Xiezhi: Toward Succinct Proofs of Solvency

Youwei Deng[1] and Jeremy Clark[1[0000−0002−3533−5965]]

Concordia University, Montreal, Canada
barry.deng1995@gmail.com,j.clark@concordia.ca

Abstract. A proof of solvency (or proof of reserves) is a zero-knowledge proof conducted by centralized cryptocurrency exchange to offer evidence that the exchange owns enough cryptocurrency to settle each of its users' balances. The proof seeks to reveal nothing about the finances of the exchange or its users, only the fact that it is solvent. The literature has already started to explore how to make proof size and verifier time independent of the number of (i) users on the exchange, and (ii) addresses used by the exchange. We argue there are a few areas of improvement. First, we propose and implement a full end-to-end argument that is fast for the exchange to prove (minutes), small in size (KBs), and fast to verify (seconds). Second, we deal with the natural conflict between Bitcoin and Ethereum's cryptographic setting (`secp256k1`) and more ideal settings for succinctness (*e.g.,* pairing-based cryptography) with a novel mapping approach. Finally, we discuss how to adapt the protocol to the concrete parameters of `bls12-381` (which is relevant because the bit-decomposition of all user balances will exceed the largest root of unity of the curve for even moderately-sized exchanges).

Keywords: zero knowledge proofs · polynomial interactive oracle proofs · proof of solvency · proof of reserves

1 Introduction

When the Bitcoin exchange Mt. Gox was declared bankrupt in 2014, a curious fact was reported in the *New York Times*—the missing 744K BTC "had gone unnoticed for years." Academics showed that proofs of solvency can be done by exchanges in strict zero-knowledge [9], and generated a stream of research papers that continues to improve efficiency [4,12,17,13,25,8] and examine the correctness of deployed proofs [5]. We believe we should continue refining these proofs toward practical implementation as they provide meaningful barriers (or friction) to fraud and incompetence by exchanges.

A proof of solvency (or proof of reserves) is a zero-knowledge proof conducted by centralized cryptocurrency exchange (or more generally, any custodian of cryptocurrencies) to offer evidence that the exchange owns enough cryptocurrency to settle each of its user's balances. The zero-knowledge component protects the exchange's proprietary information such as: number of users, balances of individual users, total balance of all users, which cryptocurrency addresses belong to the exchange, and total amount of cryptocurrency owned by the exchange. The proof itself is broken into sub-components: (π_{keys}) a proof of knowledge of private signing keys associated with public cryptocurrency addresses (hidden in a freely-composable anonymity set of addresses not belonging to the exchange); (π_{assets}) a summation of these assets into the total assets; (π_{user}) an individualized proof given to each user asserting their balance as used in the overall proof; ($\pi_{\mathsf{liabilities}}$) a summation of these individual liabilities into the total liabilities; and (π_{solvency}) a demonstration that the subtraction of total liabilities from the total assets is at least 0.

Our contributions can be summarized as:

– Xiezhi:[1] A novel and mostly succinct (constant size and verifier time) proof of solvency protocol that covers every step of the "proof,[2]" where each sub-component of the proof works with each other sub-component.

[1] Folklore creature revered in ancient Chinese culture for its ability to distinguish truth from deceit.

[2] We abuse terminology and generally do not distinguish between 'proofs' and 'arguments,' using the term 'proofs' for both. Proofs provide soundness against unbounded malicious provers, while arguments provide zero knowledge against unbounded malicious verifiers. Xiezhi is a hybrid.

– A novel technique for mapping knowledge of private keys of common blockchains, such as Bitcoin and Ethereum, from their group (secp256k1) into a pairing-friendly group (bls12-381) used for succinct arguments.
– Practical adjustments to the protocol to account for concrete parameters, such as the maximum root of unity in bls12-381.
– Proof of concept implementation of Xiezhi with performance experimentation.

Xiezhi has certain limitations: it relies on a universal (shareable with other zk-SNARK systems) trusted setup, secure with one honest participant in a decentralized computation of it [22]; it assumes the public key (as opposed to only its hash) associated with every address in an anonymity set of keys is known (achievable by spending performing at least one transaction); it is limited to funds controlled by a single public key. That said, it works for any token controlled by public keys with known balances on any such chain or layer 2. Proofs of solvency, generally, also have common limitations shared by Xiezhi: users need to check their balances in the proof, proofs complicate the cover-up of hacks and exit scams but do not prevent them, asset input to the proof can belong to colluders, and TEEs can help share proving ability without the keys themselves (but Xiezhi's π_{keys} can be adapted for complete knowledge [19]).

2 Preliminaries

Terminology. A balance sheet consists of liabilities (value owed to others) and assets (value owned). When the total asset value is the same or more than the total liabilities, the firm is called solvent. The amount by which the assets exceed the liabilities is called capital or equity (depending on context). Some literature prefers the term 'proof of reserves' to 'proof of solvency.'

Related Work. Table 1 reviews research on proofs of solvency, noting that the vast majority of work on proofs of solvency have not attempted an end-to-end proof, focusing instead on just the liabilities or just the assets. Why? We hypothesize that the biggest impediment is that Bitcoin and Ethereum assets are controlled by secp256k1 private keys. Outside of Bulletproofs (based on inner-product arguments that do not require bilinear pairings and thus, can be implemented in secp256k1), most other approaches to succinctness require a specific cryptographic setting that is not secp256k1 (*i.e.,* RSA for accumulators, pairing-based cryptography for zk-SNARKs, and lattices for zk-STARKs). If one only considers liabilities, then this problem does not have to be dealt with.

Circuit-based solutions are feasible but expensive for the prover—the authors of IZPR report about 500K constraints needed per key and proving times in the order of days for an anonymity set of 6000 keys [8]. By contrast, Xiezhi is a few minutes of work for the prover for 6000 keys. As this part is not succinct (it is based on Σ-protocols), the trade-off is that the verifier has to do a few minutes of work as well. In both cases, IZPR and Xiezhi, this step does not need to be repeated often, only when the exchange wants to introduce new keys holding its assets. It is also important to recognize IZPR can let the exchange add keys it has not used yet to π_{keys}, further reducing how often this proof needs to be redone. This is a desirable property we are not able to easily achieve in Xiezhi (in short, it is due to our use of selector polynomials instead of lookup arguments but future work could explore blending the best properties of Xiezhi and IZPR).

3 Cryptographic Building Blocks

We refer the reader to Appendix A for the following cryptographic primitives: discrete logarithm assumption (Section A.1), Pedersen commitments (Section A.2), zero-knowledge proof and its related properties

[3] V. Buterin, "Having a safe CEX: proof of solvency and beyond," vitalik.ca, 2022
[4] https://summa.gitbook.io/summa
[5] https://www.proven.tools/
[6] https://minaprotocol.com/

Σ-Protocols	The first proofs of solvency are based on Σ-Protocols which work on standard elliptic curves like `secp256k1` but are not succinct (linear proof space, linear verifier time, heavy constants).	Provisions [9]
Inner-product arguments	Protocols like bulletproofs work on standard elliptic curves like `secp256k1` and can reduce some sub-routines (e.g., range arguments) to constant space and logarithmic verifier time.	Bulletproofs [4]
Liabilities only	As it is the asset-side of solvency that ties the protocol to standard elliptic curves like `secp256k1`, proving only the liability side can be done in any cryptographic setting.	ZeroLedge [12], DAPOL+ [17], SSVT-based [13], Notus [25], SafeCex[3]
Publish assets	A trivial proof of assets is one that is not zero-knowledge. An exchange could reveal all its addresses and prove ownership by signing a proof-specific message from each address.	Summa[4]
Circuit-level	A general zk-snark can implement any arithmetic circuit, including `secp256k1` operations, which offers a proof of constant size and constant verifier time.	IZPR [8], Proven.tools[5]
Custom blockchain	If new blockchains are deployed, they could use digital signatures over pairing-friendly curves.	Mina[6]
Mapping between groups	If `secp256k1` values can be mapped to a pairing-friendly group, Poly-IOP arguments can potentially reduce the rest of the proof to constant size and constant verifier time.	COPZ [6], Xiezhi

Table 1: How to deal with the fact that Bitcoin and Ethereum use `secp256k1` digital signatures when trying to make a succinct proof of solvency.

(Section A.3), Σ-protocols (conjunction and disjunction) (Section A.4,A.5), polynomial commitment scheme (Section A.6), roots of unity (Section A.7), and polynomial interactive oracle proof (Poly-IOP) model (Section A.8).

Notation. Our protocols work on a finite field of prime order between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. Let g_s and h_s denote two independent generators in the `secp256k1` group $\mathbb{G}_s$; and we use g_b and h_b for `bls12-381`. We denote by $e([x]_1, [x]_2)$ a non-degenerate bilinear pairing, with input groups specified $[x]_i := g_i^x \in \mathbb{G}_i$ for $i = 1, 2$ (when omitted, values are assumed to be in the first group). For vectors, we use an overhead bar to denote a vector and brackets to denote the elements in this vector, e.g., $\overline{v} = \langle v_1, v_2, \ldots \rangle$. Let $\mathbf{C}(x)$ denote a Pedersen commitment to a value x with a hiding factor. Particularly, we use $\mathbf{C}_{\texttt{secp}}$ and $\mathbf{C}_{\texttt{bls}}$ to denote a commitment in `secp256k1` and `bls12-381`, respectively. To help distinguish polynomials and constants, we denote by $\mathcal{C}_f$ a polynomial commitment to f in `bls12-381`. We use ω to denote roots of unity.

3.1 Range Proof Variant

A zero-knowledge range proof (ZKRP) enables $\mathcal{P}$ to convince $\mathcal{V}$ a value x is in a specified range, e.g., $[0, 2^k)$, without revealing x. ZKRPs have three typical approaches: square decomposition, n-ary decomposition, and hash chains [7]. We use the succinct decomposition method from Boneh *et al.* [3](Section A.9) however we adapt it to prove multiple values are in the specified range (Section B.3). We will introduce and motivate our variant in Section 5.1.

Lemma 1. *The range proof from Boneh et al. [3] is complete and has knowledge soundness in the algebraic group model.*

Proof. See Section A.9.

3.2 KZG Evaluation Variants

A polynomial commitment scheme (PCS) enables a succinct demonstration that a commitment $\mathcal{C}$ to polynomial f evaluated at point a is b. We use the KZG PCS [18] in its zero-knowledge form, which requires (i) randomized commitments and (ii) committing to additional random points, so checks of polynomial relationships at random points do not leak information about the polynomial. We denote KZG with such a zero-knowledge extension as KZG_{zk}. Specifically, we claim two algorithms, $\pi_\zeta \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_1, f_2, \ldots; \zeta)$ to open multiple polynomials $f_1, f_2, \ldots$ at the point ζ, and $\{1/0\} \leftarrow \mathsf{KZG}_{zk}.\mathsf{Verify}(\mathcal{C}_{f_1}, \mathcal{C}_{f_2}, \ldots; \pi_\zeta; \zeta)$ to verify the proof at ζ. In Section A.10, we specify KZG_{zk}.

We also show it is possible to 'stop early' in KZG_{cm} and end up with a Pedersen commitment to $f(a)$, instead of revealing $f(a)$ in plaintext. We call this 'open to a commitment' variant with two algorithms like we did in KZG_{zk}, $\mathbf{C}_{\mathtt{bls}}(f(\zeta)) \leftarrow \mathsf{KZG}_{cm}.\mathsf{Prove}(f; f(\zeta); \zeta)$ to open the committed evaluation of $f(\zeta)$ in `bls12-381`, and $\{1/0\} \leftarrow \mathsf{KZG}_{cm}.\mathsf{Verify}(\mathcal{C}_f; \mathbf{C}_{\mathtt{bls}}(f(\zeta)); \zeta)$ to verify the committed value is correct. In Section B.1, we specify KZG_{cm} and prove it is complete, sound, and HVZK.

4 Proof of Assets (PoA)

We are now ready to explain Xiezhi. We will begin on the asset side, taking the example of ETH. The PoA is broken into two steps: π_{keys} and π_{assets}. In short, for π_{keys}, the exchange publishes a public vector of Ethereum public keys, consisting of its own public keys hidden amongst a larger anonymity set of keys belonging to others. Next, it will output a hiding commitment to a binary 'selector' vector (in `bls12-381`) and prove it records a 0 if the exchange is not claiming to know the secret key (in `secp256k1`) of the public key in the same position in the vector, and a 1 if the exchange can prove knowledge of the secret key. For π_{assets}, the exchange publishes a public vector of ETH balances matched to the vector of public keys. It outputs a hiding commitment to the sum of its assets, as the first element of a vector, and proves it is the sum of the subset of balances in the balance vector marked by the selector vector.

4.1 The π_{keys} proof

Before presenting our π_{keys} proof, we quickly discuss a few approaches that helped us develop it. A highly relevant Σ-protocol from the literature, COPZ, proves that two commitments in two different groups (*e.g.,* `secp256k1` and `bls12-381`) commit to the same value [6]. The paper cites proof of assets as a use-case but does not work out a protocol. COPZ allows an exchange to 'map' private keys from `secp256k1` to `bls12-381`. Two places this mapping could occur would be at the very start or the very end of the assets proof. At the end, it might look like this: an existing proof of assets protocol (*e.g.,* Provisions [9]) could be run to create a commitment to the total assets in `secp256k1`, then COPZ can be used to prove the same commitment in `bls12-381`, and finally this can be 'glued' to a succinct proof of liabilities in `bls12-381`. However, this does not leverage the fact that `bls12-381` might help make the assets proof succinct.

Alternatively, the exchange can map all their keys at the start. There is a roadblock: the exchange can only map keys they know the secret key for and the exchange cannot reveal which keys they know and which they do not. Assume there is a protocol that would allow the exchange to output a sparse vector of `bls12-381` private key values (sorted by index of known ETH public keys) containing the key value when they know it, and recording a 0 if they do not. We designed such a protocol only to realize that the key values in `bls12-381` are actually never used, we only use the fact that knowledge of them is proven (which is covered by the ability to produce the value in `bls12-381`) and the fact that unclaimed keys are zeroed-out.

This leads to our key observation: we do not need to map *values* from `secp256k1` to `bls12-381`, we just have to map the *success or failure* of a ZKP in `secp256k1` to `bls12-381`. This can be accomplished by composing (conjunction and disjunction of) Σ-protocols. The prover (exchange) will choose a set of Ethereum public keys as its anonymity set of size κ (containing its actual keys) $\{\mathsf{pk}_1, \mathsf{pk}_2, \ldots, \mathsf{pk}_\kappa\}$ and publish them

1. $\mathcal{P}$ publishes an anonymity set of public keys: $\langle \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3, \ldots, \mathsf{pk}_n \rangle$.
2. $\mathcal{P}$ constructs a selector vector: $\overline{s} = \langle s_1, s_2, s_3, \ldots, s_n \rangle$ where $s_i = 0$ unless $\mathcal{P}$ can prove knowledge of sk_i given $\mathsf{pk}_i = g_s^{\mathsf{sk}_i}$, then $s_i = 1$.
3. $\mathcal{P}$ interpolates a polynomial $f_{\mathsf{sel}}(X)$ from $\overline{s}$ and commits to f_{sel}.
4. For each i, $\mathcal{P}$ computes $\mathbf{C}_{\mathtt{bls}}(s_i) \leftarrow \mathsf{KZG}_{cm}.\mathsf{Prove}(f_{\mathsf{sel}}; s_i; \omega^i)$, and $\mathcal{V}$ aborts if $\mathsf{KZG}_{cm}.\mathsf{Verify}(\mathcal{C}_{f_{\mathsf{sel}}}; \mathbf{C}_{\mathtt{bls}}(s_i); \omega^i)$ returns 0;
5. For each $\mathbf{C}_{\mathtt{bls}}(s_i)$, $\mathcal{P}$ and $\mathcal{V}$ run Protocol 6 to prove ZKPoK$\{(\mathsf{sk}_i, s_i) : [\mathsf{pk}_i = g_s^{\mathsf{sk}_i} \wedge \mathbf{C}_{\mathtt{bls}}(s_i) = \mathbf{C}_{\mathtt{bls}}(1)] \vee \mathbf{C}_{\mathtt{bls}}(s_i) = \mathbf{C}_{\mathtt{bls}}(0)\}$.

Protocol 1: The π_{keys} proof demonstrates that $f_{\mathsf{sel}}(X)$ encodes a binary selector vector of the public keys for which the exchange can prove knowledge of the corresponding secret key.

in an ordered (indexed) way. It will also create a binary 'selector' array A_{keys} with a 1 in the same index of every key it is claiming to know and a 0 in the index of the keys it does not know (or does not want to claim for whatever reason). This vector is interpolated into the evaluation points of a polynomial $f_{\mathsf{keys}}(X)$ and committed to $\mathbf{C}_{\mathtt{bls}}(f_{\mathsf{keys}}(X))$ using the KZG polynomial commitment scheme [18]. For each index i, the prover shows the evaluation of $f_{\mathsf{keys}}(X_i)$ but instead of providing the evaluation value ($A_{\mathsf{keys},i}$) in plaintext, it provides a Pedersen commitment to it $\mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i})$ (a mild modification of the KZG showing protocol detailed in Section B.1). It then shows the value is correct with the following Σ-protocol (for details see Protocol 6 in Section B.2):

$$\text{ZKPoK}\{(\mathsf{sk}_i, A_{\mathsf{keys},i}) : \mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i}) = \mathbf{C}_{\mathtt{bls}}(0) \vee [\mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i}) = \mathbf{C}_{\mathtt{bls}}(1) \wedge \mathsf{pk}_i = g_s^{\mathsf{sk}_i}]\}$$

In plain English, the prover either: (1) puts a 0 in the selector vector; or (2a) puts a 1 in the selection vector *and* (2b) knows the private key of the given public key. (1) and (2a) are a PoK of a representation for Pedersen commitments in `bls12-381` while (2b) is a Schnorr PoK of a discrete logarithm in `secp256k1`—both well studied Σ-protocols [10,21]. The fact that (1) and (2a) are in `bls12-381` while (2b) is in `secp256k1` is not problematic because the disjunction ($\vee$) and conjunction ($\wedge$) operations for composing Σ-protocols are based only on how challenge values are constructed and both groups (`secp256k1` and `bls12-381`) can encode a large t-bit challenge (*e.g.,* $t = 254$) into their exponent groups.

As this protocol is repeated for each key, it is not succinct and will be linear in proof size and verifier time. However, once the selector array is proven correct, the exchange can re-use it every time it does a proof of solvency until it updates its keys. The full details are provided in Protocol 1.

4.2 The π_{assets} argument

The π_{keys} protocol proves that $\mathcal{C}_{f_{\mathsf{sel}}}$ is a commitment to a selector polynomial $f_{\mathsf{sel}}(X)$ (in `bls12-381`) which marks the public keys owned by the exchange. At a given time (block number), the balances of every public key included in the anonymity set will be encoded in polynomial $f_{\mathsf{bal}}(X)$. The product of $f_{\mathsf{sel}}(X) \cdot f_{\mathsf{bal}}(X)$ will preserve the balance values owned by the exchange and zero-out the balance values not claimed by the exchange. The final step is produce a summation over the values in $f_{\mathsf{sel}}(X) \cdot f_{\mathsf{bal}}(X)$. The exchange will put $f_{\mathsf{sel}}(X) \cdot f_{\mathsf{bal}}(X)$ in accumulator form $f_{\mathsf{assets}}(X)$ and prove its correctness. In this form, the total assets will sit at the head (first index) of $f_{\mathsf{assets}}(X)$, which is $f_{\mathsf{assets}}(\omega^0)$. The full details are provided in Protocol 2.

5 Proof of Liabilities

5.1 The $\pi_{\mathsf{liabilities}}$ argument

The exchange will commit to every user balance and produce a commitment of the total amount across all balances. Since the exchange is free to make-up additional users and include them, we want to make

1. $\mathcal{V}$ processes the balance of the anonymous set.
   (a) $\mathcal{V}$ queries the balance of each public key in the set $\mathcal{P}$ chose, denoted by $\bar{b} = \langle \mathfrak{bal}_1, \mathfrak{bal}_2, \ldots, \mathfrak{bal}_n \rangle$.
   (b) $\mathcal{V}$ interpolates a polynomial $f_{\mathsf{bal}}(X)$ from $\bar{b}$.
2. $\mathcal{V}$ commits to f_{bal} and sends $f_{\mathsf{bal}}, \mathcal{C}_{f_{\mathsf{bal}}}$ to $\mathcal{P}$.
3. $\mathcal{P}$ takes as input the selector vector of keys shown ownership of, $f_{\mathsf{sel}}(X)$, from π_{keys}.
4. $\mathcal{P}$ constructs an accumulative polynomial $f_{\mathsf{assets}}(X)$ such that
   (a) $f_{\mathsf{assets}}(X) - f_{\mathsf{assets}}(X\omega) = f_{\mathsf{bal}}(X) \cdot f_{\mathsf{sel}}(X), X \neq \omega^{n-1}$
   (b) $f_{\mathsf{assets}}(\omega^{n-1}) = f_{\mathsf{bal}}(\omega^{n-1}) \cdot f_{\mathsf{sel}}(\omega^{n-1})$
5. $\mathcal{P}$ commits to $f_{\mathsf{assets}}, f_{\mathsf{bal}}, f_{\mathsf{sel}}$ and sends the commitments to $\mathcal{V}$.
6. $\mathcal{V}$ replies with a random evaluation point $\zeta \in \mathbb{F} \setminus H$.
7. $\mathcal{P}$ computes $\pi_\zeta \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_{\mathsf{assets}}, f_{\mathsf{bal}}, f_{\mathsf{sel}}; \zeta)$,
   $\pi_{\zeta\omega} \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_{\mathsf{assets}}; \zeta\omega)$,
   and $\mathbf{C}_{\texttt{bls}}(f_{\mathsf{assets}}(\omega^0)) \leftarrow \mathsf{KZG}_{cm}.\mathsf{Prove}(f_{\mathsf{assets}}; f_{\mathsf{assets}}(\omega^0); \omega^0)$.
8. $\mathcal{V}$ outputs **acc** if and only if
   (a) $\mathsf{KZG}_{zk}.\mathsf{Verify}(\mathcal{C}_{f_{\mathsf{assets}}}, \mathcal{C}_{f_{\mathsf{bal}}}, \mathcal{C}_{f_{\mathsf{sel}}}; \pi_\zeta; \zeta)$, $\mathsf{KZG}_{zk}.\mathsf{Verify}(\mathcal{C}_{f_{\mathsf{assets}}}; \pi_{\zeta\omega}; \zeta\omega)$, and $\mathsf{KZG}_{cm}.\mathsf{Verify}(\mathcal{C}_{f_{\mathsf{assets}}}; \mathbf{C}_{\texttt{bls}}(f_{\mathsf{assets}}(\omega^0)); \omega^0)$ return 1.
   (b) The opening evaluations satisfy the conditions (a) and (b) in step 4.

Protocol 2: The π_{assets} proof demonstrates that the balances associated with each key in the anonymity set are included, the subset not owned by the exchange (per selector vector from π_{keys}) are zero-ed out, and remaining balances are totalled correctly in $f_{\mathsf{assets}}(\omega^0)$.

sure this does not help an insolvent exchange in any way. To do this, we force all balances to be zero or positive numbers. For a finite field, this means small integers that have no chance of exceeding the group order (modular reduction) when added together. In practice, we can limit ourselves to an even smaller range that is sufficient to capture what a balance in ETH (or fractions of ETH) might look like. These balances are expressed in binary and we use range proof from Section A.9.

However, when we turn to implement this in practice, we encounter a roadblock. If μ balances are placed as k-bit numbers side-by-side in a vector, we need a vector of size $\mu \cdot k$. If we want to optimize polynomial interpolation, we encode our array at x-coordinates that correspond to the roots of unity of the exponent group (see Section A.7) and for `bls12-381`, we can only efficiently encode data vectors of length up to $2^{32} = 4,294,967,296$.[7] Consider an exchange with $\mu = 1,000,000$ accounts, only 12 bits are left to capture account balances, say, as between 0.01 and 40.96 ETH ($30 to $150K USD at time of writing). Exchanges could have more than 1 million accounts, the largest could be more than $150K USD, and an exchange could have a long tale of accounts with balances less than $30 such that rounding them all up to $30 creates a solvency issue. Clearly $k = 2^{32}$ is not large enough for directly encoding liabilities (as binary numbers) into a single polynomial.

To deal with this issue, there are three main alternatives. (1) The exchange can encode points at arbitrary x-coordinates and use general (Laplacian) interpolation, (2) the exchange can break down what it is proving into chunks but this will require one succinct proof per chunk, or (3) the range proof could be adapted for decomposition into something larger than bits (*e.g.,* bytes or 32-bit words). The latter may be feasible with lookup arguments, but we do not pursue modifying the range proof [3] in this work. Instead we opt for (2). Specifically we will produce a polynomial argument for the first bit of every account, for the second bit of every account, *etc.* This means $\pi_{\mathsf{liabilities}}$ will be linear in proof size and verifier work but it is linear in the bit-precision of each account (k) and is in fact constant (succinct) in terms of the number of users on the exchange. For example, we will later show if accounts are captured with a precision of 32-bits, the proof size will be under 10KB and verifier time will be under 8ms independent of the number of users on the exchange (see Figure 3).

[7] The exponent group in `bls12-381` has 2-adicity of 32.

1. $\mathcal{P}$ runs Protocol 7 with $\{\mathfrak{bal}_1, \mathfrak{bal}_2, \ldots, \mathfrak{bal}_\mu\}$ to compute $\{p_1, p_2, \ldots, p_k\}$ and $\{v_1, v_2, \ldots, v_k\}$
2. $\mathcal{P}$ builds an additive accumulator $\overline{\nu}$ for p_1 where $\nu_k = \mathfrak{bal}_k = p_1(\omega^{k-1})$ and $\nu_i = \nu_{i+1} + \mathfrak{bal}_i, i \in [1, \mu]$. Remark: ν_1 will contain the total liability value.
3. $\mathcal{P}$ interpolates f_{liab} from $\overline{\nu}$ and publishes the commitments to f_{liab} and $\{p_1, p_2, \ldots, p_k\}$.
4. $\mathcal{V}$ replies with a random evaluation point $\zeta \in \mathbb{F} \setminus H$.
5. $\mathcal{P}$ shows w_1, w_2 such that

$$w_1 := [f_{\mathsf{liab}}(X) - f_{\mathsf{liab}}(X\omega) - p_1(X)] \cdot (X - \omega^{\mu-1})$$
$$w_2 := [f_{\mathsf{liab}}(X) - p_1(X)] \cdot \frac{X^\mu - 1}{X - \omega^{\mu-1}}$$

are vanishing on H by computing $\pi_\zeta \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_{\mathsf{liab}}, p_1, p_2, \ldots, p_k; \zeta)$, $\pi_{\zeta\omega} \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_{\mathsf{liab}}; \zeta\omega)$, and $\mathbf{C}_{\mathtt{bls}}(f_{\mathsf{liab}}(\omega^0)) \leftarrow \mathsf{KZG}_{cm}.\mathsf{Prove}(f_{\mathsf{liab}}; f_{\mathsf{liab}}(\omega^0); \omega^0)$.
6. $\mathcal{V}$ outputs **acc** if and only if
   (a) $\mathsf{KZG}_{zk}.\mathsf{Verify}(f_{\mathsf{liab}}, p_1, p_2, \ldots, p_k; \pi_\zeta; \zeta)$, $\mathsf{KZG}_{zk}.\mathsf{Verify}(f_{\mathsf{liab}}; \pi_{\zeta\omega}; \zeta\omega)$, and $\mathsf{KZG}_{cm}.\mathsf{Verify}(\mathcal{C}_{f_{\mathsf{liab}}}; \mathbf{C}_{\mathtt{bls}}(f_{\mathsf{liab}}(\omega^0)); \omega^0)$ return 1.
   (b) $\{w_1, w_2\}$ and $\{v_1, v_2, \ldots, v_k\}$ are vanishing on H.

Protocol 3: The $\pi_{\mathsf{liabilities}}$ proof demonstrates that each liability is either zero or a positive number, and that the balances are totalled correctly in $f_{\mathsf{liab}}(\omega^0)$.

1. $\mathcal{P}$ interpolates the identifier polynomial $f_{\mathsf{uid}}(X)$ such that $f_{\mathsf{uid}}(X_i) = \mathfrak{uid}_i$ for i from 1 to μ.
2. $\mathcal{P}$ publishes the commitments to $f_{\mathsf{uid}}(X)$ and p_1 (from $\pi_{\mathsf{liabilities}}$ above).
3. For check from user i, $\mathcal{P}$ tells the user he is at index i and opens $f_{\mathsf{uid}}(\omega^{i-1})$ and $p_1(\omega^{i-1})$.
4. $\mathcal{V}$ outputs **acc** if and only if
   (a) His user identifier is the evaluation of f_{uid} at the given point ω^{i-1}.
   (b) His balance is the evaluation of p_1 at the given point ω^{i-1}.

Protocol 4: The π_{users} proof demonstrates that to each user who checks that their balance is recorded correctly under a unique identifier for them (to mitigate clash attacks).

The protocol creates k polynomials—the kth polynomial p_k for the last bit of each of the μ accounts, the last second polynomial for the last second bit of every account, *etc.* It conducts a range proof 'vertically' (across $\{p_1(\omega^i), p_2(\omega^i), \ldots, p_k(\omega^i)\}$ for $\mathfrak{bal}_i$) for each account (for all i). It then converts the bits into integers 'vertically' $(p_j(\omega^i) - 2p_{j+1}(\omega^i) \in \{0, 1\}, j \in [1, k)$, so that $p_1(\omega^i) = \mathfrak{bal}_i)$ for each account, creating a polynomial f_{liab} of each user's balance. Last it sums up all elements 'horizontally' $(\sum_{i=1}^{\mu-1} f_{\mathsf{liab}}(\omega^i))$ in f_{liab} to produce the total liabilities (Protocol 7 in Section B.3). The bit decomposition is argued with the range proof and the summation of balances is argued with a sum-check. The full protocol is given in Protocol 3.

5.2 The π_{users} argument

The π_{users} argument is conducted between the exchange and the user, so the user can check that their balance is correctly encoded into the polynomials used in $\pi_{\mathsf{liabilities}}$. If two users have the same balance, a malicious exchange might include only one of the balances and open up the same balance for each user. Unless the users compared their proofs, they would not catch the exchange (*cf.* [5]). This attack appears in other cryptographic protocols where users need to check things, the main one being cryptographic voting schemes. It has been studied under general definitions as a 'clash attack' [20]. The solution is to label each balance with a unique user identifier [9]. Labeling can be done with an additional polynomial of labels under the assumption that a user ID and a balance need to be at the same index. A user ID can be the hash of the user's account name or email address. The full protocol is given in Protocol 4.

1. $\mathcal{P}$ computes equity eq as the total assets minus the total liabilities.
2. $\mathcal{P}$ publishes commitment to polynomial $f_{\mathsf{eq}}(X)$ where $f_{\mathsf{eq}}(\omega^0) = \mathsf{eq}$.
3. $\mathcal{P}$ generates a range proof for eq in $f_{\mathsf{eq}}(X)$ to demonstrate it is a non-negative integer.
4. $\mathcal{P}$ opens $f_{\mathsf{eq}}(\omega^0)$ through KZG_{cm} and publishes $\mathbf{C}_{\mathtt{bls}}(f_{\mathsf{assets}}(\omega^0))$, $\mathbf{C}_{\mathtt{bls}}(f_{\mathsf{liab}}(\omega^0))$ from π_{assets} and $\pi_{\mathsf{liabilities}}$.
5. $\mathcal{V}$ outputs **acc** if and only if
   (a) $\mathbf{C}_{\mathtt{bls}}(f_{\mathsf{assets}}(\omega^0)) = \mathbf{C}_{\mathtt{bls}}(f_{\mathsf{liab}}(\omega^0)) \cdot \mathbf{C}_{\mathtt{bls}}(f_{\mathsf{eq}}(\omega^0))$.
   (b) The range proof for eq is valid.

Protocol 5: The π_{solvency} proof demonstrates that the total assets exceed the total liabilities by a non-negative integer (called the equity).

5.3 The π_{solvency} argument

The final step of the proof is prove the total assets exceed the total liabilities. At the end of π_{assets}, the total assets are contained in the polynomial evaluation point $f_{\mathsf{assets}}(\omega^0)$; while at the end of $\pi_{\mathsf{liabilities}}$, the total liabilities are contained in $f_{\mathsf{liab}}(\omega^0)$. Assuming assets exceed liabilities by some amount, this amount can be added to the liability-side to provide a difference of exactly zero. The full argument is given in Protocol 5.

6 Security Analysis

We adapt the security definition of a zero-knowledge proof of solvency from Provisions [9] in Section C.1. We refer to Section C.3 for the proofs of Xiezhi's sub-components: π_{keys}, π_{assets}, $\pi_{\mathsf{liabilities}}$, π_{assets}, π_{users}, and π_{solvency}. Here we offer the security proof for the the main theorem:

Theorem 1. *Xiezhi (Xiezhi* $\leftarrow \langle \pi_{\mathsf{keys}}, \pi_{\mathsf{liabilies}}, \pi_{\mathsf{assets}}, \pi_{\mathsf{users}} \rangle$*) is a privacy-preserving proof of solvency with respect to Definition 12.*

Proof. To prove this theorem, we rely on the corollaries in Section C.3. There are no new insights, it is simply a matter of mapping what is proven in the corollaries onto what is required in the definition of a privacy-preserving proof of solvency.

1. *Correctness.* If π_{solvency} is complete (Corollary 6), then Xiezhi is correct according to Definition 12.
2. *k-Soundness.* If $\mathcal{A}$ and $\mathcal{L}$ are not a valid pair and the protocol accepts with probability greater than $\mathsf{neg}(k)$, then π_{solvency} is not sound (contradicting Corollary 6), where soundness is bounded by $k = \min[d/n, 2^{-t}]$ where $d/n = 2^{-233}$ (Schwartz-Zippel lemma for polynomial commitments in `bls12-381`) and $t = 2^{-254}$ (challenge length for NIZKPs under Fiat-Shamir for a common challenge in `secp256k1` and `bls12-381`). If $\mathcal{L}[\mathfrak{uid}] \neq \ell$ (*i.e.,* the exchange provides the user with the wrong balance) and the protocol accepts with probability greater than $\mathsf{neg}(k)$, then π_{solvency} is not sound (contradicting Corollary 6).
3. *Ownership.* Recall that ownership means that if the protocol accepts, there exists an extractor that can produce x for all $y \in \mathcal{A}$. We show such an extractor in the proof of Theorem 5.
4. *Privacy.* Roughly, this means a (statically) corrupted user cannot distinguish between an interaction using the real pair $\mathcal{A}$ and $\mathcal{L}$ and any other (equally sized) valid pair $\hat{\mathcal{A}}$ and $\hat{\mathcal{L}}$ such that $\hat{\mathcal{L}}[\mathfrak{uid}] = \mathcal{L}[\mathfrak{uid}]$ (*i.e.,* the simulated pair records the same balance for all corrupted users as the real valid pair). This follows from π_{users} and π_{solvency} being zero-knowledge (Corollary 5 and 6), where the former covers the case that the universally verifiable proof reveals private information, and the latter covers the supplementary user check proof.

Therefore Xiezhi is a privacy-preserving proof of solvency.

7 Performance Evaluation

7.1 Theoretical Performance

In this section, we analyze the performance of Xiezhi, and compare the performance of our work with other prior schemes. Our analysis ignores the relatively trivial cost like Fast Fourier Transform (FFT) and focuses on the heavy work such as multi-scalar multiplication (MSM) and group operations. Our analysis also ignores the differences in the implementations and assumes each protocol is executed in a single thread.

Proof of Assets We use κ to denote the size of the anonymity set and we assume κ is the power of two for simplicity. The performance analysis of π_{keys} and π_{users} are performed as follows:

- π_{keys}: When opening an evaluation of a KZG commitment for each public key, one MSM for the witness and one MSM for the blinding polynomial are involved. The number of scalar multiplications of the Σ-protocol is constant. Thus, the overhead proving time of π_{keys} is $O(\kappa^2)$. In terms of the verifier's work, for each key, $\mathcal{V}$ performs scalar multiplications for constant times and manipulates the batched KZG scheme to validate related polynomial constraints, which means $O(1)$ verifying time and proof size. Therefore, the overhead verifying time and proof size of π_{keys} is $O(\kappa)$.
- π_{assets}: $\mathcal{P}$ constructs the accumulator and commits to a constant number of polynomials. Since $\mathcal{P}$ opens one point of each polynomial, the proving time is $O(\kappa)$ and the proof size is $O(1)$. While it takes constant time for $\mathcal{V}$ to verify the proof of the PCS, $\mathcal{V}$ needs to interpolate the balances and commit to the balance polynomial, which means one MSM is involved. Therefore, the overhead verifier's work of π_{assets} is $O(\kappa)$.

Proof of Liability We use μ to denote the number of users and k to denote the allowed size of the range proof. When $\mathcal{P}$ computes the accumulative polynomial to prove the total liablity is correct, it can be done in linear time. Different from π_{keys}, $\mathcal{P}$ only opens each polynomial at one random evaluation point. Thus, the proving time is $O(\mu)$.

The verifier's work is broken into π_{users} and $\pi_{\mathsf{liabilities}}$:

- π_{users}: Each user verifies his balance is the evaluation of the polynomial p_1 and his user identifier is the evaluation of the polynomial f_{uid}. The user checks two KZG proofs, so the proof size and the verifying time for customers are both $O(1)$.
- $\pi_{\mathsf{liabilities}}$: Auditor verifies the constraints among polynomials $\{p_i\}$ are correct and the committed total liabilities is the evaluation of $f_{\mathsf{liab}}(\omega^0)$. The first step can be done in $O(k)$ as the number of polynomials is related to the range proof rather than the number of users. The second step involves opening one KZG commitment through KZG_{cm}, which means the verifying time and the proof size for auditors are both $O(k)$.

Comparison In Table 2, we compare this work with other prior PoA schemes. Both IZPR and this work utilize bootstrapping, but the bootstrapping of IZPR will be introduced in their following paper. We only analyze the performance of the bootstrapping for this work. In Table 3, we compare this work with prior PoL schemes.

7.2 Implementation and Benchmark Methodology

To evaluate the performance of Xiezhi, we implemented our protocols in Rust based on the popular library, arkworks[8]. Our implementation is publicly accessible on GitHub[9]. We chose the pairing-friendly elliptic curve `bls12-381` for the KZG commitment which has 128-bit security.

Our experiments were conducted on a personal computer with i9-13900KF and 32GB of memory. The experimental data including balances and `secp256k1` key pairs are randomly generated locally for simplicity.

[8] `https://github.com/arkworks-rs`
[9] `https://github.com/Shvier/proof-of-solvency`

π_{keys}			
Scheme	**Proving time**	**Verifying time**	**Proof size**
Xiezhi (**Ours**)	$O(\kappa^2)$	$O(\kappa)$	$O(\kappa)$

π_{assets}				
Scheme	**Proving time**	**Verifying time**		**Proof size**
		π_{input}	π_{proof}	
Provisions[9]	$O(\kappa)$	N/A	$O(\kappa)$	$O(\kappa)$
Bulletproofs[4]	$O(\kappa)$	N/A	$O(\kappa)$	$O(\log \kappa)$
IZPR[8]	$O(t \log t)$	$O(\kappa)$	$O(1)$	$O(1)$
Xiezhi (**Ours**)	$O(\kappa)$	$O(\kappa)$	$O(1)$	$O(1)$

Table 2: Comparison of this work with prior PoA schemes. π_{input} is the verifier processes the public inputs before validating the proof; π_{proof} is the verifier verifies the proof sent by the prover. Notation: κ is the number of keys that the exchange wants to prove. For IZPR[8], t is the throughput of the blockchain (number of addresses which have changed since the last proof).

Scheme	**Proving time**	**Verifying time**		**Proof size**	
		π_{users}	$\pi_{\mathsf{liabilities}}$	π_{users}	$\pi_{\mathsf{liabilities}}$
Provisions[9]	$O(\mu)$	$O(1)$	$O(\mu)$	$O(1)$	$O(\mu)$
DAPOL+[17]	$O(\mu \log \mu)$	$O(\log \mu)$	$O(1)$	$O(\log \mu)$	$O(1)$
SPPPOL[13]	$O(\log_\lambda \mu)$	$O(\log_\lambda \mu)$	$O(1)$	$O(\log_\lambda \mu)$	$O(1)$
Notus[25]	$O(\mu \log \mu)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Xiezhi (**Ours**)	$O(k \cdot \mu)$	$O(1)$	$O(k)$	$O(1)$	$O(k)$

Table 3: Comparison of this work with prior PoL schemes. Notation: μ is the number of users, k is the number of bits of the range proof. For SPPPOL [13], λ is the arity of the Verkle Tree it uses.

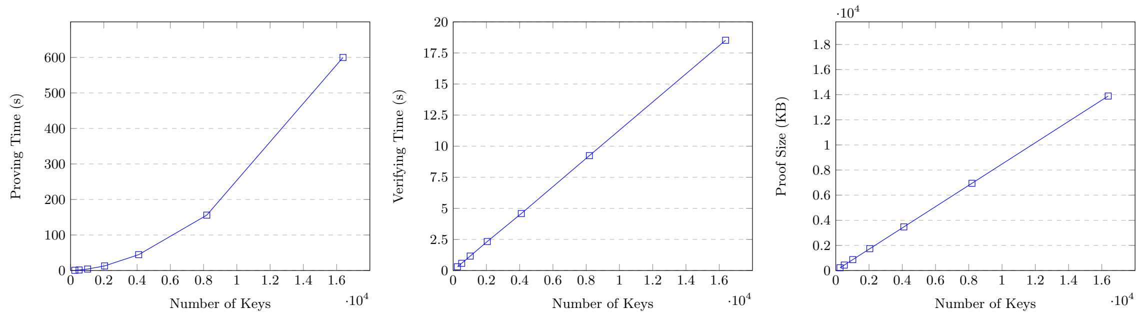
Since there is no range-proof for PoA, we tested the PoA with balances randomly distributed in $[1, 2^{64})$ to simulate the real distribution of assets, and for PoL, we tested the program with balances randomly distributed in $[1, 2^8)$, $[1, 2^{16})$, $[1, 2^{32})$, and $[1, 2^{64})$. We simulated $2^8, 2^9 \ldots, 2^{14}$ and $2^{10}, 2^{11}, \ldots, 2^{20}$ users for PoA and PoL respectively. Simulating different numbers of users for PoA and PoL is because π_{keys} was time-consuming for a larger number of users. For each protocol, we ran the test ten times with the same experimental data. Our figures are interpolated from the average performance of ten times discarding the maximum and minimum of the samples.

7.3 Experimental Evaluation

Figure 1, 2, and 3 reflect the performance of Xiezhi in single thread with i9-13900KF. The Subfigure (a) of Figure 1 suggests it takes around 600 seconds to generate the proofs for 16,384 keys with i9-13900KF for π_{keys}, and the proof size is 13,893KB. It seems impractical for the exchange to achieve such performance if it wants the maximum anonymous set. However, recall the exchange only needs to perform π_{keys} once. Also, the proving time can be reduced significantly if manipulating more efficient KZG opening schemes. See Section 8 for more detailed optimizations.

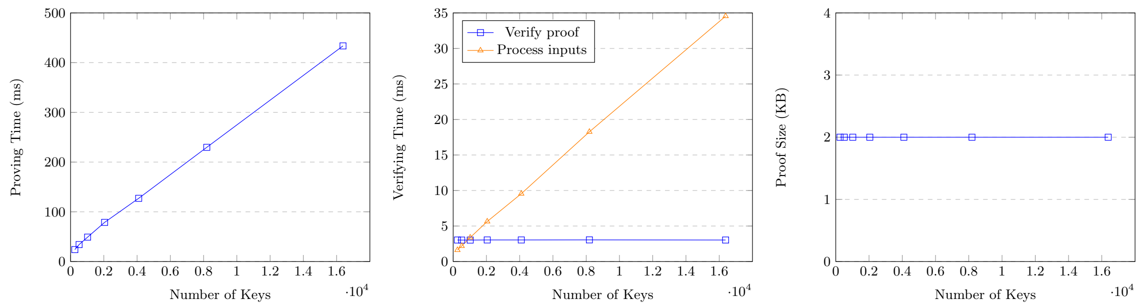
Figure 2 shows the proving time and the verifying time are linear in the number of the keys. In our experiments, it takes 433.66 milliseconds to generate the proof and 37.57 milliseconds to verify the proof for 16,384 keys. This suggests the proving time is less than 2 hours if the anonymous set is all addresses on Ethereum without any other optimizations. Since the proof size of a KZG commitment is unrelated to the degree of the polynomial (the number of keys), the proof size of π_{assets} is constant (2KB) based on our implementation.

Figure 3 illustrates the performance of our PoL with different numbers of users and allowed ranges for balance. Our experiments show the proving time grows linearly by the number of users while the verifying time and the proof size are constant. From the test result of Binance's PoL, it needs 1.5 days to generate



(a) (b) (c)

Fig. 1: Performance of π_{keys}



(a) (b) (c)

Fig. 2: Performance of π_{assets}

the proof for 100 million accounts with 100 servers [10], but our PoL requires less than 10 minutes with the same number of servers. This indicates our protocol is practical to handle real-world applications.

8 Open Research Challenges

The efficiency and succinctness of Xiezhi might be further improved. Recall the heaviest work of π_{keys} is proving each committed point is correct, and the opening scheme we demonstrated from Plonk requires $t \cdot d$ scalar multiplications for prover, where t is the number of the opening points and d is the degree bound of the polynomial. The work in BDFG20 [2] can reduce this complexity to $2n$ scalar multiplications, which means the dominating complexity will become $O(n)$ rather than $O(n^2)$. The aggregation slightly increases the verifier's work but the extra cost is trivial because of the succinctness of the KZG commitment scheme. These optimizations can be applied to both our PoA and PoL. Moreover, the proof length for multiple points of the KZG commitment will also be decreased to $O(1)$ if BDFG20 is integrated, but the total proof length is still $O(n)$ because of the proof of the Σ-protocol. Other advances in other Poly-IOP systems require future research: lookup arguments, multivariate polynomials (and corresponding commitment schemes), and folding techniques.

If blockchains like Ethereum add low-gas cost support for `bls12-381`, a topic of discussion (EIP-2537[11]), verifying proofs of solvency could move on-chain. If an exchange fails to provide a smart contract with a proof of solvency in a timely fashion, the smart contract could be called to trigger penalties or other actions.

[10] `https://github.com/binance/zkmerkle-proof-of-solvency/?tab=readme-ov-file`
[11] `https://eips.ethereum.org/EIPS/eip-2537`

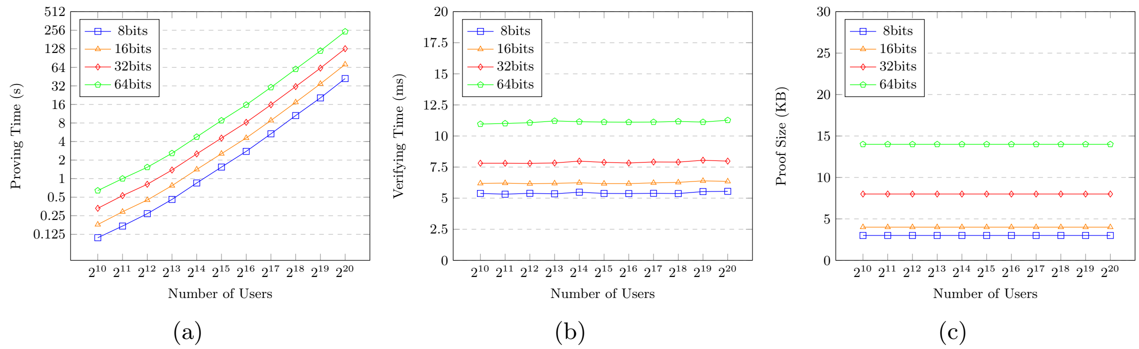


Fig. 3: Performance of $\pi_{\mathsf{liabilities}}$

Acknowledgements. We thank Dr. Xiang Fu for answering questions about IZPR and the a16z crypto research summer 2024 cohort for feedback. We thank the reviewers who helped to improve our paper. J. Clark acknowledges support for this research project from (i) the National Sciences and Engineering Research Council (NSERC), Raymond Chabot Grant Thornton, and Catallaxy Industrial Research Chair in Blockchain Technologies, (ii) the AMF (Autorité des Marchés Financiers), and (iii) NSERC through a Discovery Grant.

References

1. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. pp. 626–643. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
2. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Paper 2020/081 (2020), `https://eprint.iacr.org/2020/081`, `https://eprint.iacr.org/2020/081`
3. Boneh, D., Fisch, B., Gabizon, A., Williamson, Z.: A simple range proof from polynomial commitments (2019), `https://hackmd.io/@dabo/B1U4kx8XI`
4. 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 (SP). pp. 315–334 (2018). `https://doi.org/10.1109/SP.2018.00020`
5. Chalkias, K., Chatzigiannis, P., Ji, Y.: Broken proofs of solvency in blockchain custodial wallets and exchanges. In: Matsuo, S., Gudgeon, L., Klages-Mundt, A., Perez Hernandez, D., Werner, S., Haines, T., Essex, A., Bracciali, A., Sala, M. (eds.) Financial Cryptography and Data Security. FC 2022 International Workshops. pp. 106–117. Springer International Publishing, Cham (2023)
6. Chase, M., Orrù, M., Perrin, T., Zaverucha, G.: Proofs of discrete logarithm equality across groups. Cryptology ePrint Archive, Paper 2022/1593 (2022), `https://eprint.iacr.org/2022/1593`, `https://eprint.iacr.org/2022/1593`
7. Christ, M., Baldimtsi, F., Chalkias, K.K., Maram, D., Roy, A., Wang, J.: Sok: Zero-knowledge range proofs. Cryptology ePrint Archive, Paper 2024/430 (2024), `https://eprint.iacr.org/2024/430`, `https://eprint.iacr.org/2024/430`
8. Conley, T., Diaz, N., Espada, D., Kuruvilla, A., Mayne, S., Fu, X.: Izpr: Instant zero knowledge proof of reserve. In: Financial Cryptography and Data Security. FC 2024 International Workshops (2024), `https://api.semanticscholar.org/CorpusID:266345810`
9. Dagher, G.G., Bünz, B., Bonneau, J., Clark, J., Boneh, D.: Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 720–731. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). `https://doi.org/10.1145/2810103.2813674`, `https://doi.org/10.1145/2810103.2813674`
10. Damgard, I.: On σ-protocols (2010), `https://www.cs.au.dk/~ivan/Sigma.pdf`

11. Dao, Q., Miller, J., Wright, O., Grubbs, P.: Weak fiat-shamir attacks on modern proof systems. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 199–216. IEEE Computer Society, Los Alamitos, CA, USA (may 2023). https://doi.org/10.1109/SP46215.2023.10179408, https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179408
12. Doerner, J., Shelat, A., Evans, D.: Zeroledge: Proving solvency with privacy (2015), https://api.semanticscholar.org/CorpusID:211105655
13. Falzon, F., Elkhiyaoui, K., Manevich, Y., De Caro, A.: Short privacy-preserving proofs of liabilities. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 1805–1819. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616645, https://doi-org.lib-ezproxy.concordia.ca/10.1145/3576915.3616645
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
15. Gabizon, A., Williamson, Z.J.: fflonk: a fast-fourier inspired verifier efficient version of plonk. Cryptology ePrint Archive, Paper 2021/1167 (2021), https://eprint.iacr.org/2021/1167, https://eprint.iacr.org/2021/1167
16. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953, https://eprint.iacr.org/2019/953
17. Ji, Y., Chalkias, K.: Generalized proof of liabilities. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3465–3486. CCS '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3460120.3484802, https://doi-org.lib-ezproxy.concordia.ca/10.1145/3460120.3484802
18. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 177–194. Springer (2010)
19. Kelkar, M., Babel, K., Daian, P., Austgen, J., Buterin, V., Juels, A.: Complete knowledge: Preventing encumbrance of cryptographic secrets. Cryptology ePrint Archive, Paper 2023/044 (2023), https://eprint.iacr.org/2023/044, https://eprint.iacr.org/2023/044
20. Kusters, R., Truderung, T., Vogt, A.: Clash attacks on the verifiability of e-voting systems. In: 2012 IEEE Symposium on Security and Privacy. pp. 395–409 (2012). https://doi.org/10.1109/SP.2012.32
21. Maurer, U.: Unifying zero-knowledge proofs of knowledge. In: International Conference on Cryptology in Africa. pp. 272–286. Springer (2009)
22. Nikolaenko, V., Ragsdale, S., Bonneau, J., Boneh, D.: Powers-of-tau to the people: Decentralizing setup ceremonies. In: Pöpper, C., Batina, L. (eds.) Applied Cryptography and Network Security. pp. 105–134. Springer Nature Switzerland, Cham (2024)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)
24. Szepieniec, A., Zhang, Y.: Polynomial IOPs for linear algebra relations. Cryptology ePrint Archive, Paper 2020/1022 (2020), https://eprint.iacr.org/2020/1022
25. Xin, J., Haghighi, A., Tian, X., Papadopoulos, D.: Notus: Dynamic proofs of liabilities from zero-knowledge RSA accumulators. In: 33rd USENIX Security Symposium (USENIX Security 24). USENIX Association, Philadelphia, PA (Aug 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/xin

A Cryptographic Building Blocks

This appendix describes cryptographic building blocks we use that are from the literature and not our own contribution.

A.1 Discrete Logarithm Assumption

The discrete logarithm problem describes, given a triplet $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of order p generated by $g \in \mathbb{G}$, and an element $y \in \mathbb{G}$, for a given adversary $\mathcal{A}$, $\mathcal{A}$ needs to compute an x such that $y = g^x$. The discrete logarithm assumption holds for $\mathbb{G}$ if it is infeasible for $\mathcal{A}$ to find such x in polynomial time.

A.2 Pedersen Commitment

The Pedersen commitment scheme [23] enables $\mathcal{P}$ to *commit* to a value x without revealing it. Pedersen commitment provides perfect hiding and computational binding based on the discrete logarithm assumption. Additionally, Pedersen commitments are *additively homomorphic*: given two commitments $\mathbf{C}_1$ and $\mathbf{C}_2$, the summation of their secrets x_1 and x_2 is the secret of $\mathbf{C}_1 \cdot \mathbf{C}_2$.

A.3 Zero-Knowledge Proofs

Informally, a zero-knowledge proof is a cryptographic protocol allowing $\mathcal{P}$ to convince $\mathcal{V}$ that the claiming statement is true without revealing additional information, except the fact that the statement's truth. A zero-knowledge proof must satisfy the following properties:

1. **Completeness:** If the statement is true, $\mathcal{V}$ will be convinced.
2. **Soundness:** If the statement is false, the probability that $\mathcal{V}$ is convinced is negligible.
3. **Zero-knowledge:** If the statement is true, $\mathcal{V}$ learns nothing except the fact that the statement is true.

Definition 1 (Special Soundness). *For a Σ-protocol, if the witness w can be extracted from any two accepting conversations on the same input x with the same message but a different challenge, we call this special soundness of Σ-protocol. Particularly, special soundness implies soundness.*

Definition 2 (Knowledge Soundness in the Algebraic Group Model). *For any algebraic adversary $\mathcal{A}$ in an interactive protocol between $\mathcal{P}$ and $\mathcal{V}$ for a relation $\mathcal{R}$, there exists a p.p.t extractor $\mathcal{E}$ given access to $\mathcal{A}$'s messages during the protocol, and $\mathcal{A}$ can win the following game with negligible probability:*

1. *$\mathcal{A}$ chooses input x and outputs the message like $\mathcal{P}$.*
2. *$\mathcal{E}$, given access to $\mathcal{A}$'s outputs from the previous step, outputs the witness w.*
3. *$\mathcal{A}$ wins if*
   (a) *$\mathcal{V}$ outputs* **acc** *at the end of the protocol, and*
   (b) *$(x, w) \notin \mathcal{R}$.*

Definition 3 (Honest Verifier Zero Knowledge). *Honest verifier zero knowledge, or HVZK, is for a Σ-protocol, if there exists a p.p.t simulator $\mathcal{S}$ such that the transcript produced by $\mathcal{S}$ has the same distribution as the transcript of a conversation between the honest $\mathcal{P}$ and $\mathcal{V}$ on the same input.*

Definition 4 (Special Honest Verifier Zero Knowledge). *Special honest verifier zero knowledge, or special HVZK, is for a Σ-protocol, if there exists a p.p.t simulator such that additionally given the challenge c to $\mathcal{S}$, the transcript produced by $\mathcal{S}$ has the same distribution as the transcript of a conversation between the honest $\mathcal{P}$ and $\mathcal{V}$ on the same input, even if for the case that the witness for the statement does not exist.*

A.4 Σ-Protocol

The $\Sigma-$protocol is a three-move interactive proof system. We define the $\Sigma-$protocol similarly to [10].

Definition 5 (Σ-protocol). *Let R be a binary relation between the statement x and the witness w. Given common input x to $\mathcal{P}$ and $\mathcal{V}$, and private input (x, w) such that $(x, w) \in R$ to $\mathcal{P}$, they run the following protocol:*

1. *$\mathcal{P}$ computes a message m from (x, w) and sends m.*
2. *$\mathcal{V}$ sends a random challenge c.*
3. *$\mathcal{P}$ replies with z.*

*At the end of the protocol $\mathcal{V}$ has the data (x, m, c, z). He decides to output **acc** or **rej**; such that*

- ***Completeness**: If $\mathcal{P}$ follows the protocol to generate the message (m, c, z), $\mathcal{V}$ always accepts.*
- ***Special soundness**: If there exists a p.p.t extractor $\mathcal{E}$, given any input x and any two accepting $(m, c, z), (m, c', z')$ where $c \neq c'$, $\mathcal{E}$ can compute w where $(x, w) \in R$.*
- ***HVZK**: If there exists a p.p.t simulator $\mathcal{S}$, such that the transcript produced by $\mathcal{S}$ is indistinguishable from the messages between $\mathcal{P}$ and $\mathcal{V}$.*

A commonly well-known way to convert a $\Sigma-$protocol into non-interactive is using the Fiat-Shamir transform [14]. But we still use the standard interactive $\Sigma-$protocol to demonstrate our work for comprehension.

A.5 Disjunction of Σ-Protocols (OR Proof)

The disjunction of Σ-protocols (OR proof) allows $\mathcal{P}$ to prove the claimed x is x_1 or x_2 through a Σ-protocol. More precisely, given two inputs x_1, x_2, $\mathcal{P}$ proves he knows a w such that $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$, but $\mathcal{V}$ cannot learn which one $\mathcal{P}$ knows. We use the same definition as [10].

Definition 6 (OR Proof). *Let s equal 0 or 1. The OR proof is a $\Sigma-$protocol that $\mathcal{P}$ and $\mathcal{V}$ are given two public inputs x_1, x_2, and $\mathcal{P}$ is given w as private input. They run the following protocol:*

1. *$\mathcal{P}$ computes the message m_s using (x_s, w) as input.*
   $\mathcal{P}$ randomly generates c_{1-s} as the challenge for x_{1-s} and runs the simulator $\mathcal{S}(x_{1-s}, c_{1-s})$ to produce (m_{1-s}, z_{1-s}).
2. *$\mathcal{P}$ sends m_s and m_{1-s}.*
3. *$\mathcal{V}$ sends a master challenge c.*
4. *$\mathcal{P}$ computes $c_s = c \oplus c_{1-s}$ and z_s on inputs (x_s, c_s, m_s, w).*
   $\mathcal{P}$ sends $(c_s, c_{1-s}, z_s, z_{1-s})$.

*At the end of the protocol $\mathcal{V}$ verifies $c = c_s \oplus c_{1-s}$ and both (m_s, c_s, z_s, x_s) and $(m_{1-s}, c_{1-s}, z_{1-s}, x_{1-s})$ are valid to output **acc** or **rej**; such that*

- ***Completeness:** The case of c_{1-s} is always accepted by $\mathcal{V}$ as the definition of a simulator; on the other side, the case of c_s has no difference from the standard $\Sigma-$protocol.*
- ***Special soundness:** Let $\mathcal{P}$ execute the protocol twice. Two accepting transcripts*

$$(x_s, x_{1-s}, c, c_s, c_{1-s}, z_s, z_{1-s}), (x_s, x_{1-s}, c', c'_s, c'_{1-s}, z'_s, z'_{1-s}), c \neq c'$$

  are given. It is clear that for some $s = 0$ or 1, the witness w such that $(x_s, w) \in R$ can be extracted through an extractor $\mathcal{E}$ by the special soundness of Σ-protocol.
- ***Special HVZK**: Given a master challenge c, let $\mathcal{S}$ choose c_s or c_{1-s} randomly and the other will be determined. Then let the simulator run twice: $\mathcal{S}(x_s, c_s), \mathcal{S}(x_{1-s}, c_{1-s})$, to output $(m_s, z_s, m_{1-s}, z_{1-s})$. The outputs of $\mathcal{S}$ have the same distribution as those of $\mathcal{P}$.*

We use $x = x_1 \vee x_2$ to denote the value x is x_1 or x_2 in the context of Σ-protocol.

A.6 Polynomial Commitment Scheme

A polynomial commitment scheme (PCS) allows $\mathcal{P}$ to commit to a polynomial to convince $\mathcal{V}$ that claimed evaluations are of the committed polynomial. Particularly, our protocol requires the scheme to use an extra random polynomial to achieve unconditionally binding, *i.e.,* the KZG commitment in Pedersen form. We define the following scheme based on [18,16,2].

Definition 7 (Polynomial Commitment Scheme). *A polynomial commitment scheme consists of three moves:* ***gen****,* ***com****, and* ***open*** *such that*

1. ***gen***(d) *is an algorithm that given a random number* $\tau \in \mathbb{F}$ *and a positive integer* d*, outputs a structured reference string* ***srs*** *such that*

$$\boldsymbol{srs} = \left\langle g_1, g_1^{\tau}, \ldots, g_1^{\tau^d}, h_1, h_1^{\tau}, \ldots, h_1^{\tau^d}, g_2, g_2^{\tau} \right\rangle$$

2. ***com***$(f,\ \boldsymbol{srs})$ *outputs a commitment* $\mathcal{C} = g_1^{f(\tau)} h_1^{\hat{f}(\tau)}$ *to* f*, where* $\hat{f}$ *is a random polynomial of degree* d*, and* f *is a polynomial of degree* d *or less.*
3. ***open*** *is a protocol that* $\mathcal{P}$ *is given input* f*, and* $\mathcal{P}$ *and* $\mathcal{V}$ *are both given*
   - ***srs***
   - $\mathcal{C}$ *- the commitment to* f
   - a *- an evaluation point of* f
   - b *- the evaluation of* $f(a)$
   - $\hat{b}$ *- the evaluation of* $\hat{f}(a)$

   They run the protocol as follows:

   (a) $\mathcal{P}$ *computes the witness* w *for* $(a, b, \hat{b})$ *such that*

   $$w = g_1^{\psi(\tau)} h_1^{\hat{\psi}(\tau)}$$

   where $\psi(x) = \frac{f(X)-f(a)}{X-a}$*, and* $\hat{\psi}(x) = \frac{\hat{f}(X)-\hat{f}(a)}{X-a}$

   (b) $\mathcal{V}$ *outputs* ***acc*** *if and only if*

   $$e(\mathcal{C}/(g_1^b h_1^{\hat{b}}), [1]_2) = e(w, [\tau - a]_2)$$

- ***Completeness****: It is clear that* w *exists if and only if* $f(a) = b$ *and* $\hat{f}(a) = \hat{b}$*, which means* $\mathcal{V}$ *always accepts the proof if* $\mathcal{P}$ *follows the protocol.*
- ***Knowledge soundness in the algebraic group model****: For any algebraic adversary* $\mathcal{A}$ *in an interactive protocol of PCS, there exists a p.p.t extractor* $\mathcal{E}$ *given access to* $\mathcal{A}$*'s messages during the protocol, and* $\mathcal{A}$ *can win the following game with negligible probability:*
  1. *Given the inputs that* $\mathcal{P}$ *can access,* $\mathcal{A}$ *outputs* $\mathcal{C}$*.*
  2. $\mathcal{E}$ *outputs* $f \in \mathbb{F}_{<d}[X]$ *from* $\mathcal{A}$*'s output.*
  3. $\mathcal{A}$ *generates* w *and* b' *at a random evaluation point* a*.*
  4. $\mathcal{A}$ *wins if*
     - $\mathcal{V}$ *accepts the proof at the end of the protocol.*
     - $b' \neq b$*.*

A.7 Roots of Unity

We use the approach of encoding data vectors into polynomials, committing to them using a polynomial commitment scheme (PCS), and forming zero-knowledge arguments—a model called a polynomial-based interactive oracle proof (Poly-IOP). The zk-SNARK system Plonk popularized Poly-IOPs and has many extensions and optimizations. A one-dimensional vector of data is encoded into a univariate polynomial using 1 of 3 methods (all 3 are used at different steps of Plonk): (1) into the coefficients of the polynomial, (2) as roots of the polynomial, and (3) as the y-coordinates ($\mathsf{data}_i = f(x_i)$) of points on the polynomial.

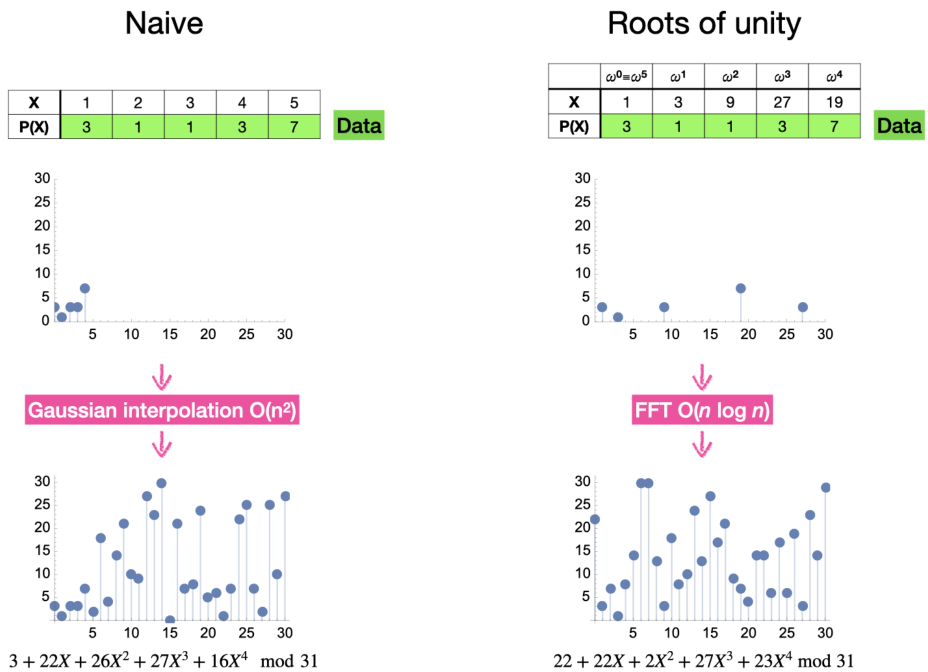


Fig. 4: Small number ($\mathbb{Z}_{31}$) example of encoding a vector of integers $\langle 3, 1, 1, 3, 7\rangle$ into (a) the first 5 points of a polynomial, and (b) into 5th roots of unity ($\omega = 3$).

Plonk mostly relies on (3) and an interpolation algorithm is used to find the corresponding coefficients of the polynomial, which is needed for the PCS. General interpolation algorithms are $O(n^2)$ work for n evaluation points but this can be reduced to $O(n \log n)$ with an optimization.

The optimization enables interpolation via the fast Fourier transform (FFT). It concerns how to choose the x-coordinates, which will serve as the index for accessing the data: evaluating $f(X)$ at x_i will reveal $\mathtt{data}_i$. First note, x-coordinates are from the exponent group ($\mathbb{Z}_q$) and the choices exceed what is feasible to use (2^{255} values in `bls12-381`). Any subset can be used and interpolated. The optimization is to choose them with a mathematical structure. Specifically, instead an additive sequence (e.g., $0, 1, 2, 3, \ldots$), we use a multiplicative sequence $1, \omega, \omega \cdot \omega, \omega \cdot \omega \cdot \omega, \ldots$ or equivalently: $\omega^0, \omega^1, \omega^2, \ldots, \omega^{\kappa-1}$. Further, the sequence is closed under multiplication which means that the next index after $\omega^{\kappa-1}$ wraps back to the first index: $\omega^{k-1} \cdot \omega = \omega^{\kappa} = \omega^0 = 1$ (this property is also useful in proving relationships between data in the vector and its neighbouring values).

For terminology, we say ω is a generator with multiplicative order κ in $\mathbb{Z}_q$. This implies $\omega^{\kappa} = 1$. Rearranging, $\omega = \sqrt[\kappa]{1}$. Thus we can equivalently describe ω as a κ-th root of 1. Finally, as 1 is the unity element in $\mathbb{Z}_q$, ω is commonly called a κ-th root of unity.

For practical purposes, κ represents the length of the longest vector of data we can use in our protocol. Where does κ come from? Different elements of $\mathbb{Z}_q$ will have different multiplicative orders but every order must be a divisor of $q - 1$. Thus κ is the largest divisor of the exact value of q used in an elliptic curve standard. The value of q in `bls12-381` has $\kappa = 2^{32}$ (for terminology, this called a 2-adicity of 32).

A.8 Polynomial Protocol

Gabizon *et al.* [16] introduced the definition of a universal polynomial protocol. Here we describe a variant of it based on the KZG commitment scheme for our work.

Definition 8 (Polynomial Protocol). *Fix positive integer d, t, l. Let $i \in [1, l]$. Let $\mathcal{R} \subseteq \mathbb{F} \times \mathbb{F} \times \cdots \times \mathbb{F}$ be a polynomial relation for one or more polynomials. Given a set of polynomials $f_1, f_2, \ldots, f_t \in \mathbb{F}_{<d}[X]$ as $\mathcal{P}$'s private input, and a set of polynomial relations $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_l$ as public input, a polynomial protocol is a three-move protocol that $\mathcal{P}$ wants to convince $\mathcal{V}$ each $\mathcal{R}_i$ holds for the certain set $F_i \subseteq \{f_1, f_2, \ldots, f_t\}$. $\mathcal{P}$ and $\mathcal{V}$ runs the protocol as follows.*

1. *$\mathcal{P}$ commits to $f_1, f_2, \ldots, f_t$, and publishes all commitments, $\mathcal{C}_{f_1}, \mathcal{C}_{f_2}, \ldots, \mathcal{C}_{f_t}$.*
2. *$\mathcal{V}$ sends a random evaluation point as the challenge.*
3. *$\mathcal{P}$ responds with the corresponding evaluation and the commitment to the witness at the evaluation point for each polynomial.*

*At the end of the protocol, $\mathcal{V}$ outputs **acc** or **rej** by checking*

1. *Each evaluation of $f_1, f_2, \ldots, f_t$ at the random point is valid through the KZG checking*
2. *Each relation $\mathcal{R}_i$ holds for the prescribed polynomials F_i. More precisely, $\mathcal{V}$ verifies the evaluations of the polynomials in F_i satisfy the equation defined by $\mathcal{R}_i$, i.e., the zero test of polynomials.*

A polynomial protocol has the following properties:

- ***Completeness:** $\mathcal{V}$ always outputs **acc** if $\mathcal{P}$ follows the protocol correctly to compute the proof π_i for the relation $\mathcal{R}_i$, and $\mathcal{R}_i$ holds for the prescribed polynomials F_i, denoted by $(F_i, \pi_i) \in \mathcal{R}_i$.*
- ***Knowledge soundness in the algebraic group model**: For any algebraic adversary $\mathcal{A}$ in a polynomial protocol, there exists a p.p.t extractor $\mathcal{E}$ given access to $\mathcal{A}$'s messages during the protocol, and $\mathcal{A}$ can win the following game with negligible probability:*
  1. *Given the inputs that $\mathcal{P}$ can access, $\mathcal{A}$ outputs $\mathcal{C}_{f_1}, \mathcal{C}_{f_2}, \ldots, \mathcal{C}_{f_t}$.*
  2. *$\mathcal{E}$ outputs $f_1, f_2, \ldots, f_t \in \mathbb{F}_{<d}[X]$ from $\mathcal{A}$'s output and $\{F_i\}$ from these polynomials.*
  3. *$\mathcal{A}$ outputs the evaluation at the random evaluation point for each polynomial and the corresponding proofs $\{\pi_i\}$.*
  4. *$\mathcal{A}$ wins if*
     - *$\mathcal{V}$ accepts the proof at the end of the protocol.*
     - *$(F_i, \pi_i) \notin \mathcal{R}_i$ or any evaluation is not correct.*

A.9 Range Proof for Single Value

$\mathcal{P}$ wants to convince $\mathcal{V}$ that a number x is in the range $[0, 2^k)$ without revealing x. They run the protocol as follows:

1. *Given input x, $\mathcal{P}$ decomposes x to a vector of binary digits $\overline{z} = \langle z_1, z_2, \ldots, z_k \rangle$, so that $x = \sum_{i=0}^{k-1} 2^i \cdot z_i$*
2. *$\mathcal{P}$ constructs a vector $\overline{x} = \langle x_1, x_2, \ldots, x_k \rangle$ such that*

$$\begin{aligned} x_1 &= x \\ x_k &= z_k \\ x_i &= 2x_{i+1} + z_i, i \in [1, k-1] \end{aligned}$$

3. *$\mathcal{P}$ interpolates a polynomial f from $\overline{x}$ over a finite field H of order n with elements $\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$*
4. *$\mathcal{P}$ proves the following polynomials are vanishing on H*

$$\begin{aligned} w_1 &:= [f(X) - x] \cdot \frac{X^n - 1}{X - \omega^0} \\ w_2 &:= f(X) \cdot [f(X) - 1] \cdot \frac{X^n - 1}{X - \omega^{n-1}} \\ w_3 &:= [f(X) - 2 \cdot f(X\omega)] \cdot [f(X) - 2 \cdot f(X\omega) - 1] \cdot (X - \omega^{n-1}) \end{aligned}$$

   (a) *$\mathcal{P}$ sends the commitment to $f(X)$*

(b) $\mathcal{V}$ *sends a random challenge* γ
(c) $\mathcal{P}$ *sends the commitment to* $q(X) = w/(X^n - 1)$*, such that*

$$w = w_1 + \gamma \cdot w_2 + \gamma^2 \cdot w_3$$

(d) $\mathcal{V}$ *sends a random evaluation point* $\zeta \in \mathbb{F}$
(e) $\mathcal{P}$ *replies with* $f(\zeta), f(\zeta\omega), q(\zeta)$
(f) $\mathcal{V}$ *outputs* ***acc*** *if and only if*
  i. $w(\zeta) = q(\zeta) \cdot (\zeta^n - 1)$
  ii. $f(\zeta), f(\zeta\omega), q(\zeta)$ *are the correct evaluations through the verifying process of KZG*

Proof. Completeness is clear by following the protocol.

For knowledge soundness, to make the equation $w(\zeta) = q(\zeta) \cdot (\zeta^n - 1)$ hold, $q(X)$ must exist. That means $w(X)$ is vanishing on H, *i.e.,* w_1, w_2, and w_3 are vanishing over H. Thus, if $f(X)$ does not satisfy any of the equations w_1, w_2, and w_3, $\mathcal{V}$ will detect the proof is invalid. By the binding property of KZG commitment, we know that the evaluations $f(\zeta), f(\zeta\omega)$, and $q(\zeta)$ are correct with overwhelmingly high probability if the KZG verifying is passed.

A.10 KZG Opening with Zero-Knowledge Extension (KZG_{zk})

Although a KZG commitment does not reveal the information of the polynomial directly due to the hiding property, the opening point leaks the evaluation of that polynomial. Assume a malicious verifier sends a different challenge point in each round, which allows him to recover the polynomial after $d + 1$ rounds (d is the degree of the polynomial). The previous works [3,24] mentioned this issue but did not clarify the solution. Here we introduce the solution formally. First, we claim two algorithms to help us explain the solution.

Claim. $\pi_\zeta \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f_1, f_2, \ldots; \zeta)$. This is an algorithm for $\mathcal{P}$ that takes as input polynomials $f_1, f_2, \ldots$ and an evaluation point ζ to output a proof π_ζ to prove the opening evaluations are correct.

Claim. $\{1/0\} \leftarrow \mathsf{KZG}_{zk}.\mathsf{Verify}(\mathcal{C}_{f_1}, \mathcal{C}_{f_2}, \ldots; \pi_\zeta; \zeta)$. This is an algorithm for $\mathcal{V}$ that takes as input the commitments to $f_1, f_2, \ldots$, π_ζ from $\mathsf{KZG}_{zk}.\mathsf{Prove}$, and the evaluation point ζ to verify π_ζ. If π_ζ is valid, it returns 1, else it returns 0.

Now we define KZG_{zk}.

Definition 9 **(KZG_{zk}).** KZG_{zk} *is a variant of KZG commitment scheme such that* $\mathcal{P}$ *is given input* $f \in \mathbb{F}_{<d}[X]$*,* $\mathcal{P}$ *wants to open* f *at* n *random points, where* $n \leq d + 1$*. They run the protocol as follows:*

1. $\mathcal{P}$ *generates* n *random numbers* $x_1, x_2, \ldots, x_n \in \mathbb{F} \setminus H$ *(*H *is the domain of* f*) and another* n *random numbers* $y_1, y_2, \ldots, y_n \in \mathbb{F}$.
2. $\mathcal{P}$ *incrementally interpolates* f *at* n *more points* $x_1, x_2, \ldots, x_n$ *such that*

$$f(x_1) = y_1, f(x_2) = y_2, \ldots, f(x_n) = y_n$$

3. $\mathcal{P}$ *publishes the commitment to* f*,* $\mathcal{C}_f$.
4. $\mathcal{V}$ *sends* n *random evaluation points* $\zeta_1, \zeta_2, \ldots, \zeta_n \in \mathbb{F} \setminus H$.
5. *For each* $\zeta_i, i \in [1, n]$*,* $\mathcal{P}$ *computes* $\pi_{\zeta_i} \leftarrow \mathsf{KZG}_{zk}.\mathsf{Prove}(f; \zeta_i)$ *to open* f *at* ζ_i.
6. $\mathcal{V}$ *outputs* ***acc*** *if and only if* $\mathsf{KZG}_{zk}.\mathsf{Verify}(f; \pi_{\zeta_i}; \zeta)$ *returns 1 for each* $\pi_{\zeta_i}, i \in [1, n]$.

Theorem 2. KZG_{zk} *is complete, sound, and HVZK.*

Proof. Completeness is clear because the new polynomial has the same evaluations as the old one over the domain.

For soundness, note the difference between the variant and the original is we reduce the field of the challenge evaluation point from $\mathbb{F}$ to $\mathbb{F} \setminus H$. The soundness error increases but is still negligible.

To verify zero knowledge, we construct a simulator $\mathcal{S}$. Let $\mathcal{S}$ construct a vanishing polynomial f^* over the same domain and randomly generate $\{x_1^*, x_2^*, \ldots, x_n^*\}$, $\{y_1^*, y_2^*, \ldots, y_n^*\}$ like $\mathcal{P}$, and then incrementally interpolate f^* such that

$$f^*(x_1^*) = y_1^*, f^*(x_2^*) = y_2^*, \ldots, f^*(x_n^*) = y_n^*$$

We can observe when $\mathcal{V}$ interacts with $\mathcal{S}$ to execute the protocol, $\mathcal{V}$ always accepts the proof from $\mathcal{S}$ because f^* has the same roots as f. Given $\{x_1^*, x_2^*, \ldots, x_n^*\}$, $\{y_1^*, y_2^*, \ldots, y_n^*\}$ are chosen uniformly at random each time, that is exactly the same as $\mathcal{P}$ incrementally interpolates f. Thus $\mathcal{V}$ cannot distinguish between the transcript from $\mathcal{S}$ and the transcript from $\mathcal{P}$.

It is worth mentioning to efficiently prove several polynomials are vanishing at several points, there are some batched KZG opening schemes [16,2,15]. Our protocol uses the batched opening scheme from [16] with the zero-knowledge extension to demonstrate how to prove the above range proof efficiently (See Section A.11).

A.11 Range Proof for Single Value with KZG_{zk}

Assume $\mathcal{P}$ is given x and $\mathcal{P}$ computes $f(X)$ using the above range proof (A.9). $\mathcal{P}$ wants to prove $f(X)$ satisfies the second and the third condition, *i.e.,* w_2 and w_3 are vanishing over H. $\mathcal{P}$ and $\mathcal{V}$ run the protocol as follows:

1. *$\mathcal{P}$ generates two random numbers $\omega', \omega'' \in \mathbb{F} \setminus H$ and another two random numbers $\alpha, \beta \in \mathbb{F}$*
2. *$\mathcal{P}$ interpolates f at two more points ω', ω'' such that*

$$f(\omega') = \alpha, f(\omega'') = \beta$$

3. *$\mathcal{P}$ computes w_2 and w_3 following the above range proof and sends the commitment to f, $\mathcal{C}_f$*
4. *$\mathcal{V}$ sends a random challenge $\gamma \in \mathbb{F}$*
5. *$\mathcal{P}$ sends the commitment to $q_w := w/(X^n - 1)$ where*

$$w := w_2 + \gamma \cdot w_3$$

6. *$\mathcal{V}$ sends a random evaluation point $\zeta \in \mathbb{F} \setminus H$*
7. *$\mathcal{P}$ sends the evaluations $f(\zeta), f(\zeta\omega), q_w(\zeta)$*
8. *$\mathcal{P}$ sends the commitments to $q_1(X), q_2(X)$, where*

$$q_1(X) := \frac{f(X) - f(\zeta)}{X - \zeta} + \gamma \cdot \frac{q_w(X) - q_w(\zeta)}{X - \zeta}$$

$$q_2(X) := \frac{f(X) - f(\zeta\omega)}{X - \zeta\omega}$$

9. *$\mathcal{V}$ chooses random $r \in \mathbb{F}$*
10. *$\mathcal{V}$ outputs **acc** if and only if*
    (a) $w_1(\zeta) + \gamma \cdot w_2(\zeta) = q_w(\zeta) \cdot (\zeta^n - 1)$
    (b) $e(F + \zeta \cdot \mathcal{C}_{q_1} + r\zeta\omega \cdot \mathcal{C}_{q_2}, [1]_2) = e(\mathcal{C}_{q_1} + r \cdot \mathcal{C}_{q_2}, [x]_2)$, *where*

$$F := \mathcal{C}_f - [f(\zeta)]_1 + \gamma \cdot (\mathcal{C}_{q_w} - [q_w(\zeta)]_1) + r \cdot (\mathcal{C}_f - [f(\zeta\omega)]_1)$$

Theorem 3. *The above range proof with KZG_{zk} is complete, sound, and HVZK.*

Proof. Completeness follows the protocol. Soundness and zero knowledge can be verified by Plonk (Section 3.1 of [16]) and Theorem 2.

B Protocols

This appendix describes cryptographic protocols that we contribute to the literature, either as novel protocols or as adaptations of existing protocols for our purposes. For space considerations, we are unable to include them in the main body of the paper but describe how they are used in the main body.

B.1 Open KZG with Committed Value (KZG_{cm})

KZG_{zk} allows $\mathcal{P}$ to prove a polynomial is vanishing over a specified domain. However, in some cases, $\mathcal{P}$ needs to prove the claimed evaluation is correct. For example, we construct a polynomial where the evaluation at ω^0 is the total assets we want to prove. Instead of revealing the evaluation directly, we publish the committed value. Now we introduce this opening scheme based on the KZG commitment in Pedersen form.

Definition 10 (KZG_{cm})**.** *KZG_{cm} is a KZG opening scheme that $\mathcal{P}$ is given input $f \in \mathbb{F}_{<d}[X]$. $\mathcal{P}$ and $\mathcal{V}$ run the protocol as follows:*

1. *$\mathcal{P}$ generates a random polynomial $\hat{f}$ with the same degree as f and computes the commitment to f, $\mathcal{C}_f = g_1^{f(\tau)} h_1^{\hat{f}(\tau)}$.*
2. *$\mathcal{V}$ sends a random evaluation point a as challenge.*
3. *$\mathcal{P}$ computes the witness w for a such that*
$$w = g_1^{\psi(\tau)} h_1^{\hat{\psi}(\tau)}$$
where $\psi(x) = \frac{f(X)-f(a)}{X-a}$, $\hat{\psi}(x) = \frac{\hat{f}(X)-\hat{f}(a)}{X-a}$.
4. *$\mathcal{P}$ sends w and $\boldsymbol{C}(b)$ such that $\boldsymbol{C}(b) = g_1^{f(a)} h_1^{\hat{f}(a)}$.*
5. *$\mathcal{V}$ outputs **acc** if and only if*
$$e(\mathcal{C}/\boldsymbol{C}(b), [1]_2) = e(w, [\tau - a]_2)$$

Theorem 4. *KZG_{cm} is complete, sound, and HVZK.*

Proof. Completeness follows the original KZG commitment scheme.

For soundness, KZG_{cm} does not violate the soundness of the original KZG commitment scheme. Recall the computational binding property of Pedersen commitment, which means it is infeasible for $\mathcal{P}$ to compute a b^* such that $f(a) \neq b^*, \mathbf{C}(b) = \mathbf{C}(b^*)$ based on discrete logarithm assumption.

To prove HVZK, let the simulator $\mathcal{S}$ compute

$$\mathcal{C}_{f^*} \xleftarrow{\$} \mathbb{G}_1, a^* \xleftarrow{\$} \mathbb{F}, \mathbf{C}(b^*) \xleftarrow{\$} \mathbb{G}_1, w^* = \mathcal{C}_{f^*}/(\mathbf{C}(b^*) \cdot [\tau - a^*]_1)$$

It is clear that the simulated conversation $(\mathcal{C}_{f^*}, a^*, \mathbf{C}(b^*), w^*)$ is always accepted by $\mathcal{V}$. We can observe that $\mathcal{C}_{f^*}, a^*, \mathbf{C}(b^*)$ are independent, uniformly distributed over their own field, and w^* is determined by $\mathcal{C}_{f^*}/(\mathbf{C}(b^*) \cdot [\tau - a^*]_1)$. Thus, the simulated conversation has the same distribution as the output of $\mathcal{P}$.

We claim two algorithms representing the step 4 and 5 in the above opening scheme, respectively.

Claim. $\mathbf{C}(b) \leftarrow \mathsf{KZG}_{cm}.\mathsf{Prove}(f; b; a)$. This is an algorithm for $\mathcal{P}$ that takes as input a polynomial f and an evaluation point a to output the committed evaluation of $b = f(a)$.

Claim. $\{1/0\} \leftarrow \mathsf{KZG}_{cm}.\mathsf{Verify}(\mathcal{C}_f; \mathbf{C}(b); a)$. This is an algorithm for $\mathcal{V}$ that takes as input the commitment to f, the committed evaluation $f(a)$, and the point a to verify the committed value. If $\mathbf{C}(b)$ is the correct evaluation at a, it returns 1, else it returns 0.

B.2 The Protocol ZKPoK in π_{keys}

Protocol 6 presents how to map the *success* or *failure* in `secp256k1` to `bls12-381`.

B.3 Range Proof for Multiple Values

Protocol 7 presents how to prove multiple values satisfy the range proof based on binary decomposition.

$\mathcal{P}$ and $\mathcal{V}$ are both given $\{\mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i})\}$. $\mathcal{P}$ has the access to $\{\mathsf{sk}_i\}, \{A_{\mathsf{keys},i}\}$ and the hiding factor of $\mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i})$, r_i.

1. Case 1: $A_{\mathsf{keys},i} = 1$ ($\mathcal{P}$ claims knowledge of sk_i)
   (a) $\mathcal{P}$ selects $e_1 \xleftarrow{\$} \{0,1\}^t; z_3, \beta \xleftarrow{\$} \mathbb{Z}_b; \alpha \xleftarrow{\$} \mathbb{Z}_s$
   (b) $\mathcal{P}$ publishes $t_1 = g_s^{\alpha}$
   (c) $\mathcal{P}$ publishes $t_2 = h_b^{\beta}$
   (d) $\mathcal{P}$ publishes $t_3 = g_b^{-e_1} h_b^{z_3 - r_i e_1}$
   (e) $\mathcal{V}$ publishes t-bit challenge $e \xleftarrow{\$} \{0,1\}^t$ (or $\mathcal{P}$ via Fiat-Shamir)
   (f) $\mathcal{P}$ computes $e_0 = e \oplus e_1$ and publishes e_0 and e_1
   (g) $\mathcal{P}$ publishes $z_1 = e_0 \mathsf{sk}_i + \alpha$
   (h) $\mathcal{P}$ publishes $z_2 = e_0 r_i + \beta$
   (i) $\mathcal{P}$ publishes z_3
2. Case 2: $A_{\mathsf{keys},i} = 0$ ($\mathcal{P}$ does not claim knowledge of sk_i)
   (a) $\mathcal{P}$ selects $e_0 \xleftarrow{\$} \{0,1\}^t; z_1 \xleftarrow{\$} \mathbb{Z}_s; z_2, \alpha \xleftarrow{\$} \mathbb{Z}_b$
   (b) $\mathcal{P}$ publishes $t_1 = g_s^{z_1} / \mathsf{pk}_i^{e_0}$
   (c) $\mathcal{P}$ publishes $t_2 = g_b^{e_0} h_b^{z_2 - r_i e_0}$
   (d) $\mathcal{P}$ publishes $t_3 = h_b^{\alpha}$
   (e) $\mathcal{V}$ publishes t-bit challenge $e \xleftarrow{\$} \{0,1\}^t$ (or $\mathcal{P}$ via Fiat-Shamir)
   (f) $\mathcal{P}$ computes $e_1 = e \oplus e_0$ and publishes e_0 and e_1
   (g) $\mathcal{P}$ publishes z_1
   (h) $\mathcal{P}$ publishes z_2
   (i) $\mathcal{P}$ publishes $z_3 = e_1 r_i + \alpha$
3. $\mathcal{V}$ outputs **acc** if and only if
   (a) $e = e_0 \oplus e_1$
   (b) $g_s^{z_1} = \mathsf{pk}_i^{e_0} t_1$
   (c) $g_b^{e_0} h_b^{z_2} = \mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i})^{e_0} t_2$
   (d) $h_b^{z_3} = \mathbf{C}_{\mathtt{bls}}(A_{\mathsf{keys},i})^{e_1} t_3$

Protocol 6: The ZKPoK proof demonstrates that $\mathcal{P}$ can prove knowledge of a secret key with the correct committed selector.

C Security Proofs for Xiezhi's Components

C.1 Definitions

Definitions 11 and 12 are taken largely verbatim from the Provisions paper at CCS 2015 [9]. Let $\mathcal{A}$ (exchange-controlled addresses) and $\mathcal{A}'$ (anonymity set of addresses) denote mappings $(y = g^x) \mapsto \mathrm{bal}(y)$ where $\mathcal{A} \subseteq \mathcal{A}'$, y is the public key corresponding to a Bitcoin address with private key x and $\mathrm{bal}(y)$ is the amount of currency, or assets, observably spendable by this key on the blockchain. Let $\mathcal{L}$ denote a mapping $\mathfrak{uid} \mapsto \ell$ where ℓ is the amount of currency, or liabilities, owed by the exchange to each user identified by the unique identity $\mathfrak{uid}$. A balance is a positive integer in $[0, \mathsf{MaxETH}]$ for a known upper-bound MaxETH. The size of $\mathcal{A}'$ is known, the size of $\mathcal{A}$ is generally unknown (beyond being less than or equal to $\mathcal{A}'$), and the size of $\mathcal{L}$ is generally unknown (see Definition 12(4) below).

Definition 11 (Valid Pair). *We say that $\mathcal{A}$ and $\mathcal{L}$ are a valid pair with respect to a positive integer* MaxETH *iff* $\forall \mathfrak{uid} \in \mathcal{L}$,

– $\sum_{y \in \mathcal{A}} \mathcal{A}[y] - \sum_{\mathfrak{uid} \in \mathcal{L}} \mathcal{L}[\mathfrak{uid}] \geq 0$
– $0 \leq \mathcal{L}[\mathfrak{uid}] \leq \mathsf{MaxETH}$

Consider an interactive protocol ProveSolvency run between an exchange $\mathcal{E}$ and user $\mathcal{U}$ such that

1. $\mathcal{P}$ takes as input some values to be proved, $\{x_1, x_2, \ldots, x_\mu\}$
2. $\mathcal{P}$ computes the binary decomposition (from most significant bit to least significant bit) of each balance, $\{z_j^{(x_i)}\}_{i\in[\mu],j\in[k]}$, such that $z_j^{(x_i)} \in \{0,1\}$ and $x_i = \sum_{j=k}^{0} 2^j \cdot z_j^{(x_i)}$.
3. $\mathcal{P}$ puts the bits into accumulator form where $\chi_k^{(x_i)} = z_k^{(x_i)}$ and $\chi_i^{(x_i)} = 2\chi_{i+1}^{(x_i)} + z_i^{(x_i)}$. (Remark: visualized as a matrix, each row is a balance where the k-th column is the least significant bit and, moving right-to-left, each bit is folded in until it accumulates to x_j in the first column.)

$$\begin{bmatrix} \chi_1^{(x_1)} & \chi_2^{(x_1)} & \chi_3^{(x_1)} & \cdots & \chi_k^{(x_1)} \\ \chi_1^{(x_2)} & \chi_2^{(x_2)} & \chi_3^{(x_2)} & \cdots & \chi_k^{(x_2)} \\ \chi_1^{(x_3)} & \chi_2^{(x_3)} & \chi_3^{(x_3)} & \cdots & \chi_k^{(x_3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \chi_1^{(x_\mu)} & \chi_2^{(x_\mu)} & \chi_3^{(x_\mu)} & \cdots & \chi_k^{(x_\mu)} \end{bmatrix}$$

4. Due to the concrete parameters of `bls12-381`, $\mathcal{P}$ will work column-by-column (proof size and verifier time will be linear in k which is the bit-precision of each account). Let column j be vector $\overline{p}_j = \{\chi_j^{(x_1)}, \chi_j^{(x_2)}, \ldots, \chi_j^{(x_\mu)}\}$. The following constraints apply (for $i \in [\mu], j \in [k]$): $\overline{p}_1[i] = x_i$; $\overline{p}_j[i] - 2 \cdot \overline{p}_{j+1}[i] \in \{0,1\}$; and $\overline{p}_k[i] \in \{0,1\}$. $\overline{p}_1$ contains $\{x_1, x_2, \ldots, x_\mu\}$.
5. $\mathcal{P}$ interpolates polynomials for $\overline{p}_j \to p_j(X)$ and publishes commitments to each.
6. $\mathcal{P}$ shows the following polynomials are vanishing for all $x \in H$ where $H = \{\omega^0, \omega^1, \ldots, \omega^{k-1}\}$

$$\begin{aligned} v_1 &:= [p_1(X) - 2p_2(X)] \cdot [1 - (p_1(X) - 2p_2(X))] \\ v_2 &:= [p_2(X) - 2p_3(X)] \cdot [1 - (p_2(X) - 2p_3(X))] \\ &\vdots \\ v_{k-1} &:= [p_{k-1}(X) - 2p_k(X)] \cdot [1 - (p_{k-1}(X) - 2p_k(X))] \\ v_k &:= p_k(X) \cdot [1 - p_k(X)] \end{aligned}$$

   $\{v_1, v_2, \ldots, v_k\}$ prove each liability is greater than or equal to 0 (range proof). To complete the proof, $\mathcal{P}$ and $\mathcal{V}$ run KZG_{zk} to open $(p_1, p_2, \ldots, p_k)$ at a random evaluation point.
7. $\mathcal{V}$ outputs **acc** if and only if
   (a) each evaluation is valid
   (b) $\{v_1, v_2, \ldots, v_k\}$ are vanishing on H

Protocol 7: The range proof for multiple values demonstrates that each value is either zero or a positive number less than a specified value.

– $\mathsf{output}_{\mathcal{E}}^{\mathsf{ProveSolvency}}(1^k, \mathsf{MaxETH}, \mathcal{A}, \mathcal{L}, \mathcal{A}') = \emptyset$
– $\mathsf{output}_{\mathcal{U}}^{\mathsf{ProveSolvency}}(1^k, \mathsf{MaxETH}, \mathcal{A}', \mathfrak{uid}, \ell) \in \{ACCEPT, REJECT\}$

For brevity, we refer to these as $\mathrm{out}_{\mathcal{E}}$ and $\mathrm{out}_{\mathcal{U}}$ respectively. Next we define, with reference to the valid pair definition, a privacy-preserving proof of solvency.

Definition 12 (Privacy-Preserving Proof of Solvency). *A privacy-preserving proof of solvency is a probabilistic polynomial-time interactive protocol ProveSolvency, with inputs/outputs as above, such that the following properties hold:*

1. *Correctness. If $\mathcal{A}$ and $\mathcal{L}$ are a valid pair and $\mathcal{L}[\mathfrak{uid}] = \ell$, then* $\Pr[out_{\mathcal{U}} = ACCEPT] = 1$.
2. *k-Soundness. If $\mathcal{A}$ and $\mathcal{L}$ are instead not a valid pair, or if $\mathcal{L}[\mathfrak{uid}] \neq \ell$, then* $\Pr[out_{\mathcal{U}} = REJECT] \geq 1 - \mathsf{negl}(k)$.

3. *Ownership. For all valid pairs $\mathcal{A}$ and $\mathcal{L}$, if $\Pr[out_{\mathcal{U}} = ACCEPT] = 1$, then the exchange must have 'known' the private keys associated with the public keys in $\mathcal{A}$; i.e., there exists an extractor that, given $\mathcal{A}$, $\mathcal{L}$, and rewindable black-box access to exchange $\mathcal{E}$, can produce x for all $y \in \mathcal{A}$.*
4. *Privacy. A potentially dishonest user $\mathcal{U}'$ interacting with an honest exchange $\mathcal{E}$ cannot learn anything about a valid pair $\mathcal{A}$ and $\mathcal{L}$ beyond its validity and $\mathcal{L}[\mathsf{ui\partial}]$ (and possibly $|\mathcal{A}|$ and $|\mathcal{L}|$); i.e., even a cheating user cannot distinguish between an interaction using the real pair $\mathcal{A}$ and $\mathcal{L}$ and any other (equally sized) valid pair $\hat{\mathcal{A}}$ and $\hat{\mathcal{L}}$ such that $\hat{\mathcal{L}}[\mathsf{ui\partial}] = \mathcal{L}[\mathsf{ui\partial}]$.*

C.2 Theorems

Theorem 5. *A Σ-protocol for relation* ZKPoK$\{(\mathsf{sk}_i, s_i) : [\mathsf{pk}_i = g^{\mathsf{sk}_i} \wedge \boldsymbol{C_{bls}}(s_i) = \boldsymbol{C_{bls}}(1)] \vee \boldsymbol{C_{bls}}(s_i) = \boldsymbol{C_{bls}}(0)\}$ *exists which is complete, has special soundness and is special HVZK.*

Proof. To demonstrate completeness, consult Protocol 6.

To demonstrate special soundness, let two accepting conversations between $\mathcal{P}$ and $\mathcal{V}$

$$(t_1, t_2, t_3, e, e_0, e_1, z_1, z_2, z_3), (t_1, t_2, t_3, e', e'_0, e'_1, z'_1, z'_2, z'_3) \text{ with } e \neq e'$$

be given. It is obvious for some $s = 0$ or 1 and $e_s \neq e'_s$, we can compute sk_i, s_i from the above conversations. Thus, the Σ-protocol for the relation ZKPoK has special soundness.

To demonstrate special HVZK, given e and randomly choose e_0, e_1 such that $e = e_1 \oplus e_2$, let the simulator compute

$$z_1 \xleftarrow{\$} \mathbb{Z}_s, z_2 \xleftarrow{\$} \mathbb{Z}_b, z_3 \xleftarrow{\$} \mathbb{Z}_b, t_1 = g_s^{z_1}/\mathsf{pk}_i^{e_0}, t_2 = g_b^{e_0} h_b^{z_2}/p_i^{e_0}, t_3 = h_b^{z_3}/p_i^{e_1}$$

and output $(t_1, t_2, t_3, e, e_0, e_1, z_1, z_2, z_3)$. Clearly, the transcript is accepted by $\mathcal{V}$. Note that e, e_0, and e_1 are random t-bit strings, which means they have the same distribution as the conversation between $\mathcal{P}$ and $\mathcal{V}$. z_1, z_2, and z_3 are uniformly distributed over their corresponding fields; moreover, given $(e_1, e_2, z_1, z_2, z_3)$, (t_1, t_2, t_3) are uniquely determined by the above equations. Therefore, the simulated transcript is not distinguishable from the real one to $\mathcal{V}$.

Corollary 1. *A Σ-protocol for relation* ZKPoK$\{(\mathsf{sk}_i, s_i) : [\mathsf{pk}_i = g^{\mathsf{sk}_i} \wedge C_{s_i} = \boldsymbol{C_{bls}}(1)] \vee C_{s_i} = \boldsymbol{C_{bls}}(0)\}$ *exists which is a non-interactive zero knowledge proof (NIZKP).*

Proof. Given the relation can proven with a "standard" Σ-protocol (Theorem 5), we can use the well-known Fiat-Shamir heuristic to compile it to a NIZKP in the random oracle model. We do not repeat the proof for this (see [10,21]) but stress that strong Fiat-Shamir [1] needs to be used here and in the Poly-IOP components of Xiezhi, or practical attacks could be leveraged against the system (*cf.* [11]).

Theorem 6. *A polynomial protocol with the zero-knowledge extension KZG_{zk} is complete, has knowledge soundness in the algebraic group model, and is HVZK.*

Proof. Completeness is clear: for an honest $\mathcal{P}$, the evaluations of polynomials are correct and the relations also hold. Thus, $\mathcal{V}$ will always accept the proofs.

We argue the knowledge soundness from two aspects: the evaluations and the relations. The binding property of KZG commitment tells us the probability that any invalid evaluation passes the verifying is negligible, which means $\mathcal{A}$ can win the first condition of the attack game with extremely low probability. By the Schwartz-Zippel lemma, the equation defined by a relation $\mathcal{R}$ has overwhelmingly low probability to hold if the evaluation at a random point does not satisfy the equation. Therefore, the knowledge soundness is proved.

Since $\mathcal{V}$ only knows the commitments to the polynomials and the witnesses and the opening evaluations, the commitments leak no information of the polynomials and the witnesses because of the hiding property of KZG commitment. By Theorem 2, the opening scheme is HVZK. Thus, the polynomial protocol with KZG_{zk} is HVZK.

C.3 Corollaries

Corollary 2. π_{keys} *is complete, sound, and HVZK.*

Proof. Recall that the π_{keys} argument contains the relation proven to be complete, sound, and HVZK in Theorem 5. It remains to be shown the rest of the protocol (Protocol 1) is secure.

Completeness follows from Protocol 1. The remainder of the protocol involves $\mathcal{P}$ demonstrating that the selector polynomial encodes a 1 at index ω^i if and only if the corresponding i-th run of the Σ-protocol used $s = 1$, and contains a 0 otherwise.

For π_{keys} to be sound, it requires (i) the polynomial commitment scheme (PCS) to be binding and (ii) the PCS to have a sound point-evaluation argument. These two properties are both demonstrated for KZG in the original paper [18]. Specifically these two properties rely on four assumptions:

- **KZG.A1**: The trusted setup outputs a structured reference string **srs** with the value of τ unknown to $\mathcal{P}$.
- **KZG.A2**: The value of τ cannot be extracted from **srs** which assumes $\mathcal{P}$ is computationally bounded and relies on (for us in `bls12-381`) the t-strong Diffie-Hellman (t-SDH) assumption.
- **KZG.A3**: If an adversary interpolates a polynomial through the point (ω^i, y) such that $y = f(\omega^i)$ but claims $y' = f(\omega^i)$ for some $y' \neq y$ then the probability that $\tau - y'$ evenly divides $y = f(\tau)$ is overwhelmingly low. This property can be demonstrated using the Schwartz-Zippel lemma by showing the number of τ values satisfying this property is bounded from above by d/q where d is the degree of the polynomial and q is the size of the exponent group. For `bls12-381` with 255-bit exponents and 2-adicity of 32, this is close to $2^{32-255} = 2^{-233}$ which is negligible.

Finally, in our protocol $\mathcal{P}$ does not reveal the evaluation of the polynomial at a point, $\mathcal{P}$ instead reveals a commitment to the evaluation through KZG_{cm}. In the original KZG opening scheme, $\mathcal{P}$ opens the commitment to the polynomial first and then opens the evaluation at the challenge point. Our modification just moves the computation work for the committed evaluation from $\mathcal{V}$ to $\mathcal{P}$. By Theorem 4, KZG_{cm} is HVZK. Therefore, π_{keys} has zero knowledge.

For future claims, we encapsulate all assumptions about KZG as a *polynomial oracle*.

Corollary 3. π_{assets} *is complete, has knowledge soundness in the algebraic group model, and is HVZK.*

Proof. Clearly, π_{assets} is a polynomial protocol for two polynomial relations (i) $f_{\mathsf{assets}}(X) - f_{\mathsf{assets}}(X\omega) = f_{\mathsf{bal}}(X) \cdot f_{\mathsf{sel}}(X), X \neq \omega^{n-1}$ and (ii) $f_{\mathsf{assets}}(\omega^{n-1}) = f_{\mathsf{bal}}(\omega^{n-1}) \cdot f_{\mathsf{sel}}(\omega^{n-1})$. The first relation proves the starting values are the same, and the second proves each successive value in the accumulative vector adds its adjacent value with the corresponding value. To check the relations, π_{assets} leverages KZG_{zk} to open $f_{\mathsf{assets}}, f_{\mathsf{bal}}, f_{\mathsf{sel}}$ at a random evaluation point ζ and f_{assets} at $\zeta\omega$. Additionally, to complete the PoA proof, π_{assets} publishes the evaluation of $f_{\mathsf{assets}}(\omega^0)$ through KZG_{cm}. We already analyzed the security of KZG_{cm} in Theorem 4. Thus, π_{assets} is complete, knowledge sound, and HVZK by Theorem 6.

Corollary 4. $\pi_{\mathsf{liabilities}}$ *is complete, has knowledge soundness in the algebraic group model, and is HVZK.*

Proof. Completeness follows from Protocol 3.

Given a *polynomial oracle*, $\mathcal{P}$ commits to a set of integers in binary form and builds a vector to accumulate the bits into the integer representation (call this the range accumulator). Knowledge soundness of this aspect follows from the knowledge soundness of the range proof by Lemma 1 which uses the *polynomial oracle* to demonstrate three constraints: that the range accumulator starts with a 0 or 1; that the binary relationship between adjacent bits in the range accumulator are 0 or 1; and that the header of the range accumulator matches a standalone commitment to the integer (we do not use this, we just use the header values directly from $p_1(X)$). To complete soundness, $\mathcal{V}$ must check that no more than k bits are used for an integer in $[0, k)$. Outside of the range proof, $\mathcal{P}$ builds a vector ($f_{\mathsf{liab}}(X)$) to accumulate the sum of each header value ($p_1(X)$) from the set of range accumulators for each user account. This is the same protocol as in π_{assets}.

For HVZK, it remains to consider what evaluation points are leaked by $\pi_{\mathsf{liabilities}}$. To check the constraints ($\{w_1, w_2\}$ and $\{v_1, v_2, \ldots, v_k\}$), $\mathcal{P}$ and $\mathcal{V}$ run KZG_{zk} to open $f_{\mathsf{liab}}, p_1, p_2, \ldots, p_k$ at a random evaluation point ζ and f_{liab} at $\zeta\omega$. Again, we already analyzed the security of KZG_{zk} in Theorem 2. Thus, $\pi_{\mathsf{liabilities}}$ is HVZK.

Corollary 5. π_{users} *is complete, has knowledge soundness in the algebraic group model, and is HVZK.*

Proof. Completeness follows from Protocol 4.

Knowledge soundness follows directly from the *polynomial oracle* which, for π_{users}, opens two points at the same index on two polynomials—one demonstrates the user's balance and one demonstrates the user's identification. The sufficiency of this to bind the balance to the user ID is already proven in Provisions which uses the same mechanism (for a different commitment scheme). The KZG assumptions already addressed in Corollary 2 cover the rest.

To verify HVZK, recall the properties of KZG commitments—seeing a polynomial commitment and an opening at a specific evaluation point reveals no further information about any other point on the polynomial. KZG does not reveal the degree of the polynomial, which would provide the number of users of the exchange, but an upperbound exists in the size of **srs** from the trusted setup (and if it can be assumed the prover will act efficiently, the largest root of unity). For each user, a p.p.t simulator can be constructed such that the evaluation at the user's index is equal to the user's balance/identification while other evaluations are random numbers. As a user (the verifier), he cannot distinguish between the simulated transcript and the real one due to the hiding property of KZG commitment.

Corollary 6. π_{solvency} *is complete, sound, and HVZK.*

Proof. Completeness follows from Protocol 5. The soundness of the argument is that $f_{\mathsf{assets}}(\omega^0)$ is sound under Corollary 3, $f_{\mathsf{liab}}(\omega^0)$ is sound under Corollary 4, and $f_{\mathsf{eq}}(\omega^0)$ is zero or positive by the soundness of the range proof (as addressed in Corollary 4). The overall constraint demonstrates that the total assets equal or exceed the total liabilities. HVZK similarly follows from the same previous corollaries (3, 4, and range proof).