Broken Proofs of Solvency in Blockchain Custodial Wallets and Exchanges

Konstantinos Chalkias[1], Panagiotis Chatzigiannis[2], and Yan Ji[3]

[1] Meta / Novi Financial
[2] George Mason University
[3] Cornell University

Abstract. Since the Mt. Gox Bitcoin exchange collapse in 2014, a number of custodial cryptocurrency wallets offer a form of financial solvency proofs to bolster their users' confidence. We identified that despite recent academic works that highlight potential security and privacy vulnerabilities in popular auditability protocols, a number of high-profile exchanges implement these proofs incorrectly, thus defeating their initial purpose. In this paper we provide an overview of *broken* liability proof systems used in production today and suggest fixes, in the hope of closing the gap between theory and practice. Surprisingly, many of these exploitable attacks are due to a) weak cryptographic operations, for instance SHA1 hashing or hash-output truncation to 8 bytes, b) lack of data binding, such as wrong Merkle tree inputs and misuse of public bulletin boards, and c) lack of user-ID uniqueness guarantees.

Keywords: blockchain, custodial wallets, solvency proofs, light clients, Merkle trees, public bulletin board, cryptographic attacks, data binding, hash-truncation, dispute resolution.

1 Introduction

It is considered a good practice for organizations that accept and manage customer funds (ranging from banks to blockchain custodial wallets) to be periodically audited for their financial solvency, i.e., showing that they have enough assets to pay back their customers. These solvency audits imply that the amount of total assets owned by the organization at least matches[4] its total liabilities, which are effectively its customers' deposits.

With many cryptocurrencies emerging during the last decade, several centralized organizations appeared offering various types of services such as exchanges, online wallets and interest accounts [29]. With blockchain technology at its infancy, these unregulated services remained opaque to their internal operations. Unsurprisingly, given the absence of a universal legal framework and centralized protection against fraudulent or malicious acts, combined with the delicacy of handling private keys required to spend cryptocurrency assets, a number of exchanges have lost their deposits and declared bankruptcy. In the most infamous case, the collapse of Mt. Gox (one of the oldest exchanges in Bitcoin's history), over $450M in customer assets were lost [34,35]. Other bad practices on behalf of those organizations include investing users' funds without their consent [19].

Towards implementing methods for preventing such fraudulent behavior against the cryptocurrency users, decentralized solutions [24,26,27,28,30,33,36] requiring customers to jointly participate on the auditing process were proposed as an alternative or complementary method to conventional auditing; therefore placing less trust on centralized auditors and empowering customers to verify that their own account and balance is indeed part of a Proof of Liabilities (which in turn is part of a Proof of Solvency), which cannot be achieved by centralized auditing. Ultimately, the goal exchanges proving their solvency is to earn users' trust in a distributed fashion while minimizing

[4] In certain cases, partial solvency might be sufficient, however for the purposes of our paper these cases are equivalent.

disclosure of additional information that could potentially expose clients' data, therefore preserving the "decentralization" and "(pseudo)-anonymity" characteristics. Naturally, we are now observing a rising demand in standardizing proof of solvency in the digital assets industry [23,28].

However, these new technologies, especially those in the blockchain space, are still being studied in terms of their security guarantees, and special considerations around potential weaknesses must be made during implementation. Proof of liabilities is no exception to this rule. All liability proving systems should serve the same goals: prove liabilities without understating obligations, while preserving privacy. However, recent works showed that these goals are not always met [28,29,32].

Our contributions. This work is the product of recent academic research on auditability and solvency, combined with extensive discussions with blockchain researchers in academia and auditing firms and stakeholders in blockchain associations and organizations. We show that improvements are still needed towards earning user trust, as we observe several cases of "broken" proof of liabilities implementations, therefore making these proofs disputable. After pinpointing to exploitable real world solvency processes, we provide the relevant discussion and context for addressing these issues towards closing the gap between theory and practice. Unfortunately, every analyzed solvency tool in this paper suffers from one or more exploitable security issues.

Paper Organization. The rest of the paper is organized as follows: In Section 2 we provide informal definitions for the related auditability proofs, and discuss different types of wallets to highlight the setting where these proofs are applicable. In Section 3 we provide a high-level overview of existing Proof of Liabilities schemes in the literature, and their potential security or privacy issues. In Section 4 we show the weaknesses of existing practical PoL implementations in the blockchain space and suggestions for fixing them. In this Section we also highlight subtle differences which exist in some of those implementations, which do not make them necessarily insecure, but implicitly operate under a different trust model. We provide our final remarks and conclusions in Section 5.

2 Background

2.1 Definitions

We first informally present the basic properties required for the auditability proofs we are exploring based on the formal definitions in [33]. The participating parties in a Proof of Liabilities (PoL) protocol are the following: a) The exchange $\mathcal{P}$ which is in the prover role and b) The exchange customers $\mathcal{U} = \{u_1, \cdots, u_n\}$ in a verifier role. In our setting, $\mathcal{P}$ publishes a commitment to a liabilities dataset L on a public bulletin board such as a blockchain. Then on a user's query, $\mathcal{P}$ proves that the user's balance with the exchange is indeed part of L. A PoL scheme should ensure:

- *Security*: $\mathcal{P}$ will not be able to "hide"/"understate" its liabilities (note that $\mathcal{P}$ has no incentive to increase the total liabilities).
- *Privacy*: Any user u_i should not learn from the proof any information besides that its account balance is indeed included in L (e.g. total number of clients, other users' balances etc.)

2.2 Types of cryptocurrency wallets/exchanges

The key management process is an important component of digital asset custody that clients should evaluate when choosing a wallet. This work focuses on custodial wallets, but as we realized many misconceptions around terminology even among experts, we provide a wallet-type categorization depending on who controls the on-chain private keys. Interestingly, blockchain wallets exist in different flavours regarding the offered capabilities and processes. There are also cases of hybrid types, where both the users and some custodian(s) mutually control the private keys [10]. Based on the above, we enlist four of the most popular wallet-types in the blockchain industry.

Pure Non-Custodial (PNC) In PNC wallets [11,14] (often mentioned as self-custody), users control their on-chain assets by having full control over an actual blockchain address. Most of these will expect a recovery key or passphrase that lets users load the wallet on other devices and wallet software. These keys must be kept safe because once compromised anyone can access and control funds contained in the wallet. There exist many kinds of PNC wallets, such as mobile, desktop, hardware, paper, secure-enclave based, browser plugins and more.

Remark 1. *PNC wallets poses account recoverability risks, especially for non-experts, given many recent examples of people losing their keys [25], but indeed there is no dependency on custodians and solvency proofs are irrelevant.*

Assisted Non-Custodial (ANC) To circumvent the key-loss issue, ANC wallets (e.g. ZenGo and Conio [22,8]) remove the burden of the single atomic private key and split the responsibility between multiple parties. Typically, these wallets use a 2-out-of-3 key share policy, where one part of the key is known to the user (i.e., a passphrase), the other part is stored encrypted in some user-controlled encrypted storage (i.e., iCloud) and the third share is controlled by a custodian. In practice, two of the shares should participate in transaction signing, which implies that the custodian is a minority and cannot spend without the user's permission. If users forget their passphrase, they can still sign a transaction using their iCloud plus custodian's shares.

Remark 2. *There are variants of the above 2-out-of-3 approach, where instead of cloud storage, the user's bank has a share (i.e., see Conio.com wallet). In the above approach, the two "custodians" need to collude in order to cheat, so although no entity controls more than 33.3% of the key material, it would be interesting to explore if legislation will enforce requiring a custodial operation license. However, it is clear that ANC wallets don't require the custodian to prove solvency either.*

Omnibus Custodial (OC) An OC wallet (e.g., Coinbase[5] [7] and Binance [3] exchanges) holds user funds in pools of funds on-chain while balances are managed on a private ledger. The bulk of user funds is usually stored in one or more on-chain addresses even though the exchange might have millions of users. This means that there is not a 1-1 matching between a user account and an on-chain address. In fact, users do not interact with the blockchain at all and typically they are not even aware of what addresses their wallet provider controls. Due to user-experience benefits, reports from Chainalysis [13] show that custodial services are very popular among current cryptocurrency owners. Clearly, this is the major wallet type where proofs of solvency should apply.

Remark 3. *In case a user loses his/her password to an OC wallet, in most cases funds can be restored by KYC (know your customer) or "forgot my password" 2FA methods. Another benefit is protection of user's privacy against outsiders (only the custodian can see who owns what).*

Segregated Custodial (SC) In SC wallets, each user is assigned with one (or more) blockchain addresses, but the custodian controls all of the private keys on behalf of the users, as in Dapper wallet [9]. It is highlighted that if wallet users cannot track which on-chain addresses correspond to their accounts, a proof of solvency solution is also recommended.

Remark 4. *In SC wallets there is no mixing of cryptocurrencies from different users under the same address. This allows for public observability of "which account owns what" and this is usually considered as more transparent, but less privacy preserving, compared to OC.*

[5] Note that Coinbase.com exchange is different from Coinbase wallet: Coinbase.com is an OC exchange, while Coinbase wallet is a PNC wallet, similar to Metamask. This subtle distinction [6] has caused confusion in the past with people losing their keys in the Wallet (and therefore their funds as well).

2.3 Proof of liabilities in wallets and exchanges

Having discussed the above flavors of wallets and exchanges, we now need to emphasise that PoL is applicable to *custodial* wallets only, as in any non-custodial wallet type, users are directly involved in their assets management. Intuitively, a PoL protocol is easier to implement in a SC approach, and is more challenging in an OC approach, as the absence of a 1-1 match between user and on-chain address makes implementing a secure PoL (that prevents an exchange from hiding liabilities) more complex. In addition, as custodial wallets are popular among cryptocurrency owners [13], implementing secure proof of reserves for custodial wallets has already been proposed as an essential tool for the blockchain industry [23].

3 Proof of Liabilities schemes

We now provide a brief overview of PoL schemes proposed in the blockchain setting [26,27,28,30,36]. All schemes follow the same paradigm we discussed in the previous section: the prover publishes a commitment and each user checks if his/her balance is properly included accordingly.

Provisions[30] proposes a PoL scheme in which $\mathcal{P}$ commits to each user's balance on a public bulletin board (PBB) so users can check if their balances are properly included. In particular, the balances are in homomorphic Pedersen commitments and proven positive in a zero-knowledge manner so that the values of users' balances are concealed. However, the commitment size on a PBB is linear in the number of users so it doesn't meet the efficiency requirement in [33]. Additionally, the number of users is public and dummy accounts need to be added to mitigate this privacy issue.

Maxwell-Todd scheme [36] initiated the line of works using a "summation" Merkle tree to prove total liabilities while minimizing the cost on PBB. In this Merkle tree variant, apart from the hash field h in each tree node, there's an additional value field v which is the sum of the values of the child nodes as shown in fig. 1. In this approach, $\mathcal{P}$ generates the summation tree and publishes the Merkle root, with the value of the root representing the amount of $\mathcal{P}$'s total liabilities. Then any u can query $\mathcal{P}$ for a Merkle proof to make sure the amount is included in the tree. For example, in fig. 1, on query by User 1, say Alice, $\mathcal{P}$ would reply with the Merkle path $(h_2, v_2), (h_6, v_6)$, which would enable Alice to verify the proof for the total liabilities in the root v_{root}.

This approach however is problematic, as it enables $\mathcal{P}$ to claim less liabilities [32] by computing each tree node value as $v = \max(v_l, v_r)$ while still being able to generate inclusion proofs and successfully pass any user queries. In addition, the value of the total liabilities is public (which is not ideal), and anyone can infer the population and/or individual liabilities though a series of inclusion proofs. Unfortunately, this scheme is still being used in existing cryptocurrency exchanges; we explain the security issues by an easy to follow example in Section 4.1.

Maxwell+[32] fixes the vulnerability in the previous protocol by simply modifying each node field to include both values and hashes of child nodes, i.e., $h = \mathcal{H}(v_l||v_r||h_l||h_r)$. This modification effectively binds the value of each node in the parent, and prevents the manipulation of the tree we discussed previously. However this scheme still does not address the privacy issues we mentioned.

Maxwell++[27] further extends Maxwell+ to conceal the population and individual liabilities by breaking and shuffling values into small units, but still without concealing $\mathcal{P}$'s total liabilities.

Camacho[26] provides privacy of liabilities by replacing the values in Maxwell-Todd by Pedersen commitments associated with zero-knowledge range proofs. The hiding and binding properties of Pedersen commitments provide privacy while preventing liability manipulation, and their homomorphic properties allows summation in commitments. The total liabilities can only be disclosed by voluntarily opening the commitment, or alternatively, $\mathcal{P}$ can prove the range of the total liabilities by utilizing zero-knowledge proofs. While Camacho hides liabilities, it leaks the number of users, and is also susceptible to the flaw we discussed in Maxwell-Todd's scheme.

DAPOL[28] and *DAPOL+* [33] improve Camacho by adopting the proposed fix in [32], and use sparse Merkle trees (SMT) to more efficiently hide the number of users.

4 Vulnerabilities in PoLs in Practice and Mitigations

Table 1: *Vulnerabilities in existing PoL implementations in practice.*

Issues identified	**Affected VASPs/auditors**	**Mitigation**
Vulnerable summation tree	BHEX, Deloitte's audits	Bind left/right values vs. sum
Short hash collisions	BHEX	Avoid hash truncation
Shared user ID	Coinfloor, Kraken, BitMEX, Armanino's audits	Ensure unique user IDs
Inconsistent root commitment	Coinfloor, BitMEX	Commit on PBB
Leak of individual liabilities	BHEX, Deloitte's audits, Coinfloor	Use Pedersen commitments
Leak of number of users	BHEX, Deloitte's audits, Coinfloor, Kraken	Use sparse Merkle tree

Despite extensive theoretical research around PoL, as discussed in the previous section, we still observe a significant gap between these works and the implementations in practice. Among the hundreds of blockchain exchanges, only a handful of them support PoL [15]. Some rely on a trusted auditor to prove their liabilities, such as Kraken [1]. However, this trust model is not desired due to the possibility of collusion between the auditor and the exchange. Such auditing might fail when the auditor is deceived such as in the infamous Enron scandal [12]. On the other hand, while other exchanges assume a decentralized and trustless model (in line with the blockchain space), their implementation suffers from exploitable attacks that could result in understated liabilities.

In this section we present our survey of the existing landscape for performing PoL by highlighting these "problematic" implementations and our suggested fixes. We highlight, that although cryptocurrency exchanges could potentially take advantage of these incorrect implementations by under-reporting liabilities, our observations by no means imply that exchanges acted through malicious intent. In fact, it is highly possible that lack of a) proper code/protocol auditing and/or b) cryptography expertise are the major factors of the observed insecure implementations.

We do believe however that our results can be taken into account by the respective exchanges, blockchain associations and auditors to further increase customer confidence and potentially result in a healthier cryptocurrency ecosystem.

4.1 Vulnerable summation tree

We first observe exchanges or auditors using the vulnerable Maxwell's summation tree [36] in practice for proof of liabilities, such as [18] or [16] used by BlueHelix Exchange (BHEX / HBTC) and [2] by Deloitte's audit for ICONOMI. In these implementations, the hash field of each node in the tree is defined as $h(v_l + v_r||h(l)||h(r))$, as shown in fig. 1, where l and r denote the child nodes. However, as we discussed in Section 3, this opens up the possibility for the exchange to claim less liabilities, as shown in fig. 2, namely computing each parent as $h(max(v_l+v_r)||h(l)||h(r))$ therefore computing a different "version" of the liabilities tree on the fly, without the clients ever realizing this. For instance, User 1 will receive a proof with a valid Merkle path $(h_2, v'_2 = 5), (h_6, v'_6 = 0)$ without detecting any misbehavior. In the worst case, a malicious VASP could only report the highest user balance as being its full liabilities (and not the sum of every balance). To mitigate this vulnerability, we can apply the technique in [32] by defining the tree as $h(v_l||v_r||h(l)||h(r))$. This construction includes both the left and right child value separately (instead of just the sum), so the trick of reduced liabilities is no longer feasible.

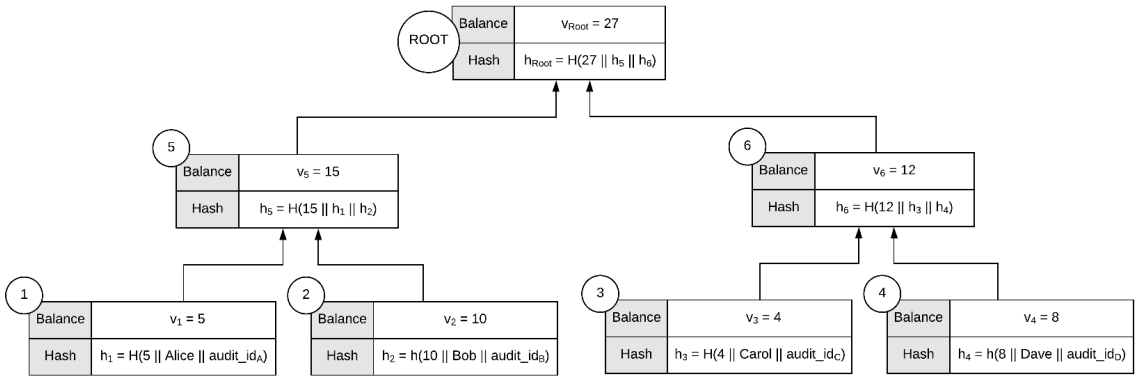

Figure 1: *Exploitable summation tree*

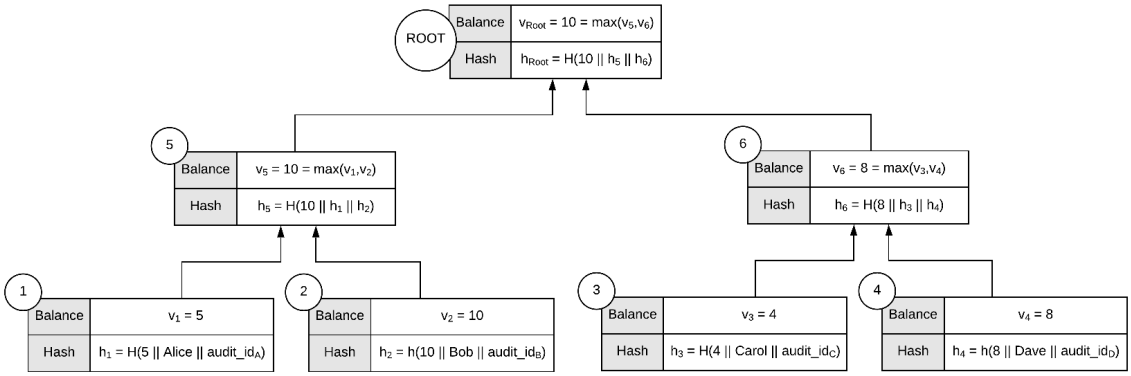

Figure 2: *Claiming reduced liabilities*

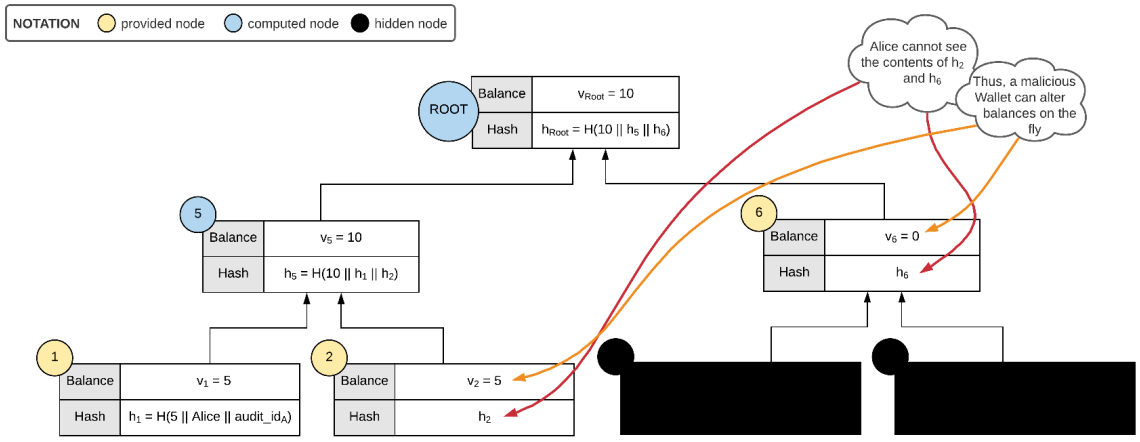

Figure 3: *Proof of reduced liabilities*

4.2 Short Hash Collisions

In BHEX's PoL implementation [18], we also observe that a 64-bit truncated SHA256 hash is used for all nodes in the tree. However, this level of truncation is insecure due to increased possibility of hash collisions. Once an adversary finds a collision in hash for two users, she can assign them to the same leaf node and return the same Merkle path to them. The total liabilities can then be under-reported without being detected by any user. The collision chance is significant especially given the vast computational power that exists in the Bitcoin mining setting. In fact, an exchange does not need to target for a specific collision, but rather search for *any* collision in the whole tree (i.e. a "multi-target attack"), which increases the likelihood of success even more given the large sizes of such trees. As a rough estimate, given that Bitcoin blocks are now produced with a 80-bit chosen prefix, a malicious wallet could produce truncated SHA256 collisions using a tiny fraction, roughly $1/2^{16}$ of today's global mining power in Bitcoin, making such attacks economically viable. However for multi-target attacks, an exchange with 16+ millions of users would require about 2^{40} of hash invocations to find a match, making this attack practically feasible. In short, truncating hashes in liability proofs is not a recommended practice for earning the trust of the community. In addition, we discourage the use of insecure hash functions such as the SHA1 used in Coinfloor [4].

4.3 Shared User ID

When verifying inclusion proofs of individual liabilities, users need to make sure proofs they receive are uniquely binding to each account. However, this is not always done correctly in practice. We observe that PoL schemes used by Coinfloor [4], Kraken [1], BitMEX [20] and Armanino's audits [17] for Gate.io [21] and Ledn.io [5] do not guarantee uniqueness of users' IDs, and can potentially let a malicious prover to reuse the same user ID for different users. This in turn would allow selecting the same user ID for different users having the same balance (and therefore mapped to the same leaf node in the Merkle tree or the same entry in the report file) and eventually hide liabilities. We believe the original intent of those exchanges was to provide privacy for the user by randomly assigning them user IDs, however having user IDs purely determined by the prover remains exploitable. Similarly, BitMEX attempts to preserve privacy of individual liabilities and number of users by splitting each user account into multiple leaf nodes as in Maxwell++ [27]. However, there's no binding between a user and the corresponding leaf nodes, so the prover can reuse the same leaf nodes for different users (and then claim less liabilities) without being detected.

On the other hand, we observe that BHEX and Deloitte PoL implementations do not suffer from such issues, as they use their users' unique usernames and e-mail addresses when deriving these user IDs, which enforces their uniqueness. In practice, to comply with EU's GDPR regulation, exchanges could ask from the users to provide a long unique-ID and avoid hashing private information, such as email addresses. All in all, we encourage binding unique user credentials with user IDs and all corresponding leaf nodes in Maxwell++ style constructions or better DAPOL+.

4.4 Multiple root commitments

In a secure PoL scheme, the prover needs to publish the commitment on a public bulletin board (PBB) to guarantee users have consistent views of the public commitment. Otherwise, a malicious prover can supply different commitments to different users so that the proofs received are all valid but are not binded to the same commitment. In this way, the prover can effectively omit some balances in each commitment without being detected. Although most PoL implementations we looked into, such as in BHEX, Deloitte and Kraken, mentioned that the commitment should be "published", the operation of *publishing on PBB* are sometimes done inappropriately. For example, Coinfloor publishes a hash of the report file on blockchain and then the transaction ID on its website [4]. Since blockchains are considered as a practical implementation of a PBB [31], the former operation guarantees non-equivocation of the commitment. However, when the prover serves

the transaction ID on its own website, the whole liability proof becomes problematic. If users only access this particular website, the prover who controls that website might present different versions of the commitment to different users, thus opening the possibility of hiding liabilities. Similarly, BitMEX publishes the commitment on an AWS server, which however remains exploitable.

To avoid equivocation, we encourage exchanges to publish these commitments on a PBB such as a blockchain, and users verifying their commitments would be confident that proof manipulation would be practically infeasible because of the blockchain's immutability properties. Users should also ensure that no more than one commitment for the same solvency round is published in the PBB. As an alternative approach, the commitment could be digitally signed by a trusted auditor.

4.5 Privacy concerns

As discussed in Section 2.1, there are also privacy concerns on PoLs in addition to the potential attacks to claim less liabilities. While DAPOL+ includes a formal general definition of privacy [33], here we particularly focus on the privacy of individual liabilities and number of users, and demonstrate how much sensitive data is leaked in each scheme.

First, the PoL implementations by BHEX, Deloitte's audits and Coinfloor leak individual user liabilities. In particular, PoL proofs received by users consists of the balance of one or multiple other users. We recommend using Pedersen commitments like in DAPOL+ to hide individual liabilities. Alternatively, the prover can split each account into multiple entries as in BitMEX, but this results in an efficiency degradation proportionally to the number of entries each account is split into.

Additionally, the schemes of BHEX, Deloitte's audits, Coinfloor and Kraken leak the total number of users. Using sparse Merkle tree as suggested in DAPOL+ could mitigate such problem. As above, the exchanges could simply split each account into multiple entries and therefore also hide the number of users, but again this approach comes at a cost of efficiency.

4.6 Dispute resolution and private verification pattern

Apart from the fundamental security and privacy requirements for PoL, there are a few additional problems that may be of concern in practice but not carried out right now. One is dispute resolution, which is still an open research problem [29]. For instance, a user might fail to verify a liability proof, however there is no mechanism or protocol in place to resolve a dispute between an exchange and a user against a third party judge, when the former claims that the user's account balance is indeed included in the liabilities proof and the latter claims the opposite.

Private verification pattern as discussed in [33] is another subtle problem to be concerned. For each audit, not all users actually query for PoL proofs and perform verification. If a malicious prover learns that some users never verifies the proofs, she can safely omit their balances in the next solvency round. Therefore it would be nice to hide users' verification pattern from the prover. Outsourcing proofs to a trusted auditor like in Kraken helps mitigate this issue but there is privacy leak to the auditor and Kraken's implementation doesn't provide verifiability of the sum but only individual inclusions. It remains an open problem to efficiently hide verification pattern without a trusted third party, although private information retrieval techniques have been proposed [28].

5 Conclusion

Having provided an extensive survey of the cryptocurrency exchange landscape in terms of their liability proofs, by pinpointing problematic implementations and their mitigations, we observe that few exchanges implement a Proof of Solvency protocol, and even fewer implement it correctly. We highlight 4 different attack types in these proofs, as well as trust models inconsistent with blockchain decentralization and privacy concepts. We believe that lack of user education combined with these few, exploitable implementations are responsible for the slow adoption of those proofs. We hope our work, being practical in nature, will directly serve as a standard towards improving cryptocurrency exchange auditability and improve user trust in the blockchain ecosystem.

References

1. Audit: Learn about kraken's audit process, https://www.kraken.com/proof-of-reserves-audit
2. Bhex 100% proof of reserve, https://medium.com/iconominet/proof-of-solvency-technical-overview-d1d0e8a8a0b8
3. Binance exchange, https://www.binance.com/
4. Bitcoin audits, https://web.archive.org/web/20210706073111/https://coinfloor.co.uk/hodl/proof/#reports
5. Check your proof of reserves in 5 simple steps, https://blog.ledn.io/en/blog/proof-of-reserves/step-by-step
6. Coinbase blog, https://blog.coinbase.com/goodbye-toshi-hello-coinbase-wallet-the-easiest-and-most-secure-crypto-wallet-and-browser-4ba6e52e4913
7. Coinbase exchange, https://www.coinbase.com/
8. Conio wallet, https://www.conio.com/en/
9. Dapper account manager, https://www.meetdapper.com/
10. Digital wallets - variations and features, https://cryptoapis.io/blog/41-digital-wallets-variations-and-features
11. Electrum bitcoin wallet, https://electrum.org
12. Enron scandal, https://en.wikipedia.org/wiki/Enron_scandal.
13. Mapping the universe of 460 million bitcoin addresses, https://blog.chainalysis.com/reports/bitcoin-addresses
14. Metamask - a crypto wallet & gateway to blockchain apps, https://metamask.io/
15. Nic's por wall of fame, https://niccarter.info/proof-of-reserves/
16. Proof of liabilities implementation, https://github.com/olalonde/proof-of-liabilities
17. Proof of reserves, https://www.armaninollp.com/software/trustexplorer/proof-of-reserves/
18. Proof of solvency: Technical overview, https://support.hbtc.co/hc/en-us/articles/360046287754-BHEX-100-Proof-of-Reserve
19. Tether's bank says it invests customer funds in bitcoin, https://www.coindesk.com/tethers-bank-says-it-invests-customer-funds-in-bitcoin
20. Tool suite for generating and validating proofs of reserves(por) and liabilities(pol), https://github.com/BitMEX/proof-of-reserves-liabilities
21. Your gateway to cryptocurrency, https://www.gate.io/
22. Zengo wallet, https://zengo.com/
23. Chamber of digital commerce: Proof of reserves – establishing best practices to build trust in the digital assets industry (2021)
24. Bitfury: On blockchain auditability (2016)
25. Blackshear, S., Chalkias, K., Chatzigiannis, P., Faizullabhoy, R., Khaburzaniya, I., Kogias, E.K., Lind, J., Wong, D., Zakian, T.: Reactive key-loss protection in blockchains. Cryptology ePrint Archive, Report 2021/289 (2021), https://ia.cr/2021/289
26. Camacho, P.: Secure protocols for provable security. https://www.slideshare.net/philippecamacho/protocols-for-provable-solvency-38501620 (2014)
27. Chalkias, K., Lewi, K., Mohassel, P., Nikolaenko, V.: Practical privacy preserving proofs of solvency. Amsterdam ZKProof Community Event (2019)
28. 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
29. Chatzigiannis, P., Baldimtsi, F., Chalkias, K.: Sok: Auditability and accountability in distributed payment systems. In: ACNS (2021)
30. Dagher, G.G., Bünz, B., Bonneau, J., Clark, J., Boneh, D.: Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In: CCS (2015)
31. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. In: NDSS. Citeseer (2014)
32. Hu, K., Zhang, Z., Guo, K.: Breaking the binding: Attacks on the merkle approach to prove liabilities and its applications. Computers & Security (2019)
33. Ji, Y., Chalkias, K.: Generalized proof of liabilities. In: CCS (2021)
34. McMillan, R.: The Inside Story of Mt. Gox, Bitcoin's $460 Million Disaster. https://www.wired.com/2014/03/bitcoin-exchange/ (2014)
35. Moore, T., Christin, N.: Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In: FC (2013)
36. Wilcox, Z.: Proving your bitcoin reserves. https://bitcointalk.org/index.php?topic=595180.0