SoK: Auditability and Accountability in Distributed Payment Systems

Panagiotis Chatzigiannis[1], Foteini Baldimtsi[1], and Konstantinos Chalkias[2]

[1] George Mason University
{pchatzig,foteini}@gmu.edu
[2] Novi Financial / Facebook Research
kostascrypto@fb.com

Abstract. Enforcement of policy regulations and availability of auditing mechanisms are crucial building blocks for the adoption of distributed payment systems. This paper reviews a number of existing proposals for distributed payment systems that offer some form of auditability for regulators. We identify two major distinct lines of work: payment systems that are not privacy-preserving such as Bitcoin, where regulation functionalities are typically tailored for organizations controlling many accounts, and privacy-preserving payment systems where regulation functionalities are typically targeted to user level. We provide a systematization methodology over several axes of characteristics and performance, while highlighting insights and research gaps that we have identified, such as lack of dispute-resolution solutions between the regulator and the entity under audit, and the incompatibility of ledger pruning or off-chain protocols with regulatory requirements. Based on our findings, we propose a number of exciting future research directions.

Keywords: Distributed payments · Regulation · Auditing · Privacy · Cryptocurrencies · Blockchain

1 Introduction

Distributed payment systems have emerged as an alternative to the centralized banking system. Starting with Bitcoin [49], a number of schemes have been proposed with the common characteristic of relying on a globally distributed, append-only public ledger (which is sometimes in the form of a blockchain), to record monetary transactions in a publicly verifiable and immutable way. These systems utilize various cryptographic primitives to secure transactions, as well as a *consensus* protocol to guarantee agreement on the ledger's state by all participants. User participation can be controlled or unrestricted, categorizing such systems in *permissioned* and *permissionless* respectively. While such systems are increasingly growing in popularity, they are often associated with fraudulent transactions or with loss of user funds due to lack of regulation [48].

Panagiotis Chatzigiannis did part of this work during an internship at Novi Financial/Facebook Research. Foteini Baldimtsi and Panagiotis Chatzigiannis were supported by NSF #1717067, NSA #204761 and a Facebook research Award.

The distributed nature of the ledger, typically accessible by the open public or by a wide base of participants (even in permissioned systems), enables external observers to access transaction information, for example sender/receiver addresses or transaction amounts. These addresses are essentially random looking strings and provide their owners a sense of anonymity, especially in permissionless payment systems where anyone can easily create multiple addresses. However, it has been shown it is possible to associate these "pseudo-anonymous" addresses with real identities, for instance using clustering techniques [47]). These concerns led to a number of privacy enhancing proposals. Some were stand-alone cryptocurrencies offering strong privacy guarantees such as Zcash [17] or Monero [55], while others were add-on functionalities to existing systems, such as CoinJoin [46] or TumbleBit [40]. But these systems in turn raised concerns for the regulatory and law-enforcement authorities, since the abuse of such strong privacy guarantees provides users the potential to circumvent regulatory controls (e.g. tax evasion or unauthorized money transmission) or even engage in fraudulent/illegal activities (e.g. money laundering, extortion or drug trafficking [6]).

Needed Features and Regulatory Goals. Ensuring compliance with regulations is crucial for any widely-accepted payment system, even in a system that is designed to preserve user privacy. The goal is to ensure that the system and/or its participants comply with financial regulations and laws (e.g. cannot conduct illicit activities such as money laundering without being accountable to the authorities). In this setting, state authorities or audit firms (e.g. Deloitte or KPMG [4,13]) will need to be convinced that the auditee "follows the rules" by meeting certain regulatory requirements. For example, all participants in a payment system should be compliant with Anti-Money Laundering and Counter-Terrorism Financing (AML/KYC) per Financial Action Task Force (FATF) Travel Rule [5], while an auditor should be able to verify compliance with regulations such as the European General Data Protection Regulation [12] or industry-specific requirements such as the Health Insurance Portability and Accountability Act (HIPAA).

Regulation in Distributed Payment Systems. As mentioned above, pseudoanonymous distributed payment systems such as Bitcoin do not hide transaction information. However, the posted information is not enough to provide regulatory control and additional regulation functionalities are still needed. For instance, in the cryptocurrency world there exist numerous types of centralized organizations which hold users' coins (e.g. online wallets, exchanges, interest accounts etc.) or even "stablecoins" that are backed by fiat currency [9]. These third-party intermediaries are typically opaque to their internal operations, and several infamous examples exist where users lost their funds without holding these intermediaries accountable [48] or organizations investing users' funds instead of focusing on solvency [10]. In fact, the Conference of State Bank Supervisors proposed a model regulatory framework including cryptographic solvency proofs as a means of demonstrating solvency [3]. In that end, some works [23,26] focused on making these services more transparent to earn users trust, and provide auditability

functionalities to authorities (i.e., proving that they are solvent) without disclosing additional information or exposing their users' privacy[1]. However preserving privacy does not come for free, as this makes proof manipulation or collusion possible to falsely convince an auditor on the organization's claims.

In the privacy-preserving setting, enforcing regulation is even more challenging, as the notions of privacy and regulation seem contradictory. In such distributed payment systems, regulation implies auditability and accountability at user level (e.g. disclosing the user's assets or past transactions) or transaction level (e.g. disclosing the participants or value associated with some transaction), while the public ledger entries hide such information from parties not associated with it. A handful of academic works attempted to provide some basic accountability or auditability functionalities, either on top of existing privacy-preserving payment systems [26,35], or as new stand-alone ones [50,57]. However, at the time of writing, no such work has attempted to provide a complete solution that would satisfy all needs from its users and regulators, and there is still a research gap for addressing regulatory concerns (e.g. enforce regulation in permissionless systems [27]).

Our Contributions. We review and unify the landscape of distributed payment systems that offer some form of auditability or accountability. Such forms can range from simple audit protocols where an auditor learns some hidden information posted on the public ledger (e.g. hidden value of a transaction), up to system policies that are automatically executed based on a system participant's behavior, while remaining consistent with the system basic security properties. We provide a non-exhaustive list of such functionalities in Section 2.2. We then categorize related work into two distinct groups: schemes that don't have underlying privacy characteristics, and schemes that preserve some privacy. These groups in turn create two major categories of regulatory functionalities: at an organization level and at a user/transaction level. Our systematization framework considers the following three axes: (1) the security guarantees of each scheme for organization-level auditing (as such audits can be easily abused or leak information), and their audit functionalities for user/transaction-level auditing (as we identify a plethora of such functionalities in privacy-preserving systems), (2) the efficiency asymptotics and (3) the overall properties and attributes. Through our systemic categorization, we identify a number of insights and research gaps which pave the path for future research directions.

Systematization scope. We focus on *distributed* payment systems (using a common public ledger) offering *some* form of auditability or accountability as regulation functionalities with any level of privacy guarantees.

We include works that either propose the above functionalities as stand-alone systems [50,57] or as extensions/add-ons to existing systems [26,35]. We include both permissioned and permissionless systems and with different "bookkeeping" formats (e.g. UTXO or account-based). Our work remains orthogonal to the

[1] In some scenarios, non-private auditing might suffice. However, such a protocol would be trivial from a security standpoint, and to our knowledge no related proposal exists.

underlying data structures used by the common ledger and to the underlying consensus protocol.

The rest of this paper is organized as follows. In Section 2 we provide the definitions and security properties used for our systematization. In Section 3 we discuss our categorization for the works we consider. In Sections 4 we present regulation in works that do not have any underlying privacy preserving mechanisms and highlight insights and research gaps, while in Section 5 we follow a similar pattern for privacy preserving systems. We conclude in Section 6 with a summary of proposed research directions in this field.

2 Background

We informally present the necessary concepts and definitions required throughout this paper and provide a more detailed cryptographic background in Appendix A.

2.1 (Private) Distributed Payment Systems

Assuming the existence of a consensus layer, we define a basic distributed payment system (DPS) to consist of the following algorithms: $\mathsf{Setup}()$, $\mathsf{CreateAcc}()$, $\mathsf{CreateTx}()$ and $\mathsf{VerifyTx}()$ with a public ledger L as common input and output. A secure system has to satisfy some basic properties, such as asset theft prevention and maintaining value balance. Appendix A provides more detailed definitions.

Private DPSs come with the following "privacy-preserving" properties.

- *Confidentiality:* Only the sender S and the receiver R of a transaction tx can learn the value associated with tx.
- *Anonymity:* An external observer of tx cannot derive the public keys or identifiers of S and R associated with that transaction[2].

We note that some systems might offer only one of the above privacy characteristics, but not both. However, if the system provides both confidentiality and anonymity, we say that it is *fully private*. We also call a system *pseudoanonymous* if it does not provide any of the above.

2.2 Enforcing Regulation in Distributed Payment Systems

We now organize the different types of regulation that we encounter in the related literature. These are protocols or policies that disclose information to the auditor, where as an auditor we consider some type of a regulation authority or an audit firm. Disclosed information can include (but not limited to) [25,50,57]:

- Transaction sender and/or receiver
- Transaction value

[2] This property is sometimes referred to as "transaction graph obfuscation".

- Tax compliance
- Total value of assets

Note that as we discuss below, some functions are mostly applicable to private DPS (e.g. transaction value) while others are applicable to pseudoanonymous systems as well (e.g. total value of assets).

Transaction and User-level Regulation. As a starting point, an auditor or a regulation authority would focus on inquiries that involve single transactions or the transaction history of certain system participants[3]. At first glance it might seem that a privacy-preserving system (which hides transaction values and/or transacting parties) cannot comply with such regulatory requirements. However, via the use of cryptographic techniques we can allow for auditability and/or accountability properties. We informally define such properties as follows:

- *Auditability:* There exists a protocol where an external auditor A having access to the common public ledger can provably learn the requested information to be audited (e.g. the participating parties in a transaction). This protocol can be either interactive with the audited parties (requiring their consent) or non-interactive (where A learns the information at will).
- *Accountability:* There exists a system functionality which enforces automatic execution of policies (as defined in its parameters) when a certain predicate is satisfied. For instance, these policies might automatically reject transactions from a specific user or transactions that do not comply with a spending limit and potentially also automatically disclose private information to a designated authority.

Auditability vs Accountability. In general, accountability does not require active participation of an auditor, and accountability policies are typically enforced during the verification phase of a transaction (which usually happens at the consensus layer). For instance, the system might perform certain actions based on the value of the transaction or the cumulative value of recent ones, such as involuntary leakage of information from that transaction or prevent acceptance of subsequent transactions. The system might also enforce other system-wide policies (e.g. tax payment to a pre-determined address). Therefore, accountability can be thought of as a stronger version of auditability, as defined in a general context in [32,36,39,44]. Another distinctive characteristic is that accountability is *proactive* in nature, while auditability is *reactive*.

Auditability and Accountability vs. System security. We note that in some works, the notions of auditability and accountability are implicitly used as a "dishonesty" detection mechanism (e.g. breaking the ledger's immutability property), aimed to incentivize a party to "follow the rules" and holding it accountable when it attempts to violate the system's security [11,22,43]. For instance, in Bitfury's whitepaper on Blockchain auditability [11] both of these notions are used interchangeably, and are associated with a system service that detects such

[3] In typical DAPs, a "human user" might control multiple payment addresses. By user/participant regulation below we refer to address-level regulation, unless we explicitly explain otherwise in certain permissioned schemes.

malicious activities even in case of collusion with ledger maintainers. However, we don't include such a role for an auditor within our scope, and we assume any such activity would be promptly detected and/or prevented by the consensus layer. In fact, in a regulatory context a party might violate laws or regulations (e.g. launder money, transfer more than $10k in a day etc.) without ever violating the system protocols or breaking its security properties.

Organization-level Regulation. In the case of an organization controlling a number of payment system accounts on behalf of its customers (i.e., a bank or custodial service), a regulatory concern is if the organization is *solvent*. Such a proof of solvency can be considered as a form of auditability, which convinces an auditor that the organization indeed controls sufficient funds reflected on the public ledger, without however disclosing more information other than this fact is true (e.g. number of its clients, total assets etc.). A solvency proof typically consists of two parts: a Proof of Assets (PoA) and a Proof of Liabilities (PoL), which when combined prove that an organization's assets exceed its liabilities, thus proving solvency. We discuss both of these proofs below. Note in some cases it might be sufficient to prove "partial" solvency (as it is typically done in the real-world) but from a technical standpoint, it's trivial to convert a full solvency to a partial solvency proof. However, an organization would typically prefer to only prove its solvency without disclosing additional information (e.g. specific asset amounts or account public keys). Also, while regulation at user or transaction level is trivial, in pseudoanonymous systems there are many factors to consider at organization-level, regardless of the system's privacy properties.

Proof of Assets (PoA). As in auditability protocols, a Proof of Assets needs to convince a verifier that an organization controls *at least* a certain amount of funds. Note than this proof does not necessarily need to disclose the exact amount to the auditor, and a lower bound will be sufficient. In some cases however, an upper bound could be provided as well (e.g. in a tax scenario, an organization might want to prove its total assets value lies within a "tax bracket").

For a pseudoanonymous payment system, a naive PoA is to provide signatures to the auditor for some (or all) of the accounts it controls, by also including some challenge value or nonce-timestamp in the signature to ensure freshness. Proving Assets in privacy-preserving systems however might require more complex protocols that involve cryptographic primitives such as ZK proofs.

Proof of Liabilities (PoL). Here the organization needs to periodically publish information on its liabilities (e.g. user balances in banks or exchanges [8]). This information can be either provided directly from the organization, or stored in a public bulletin board. However, with this information publicly available, it is desirable to leak as little side-information as possible (e.g. without exposing the exact value of the organization's liabilities or other information related to its clients). The published information is verified by clients in a probabilistic fashion (i.e., not all of the clients need to actively check for the validity of the published information). We note that publishing this information can be seen equivalent

to an auditor "reply" step (where any client can assume the auditor role), so it is possible to reduce the PoL functionality to an auditability protocol.

2.3 Security Properties and Threats of Regulatory Functions

Based on the above discussion of required regulatory functions for DPS we informally define the related security properties.

- *Regulation Correctness.* An honest auditee following the regulation protocols should always be able to convince an auditor and transact under the correct system policies.
- *Regulation Soundness.* An auditor should always reject false claims for a malicious auditee, while the system should always apply the corresponding policies to the system participants that should be affected by those policies.
- *Minimal Information Disclosure.* When implementing a regulation functionality, the auditee should only disclose the needed information, without suffering any "collateral damage" in terms of privacy. For instance, when a user is asked if ever transacted with a specific party, it should not reveal the associated values; an exchange proving solvency should not leak its number of clients.

Beyond security, some additional desired properties are the following:

- *Offline Auditors.* A system compatible with offline auditors, who do not need to always maintain and track its entire public state.
- *Out-of-band Communication and Storage.* Regulatory functions are preferably performed using system-maintained information, minimizing the use of out of band protocols.
- *Dispute Resolution.* There exists a mechanism to resolve disputes between an auditor and auditee, in case a malicious auditor falsely accuses the auditee of non-compliance. Such mechanism might even include holding an auditor accountable for its own actions.

3 Systematization Methodology

We first categorize all works and systems we consider into two major taxonomies. The first considers pseudoanonymous DPSs (where audits are typically performed at an organization level), and are discussed in detail in Section 4. The second includes schemes that have privacy-preserving attributes (where they typically perform auditing at a transaction or user level), and are discussed in detail in Section 5 (from our previous discussion, user or transaction-level auditing in pseudoanonymous systems is trivial from a cryptographic standpoint).

Our core systematization is performed over the following three axes (we consider systems with respect to their privacy guarantees separately within each axis):

Audit properties axis: We first consider pseudoanonymous systems. In such schemes the basic audit functionalities are Proof of Assets (PoA) and Proof of Liabilities (PoL) which happen in an organization level. In Table 1 we consider schemes that provide PoA and/or PoL functionalities. Additionally, we also

	Functions		Auditor security		Auditee privacy				Gener-alize
	PoA	PoL	PoA Collusion	Hidden Liab.	Value hiding	Popul. hiding	Account Leakage	Multi Proof Leakage	
Provisions [26]	●	●	●	●	●	✕	●	●	●
ZeroLedge [29]	N/A	●	N/A	●	●	✕	●	●	●
DaPoL [23]	N/A	●	N/A	●	●	●	●	●	●
Maxwell [7]	●	●	✕	✕	✕	✕	✕	✕	●
Blockstream [54]	●	N/A	✕	N/A	✕	N/A	✕	●	✕
Wang [56]	●	N/A	●	N/A	✕	N/A	N/A	●	●

Table 1. Organization-level Regulation of pseudoanonymous schemes. By ● we denote support of a functionality, by ✕ a vulnerability and by "N/A" non applicability.

	Transaction-level					Tx/User level	User level			
Scheme	Lim	Trace	Value	Sender-Receiver	Tax	Non-participation	Sum	Blacklist	Stats	Anonym. revocation
Zcash ext [35]	●	●	○	○	●	○	○	○	○	●
ZKLedger [50]	○	N/A	●	●	○	○	●	○	●	N/A
PRCash [57]	●	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	●
PGC [25]	●	○	●	N/A	●	N/A	●	N/A	○	N/A
ACCDET [15]	N/A	N/A	●	●	N/A	○	N/A	N/A	N/A	N/A
ATRA [19]	N/A	N/A	●	●	N/A	N/A	N/A	N/A	N/A	N/A
MProve [30]	N/A	N/A	N/A	N/A	N/A	N/A	●	N/A	N/A	N/A
Damgård et al. [27]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	●
Barki-Gouget [16]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	●

Table 2. Regulation Functionalities in Private Schemes. By ● we denote supporting functionality, by ○ that although not explicitly defined, the scheme can be trivially extended to support that audit type and by "N/A" no support.

look at security and privacy properties, as existing works in the literature have different security guarantees, both for the auditor and the auditee. From the auditor's point of view, security implies Regulation Soundness, i.e. rejecting false PoA due to collusion among organizations (denoted as "PoA Collusion" in Table 1) or false PoL due to "Hidden Liabilities" not included in the proof. From the auditee's point of view, security implies Minimal Information Disclosure, i.e. the actual values of total assets and liabilities remain hidden ("Value hiding"), no further information is leaked from the auditee such as the total client population ("Population hiding") or details on its accounts ("Account Leakage"), and the auditor cannot infer any additional information from subsequent proof executions ("Multi Proof Leakage"). In Table 1 we present a classification of systems according to the above properties, and discuss them in detail in Section 4. We also classify a scheme based on whether it can be generalized to work for any distributed ledger-based payment system or if it is designed for a specific one.

For regulating privacy-preserving systems (at a transaction or user level), we desire a scheme to offer a wide variety of auditability and accountability functionalities. In Table 2, we first distinguish between regulatory functions in a transaction level, for instance value limits (if value is under or over a threshold),

Scheme	Proof size		Tx Create	Audit Prove		Audit Verify	
	PoA	PoL		PoA	PoL	PoA	PoL
Provisions [26]	$O(k)$	$O(n)$	N/A	$O(k)$	$O(n)$	$O(k)$	$O(n)$
ZeroLedge [29]	N/A	$O(n)$	N/A	N/A	$O(n)$	N/A	$O(n)$
DAPOL [23]	N/A	$O(1)$	N/A	N/A	$O(\lg N)$	N/A	$O(\lg N)$
Maxwell [7]	$O(n)$	$O(1)$	N/A	$O(n)$	$O(\lg n)$	$O(n)$	$O(\lg n)$
Blockstream [54]	$O(n)$	N/A	N/A	$O(n)$	N/A	$O(n)$	N/A
Wang [56]	$O(n)$	N/A	N/A	$O(n)$	N/A	$O(n)$	N/A

Scheme	Ledger Storage	Tx Create	Audit Prove	Audit Verify
Zcash ext [35]	$O(n)$	$O(\lg m(\lg\lg m))$	Embedded in tx	$O(1)$
ZKLedger [50]	$O(mn)$	$O(n)$	$O(1)$	$O(1)$
PRCash [57]	$O(k)$	$O(k)$	Embedded in tx	$O(1)$
PGC [25]	$O(n)$	$O(1)$	$O(m_u)$	$O(m_u)$
ACCDET [15]	$O(n)$	$O(k)$	Embedded in tx	$O(n)$
ATRA [19]	$O(n)$	$O(1)$	Embedded in tx	$O(n)$
MProve [30]	$O(k)$ proof size	N/A	$O(k)$	$O(k)$

Table 3. Efficiency asymptotics. n: # of owned/system accounts (for pseudoanonymous and privacy preserving schemes respectively). N: maximum number of supported accounts. m: number of recorded transactions. k: size of anonymity set. m_u: Txs of user. For Audit Proofs we consider the most basic audit in each scheme (i.e proof of a value). If audit proof is non-interactive, cost is embedded in tx creation costs).

tracing (providing links between transactions), revealing the transaction's value or participants (sender and receivers) and tax payments (deducting a value's portion towards a pre-determined account). We also distinguish them in a user level, for example auditing user's sum of values (equivalent to PoA in pseudoanonymous systems), user revocation (i.e. applying policies only to users in a "blacklist"), deriving statistical information from user's past transactions (e.g. learning the average transacted value in a time period) or revoking a non compliant user's anonymity. It might also be desirable to prove user non-participation in some transaction, which is an audit across both levels mentioned above.

Efficiency axis: Our second axis is based on the efficiency asymptotics for both pseudoanonymous and private systems, namely their space requirements, transaction creation costs and audit proving and verification costs. We do not consider concrete metrics because of the variety (or absence) of implementations. Regarding space efficiency, we consider the proof size for organization-level auditing and the overall ledger storage costs for privacy-preserving schemes with user or transaction level auditing (denoted as "proof storage" and "ledger storage" respectively in Table 3). Then we capture the transaction creation costs (which are not applicable for schemes under the first category as the transactions have already been created). Finally for capturing the auditee's proving and the auditor's verification costs, we consider the most "basic" audit functionality each scheme offers. We note that not all schemes have the same "basic" audit functionality, thus Table 3 should not be seen as a comparison table. Also, audit proofs are not applicable for schemes offering only accountability, as the neces-

Scheme	Audit Scope	Model	Consensus	Partic-ipation	Identity authority	Auditing: Inter-action	Auditing: Auditing authority	Auditing: Fine-coarse	Privacy	Assumptions
Provisions [26]	O	Both	⊥	N/A	N	Y	N	Coarse	◐	
ZeroLedge [29]	O	Both	⊥	N/A	N	N	N	Coarse	◐	
DAPOL [23]	O	Both	⊥	N/A	N	N	N	Coarse	◐	
Maxwell* [7]	O	Both	⊥	N/A	N	Y	N	Coarse	○	
Blockstream* [54]	O	UTXO	🌍	🌍	N	N	N	Fine	○	
Wang [56]	U/O	Both	⊥	N/A	N	Y	N	Fine	○	Secure channel between prover-auditor
Zcash ext [35]	T/U	UTXO	🌍	🌍	Y	N	Y	Both	●	Trusted setup, Spending authority
ZKLedger [50]	T/U/O	Acc	⊥	🔒	Y	Y	N	Both	●	Out of band comm
PRCash [57]	T/U	Acc	⊥	🔒	Y	N	Y	Coarse (limit)	◐	Trusted validators, Out of band comm, sender-receiver interaction
PGC [25]	T	Acc	⊥	🔒	Y	Y	N	Fine	◑	
ACCDET [15]	U	UTXO	🔒	🔒	Y	N	Y	Fine	●	
ATRA [19]	U	N/A	🔒	🔒	Y	N	Y	Fine	●	
MProve [30]	O	UTXO	🌍	🌍	N	N	N	Fine	◐	
Damgård et al. [27]	U	N/A	⊥	🔒	Y	N	Y	N/A	●	Anonymity revoker
Barki-Gouget [16]	U	N/A	⊥	🔒	Y	N	Y	N/A	●	Anonymity revoker

Table 4. General categorization of auditable schemes. Scope: T Transaction level, U User level, O Organization level. Model: Acc: account-based, UTXO: Unspent Transaction Output based. By ● we denote full privacy, by ◐ set anonymity, by ◑ confidentiality and by ○ pseudoanonymity (privacy is against auditor and against all observers for pseudoanonymous and privacy-preserving schemes respectively). By 🌍 we denote permissionless, by 🔒 permissioned and by ⊥ orthogonality to consensus layer. By ⋆ we denote non-academic works.

sary information is included in the transaction itself, while transaction creation costs do not apply for auditing at an organization level (as they are always independent of PoA or PoL protocols).

General properties axis: This axis is based on several different properties and attributes of all existing proposals. We compare them in terms of audit scope, i.e. user level (U) if auditing information of a particular user, transaction level (T) if auditing a transaction's full details and organization (O) if auditing an organization's assets, liabilities or solvency, underlying system architectures (e.g. UTXO or account-based), consensus and participation models (permissioned vs permissionless), audit granularity and their privacy characteristics. More importantly, we clarify the picture in terms of their underlying assumptions, and we particularly highlight the need of various trusted centralized authorities, which is typically not desired. The comparison is shown in Table 4. We note that requiring interaction for transaction/user level audits implies auditee consent, while *all* organization-level audits always require consent from the organization. Another observation is that non-interactive user/transaction audits typically require the existence of an auditing authority, as the non-interactive data should only be accessible to such authorities (else this would compromise privacy).

4 Regulation Functions in Pseudoanonymous Systems

We now discuss the first major category of payment auditable systems; those with a pseudoanonymous underlying system where the focus is typically on organization level regulation. As discussed in Section 2.2 the common goal here is for an organization to prove that is solvent.

Insight 1 *Pseudoanonymous systems inherently expose their users' privacy (due to the public nature of the distributed ledger), making user or transaction-level auditing trivial.*

Perhaps the first attempt to prove an organization's digital assets was Bitstamp's Proof of Reserves [2], a procedure on top of Bitcoin (also applicable to any pseudoanonymous system), involving a single entity asking the organization to provide a signature using its private keys on a message selected by that entity. The obvious drawback of this approach however is lack of auditee privacy, i.e. it does not satisfy Minimal Information Disclosure.

Insight 2 *As a naive PoA method, one might prove ownership of accounts associated with those assets by signing a message using the respective private keys. This method however does not accomplish Minimal Information Disclosure, as it also discloses the organization's exact assets as well as the respective accounts.*

Towards providing a way to prove solvency in a more distributed fashion, Maxwell's PoL [7] consists of a "summation" Merkle tree with each leaf contains a client's balance in plaintext, summing with the siblings up to the root as in a plain Merkle tree. Then, the organization's clients *can* check their inclusion in the organization's liabilities through Merkle proofs, which implies this method of PoL has a probabilistic nature. However Minimal Information Disclosure was still unresolved, as Maxwell's PoL publicly exposes the total liabilites and the number of the users, and leaks information of sibling nodes by multiple execution of proofs. In addition, an attack was identified (and subsequently fixed) in this scheme which could potentially enable a participant to claim less liabilities [41].

Gap 1 *Given the probabilistic nature of PoL, an organization can collect information on client queries in the network or application layer and manipulate subsequent proofs by not including user balances with low-probability retrievals. Can such behavior be prevented without having to publicly disclose liabilities data, thus disclosing the population of an organization's accounts?*

Private Information Retrieval techniques have been proposed [23] to efficiently mitigate Gap 1, however such techniques have not yet been deployed.

As in [2], Maxwell's PoA protocol simply requires signing some message using the private keys associated with the controlled assets. As discussed in Section 2.2, Maxwell's PoA and PoL combined constitute a proof of the organization's solvency. We also mention an early implementation for Proof of Solvency using similar cryptographic techniques [1].

Provisions [26] was among the first academic works to present a complete proof of solvency solution, and augmented both PoA and PoL protocols with privacy-preserving characteristics. For the PoA part, the organization chooses adds a number account public keys to those it already controls (Provisions assumes these keys are not hashed), essentially forming an anonymity set, then creates a Pedersen commitment for each (with a commitment to zero for those it does not own) and publishes a homomorphic addition of those commitments. Then using standard Zero Knowledge (ZK) protocols, it proves that it either knows the secret key for the respective commitment public key *or* it is a commitment to zero. This ensures that the verifier does not really learn which accounts the organization owns. Note that using standard, efficient ZK proofs is not compatible with hashed public keys (since in systems like Bitcoin it would be required to spend at least once from a wallet address to reveal the account public key). Customized ZK proofs would have to be created for such statements to overcome this limitation, and works like [24] could serve as a starting point.

For the PoL part, the organization constructs and publishes a Pedersen commitment representing each one of its clients' balances, which can be homomorphically added to form a commitment of the organization's total liabilities. At any point, each client can check it is included in the liabilities commitment by asking for an inclusion proof, while ensuring overall that no commitments have been added with negative amounts though appropriate range proofs. However, as shown in Table 1, subsequent execution can still potentially leak some information to external parties (e.g. number of user accounts of organization at a specific time). Given the two above published commitments, the organization proves that their difference is positive (which constitutes the overall proof of solvency). As [26] was published in 2015, there is room for efficiency improvements by i.e. utilizing recent range proof constructions (Bulletproofs [21]).

Insight 3 *Hashed public keys in systems like Bitcoin are not fully compatible with more complex PoA techniques as in [26] using standard Zero-Knowledge proofs. Customized ZK circuits for SNARKs need to be designed in such cases.*

Gap 2 *During PoA, organizations can collude with each other to manipulate these proofs by including their assets to each other's proofs, violating Regulation Soundness. Does a mitigation strategy exist to prevent this?*

While performing PoA in a synchronous manner could trivially prevent organization collusions, this approach is impractical.

Gap 3 *Can we design a PoA protocol on top of a pseudoanonymous system that is fully private for the auditee with sublinear costs?*

We observe that all current PoA protocols form some anonymity set to hide the actual audited organization's accounts. Implementing a zk-SNARK in a system might enable full privacy and sublinear proof size and verification costs, but at the cost of introducing additional assumptions like trusted setup.

ZeroLedge [29] focuses on the PoL aspect, aiming to address the weaknesses of Maxwell's protocol, most notably its "hidden liabilities" attack. In this approach,

the organization creates commitments to identifier-value pairs for each of its accounts along with zero-knowledge proofs of their validity as well as to the total liabilities amount. Through the zero-knowledge properties, it prevents collusion attacks and preserves verifier anonymity, however it still leaks some information about the total number of accounts. Similar to Provisions, this scheme could benefit from newer, more efficient range proof techniques such as Bulletproofs [21]. Even so, as shown in Table 3 its asymptotic costs remain linear in the total number of accounts, which might make it very costly in large deployments.

The recent PoL proposed standard (DAPOL) [23] is inspired by previous works [7,26]. It addresses the majority of their privacy-related issues and currently has the best PoL Minimum Information Disclosure possible, without leaking information about other addresses' balances, total liabilities or total number of addresses (which even the "flat" version of Provisions was leaking). To achieve this, as the Merkle Tree approach always leaks such information, it augments it with more sophisticated primitives and constructions, such as Zero Knowledge Proofs, VRFs, and sparse Merkle Trees, and has a "layered" PoL that supports a very large number of addresses. In addition, it presents itself in use cases outside financial applications and solvency proofs, such as disapproval votes, virus outbreak reports or referral programs.

Insight 4 *To achieve Regulation Soundness and Minimal Information Disclosure, more advanced cryptographic primitives and complex constructions have to be employed, such as Zero Knowledge proofs, VRFs and sparse Merkle trees.*

Wang et al. [56] provides a simple protocol for a potential buyer proving assets to a vendor before conducting a transaction, using that transaction's external data as a "challenge". Although this protocol could be extended for providing PoA at an organization level as well, the use-case of such a protocol seems to be limited to a "buyer-vendor" scenario as it is associated with a specific transaction. More importantly, it does not have strong Minimal Information Disclosure as other PoA/PoL protocols like Provisions or DAPOL[23,26].

Finally we briefly mention a few additional works related to organization-level regulation. Hu et al. [41] highlighted a "mix and match" attack on the Maxwell protocol, while proposing its mitigation technique. Blockstream proof of reserves [54] propose an alternative to the naive PoA approach by creating and signing invalid transactions using all owned UTXOs, which however degrades the organization's privacy. Moore and Christin [48] provide a risk analysis for cryptocurrency exchanges, highlighting the need of Solvency Proofs. Finally, Decker et al. [28] proposed a variant of the Maxwell Protocol based on a trusted platform module (TPM) to securely execute the necessary computations while ensuring honest computation. This approach can also facilitate the computation between PoA and PoL for proving solvency.

An open problem in Proofs of Liabilities approach commonly used in these works is Dispute Resolution, i.e. a client claiming that his balance within the organization is not included in the proof. This is problematic in both ways, i.e. a malicious organization simply rejecting an honest client's claim, or a malicious client falsely accusing an honest organization of misbehavior.

Gap 4 *Can disputes be resolved by a third-party judge at an organization-level regulation, when a client claims that their balance with the organization has not been included in a PoL?*

Mutual contract-signing solutions (i.e. both client and the organization signing a user's transaction and signature) might be helpful to solve this gap, however such an approach also needs to be practically deployable.

5 Regulation Functions in Privacy-preserving Systems

In this section, we provide an overview of distributed payment systems that have privacy-preserving characteristics *and* some form of auditability and/or accountability functionalities focused on a transaction or user-level. In our discussion, we further divide such systems into two categories: the ones that require designated auditors and the ones where no explicit auditors are assumed, as shown in the "Auditing authority" column in Table 4.

5.1 Centralized - Designated Authority

A common approach when designing audit and accountability functionalities for privacy-preserving DPSs is to add a system-designated, centralized authority (or group of authorities). Such authority could either enforce accountability rules or take the role of an external auditor as defined in Section 2.2. This approach was adopted in one of the first works [35] to address regulatory concerns in privacy preserving cryptocurrencies in the permissionless model by extending Zerocash [17]. It assumes the existence of various types of authorities where each one is designated to enforce different policies, offering a wide range of auditability functionalities as seen in Table 2. The main idea is to embed *auxiliary information* to coins, such as counters or coin types, and define policies as algorithms that are executed each time a coin is spent. Then, a designated authority can be used to verify a policy at the time of transaction generation, for example it could check that a transaction value does not exceed a certain limit and sign the transaction to certify it. This is a type of accountability, since it can proactively check transactions before being posted in the ledger. We note that this technique can be easily adapted for pretty much any type of policy but as discussed below comes with efficiency and privacy costs. [35] also presents techniques for coin tracing assuming that coins include tracing information encrypted under an authority's public key. Then, an authority could at will trace those coins (and all subsequent coins resulting from transactions of the original coins) without any interaction with the users. Interestingly, [35] also provides techniques for *accountable authorities*, the actions of which would be transparent to users. For instance, users could check if a tracing authority has traced their coins.

The techniques of [35] are quite effective, easily allowing the implementation of a wide range of policies. However, there exist both efficiency and privacy limitations. Requiring transactions to be validated by an authority *before posted*,

requires extra communication and computation costs which can be a burden in practice – especially on top of an already computationally intensive system such as Zerocash. Most importantly though, the main issue with the given approach is that it gives too much power to the designated authorities. Not only such authorities can learn transaction information that would otherwise remain private, but they could also arbitrarily refuse to validate certain transactions.

Insight 5 *Embedding encrypted auxiliary information in transactions that can be decrypted by designated authorities at any point, is a trivial solution for accountability and auditing but has negative impact to user privacy.*

Insight 6 *Accountability policies rely on the transaction verifiers (i.e. the consensus layer) to enforce them. Any reactive auditability functionality can be converted to an equivalent proactive accountability policy through auxiliary data and interaction with an authority in the transaction creation phase.*

Although [35] only included a vague description of how to add audit functionalities in Zerocash (without providing a concrete construction or evaluation), it offered some of the first insights and problems for designing auditable/accountable privacy-preserving payment systems. PRCash [57] was developed as a *standalone* fully-private payment system with built-in accountability for spending limits. Specifically, it only allows transactions up to a specified amount, while leaving the option for users to de-anonymize themselves against a centralized authority if wishing to transfer larger amounts. PRCash works in a permissioned setting, with an anonymous credential issued by a centralized identity authority to prevent circumventing limits by creating sybil identities. Consequently, this same authority is responsible for both identity management and regulatory functions in the system (although these roles could be decoupled in separate "identity" and "auditing" authorities). PRCash is inspired by the private cryptocurrency Mimblewimble [33,52], however its construction follows a more complex design. Namely it uses two commitments and a public key pair (as opposed to one commitment and no public key cryptography in Mimblewimble) in order to connect participants to identity credentials. The additional commitment is used to generate "re-randomizeable" authority certificates (that permit participating in the system) and a secret key is used to derive a unique transaction ID. For transacting within the limit, the sender would include an appropriate range proof in the transaction. To exceed the limit, instead of the range proof, the sender would need to encrypt his public key under the authority's public key (thus deanonymizing himself to the authority). PRCash as shown in Table 2 is limited to this specific accountability functionality by design, while it relies on a centralized authority for its accountability aspects. In addition, although an external observer of the ledger cannot link transacting parties or learn transaction values, the system is not fully private against transaction Validators in the consensus layer, as each transaction leaks information (pseudo-identifiers) to the Validators which could be used to generate links between transactions.

Insight 7 *In a permissionless setting, identity authorities associating real identities with public keys are required to preserve regulation connected to value limits, or else such regulation can be trivially circumvented through sybil identities.*

In the permissioned setting, an anonymous, auditable payment scheme was presented in [15] (we will refer to it by ACCDET). The goal of ACCDET is to hide the content of transactions without preventing authorized parties from auditing them. The paper considers a centralized identity authority for associating real-world identities with system credentials, similar to PRCash [57], as well as a designated trusted auditor for each system participant who can learn any hidden transaction the participant is involved in at will. This is ensured by including encrypted transaction information under the designated auditor's public key, along with a standard ZK proof of correct encryption under the correct key. Value tokens are hidden using Pedersen commitments, while they are checked for validity and blind-signed by a certifier when spending them. The paper includes an evaluation based on Hyperledger Fabric [14] as a consensus layer and provides an analysis of the computational costs for each required operation. Besides the fact that the trusted auditor is again very powerful, we also note that the auditing verification cost is linear to the number of transactions ever happened in the system, as shown in Table 3. This is because ACCDET does not assume user consent during audit, thus an auditor would have to decrypt the whole ledger in order to trace any user. A more generic approach that does not restrict itself to payment systems [19] (we will refer to it by ATRA), considers the issuance of anonymous credentials in permissioned blockchain systems which could be transformed into a payment scheme. It also implements auditability by assuming the existence of a centralized auditor and encrypting all private transaction information under the auditor's keys. ATRA, similar to ACCDET, has inefficient auditing, as the auditor has to decrypt all ciphertexts in the ledger.

Insight 8 *Auditing a fully-private DPS without any aid from the auditee, is generally inefficient, as the auditor would need to audit the whole ledger to retrieve the desired audits.*

Gap 5 *Can we design a private DPS with centralized authorities that reveal a user identity only when a user misbehaves according to well-defined policies (as done in traditional Chaum e-cash protocols [20])?*

Note that PRCash [57] seems to accomplish this, as it de-anonymizes a user against an authority if the user exceeds a transaction limit. Still PRCash is tailored to support this specific policy, and designing a system that can make users accountable without their consent for arbitrary policies is challenging.

Damgård et al. [27] provide "design principles" for private and accountable distributed payment systems, rather than building a standalone one. Their main focus is to support auditing as anonymity revocation of participants in the identity layer, rather than auditing the aspects of a payment system (e.g. transactions or assets) in the transaction layer. To accomplish this, they propose two kinds of authorities: An Identity provider who provides a digital identity

with attributes to account holder and stores registration information, and an anonymity revoker who can revoke anonymity at will on accounts created by the account holder using the registration information. While threshold encryption is proposed to avoid giving too much power to a single anonymity revoker, this approach might be still problematic in the same manner as in Zerocash extension [35]. In a similar manner, Barki-Gouget [16] also focus in a physical entity's anonymity revocation by a centralized authority, while presenting their work as a standalone accountable-anonymous system. As both of above works focus on the identity layer and they only provide an abstract way of building a complete distributed payment system, we omit concrete efficiency asymptotics in Table 3.

5.2 General Auditor

A designated auditor (or set of auditors) makes an implementation simpler, but is a strong assumption for any payment system. Some schemes were proposed making auditing possible by *any* auditing authority typically with the auditee's consent. This approach is generally preferred to address Insight 5 concerns, while still being compatible with regulatory practices as we discuss below.

zkLedger [50] is a *permissioned*, privacy-preserving payment system with built-in auditability functions that does not require a designated auditor. Its shared transaction ledger takes a unique approach, and instead of the typical blockchain format, it employs a two-dimensional table recording all participating parties (columns) and all posted transactions (rows). Transactions are formed via a combination of commitments and ZK proofs and, whenever a transaction happens, a new row is generated in the table including information for *all* system participants, even for those who do not participate in that transaction by committing to a zero value (in order to ensure transaction and participant privacy).

zkLedger's basic auditing functionality is an *interactive* ZK protocol between an account holder and an auditor, where the account holder reveals the value hidden in a commitment in a verifiable manner, without disclosing any other information (such as its private key) or needing to open the commitment. Based on this basic audit functionality, several other audits can be implemented at a transaction or participant level (e.g. transaction limits or statistical information). Statistical audits can be easily derived from the basic value audit using auxiliary bit flags, while limit audits can be implemented using appropriate range proofs. Value audits can also be used to derive participation audits (as well as non-participation proofs) utilizing zkLedger's architecture. zkLedger could be trivially extended to accommodate tax compliance either proactively as an accountability functionality (by verifying a ZK proof that the relation between the total recipient values and the tax authority recipient is equal to a fixed ratio, per Insight 6) or reactively as an auditability functionality (by verifying the same ZK proof as before during the auditing process). Also zkLedger's audits can be seen as organization-level audits (i.e. PoA) due to its participants mainly considered being "Banks". Transaction types and tracing could also be easily added as system add-ons through an additional commitment and an appropriate ZK proof of consistency between sent and received values, but this would further

increase storage needs and computational costs. However, its structure circumvents the limitation of Insight 8 which enables efficient auditing with as low as $O(1)$ asymptotic costs (assuming an online auditor tracking ledger state).

However it is obvious from Table 3 that zkLedger suffers from very limited scalability, as requiring each transaction to include a commitment (as well as the needed auxiliary information) for *all* participants, combined with an ever-increasing ledger of such transactions, results in large computational and storage costs, which makes it viable for up to about a hundred participants. It also requires participants to communicate out-of-band and be online at all times, which further limit its practical applications. Nevertheless, it was the first system to depart from the naive approach described in Insight 5 while offering a wide range of auditing functionalities, which inspired subsequent academic works.

Insight 9 *An interactive auditing protocol implies the auditee's consent and cooperation with an auditing authority. This requirement is not necessarily a drawback for such schemes, as refusal to cooperate with authorities can be considered as equivalent to a failed audit. In addition, audit by consent typically enables more efficient auditing.*

Insight 10 *Many audit functions such as transaction limit or tax compliance can be reduced to a "basic" transaction value audit.*

PGC [25] is a confidential payment system with an auditing functionality on transaction values. It uses similar cryptographic techniques to zkLedger. Specifically, it uses an El Gamal encryption variant equivalent to a Pedersen Commitment and the auxiliary information used in zkLedger, and relies on ZK proofs composed of Σ-protocols. PGC by using encryption instead of commitments does not need to rely on out-of-band communication assumptions (which are needed to open the commitments), which could potentially enable it to work in a permissionless setting as well (although not explicitly discussed in the paper). It proposes three different audit functions, namely transaction limit (using appropriate range proofs), value disclosure and tax payments (which can be derived from value disclosure as discussed in Insight 10). Recall that these reactive auditability functions could be converted into equivalent proactive accountability functions enforced by the consensus layer (Insight 6). Since PGC is not anonymous, transaction participant auditing is not applicable. Also transaction types or tracing functionalities can be added in PGC by including the necessary auxiliary information in the transaction structure as in Zcash extension [35]. Although PGC does not suffer from the scalability issues of zkLedger, it requires each transaction to include a unique serial number to prevent replay attacks (as Zcash [17]), which asymptotically results in linearly-increasing blockchain storage to the number of transactions. PGC can be considered as a special case of Insight 8 - by trading off anonymity, it achieves highly efficient auditing only dependent to the number of past user transactions, as shown in Table 3.

Gap 6 *Is auditing of transactions that do not appear in the ledger (or happen "off the chain") possible?*

No system that enforces regulation policies is compatible with ledger compression techniques, as auditing in DPSs needs to refer to some published data on the ledger. Designing new pruning techniques would be needed to fill this gap. At the same time, no system exists that achieves Regulation Correctness and Regulation Soundness for locked funds. While it is unclear how to provide regulatory control for distributed ledger systems implementing payment channels or cross-chain atomic swaps a first approach might be to consider a centralized authority.

Gap 7 *How can a dispute between an auditor and an auditee be resolved?* [4]

We note most of DPSs ignore the problem of dispute resolution in case an auditor misbehaves (e.g. accuses an auditee of failing an audit). While the auditee could publish all their secret information to rebut the accusation, this would fully compromise their privacy.

MProve [30] is a PoA protocol tailored for Monero [55]. As the Provisions protocol won't work for Monero because of ring signature obfuscation, MProve provides a proof that the key images of the one-time addresses controlled by the exchange (which they sum to its total assets) have not previously appeared on the public ledger. While this approach provides a proof of non-collusion as well (as the one-time nature of the key images would trivially expose collusion), it exposes the sender's identity when those key images are spent which might eventually enable public tracing of transactions (especially in cases where such a PoA protocol is used frequently).

Gap 8 *Can PoA/PoL be implemented on privacy-preserving payment systems without degrading the participant's privacy?*

6 Conclusion

We observe increasing efforts towards implementing regulatory control in distributed payment systems. However, existing lines of research approach the problem from different angles, with different goals, assumptions and use-cases. Our systematization identifies a number of exciting open problems on providing mechanisms for regulation of DPSs which we summarize below.

Despite auditing in pseudoanonymous systems intuitively being straightforward, no solution exists that prevents collusions among organizations when proving assets, while allowing them to exclude liabilities with low-probability of access (Gaps 1 and 2). In addition, proving assets efficiently, in a fully-anonymous manner and without introducing additional assumptions remains an open problem (Gap 3). In privacy-preserving systems, we note that mapping regulation requirements and laws to automated computations is not trivial, and designing fully-private DPSs supporting many different regulations currently seems out of reach. Proof of solvency on privacy-preserving systems is challenging (Gap 8), while there is no *fully-private* DPS that is scalable, computationally efficient

[4] Gap 7 is the equivalent of Gap 4 for privacy-preserving systems.

and without strong assumptions such as designated auditing authorities that can violate user privacy. We observe all schemes attempting to enforce regulatory functions are designed for "on-chain" protocols, and are not compatible with information that either lives off the ledger and/or locks funds, or with information that has been pruned entirely from the common ledger (Gap 6). Finally, dispute resolution without compromising privacy against a third-party judge is an open problem in both pseudoanonymous and privacy-preserving systems (Gaps 4, 7).

References

1. Bitgo announces "verified by bitgo" proof of asset service, https://www.businesswire.com/news/home/20150630005466/en/BitGo-Announces-%E2%80%9CVerified-BitGo%E2%80%9D-Proof-Asset-Service#.VZKYwO1Viko
2. Bitstamp proof of reserves, https://www.bitstamp.net/s/documents/Bitstamp_proof_of_reserves_statement.pdf
3. CSBS state regulatory requirements for virtual currency activities, https://www.csbs.org/sites/default/files/2017-11/CSBS%20Draft%20Model%20Regulatory%20Framework%20for%20Virtual%20Currency%20Proposal%20--%20Dec.%2016%202014.pdf
4. Deloitte COINIA and the future of audit, https://www2.deloitte.com/us/en/pages/audit/articles/impact-of-blockchain-in-accounting.html
5. FATF travel rule: What you need to know, https://complyadvantage.com/knowledgebase/fatf-travel-rule/
6. IRS is trying to deanonymize privacy coins like monero and zcash, https://www.forbes.com/sites/shehanchandrasekera/2020/07/06/irs-is-trying-to-deanonymize-privacy-coins-like-monero-and-zcash/#4607506c4174
7. Maxwell summation trees, https://bitcointalk.org/index.php?topic=595180.0
8. Proof of solvency: Technical overview, https://medium.com/iconominet/proof-of-solvency-technical-overview-d1d0e8a8a0b8
9. Tether: Fiat currencies on the bitcoin blockchain, https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf
10. Tether's bank says it invests customer funds in bitcoin, https://www.coindesk.com/tethers-bank-says-it-invests-customer-funds-in-bitcoin
11. On blockchain auditability (2016), https://bitfury.com/content/downloads/bitfury_white_paper_on_blockchain_auditability.pdf
12. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). Official Journal of the European Union L119 pp. 1–88 (2016)
13. Deloitte's 2020 global blockchain survey (2020), https://www2.deloitte.com/content/dam/insights/us/articles/6608_2020-global-blockchain-survey/DI_CIR%202020%20global%20blockchain%20survey.pdf
14. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou,

C., Vukolic, M., Cocco, S.W., Yellick, J.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018. pp. 30:1–30:15 (2018)

15. Androulaki, E., Camenisch, J., De Caro, A., Dubovitskaya, M., Elkhiyaoui, K., Tackmann, B.: Privacy-preserving auditable token payments in a permissioned blockchain system. Cryptology ePrint Archive, Report 2019/1058 (2019), https://eprint.iacr.org/2019/1058
16. Barki, A., Gouget, A.: Achieving privacy and accountability in traceable digital currency. Cryptology ePrint Archive, Report 2020/1565 (2020), https://eprint.iacr.org/2020/1565
17. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014). https://doi.org/10.1109/SP.2014.36
18. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_18
19. Bogatov, D., De Caro, A., Elkhiyaoui, K., Tackmann, B.: Anonymous transactions with revocation and auditing in hyperledger fabric. Cryptology ePrint Archive, Report 2019/1097 (2019), https://eprint.iacr.org/2019/1097
20. Brands, S.: Untraceable off-line cash in wallets with observers (extended abstract). In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_26
21. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
22. Cecchetti, E., Zhang, F., Ji, Y., Kosba, A.E., Juels, A., Shi, E.: Solidus: Confidential distributed ledger transactions via PVORM. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 701–717. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134010
23. Chalkias, K., Lewi, K., Mohassel, P., Nikolaenko, V.: Distributed auditing proofs of liabilities. Cryptology ePrint Archive, Report 2020/468 (2020), https://eprint.iacr.org/2020/468
24. Chase, M., Ganesh, C., Mohassel, P.: Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 499–530. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_18
25. Chen, Y., Ma, X., Tang, C., Au, M.H.: PGC: Decentralized confidential payment system with auditability. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part I. LNCS, vol. 12308, pp. 591–610. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-58951-6_29
26. Dagher, G.G., Bünz, B., Bonneau, J., Clark, J., Boneh, D.: Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 720–731. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813674

27. Damgård, I., Ganesh, C., Khoshakhlagh, H., Orlandi, C., Siniscalchi, L.: Balancing privacy and accountability in blockchain transactions. Cryptology ePrint Archive, Report 2020/1511 (2020), https://eprint.iacr.org/2020/1511
28. Decker, C., Guthrie, J., Seidel, J., Wattenhofer, R.: Making bitcoin exchanges transparent. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) ESORICS 2015, Part II. LNCS, vol. 9327, pp. 561–576. Springer, Heidelberg (Sep 2015). https://doi.org/10.1007/978-3-319-24177-7_28
29. Doerner, J., Shelat, A., Evans, D.: Zeroledge: Proving solvency with privacy
30. Dutta, A., Vijayakumaran, S.: Mprove: A proof of reserves protocol for monero exchanges. In: 2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019. pp. 330–339. IEEE (2019). https://doi.org/10.1109/EuroSPW.2019.00043, https://doi.org/10.1109/EuroSPW.2019.00043
31. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
32. Frankle, J., Park, S., Shaar, D., Goldwasser, S., Weitzner, D.J.: AUDIT: Practical accountability of secret processes. Cryptology ePrint Archive, Report 2018/697 (2018), https://eprint.iacr.org/2018/697
33. Fuchsbauer, G., Orrù, M., Seurin, Y.: Aggregate cash systems: A cryptographic investigation of Mimblewimble. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 657–689. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_22
34. Garay, J.A., Kiayias, A.: Sok: A consensus taxonomy in the blockchain era. IACR Cryptology ePrint Archive (2018), https://eprint.iacr.org/2018/754
35. Garman, C., Green, M., Miers, I.: Accountable privacy for decentralized anonymous payments. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 81–98. Springer, Heidelberg (Feb 2016)
36. Goldwasser, S., Park, S.: Public accountability vs. secret laws: Can they coexist? Cryptology ePrint Archive, Report 2018/664 (2018), https://eprint.iacr.org/2018/664
37. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 05. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (Jun 2005). https://doi.org/10.1007/11496137_32
38. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
39. Guts, N., Fournet, C., Nardelli, F.Z.: Reliable evidence: Auditability by typing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 168–183. Springer, Heidelberg (Sep 2009). https://doi.org/10.1007/978-3-642-04444-1_11
40. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In: NDSS 2017. The Internet Society (Feb / Mar 2017)
41. Hu, K., Zhang, Z., Guo, K.: Breaking the binding: Attacks on the Merkle approach to prove liabilities and its applications. Cryptology ePrint Archive, Report 2018/1139 (2018), https://eprint.iacr.org/2018/1139
42. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 357–388. Springer International Publishing, Cham (2017)

43. Küsters, R., Rausch, D., Simon, M.: Accountability in a permissioned blockchain: Formal analysis of hyperledger fabric. IACR Cryptol. ePrint Arch. **2020**, 386 (2020), https://eprint.iacr.org/2020/386
44. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 526–535. ACM Press (Oct 2010). https://doi.org/10.1145/1866307.1866366
45. Lamport, L., Shostak, R.E., Pease, M.C.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
46. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world (2013), https://bitcointalk.org/index.php?topic=279249.0
47. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Papagiannaki, K., Gummadi, P.K., Partridge, C. (eds.) Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013. pp. 127–140. ACM (2013). https://doi.org/10.1145/2504730.2504747, https://doi.org/10.1145/2504730.2504747
48. Moore, T., Christin, N.: Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In: Sadeghi, A.R. (ed.) FC 2013. LNCS, vol. 7859, pp. 25–33. Springer, Heidelberg (Apr 2013). https://doi.org/10.1007/978-3-642-39884-1_3
49. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009), http://bitcoin.org/bitcoin.pdf
50. Narula, N., Vasquez, W., Virza, M.: zkledger: Privacy-preserving auditing for distributed ledgers. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). pp. 65–80. USENIX Association, Renton, WA (Apr 2018), https://www.usenix.org/conference/nsdi18/presentation/narula
51. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
52. Poelstra, A.: Mimblewimble (2016), https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf
53. Poelstra, A., Back, A., Friedenbach, M., Maxwell, G., Wuille, P.: Confidential assets. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC 2018 Workshops. LNCS, vol. 10958, pp. 43–63. Springer, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-662-58820-8_4
54. Roose, S.: Standardizing bitcoin proof of reserves, https://blockstream.com/2019/02/04/en-standardizing-bitcoin-proof-of-reserves/
55. Van Saberhagen, N.: Cryptonote v 2.0 (2013), https://cryptonote.org/whitepaper.pdf
56. Wang, H., He, D., Ji, Y.: Designated-verifier proof of assets for bitcoin exchange using elliptic curve cryptography. Future Gener. Comput. Syst. **107**, 854–862 (2020). https://doi.org/10.1016/j.future.2017.06.028, https://doi.org/10.1016/j.future.2017.06.028
57. Wüst, K., Kostiainen, K., Capkun, V., Capkun, S.: PRCash: Fast, private and regulated transactions for digital currencies. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 158–178. Springer, Heidelberg (Feb 2019). https://doi.org/10.1007/978-3-030-32101-7_11

A Cryptographic Background

A.1 Consensus

A consensus protocol allows a number of nodes to output a common agreement on input of a sequence of messages. In our setting, the commonly agreed value is typically recorded on a public ledger. The basic properties of a consensus protocol are [34] a) *Consistency:* On some input, all honest nodes make the same output. b) *Liveness:* An input proposed by some honest node will be eventually processed by all honest nodes after a finite number of rounds. A common distinction among consensus protocols is according to their failure model, where *crash tolerant* protocols assume failed nodes may become offline or otherwise stop interacting with the system, while *Byzantine tolerant* [45] protocols assume such nodes might also engage into malicious activity in order to defeat the above properties. These models typically assume different levels of adversarial power needed for the system to fail. Another distinction is based on the participation model, where *permissioned* consensus participation is open only to a closed set of parties, while *permissionless* is open to anyone, which however needs a mechanism to prevent attacks through "sybil" identities such as Bitcoin's Proof of Work [49] or Proof of Stake protocols [42].

A.2 Distributed Payment Systems

A distributed payment system DPS (also known as ledger-based payment system) can be simply defined by the following algorithms and protocols when already assuming the existence of a consensus layer.

- $\mathsf{pp}, L \leftarrow \mathsf{Setup}(\lambda)$: on input of security parameter λ, outputs public parameters pp and initializes a public ledger L to be maintained by the consensus layer. This algorithm is executed once in the setup phase of the system, and is run by either a single party or a quorum of parties in a multi-party computation (MPC) protocol. In the following algorithms and protocols, pp and L are default inputs and are omitted for simplicity.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CreateAcc}()$: Run by any party wishing to transact in the system[5], outputs a public key pair.
- $\mathsf{tx} \leftarrow \mathsf{CreateTx}(\mathsf{sk_S}, \mathsf{pk_R}, v)$: Run by a sender wishing to send value v to receiver, and outputs a transaction tx. Although here for simplicity we assume a single sender and receiver, a transaction can generally accommodate multiple senders and receivers. tx is sent to the consensus layer in order to be included in L after verification.
- $\mathsf{VerifyTx}(\mathsf{tx}) := \{0, 1\}$ Verifies the validity of a transaction tx, given the state of the ledger L. Verification is typically performed in a distributed fashion in the consensus layer among all verifiers (often called "miners"), where agreement results in the update of the ledger's state to L' which contains tx.

[5] Although participation in payment systems is typically achieved through public key cryptography, some systems achieve it through other primitives (e.g. spending in Mimblewimble [33,52] requires knowledge of a commitment's blinding factor).

A DPS can be *permissioned* where all participants are known (typically controlled by a single entity or organization), or *permissionless* where participation is open to anyone, resulting in more decentralization (in the above generic definition we do not distinguish between a permissioned and a permissionless system). Note that although participation in the system is typically consistent with the consensus protocol (i.e. transacting in a permissioned payment system implies a permissioned consensus, similarly for permissionless), this is not always the case. For example, it is possible to run a permissionless payment system with a permissioned consensus layer, such as a permissionless "Fabcoin" on top of Hyperledger Fabric [14].

A distributed payment system *must* satisfy the following core properties, which are typically safeguarded by verifiers participating in the consensus algorithm (e.g. "miners")

- *Theft prevention:* Spending values from a sender account S requires knowledge of private information associated with that account (typically a secret key sk_S).
- *Balance:* A transaction which transfers a value v from a sender S to a receiver R, should always increase a receiver's total assets by v and adjust S's total assets by $-v$.
- *Non-negative assets:* A transaction that spends v from sender's total assets, should not result in negative assets for S which would allow S to overspend.

A.3 Commitment schemes

are very commonly used in private DPSs, to hide transaction information. A non-interactive commitment scheme $\mathsf{Com}(\mathsf{pp}, m, r)$ takes as input public parameters pp, a message m and randomness r and outputs a commitment value cm. This value reveals no information about the message (*hiding* property) while it is hard to find (m', r') such that $\mathsf{Com}(\mathsf{pp}, m, r) = \mathsf{Com}(\mathsf{pp}, m', r')$, when $m' \neq m$ (*binding* property). Certain commitment schemes, i.e. Pedersen commitments [51] allow for homomorphic operations over committed values, a useful property in private DPSs.

A.4 Zero Knowledge Proofs

A Zero Knowledge proof is an interactive protocol between a prover P and a verifier V where P based on a common input statement proves knowledge of a witness w without revealing to V any additional information other than this fact alone. In DPSs, zero-knowledge proofs are used extensively to provide privacy-preserving attributes, with transacting parties proving validity of a transactions based on a public ledger without revealing the full transaction details, while in recent works they are also used to prove compliance with regulatory requirements.

Range Proofs are Zero Knowledge protocols proving that a committed value v lies within some interval (a, b), with v as the witness. In a payment system

setting, such proofs are typically used to show that v is positive or does not overflow a maximum presentable value. Most well-known construction families for range proofs include square decomposition [37], multi-base decomposition [53] and Bulletproofs [21], with the latter being the most efficient in terms of proof size. Obviously, one can generate constant size range proofs from trusted-setup based SNARKs like Groth16 [38]. In privacy-preserving DPSs they are often used to ensure their basic core properties discussed in Section A.2, but they are also used for regulation purposes (e.g. distinguish between transactions that exceed a value limit).

A.5 Zero Knowledge Proofs

An interactive zero-knowledge proof (ZKP) for statement $\{w : f(w, x)\}$ where x is publicly known and witness w is known only to prover P, is a protocol between P and verifier V that proves P's knowledge of w such that $f(w, x)$ holds. This protocol needs to satisfy the following:

- **Completeness:** Honest V is always convinced by an honest P who knows a valid witness w.
- **Soundness:** A malicious prover P^* cannot convince a verifier for a false statement.
- **Zero Knowledge:** After executing the protocol, a verifier does not learn any additional information other than the validity of the statement.

An interactive ZKP can be converted to a non-interactive zero knowledge proof (NIZK) using the Fiat-Shamir heuristic [31]. In turn, a ZK - Succinct Non-interactive ARgument of Knowledge (zk-SNARK) is a non-interactive zero-knowledge proof that is succinct, namely its proofs are very short ($O(\lambda)$ with efficient verification $O(\lambda|x|)$ [18].