^{r1d}, pk^{r1d}$	$g^{0\$} h^{r1e}, pk^{r1e}$

2024 Jan: $tx2$

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{-2,000\$} h^{r2a}, pk^{r2a}$	$g^{0\$} h^{r2b}, pk^{r2b}$	$g^{0\$} h^{r2c}, pk^{r2c}$	$g^{0\$} h^{r2d}, pk^{r2d}$	$g^{2,000\$} h^{r2e}, pk^{r2e}$

Broker broadcast a $tx3$ on PADL, a single exchange of bonds unit with lending to the bond issuer.

- Investor M sends 1,000\$ to the bond issuer and gets 100X units,
- Investor N gets 200X units for 2,000\$, the bond issuer sees 3,000\$ received in the t .

No one besides the broker can tell what the blending values (r 's) are or what are the invested \$/bond values of the other participants. They can only verify their values. For example, the issuer can only infer that the money it received in the t is 3,000\$, and the bond unit it sent is at a value of 300X.

2024 Jan: $tx3$, exchange

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{0\$} h^{r3a}, pk^{r3a}$	$g^{3,000\$} h^{r3b}, pk^{r3b}$	$g^{0\$} h^{r3c}, pk^{r3c}$	$g^{-1,000\$} h^{r3d}, pk^{r3d}$	$g^{-2,000\$} h^{r3e}, pk^{r3e}$
1	$g^{0X} h^{r4a}, pk^{r4a}$	$g^{-300X} h^{r4b}, pk^{r4b}$	$g^{0X} h^{r4c}, pk^{r4c}$	$g^{100X} h^{r4d}, pk^{r4d}$	$g^{200X} h^{r4e}, pk^{r4e}$

Broker also sends to the issuer of the bond, the future coupons t so the issuer can broadcast these t every year. Only the broker knows the values put for investor B and M , the bond issuer knows the total, and can verify the rate but does not know about the individual values of the investors.

The bond issuer keeps the encrypted t and will send the first coupon t to the ledger at the end of the year and the second at the end of the second year. The

2025 Jan: tx4, Year 1 coupon

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{0\$}h^{r5a}, pk^{r5a}$	$g^{-300\$}h^{r5b}, pk^{r5b}$	$g^{0\$}h^{r5c}, pk^{r5c}$	$g^{100\$}h^{r5d}, pk^{r5d}$	$g^{200\$}h^{r5e}, pk^{r5e}$
1	$g^{0X}h^{r6a}, pk^{r6a}$	$g^{0X}h^{r6b}, pk^{r6b}$	$g^{0X}h^{r6c}, pk^{r6c}$	g^0h^{r6d}, pk^{r6d}	$g^{0X}h^{r6e}, pk^{r6e}$

2026 - Jan: tx5, Year 2 coupon

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{0\$}h^{r7a}, pk^{r7a}$	$g^{-300\$}h^{r7b}, pk^{r7b}$	$g^{0\$}h^{r7c}, pk^{r7c}$	$g^{100\$}h^{r7d}, pk^{r7d}$	$g^{200\$}h^{r7e}, pk^{r7e}$
1	$g^{0X}h^{r8a}, pk^{r8a}$	$g^{0X}h^{r8b}, pk^{r8b}$	$g^{0X}h^{r8c}, pk^{r8c}$	g^0h^{r8d}, pk^{r8d}	$g^{0X}h^{r8e}, pk^{r8e}$

issuer cannot tell how the money is distributed between the investors, only that its total rate, 300\$ is the right amount recorded in these t .

Proof of the coupon rate. This is an additional proof to make sure broker and participants are honest in the coupon t , even though everyone may be fine with its money received. All participants in addition to the general proofs in PADL t , creates a proof of ratio equivalence. Each participant add this prove to the cell when they add the rest of the proofs of the cell.

Finally, in maturity of the bond issuer returns the money and investor return the bonds units. The Broker also gets 0.1% fees of the total bonds for its hard work. The fees proof can be also added to the cell using the rate proof to ensure the Broker is paid exactly 0.1% of everyone.

2026 - maturity: Transaction Row, tx6, return assets and fees

Asset	Custodian	Bond Issuer	Broker	Investor M	Investor N
0	$g^{0\$}h^{r9a}, pk^{r9a}$	$g^{-3,003\$}h^{r9b}, pk^{r9b}$	$g^{6\$}h^{r9c}, pk^{r9c}$	$g^{999\$}h^{r9d}, pk^{r9d}$	$g^{1998\$}h^{r9e}, pk^{r9e}$
1	$g^{0X}h^{r10a}, pk^{r10a}$	$g^{300X}h^{r10b}, pk^{r10b}$	$g^{0X}h^{r10c}, pk^{r10c}$	$g^{-100X}h^{r10d}, pk^{r10d}$	$g^{-200X}h^{r10e}, pk^{r10e}$

B.2 Settlement Bank Example

The following transactions are exemplary of Sec. 4.2. Assume we have a settlement bank and two counter-party banks. In total the ledger has 3 participants. The two counter-party banks, Bank A and Bank B, wish to make a payment on assets. For example, Bank A wants to trade money market funds (short term debt) for USD-tokens from Bank B, to cover short term liquidity requirements. They can do this through a trusted participant (settlement bank) that will ensure that both parties have adequate funds to complete the transaction. The

settlement party has to be able to verify the balance without having access to the amounts or assets traded.

Privacy assumptions:

- A trusted party (p) to act as the settlement bank and be able to audit the transactions (t).
- The two counterparty banks (Bank A and Bank B) will not share their private keys (sk) with the settlement party.

Similarly to the **Bond Coupon Simple Example**, PADL creates a ledger with an initialisation line ($tx0$).

2024 Jan: $tx0$			
Asset	Settlement Bank (Issuer)	Bank A	Bank B
0	$g^{0\$}h^{r0a}, pk^{r0a}$	$g^{0\$}h^{r0b}, pk^{r0b}, pk_I^{r0b}$	$g^{2000\$}h^{r0c}, pk^{r0c}, pk_I^{r0c}$
1	$g^{0\$}h^{r1a}, pk^{r1a}$	$g^{10MM\$}h^{r1b}, pk^{r1b}, pk_I^{r1b}$	$g^{0\$}h^{r1c}, pk^{r1c}, pk_I^{r1c}$

Compared to the previous example, we add an additional token (tk_I) that signs $r_{t,a,p}$, by the public key to the tokens of the issuer I . The initial $tx0$ is USD-tokens with Bank B and Bank A starts with market fund $10MM$. In the subsequent transaction ($tx1$), Bank A will send 10 money market funds ($10MM$), with value $200\$$ each, for exchange for $2,000\$$ USD tokens. The settlement bank keeps the money market funds ($10MM$) until Bank A repays back the $2,000\$$ in **a months' time**.

2024 Jan: $tx2$			
Asset	Settlement Bank	Bank A	Bank B
0	$g^{0\$}h^{r2a}, pk^{r2a}$	$g^{2,000\$}h^{r2b}, pk^{r2b}, pk_I^{r2b}$	$g^{-2,000\$}h^{r2c}, pk^{r2c}, pk_I^{r2c}$
1	$g^{10MM\$}h^{r3a}, pk^{r3a}$	$g^{-10MM\$}h^{r3b}, pk^{r3b}, pk_I^{r3b}$	$g^0h^{r3c}, pk^{r3c}, pk_I^{r3c}$

At the end of the month, Bank A sends to Bank B the $2,000\$$ value and the settlement bank releases the money market funds back to Bank A.

In all cells, the settlement bank can extract the value using the commit and its corresponding tk_I .

C Security and Privacy Analysis

The security and privacy guarantees of PADL is stated formally here. First, we recall the standard cryptographic definitions that are omitted in the main body for both the commitment scheme and NIZK. Then, proof is given for both Integrity and Anonymity properties of PADL.

2024 Feb: tx3

Asset	Settlement Bank	Bank A	Bank B (Issuer)
0	$g^{0\$} h^{r4a}, pk^{r4a}$	$g^{-2,000\$} h^{r4b}, pk^{r4b}, pk_I^{r4b}$	$g^{2,000\$} h^{r4c}, pk^{r4c}, pk_I^{r4c}$
1	$g^{-10MM} h^{r5a}, pk^{r5a}$	$g^{10MM} h^{r5b}, pk^{r5b}, pk_I^{r5b}$	$g^{0MM} h^{r5c}, pk^{r5c}, pk_I^{r5c}$

C.1 Additional Background

Definition 5 (Properties of Commitment Scheme). A non-interactive scheme $(\text{CKeyGen}, \text{Com})$ is called a commitment scheme if it satisfies the following properties.

Hiding. Let the advantage $\text{Hid}^A(\lambda)$ of breaking the binding property be the probability defined below. A scheme satisfies hiding if for all PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that

$$\left| \Pr \left[\mathcal{A}(\text{cm}) = b : \begin{array}{l} \text{ctk} \leftarrow \text{CKeyGen}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(\text{ctk}) \\ b \leftarrow_{\$} \{0, 1\}; \text{cm} \leftarrow \text{Com}(\text{ctk}, m_b) \end{array} \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

Binding. Let the advantage $\text{Bind}^A(\lambda)$ of breaking the binding property be the probability defined below. A scheme satisfies binding if for all PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that

$$\Pr \left[m_0 \neq m_1 \wedge \text{Com}(m_0, r_0) = \text{Com}(m_1, r_1) : \begin{array}{l} \text{ctk} \leftarrow (\text{CKeyGen}(1^\lambda); \\ (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\text{ctk}) \end{array} \right] \leq \text{negl}(\lambda)$$

Definition 6 (Properties of NIZK). A NIZK proof system has the following properties.

Completeness. For all security parameters λ , all statements $\text{stmt} \in \mathcal{L}_{\mathcal{R}}$, all witness wit with $\mathcal{R}(\text{stmt}, \text{wit}) = 1$, let $\text{crs} \leftarrow \text{ZKSetup}(\lambda)$ and $\pi_{\text{ZKP}} \leftarrow \text{ZKProve}(\text{crs}, \text{stmt}, \text{wit})$, it holds that $\text{ZKVerify}(\text{crs}, \text{stmt}, \pi_{\text{ZKP}}) = 1$.

Computational Soundness. The proof system is computationally sound if for all PPT adversaries \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that the following holds:

$$\Pr \left[\begin{array}{l} \text{ZKVerify}(\text{crs}, \text{stmt}, \pi_{\text{ZKP}}) = 1 : \\ \text{crs} \leftarrow \text{ZKSetup}(\lambda); \\ (\pi_{\text{ZKP}}, \text{stmt}) \leftarrow \mathcal{A}(\text{crs}); \\ \text{stmt} \notin \mathcal{L}_{\mathcal{R}} \end{array} \right] \leq \text{negl}(\lambda)$$

Zero-Knowledge The advantage $\text{ZK}^A(\lambda)$ of an adversary \mathcal{A} in breaking the zero-knowledge property is the probability defined below. The proof system is zero-knowledge if there exists simulator algorithms $(\text{Sim}_1, \text{Sim}_2)$ such that for all PPT adversaries \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that $\text{ZK}^A(\lambda) \leq$

$\text{negl}(\lambda)$.

$$\left| \Pr \left[\begin{array}{l} \mathcal{A}(\pi_{\text{ZKP}}^*) = 1 : \\ \text{crs} \leftarrow \text{ZKSetup}(\lambda); \\ (\text{stmt}, \text{wit}) \leftarrow \mathcal{A}(\text{crs}); \\ \mathcal{R}(\text{stmt}, \text{wit}) = 1; \\ \pi_{\text{ZKP}}^* \leftarrow \text{ZKProve}(\text{crs}, \\ \text{stmt}, \text{wit}) \end{array} \right] - \Pr \left[\begin{array}{l} \mathcal{A}(\pi_{\text{ZKP}}^*) = 1 : \\ (\text{crs}, \text{st}) \leftarrow \text{Sim}_1(\lambda); \\ (\text{stmt}, \text{wit}) \leftarrow \mathcal{A}(\text{crs}); \\ \mathcal{R}(\text{stmt}, \text{wit}) = 1; \\ \pi_{\text{ZKP}}^* \leftarrow \text{Sim}_2(\text{crs}, \text{stmt}, \text{st}) \end{array} \right] \right|$$

Generalized Zero-Knowledge Proof for Group Homomorphism. We rely on the following generalized special honest-verifier zero-knowledge proof of knowledge (ZKPoK) Σ -protocol for preimage of a group homomorphism by Maurer [29].

Theorem 3. *Let (G_1, \star) and (G_2, \circ) be two groups, and $\phi : G_1 \rightarrow G_2$ be a group homomorphism such that $\phi(a \star b) = \phi(a) \circ \phi(b)$. Let $\ell \in \mathbb{Z}$, $u \in G_1$, and $z = \phi(x)$ are known such that:*

- (1) $\forall c_1, c_2 \in \mathcal{C}$ with $c_1 \neq c_2$, $\text{gcd}(c_1 - c_2, \ell) = 1$
- (2) $\phi(u) = z^\ell$

then a ZKPoK for a given z , of a value x such that $z = \phi(x)$ can be constructed by having the prover samples $k \leftarrow G_1$ and computes $t := \phi(k)$, verifier then generates a challenge $c \leftarrow \mathcal{C}$, prover then outputs $r := k \star x^c$, and finally the verifier checks $\phi(r) \stackrel{?}{=} t \circ z^c$.

Using the theorem defined above, we can now show that proof of consistency defined in Section 5.3 is a ZKPoK protocol.

Theorem 4. *Let \mathbb{G} to be p -prime ordered group and $g, h, \text{pk} \in \mathbb{G}$. Then, proof of consistency defined in Section 5.3 is a ZKP protocol for the language $\mathcal{L}_{\mathcal{R}_c} := \{(\text{cm}, \text{tk}) : \exists v, r \text{ s.t. } \text{cm} = g^v h^r \wedge \text{tk} = \text{pk}^r\}$.*

Proof. Let $G_1 = \mathbb{Z}_p \times \mathbb{Z}_p$ with its group operation being component-wise addition and $G_2 = \mathbb{G} \times \mathbb{G}$ with its group operation being component-wise. Then $\phi(v, r) := (g^v h^r, \text{pk}^r)$ is a group homomorphism as $\phi(v_1, r_1) \circ \phi(v_2, r_2) = (g^{v_1+v_2} h^{r_1+r_2}, \text{pk}^{r_1+r_2}) = \phi(v_1 + v_2, r_1 + r_2)$. Let $\ell = p$ and $u = (0, 0)$, $\forall z \in G_2$, $z^\ell = (1, 1) = \phi(u)$ satisfied the requirements for Theorem 3. We can now apply Theorem 3 to obtain a ZKP protocol for $\mathcal{L}_{\mathcal{R}_c}$ and then apply Fiat-Shamir transform to obtain the corresponding NIZK. \square

C.2 Model and Analysis

The state of PADL ledger is interacted with exp oracles described in Definition 7. As discussed in Section 6, integrity property is derived to maintain the invariant

defined in Definition 4. Integrity property ensures that for each transaction, it always preserves asset balance, transaction balance and ownership endorsement requirements.

Definition 7 (Oracles). *The adversary will be given oracles with state of transactions \mathcal{L}_T , account public key list \mathcal{L}_{PK} , account key pair list \mathcal{L}_{ACC} , invalid transaction list \mathcal{L}_{CTx} , and corrupted key pair list \mathcal{L}_C defined as below depending on the experiments:*

- $\text{Exp.AddAcc}()$: the oracle executes $(sk, pk) \leftarrow \text{PACT.KeyGen}(1^\lambda)$, registers a new account under pk by updating its state with $\mathcal{L}_{PK} = \mathcal{L}_{PK} \cup \{pk\}$, appends (pk, sk) to \mathcal{L}_{ACC} , and returns address pk .
- $\text{Exp.PostTx}(tx, \pi)$: on input a transaction tx and a validity proof π , if $\text{PACT.Verify}(tx, \pi, \mathcal{L}_T, \mathcal{L}_{PK}) = 0$ or $(tx, \pi) \in \mathcal{L}_{CTx}$, the oracle returns failure bit $b = 0$ without appending the ledger, else the oracle appends its ledger \mathcal{L}_T with tx and returns success bit $b = 1$.
- $\text{Exp.Spend}(\mathcal{V})$: on input a transaction amount list \mathcal{V} , executes $(tx, \pi) \leftarrow \text{PACT.Spend}(\mathcal{V}, \mathcal{L}_T, \mathcal{L}_{PK})$, $(b) \leftarrow \text{Exp.PostTx}(tx, \pi)$, and the oracle returns \perp if $b = 0$, else returns (tx, π) .
- $\text{Exp.Corrupt}(pk)$: on input an account public key pk , the oracle retrieves account key pair $(pk, sk) \in \mathcal{L}_{ACC}$, appends (pk) to \mathcal{L}_C and returns sk .
- $\text{Exp.Policy}(p, \mathcal{F})$: on input a party identifier p , and a predicate function \mathcal{F} , the oracle executes $\text{Pol}_p := \mathcal{F}$.
- $\text{Exp.Latest}()$: the oracle returns $(\mathcal{L}_{PK}, \mathcal{L}_T)$.

Definition 8 (Integrity). *It holds that every PPT adversary \mathcal{A} has at most negligible advantage in the following experiment, where we define the advantage as $\text{Adv}_{\text{Integrity, PACT}} := \Pr[\text{Integrity}^{\mathcal{A}}(\lambda) = 1]$.*

$\text{Integrity}^{\mathcal{A}}$
$pp \leftarrow \text{PACT.Setup}(1^\lambda)$ $\mathcal{O} := \{\text{Exp.AddAcc}, \text{Exp.PostTx}, \text{Exp.Spend}, \text{Exp.Corrupt}, \text{Exp.Policy}, \text{Exp.Latest}\}$ $(tx, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(pp)$ $(\mathcal{PK}, \mathcal{T}) \leftarrow \text{Exp.Latest}()$ return 1 if $\text{PACT.Verify}(tx, \pi, \mathcal{T}, \mathcal{PK}) = 1$ and one of the following hold for any asset a in the transaction tx : (i) $\exists p$ s.t. $\sum_{i=1}^{ \mathcal{T} } v_{i,p,a} < v_{tx,p,a} $ when $v_{tx,p,a}$ is negative , (ii) $\exists p$ s.t. $pk_p \notin \mathcal{L}_C$, $\pi_{tx,p,a}^A$ and $\pi_{tx,p,a}^{EQ}$ are not generated by the challenger, (iii) $\sum_{i=1}^{ \mathcal{PK} } v_{tx,i,a} \neq 0$.

Theorem 5. *If the underlying proof system is sound and zero-knowledge, and the commitment scheme used is binding, then PADL is integrity preserving.*

Proof. Suppose there exists an efficient adversary \mathcal{A} that can break the integrity property of PADL with non-negligible probability, we exhaust three cases where (i) proof of asset asserts enough account balance even though there is not enough account balance to execute the transaction (ii) proof of asset is generated without account ownership (secret key) (iii) proof of balance for commitment zero out to group identity despite the committed value does not zero out.

Case 1 Asset Overspending: Given a transaction history \mathcal{T} , current transaction tx , and a party p , asset balance states that $\sum_{i=1}^{|\mathcal{T}|} v_{i,p,a} + v_{\text{tx},p,a} < 2^N$ where 2^N is some predetermined positive limit. \mathcal{A} wins in this case means $\sum_{i=1}^{|\mathcal{T}|} v_{i,p,a} + v_{\text{tx},p,a} \geq 2^N$ for some party p . Following the soundness of equivalence proof and the binding property of commitment scheme, the value of $\sum_{i=1}^{|\mathcal{T}|} v_{i,p,a} + v_{\text{tx},p,a}$ is committed under cm'_{Π} . We now construct a reduction that breaks the soundness of the proof system used for the range proof in proof of asset. The reduction simulates the whole system honestly. It follows that π_p^A is a proof for a false statement. Thus, by returning $((\text{cm}'_{\Pi}, 2^N), \pi_p^A)$, the reduction will break the soundness property of the proof system for the range proof used in proof of asset which contradicts the assumed soundness of the proof system used.

Case 2 Endorsement Forgery: Let π^A and π^{EQ} be the proofs that are not generated by the challenger and pk be the corresponding public key account. Following the discrete log hardness of h^{sk} , zero-knowledge property of π^A and π^{EQ} for previous transactions, if pk is never queried to the key retrieval oracle, then the challenger runs \mathcal{A} until the same inputs are queried twice to the random oracle with two different proof outputs given by \mathcal{A} where the proofs are forged for the same public key account satisfying the above criteria. From here, run the extractor algorithm from the Schnorr discrete log ZKP to obtain a solution for discrete log contradicting the assumed hardness of discrete log.

Case 3 Transaction (Im)Balance: Given a set of commitments $\mathcal{CM} := \{\text{cm}_1, \text{cm}_2, \dots, \text{cm}_{|\mathcal{PK}|}\}$ with the corresponding set of values $\mathcal{V} := \{v_1, v_2, \dots, v_{|\mathcal{PK}|}\}$ and randomness used $\mathcal{R} := \{r_1, r_2, \dots, r_{|\mathcal{PK}|}\}$ such that $\text{cm}_{\Pi} := \prod_{\text{cm} \in \mathcal{CM}} \text{cm} = g^{v_1 + v_2 + \dots + v_{|\mathcal{PK}|}} \cdot h^{r_1 + r_2 + \dots + r_{|\mathcal{PK}|}} = g^0 h^0 = 1$. Let \mathcal{V}^* and \mathcal{R}^* be different sets of values and randomness that one used in the transaction such that $v_1^* + v_2^* + \dots + v_{|\mathcal{PK}|}^* = v_{\Sigma}^* \neq 0$ and $r_{\Sigma}^* = \sum_{r^* \in \mathcal{R}^*} r^*$, then $g^{v_1^* + v_2^* + \dots + v_{|\mathcal{PK}|}^*} \cdot h^{r_1^* + r_2^* + \dots + r_{|\mathcal{PK}|}^*} = g^{v_{\Sigma}^*} g^{x(r_{\Sigma}^*)} = g^0 = 1$ which means $x = -(v_{\Sigma}^*)/r_{\Sigma}^*$ where $g^x = h$ and $r_{\Sigma}^* \neq 0$. Note that knowledge of v^*, r^* is used to generate π^C . The challenger always rewinds \mathcal{A} once per π^C generation to obtain two different proof outputs with the same inputs and runs the extractor algorithm to obtain v^*, r^* which allows the computation of discrete log when the above mentioned criteria is met. Thus, contradicting the assumed hardness of discrete log assumption.

With the case exhaustion, the adversary has no way to win the game anymore. Therefore, we proved the theorem. \square

The (k)-anonymity of PACT prevents an adversary from distinguishing transaction amounts and type of assets for accounts that the adversary does not own

among a set of spenders, receivers, and neutral parties. The adversary also will not be able to distinguish between spenders, receivers and neutral parties as the transaction amount and its sign are hidden. This property is captured by the Anonymity defined in Definition 9. Informally, PACT safeguards the anonymity of all participants (including spenders and recipients) among the selected set. In a deployed system, the adversary would be able to learn some information (such as insufficient balance) about the transaction if the adversary is freely allowed to choose the transaction amount details that an honest party will always accept. This situation is prevented in the model by requiring the adversary to distinguish transactions without the side effect of the transaction being queryable. In the post-challenge query phase, the adversary is not allowed to query the account secret key anymore to exclude the trivial case of distinguishing by extracting the transaction value directly. Trivial cases such as differences in adversary accounts and differences in account policy enforcement for the two given inputs are also excluded.

Definition 9 (Anonymity). *It holds that every PPT adversary \mathcal{A} has at most negligible advantage in the following experiment, where we define the advantage as $\mathbf{Adv}_{\text{ANON,PACT}} := |\Pr[\text{ANON}^{\mathcal{A}}(\lambda) = 1] - 1/2|$.*

$\text{ANON}^{\mathcal{A}}$

$pp \leftarrow \text{PACT.Setup}(1^\lambda)$
 $\mathcal{O} := \{\text{Exp.AddAcc}, \text{Exp.PostTx}, \text{Exp.Spend}, \text{Exp.Corrupt}, \text{Exp.Policy}, \text{Exp.Latest}\}$
 $(\mathcal{V}_0, \mathcal{V}_1) \leftarrow \mathcal{A}^{\mathcal{O}}(pp)$
 $b \leftarrow_{\$} \{0, 1\}$
 $(\mathcal{PK}, \mathcal{T}) \leftarrow \text{Exp.Latest}()$
 return \perp if $\exists p$ s.t. $\text{pk}_p \in \mathcal{L}_C, v_p \in \mathcal{V}_0, v'_p \in \mathcal{V}_1, v_p \neq v'_p$
 return \perp if $\exists p$ and $\exists b^* \in \{0, 1\}$ s.t. $v_p \in \mathcal{V}_{b^*} \wedge v'_p \notin \mathcal{V}_{1-b^*}$
 return \perp if $\exists p$ s.t. $v_p \in \{\mathcal{V}_0 \cup \mathcal{V}_1\}, \text{Pol}_p(v_p, \text{aux}_p) \neq 1$
 $(\text{tx}, \pi) \leftarrow \text{PACT.Spend}(\mathcal{V}_b, \mathcal{T}, \mathcal{PK})$
 $\mathcal{L}_{C_{\text{TX}}} := \mathcal{L}_{C_{\text{TX}}} \cup (\text{tx}, \pi)$
 $b' \leftarrow \mathcal{A}^{\mathcal{O} \setminus \{\text{Exp.Corrupt}\}}(pp, \text{tx}, \pi)$
 return 1 if $b' = b$

Theorem 6. *Let us define the advantage of an adversary \mathcal{A} in winning the experiment in Definition 9 as:*

$$\mathbf{Adv}_{\text{ANON,PACT}}^{\mathcal{A}} := |\Pr[\text{ANON}_{\text{PACT}}^{\mathcal{A}}(\lambda) = 1] - 1/2|$$

For the PADL defined in Table 1, we have:

$$\mathbf{Adv}_{\text{ANON,PACT}}^{\mathcal{A}} \leq \left(\sum_{a=1}^{|\mathcal{AS}|} |\mathcal{V}_a^*| \right) \cdot \text{Hid}_{\text{HCOM}}^{\mathcal{A}}(\lambda) + 4 \left(\sum_{a=1}^{|\mathcal{AS}|} |\mathcal{V}_a^*| \right) \cdot \text{ZK}^{\mathcal{A}}(\lambda)$$

Informally, PADL has anonymity as long as the proofs used in the transaction scheme is zero-knowledge and the commitment scheme is hiding.

Proof. The proof proceeds via a series of game changes. The first game is exactly the original game defined and the last game completely hide the information about b .

Game₀ : This is exactly the original game defined in Definition 9.

Game₁ : In the PACT.Spend algorithm, the challenger replaces all transaction amount values v_i in \mathcal{V} (except values belonging to accounts in corrupted list \mathcal{L}_C) with a random value v_i^* while making sure entries in the modified transaction summed to zero and it does not spend over the account balance limit. Note that transaction values v for adversary corrupted accounts in both \mathcal{V}_0 and \mathcal{V}_1 is the same to prevent trivial cases of distinguishing. Let $\mathcal{V}^* \subseteq \mathcal{V}$ be the sub-list containing only the modified values and let \mathcal{AS} be the set of assets in the transaction. The distribution of the resulting commitments is the same from \mathcal{A} 's point of view. The game is indistinguishable from **Game₀** by the commitment hiding property, it follows that

$$|\Pr[\text{Game}_1(\mathcal{A}) = 1] - \Pr[\text{Game}_0(\mathcal{A}) = 1]| \leq \left(\sum_{a=1}^{|\mathcal{AS}|} |\mathcal{V}_a^*| \right) \cdot \text{Hid}_{\text{HCOM}}^A(\lambda)$$

Game₂ : Let $(\text{Sim}_0, \text{Sim}_1)$ be the simulator for the zero-knowledge experiment of the underlying proof system, and $|\mathcal{PK}|$ be the number of participants. During the challenge phase where two lists of potentially different $\mathcal{V}_1, \mathcal{V}_2$ is given by the adversary, instead of computing the proof normally using $\text{ZKProve}_{\mathcal{L}_{\mathcal{R}}}(\text{crs}, \text{stmt}, \text{wit})$, we use simulator $\text{Sim}_2(\text{crs}, \text{stmt}, \text{st})$ where $(\text{crs}, \text{st}) \leftarrow \text{Sim}_1(\lambda)$. Since all commitments and tokens are generated honestly as well as spending amounts are ensured to be lower than account balance, $\text{stmt} \in \mathcal{L}$ for all languages used $(\mathcal{L}_{\mathcal{R}_C}, \mathcal{L}_{\mathcal{R}_C'}, \mathcal{L}_{\mathcal{R}_{EQ}}, \mathcal{L}_{\mathcal{R}_A})$. The game is indistinguishable from **Game₁** by the zero-knowledge property of the proof system, it follows that

$$|\Pr[\text{Game}_2(\mathcal{A}) = 1] - \Pr[\text{Game}_1(\mathcal{A}) = 1]| \leq 4 \left(\sum_{a=1}^{|\mathcal{AS}|} |\mathcal{V}_a^*| \right) \cdot \text{ZK}^A(\lambda)$$

Note that now (tx, π) is generated independent of $\mathcal{V}_0, \mathcal{V}_1$ and secret keys of \mathcal{PK} . Therefore, the spending with (tx, π) in this game leaks no information about b and $\Pr[b' = b] = 1/2$.

□

D Threat Model

PADL assumes that participants can be malicious by trying to either steal, hide or modify values in assets or transaction. Participants can also collaborate to

collude and hide transaction information from auditors, or try to break the consistency of a distributed ledger. We identify several points for the threat model and discuss their risk and resolutions.

D.1 Ledger consistency

We obtain the state of the ledger with each transaction, as the hash of all commits and tokens of the ledger. In order to verify previous transactions, one can recalculate the hash of the ledger to verify that transactions are not omitted. It is also reasonable to assume that one cannot generate a new ledger, by ‘picking’ transactions, as participants’ proof of assets would fail if a transaction is omitted. Removing last transactions is possible, but for that all participants need to agree on. Though if the ledger is deployed on smart contract, it would also require changing the history of the blockchain service which is down to the security level of the service.

D.2 Cryptography Setup

To ensure that $\log_g h$ is hard, all parties could contribute to the randomness of h . For example, each party can choose h^{r_p} on the curve and the $\sum_p h^{r_p}$ is used as the ledger generator point, with no-trust.

D.3 Quantum attack

The current commit scheme is not considered quantum safe, but the combined system can be adapted to PQC primitives. This is intended to be explored in future study, for example by using lattice based cryptography for commitment scheme and Σ -protocol ZKPs.

D.4 Other Attacks Consideration

A replay attack by copying a transaction is improbable because the state of the ledger used to verify the proofs would be different. Assets in a transaction cannot be separated into individual transactions, as the hash of the transaction intent would be different. Similarly, generated proofs cannot be separated from the transaction it is proved for as the transaction identifier is a part of the statement. In addition, the broadcaster cannot append a transaction by itself without endorsement from account owners, as the proof of assets and complementary commits and tokens would be missing.