IZPR: Instant Zero Knowledge Proof of Reserve

Trevor Conley[1], Nilsso Diaz[1], Diego Espada[1], Alvin Kuruvilla[1], Stenton Mayne[2], Xiang Fu[1][0000−0002−6608−1654]⋆

[1] tconley,ndiaz5,despada1,akuruvilla1@pride.hofstra.edu; Xiang.Fu@hofstra.edu
Hofstra University
[2] stentonian@kn0x1y.xyz
Kn0x1y

Abstract. We present a non-interactive and public verifier scheme that allows one to assert the asset of a financial organization *instantly* and *incrementally* in zero knowledge with high throughput. It is enabled by the recent breakthrough in lookup argument, where the prover cost can be independent of the lookup table size after a pre-processing step. We extend the `cq` protocol [21] and develop an aggregated non-membership proof for zero knowledge sets. Based on it, we design a non-intrusive protocol that works for pseudo-anonymous cryptocurrencies such as BTC. It has $O(n\log(n))$ prover complexity and $O(1)$ proof size, where n is the platform throughput (instead of anonymity set size). We implement and evaluate the protocol. Running on a 56-core server, it supports 1024 transactions per second.

1 Introduction

First defined in Provisions [15], the zero knowledge (zk) proof-of-solvency (PoS) problem is to prove in zero knowledge that the *asset* of an organization is greater than its *liability*. Due to the volatility of financial market, we are interested in a variation called *instant zk-solvency* problem, i.e., to prove solvency of an organization instantly and incrementally, e.g., at a frequency of $1Hz$. This seems like a daunting task that requires very expensive computing resources. For instance, in [15] to generate and then verify a single zk-solvency proof for an anonymity set of $500k$ BTC addresses needs about 1 hour. In our work, we use the entire 80 million BTC addresses as the anonymity set, and in fact it can be arbitrarily large. The key observation of our work is that *the ownership of an account needs to be proved only once*, and then the total asset can be proved incrementally. We assume that the registration and verification of liability is handled by frameworks such as DAPOL(+) [10, 27], then we just need to focus on the instant zk-proof for asset only. In the rest of the paper we use terms "proof-of-reserve" and "proof-of-asset" (PoA) interchangeably. The "asset" ("reserve") here refers to the asset that is in control by the organization.

We consider pseudo-anonymous cryptocurrencies such as Ethereum and BTC where transaction details are public. Here, we abstract away platform specifics

⋆ Corresponding Author

and use "account" to denote e.g., BTC wallet address and Ethereum account. [3] Let $\mathbf{S}$ represent the account set owned by an organization. At each transaction cycle, the market (or a trusted smart contract) discloses two vectors: (1) $\mathbf{a}$: the list of all accounts that are involved in any transaction in the current cycle; and (2) $\boldsymbol{\Delta}$ where each $\boldsymbol{\Delta}_i$ represents the balance change of the i'th account in $\mathbf{a}$. Let v be the total balance change for $\mathbf{a} \cap \mathbf{S}$, and let $\mathbf{C_S}$ be the Pedersen commitment to $\mathbf{S}$. We present a constant size zk-proof which convinces the verifier that a commitment $\mathbf{C}_v$ hides the valid value for v, given $\mathbf{a}$, $\boldsymbol{\Delta}$, and $\mathbf{C_S}$. Since $\mathbf{C_S}$ does not disclose information of $\mathbf{S}$, the incremental balance change of $\mathbf{S}$ is proved in zero knowledge.

The main challenge here is the concrete performance. We leverage the recent break-through in lookup arguments [24, 36, 31, 22, 37, 26, 21]. Built upon the `cq` protocol [21], we develop an aggregated non-membership proof for zero knowledge sets, i.e., to show that the intersection of two committed zk-sets is empty, which is used to demonstrate accumulated balance over $\mathbf{a} \cap \mathbf{S}$. Because we leverage the pre-processed lookup arguments, the prover complexity is independent of the size of $\mathbf{S}$. Let n denote the throughput of the platform, the prover cost is $O(n\mathsf{log}(n))$ field operations and a constant number of group multi-exponentiation of size $O(n)$. The verifier complexity and proof size are both $O(1)$. Concretely, running over a 56-core server at Google Cloud Computing, our protocol has throughput of 1024 transactions per second (TPS), which is half of Visa Network and 140 times faster than BTC. A second protocol works for privacy preserving cryptocurrencies such as Monero and ZeroCoin and we defer the details to an extended version of this paper.

2 Related Work

The earliest attempt of asset proof is [35] which is not zero knowledge. In [16], the solution depends on special TPM hardware. Provisions [15] provides a zk-solution to both the proof of reserve and liability problems, mainly relying on zk-Σ-protocols. Its prover and verifier complexity are both linear with the size of anonymity set, and it addresses pay-to-pubkey mode of BTC only. MProve(+) [19, 17], Revelio [20], and Nummatus [18] each targets at a specific platform (e.g., Monero and Quisquis), and their design is tightly coupled with the platform (e.g., exploiting the ring signature in Monero). Their prover and verifier complexity are both linear. In [1], zk-SNARK and double discrete logarithm proofs (DDLOG) are combined to prove asset, where DDLOG is very expensive (e.g., costing over 2000 group elements for the proof of one coin). gOTzilla [3] employs MPC-in-the-head and 1-out-of-n oblivious transfer. It can handle anonymity set size of

[3] Even though BTC uses UTXO and allows mixing, from a BTC transaction and its input, one can extract its sender/receiver BTC addresses and the change of balance for each address. In case no-reuse principle is practiced, a BTC address is regarded as a use-once account. In this case, to make out model work, an organization can pre-compute a fixed set of BTC addresses for future use, and periodically expand this set. We discuss its impact on the overall cost of the scheme in Section 5.1.

80 million, but lacks non-interactiveness (i.e., it needs the verifier to be online with the prover).

Scheme	Prover Work	Proof Size	Hashed	NI	Incremental
Provisions [15]	$O(n)$	$O(n)$	N	Y	N
MProve [19, 17, 20]	$O(n)$	$O(n)$	Y	Y	N
zkSNARK e.g. [30]	$O(n\mathsf{log}(n))$	$O(1)$	Y	Y	N
CompositeNIZK [1]	$O(n\mathsf{log}(n))$	$O(n)$	Y	Y	N
gOTzilla [3]	$O(n)$	$O(\mathsf{log}(n))$	Y	N	N
This Work	$O(t\mathsf{log}(t))$	$O(1)$	Y	Y	Y

Table 1. Comparison of Related Work. n: anonymity set size, t: blockchain throughput, m: number of accounts controlled by the organization. In practice, $t << n$, e.g., for BTC t is 7 and n is 80 million. `Hashed:` whether the scheme supports hash in generating account address. Complexity includes field operation cost. `NI:` non-interactive. The complexity of gOTzilla is for the case $m = 1$. We do not know if its proof size scales linearly or sub-linearly with m.

In Table 1, we present a brief comparison of the asymptotic complexity of all the aforementioned related works. Compared with them, the proof size and verifier complexity of our protocol are $O(1)$. In addition, because the platform throughput (e.g., 7 for BTC, or 2000 in Visa network) is much smaller than the anonymity set size (e.g., 80 million for BTC), the concrete cost of prover work in our scheme is much smaller. None of the aforementioned related work is concretely efficient for supporting an incremental, non-interactive, public verifier, and instant zk-asset proof scheme for the scale of the throughput as presented in this paper. On the other hand, the protocol presented in this paper is only used for the incremental zk-proof, and its bootstrap process (see Section 4.1) needs to leverage the "classical" PoA technique. We provide an open-source implementation of our protocol.[4]

In the cryptocurrency industry, there is a recent increase of interest in PoS systems, after the FTX scandal. For instance, Binance provides a PoA system since late 2022 [4]. Its zk-protocol works only for proving liability, and its corporate holdings are manually verified by a third party auditor Mazar [14], who backed off in December 2023. The solution provided in this paper helps to rule out unreliable human factors and it provides *instant* asset report instead of monthly updates. OKX [29] provides a similar PoS system as Binance, where zkSTARK is used to assert user contribution to total liability. It differs in the verification of corporate holdings - OKX publishes all of its wallet addresses, hence losing privacy of its assets. Summa [34] takes a similar approach where the asset addresses are disclosed. More industrial efforts in PoA can be found in [9], and a comprehensive survey of academic work can be found in [11].

[4] `https://github.com/xfu2006/izpr`

3 Preliminaries

3.1 Notations

We use $\mathbf{P}$ and $\mathbf{V}$ to denote the prover and the verifier. Let $\mathcal{G}$ be a generator of bilinear groups, i.e., $(q, \mathbf{g}_1, \mathbf{g}_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, where all groups have order q, and $\mathbf{g}_1$ ($\mathbf{g}_2$) the generator of $\mathbb{G}_1$ ($\mathbb{G}_2$). $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the bilinear map. Let $\mathbb{F}$ be the scalar field of $\mathbb{G}_1$. $s \xleftarrow{\$} \mathbb{F}$ represents sampling s uniformly from $\mathbb{F}$. We use n and N to denote the size of query and lookup tables. We assume the existence of n-th and N-th roots of unity so FFT is applicable. We follow the notations in Groth16 [25]. For instance, ${\mathbf{g}_1}^a {\mathbf{g}_1}^b$ is written as $[a]_1 + [b]_1$. We use $[n]$ to denote range $[1, n]$. A vector $\mathbf{t} \in \mathbb{F}^n$ is written as $(\mathbf{t}_1, \ldots, \mathbf{t}_n)$. $\mathbf{u}+\mathbf{v}$ denotes $(\mathbf{u}_1+\mathbf{v}_1, \ldots, \mathbf{u}_n+\mathbf{v}_n)$, and $\alpha\mathbf{u}$ is $(\alpha\mathbf{u}_1, \ldots, \alpha\mathbf{u}_n)$. We use ω_n for the n'th root of unity, i.e., $(\omega_n)^n = 1$. We define $\mathbb{H}_n = \{\omega_n^1, \ldots, \omega_n^n\}$, and the corresponding set of Lagrange base polynomials is denoted as $\{L_{n,i}(X)\}_{i=1}^n$, where $L_{n,i}(\omega_n^i) = 1$ and $L_{n,i}(\omega_n^j) = 0$ for $j \neq i$. Given a multi-set S, its vanishing polynomial $z_S(X)$ is defined as $z_S(X) = \prod_{s \in S}(X - s)$. It is known that $z_{\mathbb{H}_n}(X) = X^n - 1$. Given $s \in \mathbb{F}$, its Pedersen commitment $\mathbf{C}_s$ is defined as $\mathbf{C}_s = [s]_1 + r[h]_1$ for a random r, where h is unknown to the prover.

Given $\mathbf{t} \in \mathbb{F}^n$, its encoding polynomial $t(X)$ is defined as a $n-1$ degree polynomial: $t(X) = \sum_{i=1}^n \mathbf{t}_i L_{n,i}(X)$. We use the univariate zk-VPD scheme [38] to provide zk-vector commitment. Sample $r_t \in \mathbb{F}$, and the masked polynomial for $\mathbf{t}$ is defined as: $\hat{t}(X) = t(X) + r_t z_{\mathbb{H}_n}(X)$, and the vector commitment to $\mathbf{t}$ is: $\mathbf{C}_\mathbf{t} = [\hat{t}(s)]_1$, where s is the trapdoor of a KZG commitment key [28]. Given a public point z, using the univariate port of the zk-VPD scheme, the prover can convince the verifier that a Pedersen commitment $\mathbf{C}_y$ hides a value $y = t(z)$ without disclosing y. Note that $\hat{t}(X)$ is also a 1-leaky masked polynomial of $t(X)$ [13, Section 3.6] (also in [23, 8]), in the sense that running standard KZG opening proof for one evaluation of $\hat{t}(X)$ still keeps $\mathbf{t}$ zero knowledge. 2-leaky masking is similarly achieved by defining $\hat{t}(X)$ as $\hat{t}(X) = t(X) + (r_{t_1}X + r_{t_0})z_{\mathbb{H}_n}(X)$. In summary, for any vector $\mathbf{u} \in \mathbb{F}^n$, we use $u(X)$, $\hat{u}(X)$, and $\mathbf{C}_\mathbf{u}$ to denote its encoding polynomial, masked polynomial, and vector commitment.

We use the notation from [6] to specify zk-protocols. Consider Schnorr's DLOG as an example: $\pi_{\mathsf{DLOG}}(\mathbf{h})\{(x) : \mathbf{h} = [x]_1\}$. Here $\mathtt{DLOG}$ in π_{DLOG} is a mnemonic. $(\mathbf{h})$ is the public information. The tuple before ":" is the secret known by prover only, i.e., (x). Then the statement inside curly braces states the relation: the prover knows the secret discrete logarithm of $\mathbf{h}$.

3.2 Lookup Argument

Let $\mathbf{t} \in \mathbb{F}^n$ and $\mathbf{T} \in \mathbb{F}^N$, a lookup argument for $\mathbf{t} \hat{\subset} \mathbf{T}$ asserts that for each $i \in [n]$: $\mathbf{t}_i \in \mathbf{T}$. Note that both $\mathbf{t}$ and $\mathbf{T}$ may contain duplicates. We need a homomorphic and zero knowledge look-up argument, and a slight zk-enhancement of $\mathtt{cq}$ [21] satisfies our needs. The basic idea of $\mathtt{cq}$ is to pre-compute commitments of quotient polynomials in $O(N\mathsf{log}(N))$ time. Then for arbitrary query table $\mathbf{t}$ of size n, the prover work is $O(n\mathsf{log}(n))$, thus independent of N. We need to

enhance cq so that it is zero knowledge. In our extension, we exploit bounded leaky-zk as in [12, 13], and Schnorr style Σ-protocols as in [33]. We will present its full technical details in the extended version of this paper. Similar constructions for the zk-enhancement of cq can be found in two concurrent work: segment lookup [13] and matrix lookup [7]. We denote it as $\pi_{\texttt{ZK\_LOOKUP}}$, as shown below:

1. $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}\ (N, \lambda)$: a trusted set-up given security parameter λ and lookup table size limit N, samples bilinear groups and generates the prover and verifier keys $(\mathsf{pk}, \mathsf{vk})$ for KZG.
2. $(\mathsf{aux}_{\mathbf{T}}, \mathbf{C_T}) \leftarrow \texttt{Preprocess}(\mathbf{T}, \mathsf{pk})$: Given the lookup table $\mathbf{T}$, it generates $\mathsf{aux}_{\mathbf{T}}$ (the preprocessed information) and $\mathbf{C_T}$ (a commitment to $\mathbf{T}$).
3. $\pi \leftarrow \texttt{Prove}(\mathsf{pk}, \mathsf{aux}_{\mathbf{T}}, \mathbf{T}, \mathbf{t})$: produces a zero knowledge proof π for $\mathbf{t} \hat{\subset} \mathbf{T}$.
4. $1/0 \leftarrow \texttt{Verify}(\mathsf{vk}, \mathbf{C_T}, \mathbf{C_t}, \pi)$: checks that π is valid.
5. $\pi \leftarrow \texttt{FoldProve}(\mathsf{pk}, \mathsf{aux}_{\mathbf{U}}, \mathbf{U}, \mathsf{aux}_{\mathbf{V}}, \mathbf{V}, \mathbf{u}, \mathbf{v}, \alpha)$: produces a zero knowledge proof π for $\mathbf{u} + \alpha\mathbf{v} \hat{\subset} \mathbf{U} + \alpha\mathbf{V}$.

Theorem 1. *Under the Q-DLOG assumption [21] in the Algebraic Group Model (AGM) and Random Oracle Model (ROM), $\pi_{\texttt{ZK\_LOOKUP}}$ is perfectly complete, computational knowledge sound, and zero knowledge.*

The proof generally follows the analysis of cq [21], and is similar to that of [13, 7]. We delay the details to the extended version. Our zk-cq implementation can be replaced by the one in [7], without impacting the rest of the protocol.

Based upon $\pi_{\texttt{ZK\_LOOKUP}}$, we develop $\pi_{\texttt{RANGE}}(\mathbf{C_a}, 2^B)$, a batched zk-range proof which asserts that $\mathbf{C_a}$ is a Pedersen vector commitment to $\mathbf{a} \in \mathbb{F}^n$, and each $\mathbf{a}_i$ in $\mathbf{a}$ is in range $[0, 2^B)$. It has $O(1)$ proof size and $O(n\mathsf{log}(n))$ prover complexity.

4 IZPR Protocol

4.1 Overview

We first provide a high level overview of the IZPR protocol. As the scheme is based upon cq, IZPR needs a one-time trusted setup to provide prover and verifier keys. It has two stages: a bootstrap stage and an incremental proof stage. **Bootstrap:** the goal of the initial stage is to provide the proof for the ownership of accounts and the initial total balance. In the discussion below, we use "prover" to denote the organization who provides the asset proof.

The prover has the following secret information: (1) $\mathbf{s} \in \mathbb{F}^N$: secret keys. (2) $\mathbf{S} \in \mathbb{F}^N$: public accounts (e.g., for BTC each $\mathbf{S}_i = \mathsf{hash}(g^{\mathbf{s}_i})$ where g is a generator in curve secp256k1 and hash is a combination of SHA256 and RIPMD160). (3) $\mathbf{v} \in \mathbb{F}^N$: initial account balances for a specific timestamp ts, i.e., $\mathbf{v}_i$ is the asset value of $\mathbf{S}_i$.

Here, we specifically note that for the case when the organization practices the no-reuse principle of BTC address, the prover has to pre-compute sufficient private key/public account pairs for future use. If an account does not appear in the blockchain yet, its value is set to 0.

The verifier is provided with the following information: (1) the snapshot of the entire cryptocurrency platform at timestamp ts, which includes the list of all accounts and their balances. Usually a succinct commitment, e.g., the root of a Merkle tree of such information, is provided. We denote such succinct commitment as $\mathbf{C}_{\mathbb{T}}$, and we use $(\mathbf{S}_i, \mathbf{d}_i) \in \mathbf{C}_{\mathbb{T}}$ to indicate that the value of $\mathbf{S}_i$ in $\mathbf{C}_{\mathbb{T}}$ is $\mathbf{d}_i$, and we use the notation $\mathbf{S}_i \notin \mathbf{C}_{\mathbb{T}}$ for $\mathbf{S}_i$ does not appear in $\mathbf{C}_{\mathbb{T}}$. (2) The information provided by the prover: $\mathbf{C_S}$: the Pedersen vector commitment to $\mathbf{S}$, and $\mathbf{C}_{\mathcal{I}}$: a Pedersen commitment to the initial asset value $\mathcal{I}$.

The goal of the prover is to convince the verifier that: the prover has the knowledge of $\mathbf{S}$, $\mathbf{d}$, $\mathbf{s}$ such that all of the following are true:

1. *Commitments Validity:* knowledge of $\mathbf{S}$ behind $\mathbf{C_S}$ and $\mathcal{I}$ behind $\mathbf{C}_{\mathcal{I}}$.
2. *Ownership*: knowledge of private keys $\mathbf{s}$, which generate accounts, e.g., for BTC: $\forall i \in [1, n] : \mathbf{S}_i = \mathsf{hash}(g^{\mathbf{s}_i})$.
3. *Account Existence:* $\forall i \in [1, n] : (\mathbf{S}_i \notin \mathbf{C}_{\mathbb{T}} \ \wedge \ \mathbf{d}_i = 0) \ \vee \ (\mathbf{S}_i, \mathbf{d}_i) \in \mathbf{C}_{\mathbb{T}}$.
4. *Initial Asset*: $\mathcal{I} = \sum_{i=1}^{N} \mathbf{d}_i$

Incremental Stage: At any time, the prover maintains a Pedersen commitment $\mathbf{C}_{\mathcal{B}}$ to the total balance $\mathcal{B}$ of all accounts. $\mathcal{B}$ can be kept as a secret depending on business needs. At the bootstrap, the prover has established that $\mathbf{C}_{\mathcal{B}} = \mathbf{C}_{\mathcal{I}}$. The prover also publishes $\mathbf{C_S}$.[5]

Then at each cycle (blockchain epoch), the prover executes a $\pi_{\texttt{IZPR}}$ protocol (details in Section 4.3), based on the public information of the cryptocurrency platform, as shown in the following.

1. The cryptocurrency platform publishes the following information: (1) $\mathbf{a}$: the list of all accounts that are involved in any transaction in the current cycle; (2) $\mathbf{\Delta}$: the change of balance for each account in $\mathbf{a}$. Let p be the order of the scalar field $\mathbb{F}$ used. If the balance change is negative, e.g., $-c$, then it is represented as $p - c$ in $\mathbb{F}$. To reduce the verifier complexity, the Pedersen vector commitment to the above are provided and let them be: $\mathbf{C_a}$ and $\mathbf{C_\Delta}$.
2. The prover provides a constant-size proof $\pi_{\texttt{IZPR}}$ which asserts that: given $\mathbf{C_S}$, $\mathbf{C_a}$, $\mathbf{C_\Delta}$, the total balance change for $\mathbf{a} \cap \mathbf{S}$ according to $\mathbf{\Delta}$ is a value v hiding in a Pedersen commitment $\mathbf{C}_v$. The $O(1)$ size proof is made public and can be verified non-interactively by any verifier.
3. The prover then updates $\mathbf{C}_{\mathcal{B}} \leftarrow \mathbf{C}_{\mathcal{B}} + \mathbf{C}_v$, leveraging the homomorphic property of Pedersen commitment.
   Assuming that $\mathbf{C}_{\mathcal{L}}$ commits to the total liability $\mathcal{L}$. $\mathbf{C}_{\mathcal{B}} - \mathbf{C}_{\mathcal{L}}$ commits to $\mathcal{B} - \mathcal{L}$. The zk-range proof to show $\mathcal{B} - \mathcal{L} > 0$ incurs trivial cost. Thus, the organization can prove its solvency without disclosing the total asset and liability value. In certain market situation, this might be desirable. With slight change, one can also prove that the secret total asset value $\mathcal{B}$ is 10% greater than total liability in zero knowledge, providing more assurance to clients of the organization.

[5] It is assumed that $\mathbf{S}$ is fixed. It is possible to periodically expand $\mathbf{S}$, which is briefly discussed in Section 5.1.

In practice, the concern is the scalability of the scheme. We extend the cq protocol, and develop a number of constructions for realizing the IZPR scheme. The rest of this section delves into the technical details.

4.2 Positive-Negative Lookup

We first need an enhanced version of lookup argument. Given the secret query table $\mathbf{t}$, and a public lookup table $\mathbf{T}$, the goal is to assert that a Pedersen commitment $\mathbf{C_o}$ encodes a Boolean vector $\mathbf{o}$ such that $\mathbf{o}_i$ indicates whether $\mathbf{t}_i$ is contained in $\mathbf{T}$. We call it Zero Knowledge Positive-Negative Lookup argument ("zk-PN-Lookup" for short). The basic idea is straight-forward. Let $\mathbf{T}$ be sorted in ascending order. When $\mathbf{t}_i \notin \mathbf{T}$, we simply identify $\mathbf{T}_j$ and $\mathbf{T}_{j+1}$ s.t. $\mathbf{T}_j < \mathbf{t}_i < \mathbf{T}_{j+1}$.

Formally, we assume that all elements in lookup tables are in range $[0, 2^B)$, e.g., for BTC, B is 160. Given a private and sorted table $\mathbf{A} = \{\mathbf{A}_1 < ... < \mathbf{A}_{N-1}\}$ with all elements in range $(0, 2^B)$, define $\mathbf{T} = \{0, \mathbf{A}_1, \ldots, \mathbf{A}_{N-1}\}$, and $\mathbf{T}' = \{\mathbf{A}_1, \ldots, \mathbf{A}_{N-1}, 2^B\}$. We call $(\mathbf{T}, \mathbf{T}')$ the *sorted vector pair* for $\mathbf{A}$ with bound 2^B. Their relation can be proved with a separate zk-proof. Then $\pi_{\texttt{PN\_LOOKUP}}$ is defined below:

$$\pi_{\texttt{PN\_LOOKUP}}(\mathbf{C_T}, \mathbf{C_{T'}}, \mathbf{C_t}, \mathbf{C_o}, B)\{(\mathbf{T}, \mathbf{T}', \mathbf{t}, \mathbf{o}) :$$
$$\mathbf{C_t} = [\hat{t}(s)]_1 \ \wedge \ \mathbf{C_o} = [\hat{o}(s)]_1 \ \wedge \ \mathbf{C_T} = [\hat{T}(s)]_1 \ \wedge \ \mathbf{C_{T'}} = [\hat{T'}(s)]_1 \ \wedge$$
$$\forall i \in [n] : ((\mathbf{o}_i = 1 \ \wedge \ \exists j \ s.t. \ \mathbf{T}_j = \mathbf{t}_i) \ \vee \ (\mathbf{o}_i = 0 \ \wedge \ \exists j \ s.t. \ \mathbf{T}_j < \mathbf{t}_i < \mathbf{T'}_j))\}$$

Figure 1 presents the details of the $\pi_{\texttt{PN\_LOOKUP}}$ protocol. We now explain its design idea and informally reason about its security.

There are two cases to cover: (1) $\mathbf{t}_i \notin \mathbf{T}$, i.e., there exists j s.t. $\mathbf{T}_j < \mathbf{t}_i < \mathbf{T'}_j$; and (2) $\mathbf{t}_i \in \mathbf{T}$, i.e., there exists a k s.t. $\mathbf{T}_k = \mathbf{t}_i$. The prover derives two private vectors $\mathbf{u}$ and $\mathbf{v}$ to assist the proof. Consider the following example.

Example 1. Let $\mathbf{T} = (0, 100, 200, 300)$ and $\mathbf{T'} = (100, 200, 300, 2^B)$. Let $\mathbf{t} = (100, 250)$, and the resulting $\mathbf{o} = (1, 0)$. To reason about the position of elements in $\mathbf{t}$, the prover computes in private: $\mathbf{u} = (100, 200)$ and $\mathbf{v} = (200, 300)$. Note that for each i: $(\mathbf{u}_i, \mathbf{v}_i)$ constitutes a pair of elements in $\mathbf{T}$ and $\mathbf{T'}$ of the same index. This property is proved via a folded lookup argument (steps 1-3 in Figure 1): Given the vector commitments to $\mathbf{u}$ and $\mathbf{v}$, the prover samples a random challenge α and then the prover proves that each element of vector $\mathbf{u} + \alpha\mathbf{v}$ belongs to $\mathbf{T} + \alpha\mathbf{T'}$. Given α is random, the probability that the property being violated is negligible. Then the prover needs to argue that for each i: $\mathbf{u}_i \leq \mathbf{t} \leq \mathbf{v}_i$. This is accomplished using two batched zk-range proofs (step 5 in Figure 1).

Next, in Step 6 of Figure 1, the prover convinces the verifier that $\mathbf{o}$ is a Boolean array, i.e., for $i \in [n]$: $\mathbf{o}_i(\mathbf{o}_i - 1) = 0$, which enforces that $\mathbf{o}_i$ is either 1 or 0. Consider the encoding polynomial $o(X) = \sum_{i=1}^{n} \mathbf{o}_i L_{n,i}(X)$, based on the property of Lagrange base polynomials, we have for $i \in [n]$: $o(\omega_n^i) = \mathbf{o}_i$. Thus the prover needs show the existence of a polynomial $q_o(X)$ s.t. $o(X)(o(X) - 1) =$

1. $\mathbf{P}$ computes $\mathbf{f}$ and $\mathbf{o}$ of size n for $\mathbf{t}$:

$$(\mathbf{f}_i, \mathbf{o}_i) = \begin{cases} (\text{j},1) & \text{if } \exists j \text{ s.t. } \mathbf{T}_j = \mathbf{t}_i \\ (\text{k},0) & \text{if } \exists k \text{ s.t. } \mathbf{T}_k < \mathbf{t}_i < \mathbf{T'}_k \end{cases}$$

   Then $\mathbf{P}$ computes $\mathbf{u} = (\mathbf{T}_{\mathbf{f}_i})_{i=1}^n$ and $\mathbf{v} = (\mathbf{T'}_{\mathbf{f}_i})_{i=1}^n$, and the vector commitments: $\mathbf{C_u}, \mathbf{C_v}, \mathbf{C_o}$. $\mathbf{P} \to \mathbf{V} : (\mathbf{C_u}, \mathbf{C_v}, \mathbf{C_o})$.
2. $\mathbf{V} : \alpha \xleftarrow{\$} \mathbb{F}$. $\mathbf{V} \to \mathbf{P} : \alpha$.
3. $\mathbf{P}$ computes $\pi \leftarrow \mathtt{FoldProve}(\mathsf{pk}, \mathsf{aux}_{\mathbf{T}}, \mathbf{T}, \mathsf{aux}_{\mathbf{T}'}, \mathbf{T}', \mathbf{u}, \mathbf{v}, \alpha)$. $\mathbf{P} \to \mathbf{V} : \pi$.
4. $\mathbf{V}$ aborts if $\mathtt{Verify}(\mathsf{vk}, \mathbf{C_T} + \alpha\mathbf{C_{T'}}, \mathbf{C_u} + \alpha\mathbf{C_v}, \pi)$ returns 0.
5. $\mathbf{P}$ and $\mathbf{V}$ run $\pi_{\mathtt{RANGE}}(\mathbf{C_t} - \mathbf{C_u}, 2^B)$, and $\pi_{\mathtt{RANGE}}(\mathbf{C_v} - \mathbf{C_t}, 2^B)$.
6. $\mathbf{P}$ shows $\mathbf{o}$ is a Boolean array by proving that there exists a $q_o(X)$ s.t.

$$o(X)(o(X) - 1) = q_o(X)z_{\mathbb{H}_n}(X)$$

7. $\mathbf{P}$ proves $o(X)$ is correct. $\mathbf{P}$ computes $\mathbf{d} = \mathbf{t} - \mathbf{u}$ (note $\mathbf{C_t} - \mathbf{C_u}$ commits to $\mathbf{d}$). Then $\mathbf{P}$ shows there are $q(X)$ and $v(X)$ s.t.

$$o(X)d(X) + (1 - o(X))(v(X)d(X) - 1) = q(X)z_{\mathbb{H}_n}(X)$$

   Here $v(X)$ encodes the inverse of each $\mathbf{d}_i$ if it exists.

Fig. 1. PN-Lookup Argument

$q_o(X)z_{\mathbb{H}_n}(X)$. The prover cannot disclose $o(X)$ or $q_o(X)$, instead, their masked polynomials are used, to preserve zero knowledge. Similarly, Step (7) in Figure 1 proves that $o(X)$ encodes the correct information, i.e., when $\mathbf{t}_i \in \mathbf{T}$, $o(\omega_n^i) = 1$, otherwise $o(\omega_n^i) = 0$.

$\pi_{\mathtt{PN\_LOOKUP}}$ immediately leads to an aggregated non-membership proof. Given a preprocessed $\mathbf{C_T}$, to prove that $\mathbf{t} \cap \mathbf{T} = \emptyset$ can be achieved by running $\pi_{\mathtt{PN\_LOOKUP}}$ first and then showing that $\mathbf{C_o}$ is a Pedersen vector commitment to $(0)_{i=1}^n$. The prover complexity of $\pi_{\mathtt{PN\_LOOKUP}}$ $O(n\mathsf{log}(n))$ and its verifier complexity is $O(1)$.

Using Fiat-Shamir transform, $\pi_{\mathtt{PN\_LOOKUP}}$ can be converted into a non-interactive proof. We then have the following:

Lemma 1. *Under the Q-DLOG assumption in the AGM and ROM, $\pi_{\mathtt{PN\_LOOKUP}}$ is perfectly complete, computational knowledge sound, and zero knowledge.*

4.3 $\pi_{\mathtt{IZPR}}$ Protocol

We now consider $\pi_{\mathtt{IZPR}}$, the asset proof for pseudo-anonymous cryptocurrencies. It is *non-intrusive* in the sense that there is no change needed on the blockchain, or any participants who have no need for asserting assets. The protocol assumes that at the bootstrap step, the organization has provided the ownership proof for each account in $\mathbf{S}$. Let $\mathbf{T}$ and $\mathbf{T}'$ be the corresponding sorted vector pair for $\mathbf{S}$. To verifier, only their commitments ($\mathbf{C_T}$ and $\mathbf{C_{T'}}$) are visible.

For each transaction cycle, the blockchain makes two vectors public: $\mathbf{a}$: the list of accounts, and $\mathbf{\Delta}$, the corresponding balance change of each account in $\mathbf{a}$. We use $[0, 2^B)$ as positive and $[|\mathbb{F}|-2^B, |\mathbb{F}|)$ as negative values. Let $n = |\mathbf{a}|$, [6] and $N = |\mathbf{T}|$. Typically, the value of n is not large and determined by the throughput of the platform. For instance, BTC, ETH, Visa Network and NYSE operate at 7, 30, 1700, and 24000 TPS, respectively. In this work, we aim at accomplishing 1000 TPS, and set $n = 2048$ (assuming each transaction causing updates on two accounts) and $N = 8$ million. Define $\mathbf{C_a} = [a(s)]_1$ (without blinding factor) and similarly is $\mathbf{C_\Delta}$ defined. They are used only as succinct representation of $\mathbf{a}$ and $\mathbf{\Delta}$ and need to be publicly computable (thus no hiding property is needed). On the other hand, $\mathbf{C_T}$ and $\mathbf{C_{T'}}$ are hiding.

Intuitively, $\pi_{\texttt{IZPR}}$ states that $\mathbf{C}_v$ commits to a value v that is the sum of balance changes for all accounts that appear in the intersection of $\mathbf{S}$ and $\mathbf{a}$ as sets. It is formally defined below.

$$\begin{aligned}
&\pi_{\texttt{IZPR}}(B, \mathbf{C_T}, \mathbf{C_{T'}}, \mathbf{C_a}, \mathbf{C_\Delta}, \mathbf{C}_v)\{(\mathbf{T}, \mathbf{T'}, \mathbf{a}, \mathbf{\Delta}, v, r) : \\
&\mathbf{C}_v = [v]_1 + r[h]_1 \ \wedge \ v = \textstyle\sum_{\mathbf{a}_j \in \mathbf{S} \cap \mathbf{a}} \Delta_j \\
&\mathbf{C_a} = [a(s)]_1 \ \wedge \ \mathbf{C_\Delta} = [\Delta(s)]_1 \ \wedge \ \mathbf{C_T} = [\hat{T}(s)]_1 \ \wedge \ \mathbf{C_{T'}} = [\hat{T'}(s)]_1\}
\end{aligned}$$

The protocol is built upon $\pi_{\texttt{PN\_LOOKUP}}$, and the details are presented in Figure 2. In the following we present its design idea and an informal analysis of its security.

(Steps 1-2): Given $\mathbf{a}$, the prover first computes a Boolean vector $\mathbf{o}$ where each $\mathbf{o}_i$ indicates if $\mathbf{a}_i$ appears in $\mathbf{T}$. Note that the validity of $\mathbf{o}$ is established in step 7. Then the prover defines an accumulator vector $\mathbf{u}$ s.t. $\mathbf{u}_1 = 0$ and its last element is the sum of all account balance changes, i.e., the value of v. Therefore, for $i \in [1, n-1]$, we have: $\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{o}_i \Delta_i$, i.e., the Boolean $\mathbf{o}_i$ decides whether to include Δ_i in the sum. Let $u(X)$ be the encoding polynomial of vector $\mathbf{u}$, we then have the following listed in Step 2:

$$u(X\omega_n) - u(X) - o(X)\Delta(X) = q(X)z_{\mathbb{H}_n}(X)/(X - \omega_n^n) \tag{1}$$

Here ω_n is the n'th root of unity, and the $u(X\omega_n)$ and $u(X)$ intuitively model the relation between $\mathbf{u}_{i+1}$ and $\mathbf{u}_i$. The last item $z_{\mathbb{H}_n}(X)/(X - \omega_n^n)$ expresses the range condition $[1, n-1]$, i.e., given that $z_{\mathbb{H}_n}(X) = \prod_{i=1}^{n}(X - \omega^i)$, the formula excludes the case $i = n$.

Note that for each vector, e.g., $\mathbf{u}$, the commitment to its *masked* polynomial $\hat{u}(X)$ is sent to the verifier, for preserving zero knowledge.

(Steps 3-5) The verifier samples a random challenge t, and the prover uses KZG polynomial commitment to assert the evaluation of each related polynomial at point t (step 4), and then the verifier use these values to check Equation 1 (step 5). Given that for each $i \in [n] : u(\omega_n^i) = \hat{u}(\omega_n^i)$, and by Schwartz-Zippel, the probability of failing soundness is negligible. Also note that since each polynomial is involved in KZG evaluation proof for up to 2 times, and their

[6] In the implementation $|\mathbf{a}| = |\mathbf{u}| - 1$

1. $\mathbf{P}$ receives input: $\mathbf{S}$, $\mathbf{a}$, and $\mathbf{\Delta}$.
   $\mathbf{V}$ receives input: B, $\mathbf{C_a}$, $\mathbf{C_\Delta}$, $\mathbf{C_T}$, $\mathbf{C_{T'}}$, and $\mathbf{C}_v$.
2. $\mathbf{P}$ computes array $\mathbf{o}_i$ as shown in Figure 1, and v the accumulation of balance changes of $\mathbf{a} \cap \mathbf{S}$. Define polynomial $u(X)$ s.t.

$$u(\omega_n^1) = 0$$
$$u(\omega_n^n) = v$$
$$u(X\omega_n) - u(X) - o(X)\Delta(X) = q(X)z_{\mathbb{H}_n}(X)/(X - \omega_n^n)$$

   $\mathbf{P}$ samples random openings and computes masked polynomials $\hat{u}(X)$, and $\hat{o}(X)$. In particular, make $\hat{u}(X)$ 2-leaky by:
   $\hat{u}(X) = u(X) + (r_{u_1}X + r_{u_0})z_{\mathbb{H}_n}(X)$. Define $\hat{\Delta}(X) = \Delta(X)$. Define $\hat{q}(X) = \frac{\hat{u}(X\omega_n) - \hat{u}(X) - \hat{o}(X)\hat{\Delta}(X)}{z_{\mathbb{H}_n}(X)/(X-\omega_n^n)}$. Compute $\mathbf{C_u} = [\hat{u}(s)]_1$, $\mathbf{C_o} = [\hat{o}(s)]_1$, and $\mathbf{C_q} = [\hat{q}(s)]_1$. $\mathbf{P} \to \mathbf{V} : (\mathbf{C_u}, \mathbf{C_q}, \mathbf{C_o})$.
3. $\mathbf{V}$ samples $t \xleftarrow{\$} \mathbb{F}$. $\mathbf{V} \to \mathbf{P} : t$.
4. $\mathbf{P}$ computes $t_{u2} = \hat{u}(t\omega_n)$, $t_u = \hat{u}(t)$, $t_o = \hat{o}(t)$, $t_\Delta = \hat{\Delta}(t)$, and $t_q = \hat{q}(t)$. $\mathbf{P}$ and $\mathbf{V}$ run standard KZG opening proof for them, where the last 4 can be batched. $\mathbf{P}$ and $\mathbf{V}$ run another KZG opening for $\hat{u}(\omega_n^1) = 0$.
5. $\mathbf{V}$ checks $t_{u2} - t_u - t_o t_\Delta = t_q(t^n - 1)/(t - \omega_n^n)$.
6. $\mathbf{P}$ and $\mathbf{V}$ engage in a univariate zk-VPD [38] opening proof to prove that $\mathbf{C}_v = [v]_1 + r_v[h]_1$ for some r_v and $u(\omega_n^n) = v$, but without disclosing the value of v.
7. $\mathbf{P}$ and $\mathbf{V}$ run $\pi_{\texttt{PN\_LOOKUP}}(\mathbf{C_T}, \mathbf{C_{T'}}, \mathbf{C_a}, \mathbf{C_o}, B)$.

Fig. 2. $\pi_{\texttt{IZPR}}$ Protocol

masked polynomials have their degrees raised correspondingly, which preserves zero knowledge. The same technique is used in [23, 8].

(Step 6) Finally, the prover needs to convince the verifier that $\mathbf{u}_n = v$, without disclosing the value of v. KZG cannot be used here, because it discloses the evaluation itself. Instead, we employ a univariate instantiation of the zk-VPD [38] scheme, which given the KZG commitment to a polynomial p, a point t and proves that a Pedersen commitment $\mathbf{C}$ hides the value of $p(t)$. This finally concludes the proof that $\mathbf{C}_v$ is a Pedersen commitment to the sum of balance changes of $\mathbf{a} \cap \mathbf{S}$.

Theorem 2. *Under the Q-DLOG assumption in AGM and ROM, $\pi_{\texttt{IZPR}}$ is perfectly complete, computational knowledge sound, and zero knowledge. Its prover complexity is $O(|\mathbf{a}|log(|\mathbf{a}|))$. Its verifier complexity and proof size are both $O(1)$.*

5 Implementation and Evaluation

We implemented and evaluated $\pi_{\texttt{IZPR}}$ over BLS12-381 (providing 128-bit security). Our implementation consists of 4600 lines of Rust, and is based on the

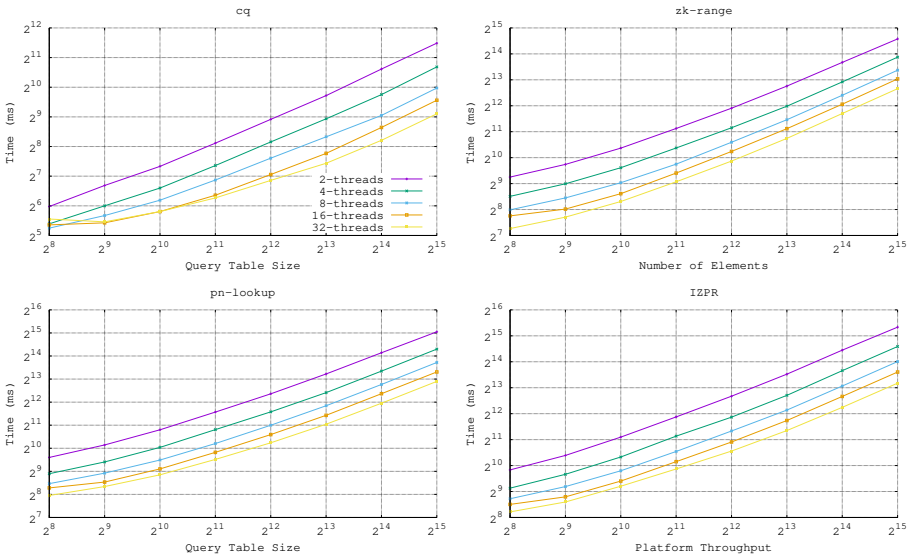

Fig. 3. Scalability of $\pi_{\texttt{IZPR}}$

`arkworks` library [2] (which provides parallelism via multi-threaded Rayon). Figure 3 shows the scalability of the zk-`cq` protocol, the batched zk-range proof based on `cq`, the zk-pn-lookup protocol and the $\pi_{\texttt{IZPR}}$ protocol. The evaluation data is collected over a GCP C2D instance with 56 cores, and the best performance is achieved at $2^{\lfloor \log(56) \rfloor} = 32$ threads. For the zk-range proof, we assert range $[0, 2^{160})$ and split each field element into 8 chunks of 20-bit numbers. Therefore given n elements in a batched zk-range proof, it results in a lookup argument for query table size of $16n$. The concrete prover cost of zk-range proof for processing $32k$ 160-bit field elements needs 24.5 seconds with 32 threads, equivalent to 6.7bit/(thread,ms), this is faster than the bullet proof [5] (3.2bit/(thread,ms)), and our proof size and verifier cost are both $O(1)$.[7]

The lookup table has 8 million entries (simulating an exchange owning 10% of existing BTC addresses). The one-time preprocessing for $\pi_{\texttt{IZPR}}$ (mainly for the `cq`) takes 4 hours.[8] The prover cost for 2048 account updates is 0.935 second (32-threads), and the verification cost is 45 ms (1-thread). Assuming that each transaction results in change of 2 accounts, this is equivalent to supporting TPS of 1024, which is half of the speed of Visa Network and 146 times of BTC. The proof size is 3.4 kb.

[7] For range proof, if `cq` is replaced by plookup [24] or logUp [26], or more recent Lasso [32], additional gain of performance can be achieved.

[8] The cost does not include the ownership proof for each BTC address.

5.1 Discussion on Bootstrap Cost

There are many ways to realize the bootstrap process, which is out of the scope of this paper, as the bootstrap is orthogonal to the incremental proof protocol. We briefly describe a way to do the bootstrap using a ZK-SNARK[9], and then state the estimated cost.

The basic idea of the SNARK is to hide the organization's account set $\mathbf{S}$ inside a bigger, public set of accounts called the anonymity set, $\mathbf{S}'$. Ownership of each account in $\mathbf{S}$ is proved via a signature, so the organization is expected to be able to produce a set of signatures $\mathbf{E}$ such that the mapping between $\mathbf{S}$ and $\mathbf{E}$ is a bijection. After verifying the signatures the SNARK calculates the sum of balances of $\mathbf{S}$, and produces a commitment to this value. Finally, the SNARK produces the commitment $\mathbf{C_S}$, which is how it is linked to the IZPR protocol.

We have the following input signals to the SNARK:

1. pub $\mathbf{S}'$ (contains addresses & asset balances)
2. pvt $\mathbf{E}$

The SNARK does these computations:

1. Verify each signature in $\mathbf{E}$ and check that it corresponds to exactly one account in $\mathbf{S}'$
2. Sum up the balances for each account related to a signature in $\mathbf{E}$, V
3. Create a commitment to V and output this as a public signal, $\mathbf{C}_V$
4. Create a commitment to $\mathbf{S}$ and output this as a public signal, $\mathbf{C_S}$

The verifier needs to check that each address and balance in the anonymity set are the same as found on the blockchain. They then need to check that the public signal $\mathbf{C_S}$ is the same as the one used in the IZPR protocol. To update the asset sum, the verifier can add the public signal $\mathbf{C}_V$ from the SNARK to the commitment $\mathbf{C}_v$ from IZPR.

There are some hurdles to overcome with the above SNARK. The first issue is the size of the anonymity set: since this is a public signal, it must form part of the proof, so the size of the proof is $O(|\mathbf{S}'|)$. If we would like to keep the proofs succinct then we can use a Merkle tree for the anonymity set. Instead of having $\mathbf{S}'$ as a public signal, we have only the root as a public signal, and then have Merkle proofs as private signals, which are verified inside the SNARK. Using a Merkle tree allows us to make our anonymity set very large without taking up too many constraints. Even if our anonymity set was 10^{15} it would only need $2\times$ more constraints than if the set had 100M elements.

Another problem is the cost of verifying ECDSA signatures (the signature scheme for Bitcoin) inside a SNARK. The size of the secp256k1 base field is greater than the native field sizes used for SNARKS, which means the values of coordinates for points on secp256k1 need to be stored using at least 2 SNARK

[9] A full description of the design can be found here: https://hackmd.io/@JI2FtqawSzO-olUw-r48DQ/rJXtAeyLT

field values. Basic operations like addition and multiplication thus become expensive. If we stick to the Circom ecosystem and use Groth16 as our proving system then the number of R1CS constraints to verify x ECDSA signatures is about $0.45x + 1$ (in millions of constraints)[10]. Given the current constraint limit of 256M (limited by largest available PTAU file) this means we are limited to about 600 signatures. We can get around this issue by using recursive SNARKs.

We can split up $\mathbf{S}$ into batches and input each batch into a different SNARK, which can be computed in parallel. We need a final SNARK to verify the proofs of the batches of SNARKS (this is the recursive step) and sum up the intermediate sums produced by each batch. With the current recursive implementation [11] the max number of signatures supported is about 6000, although this number can be increased using snarkpack.

Regarding computational cost, if we assume $|\mathbf{S}| = 6000$ then the computational time is dominated by proving key generation. This takes around 155 hours on an EC2 m7g.8xlarge. The other expensive operation is compilation, which takes around 4 hours. We need 3 m7g.8xlarge's to take advantage of the recursive and parallel nature of the design. The total USD cost is around $875. This amounts to about 0.14 US dollar for the initial asset proof per BTC address.

6 Conclusion

Based on the recent progress of lookup arguments, we develop a zero knowledge protocol for asserting the asset of an organization incrementally. The prover cost does not depend on the number of the accounts owned by an organization (which can be arbitrarily large), but on the *throughput* of the platform. With modest computing resources, the protocol can support regular business transaction systems at 2^{10} TPS. We foresee that with some light implementation efforts, e.g., leveraging MPI, the protocol can be further scaled.

Acknowledgment: We would like to thank Dr. Ariel Gabizon for the recommendation of appropriate lookup arguments for this work. We appreciate the constructive suggestions from the reviewers for improving this manuscript. This research is supported by an unrestricted research gift from Chan Zuckerberg Initiative, and a Hofstra SEAS FRDG grant. Trevor Conley is supported by the Hofstra SEAS AsPire'23 Summer Research Program, Fall'23 Stuart and Nancy Rabinowitz Honors College Research Assistant Program, and various student travel support resources from Hofstra University.

References

1. S. Agrawal, C. Ganesh, and P. Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *CRYPTO*, pages 643–673, 2018.
2. arkworks contributors. `arkworks` zksnark ecosystem, 2022.

[10] https://github.com/puma314/batch-ecdsa
[11] https://github.com/silversixpence-crypto/zk-proof-of-assets

3. F. Baldimtsi, P. Chatzigiannis, S. Gordon, P. Le, and D. McVicker. gOTzilla: Efficient disjunctive zero-knowledge proofs from MPC in the head, with application to proofs of asset in cryptocurrencies. *PoPETs*, 4:229–249, 2022.
4. Binance, 2023.
5. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *SSP*, pages 315–334, 2018.
6. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO*, LNCS 1296, pages 410–424, 1997.
7. M. Campanelli, A. Faonio, D. Fiore, T. Li, and H. Lipmaa. Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. `https://hal.science/hal-04234948/document`, 2023.
8. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodriguez. Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions. In *ASIACRYPT*, pages 3–33, 2021.
9. N. Carter. Nic's PoR wall of fame. `https://niccarter.info/proof-of-reserves/`, 2023.
10. K. Chalkias, K. Lewi, P. Mohassel, and V. Nikolaenko. Distributed auditing proofs of liabilities. IACR Cryptol. ePrint Arch. `https://eprint.iacr.org/2020/468`, 2020.
11. P. Chatzigiannis, F. Baldimtsi, and K. Chalkias. SoK: auditability and accountability in distributed payment systems. In *ACNS*, pages 311–337, 2021.
12. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In *EUROCRYPT*, pages 738–768, 2020.
13. A. Choudhuri, S. Garg, A. Goel, S. Sekar, and R. Sinha. Sublonk: Sublinear prover plonk. `https://eprint.iacr.org/2023/902`, 2023.
14. CoinDesk. Binance's bitcoin reserves are overcollateralized, new report says, 2023.
15. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *CCS*, pages 720–731, 2015.
16. C. Decker, J. Guthrie, J. Seidel, and R. Wattenhofer. Making Bitcoin exchanges transparent. In *ESORICS*, pages 561–576, 2015.
17. A. Dutta, S. Bagad, and S. Vijayakumaran. MProve+: privacy enhancing proof of reserves protocol for monero. *IEEE Transactions on Information Forensics and Security*, 16:3900–3915, 2021.
18. A. Dutta, A. Jana, and S. Vijayakumaran. Nummatus: a privacy preserving proof of reserves protocol for Quisquis. In *INDOCRYPT*, pages 195–215, 2019.
19. A. Dutta and S. Vijayakumaran. MProve: a proof of reserves protocol for monero exchanges. In *Euros&p Workshops*, pages 330–339, 2019.
20. A. Dutta and S. Vijayakumaran. Revelio: a MimbleWimble proof of reserves protocol. In *CVCBT*, pages 7–11, 2021.
21. L. Eagen, D. Fiore, and A. Gabizon. cq: Cached quotients for fast lookups. IACR Cryptol. ePrint Arch., 2022.
22. A. Gabizon and D. Khovratovich. Flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. IACR Cryptol. ePrint Arch., 2022.
23. A. Gabizon, Z. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. `https://eprint.iacr.org/2019/953`, 2019.
24. A. Gabizon and Z. J. Williamson. Plookup: A simplified polynomial protocol for lookup tables. IACR Cryptol. ePrint Arch., 2020.