Short Privacy-Preserving Proofs of Liabilities

Francesca Falzon
ffalzon@ethz.ch
ETH Zürich

Kaoutar Elkhiyaoui
kao@zurich.ibm.com
IBM Research - Zürich

Yacov Manevich
yacov.manevich@ibm.com
IBM Research - Zürich

Angelo De Caro
adc@zurich.ibm.com
IBM Research - Zürich

ABSTRACT

In the wake of fraud scandals involving decentralized exchanges and the significant financial loss suffered by individuals, regulators are pressed to put mechanisms in place that enforce customer protections and capital requirements in decentralized ecosystems. Proof of liabilities (PoL) is such a mechanism: it allows a prover (e.g., an exchange) to prove its liability to a verifier (i.e., a customer).

This paper introduces a fully privacy-preserving PoL scheme with *short* proofs. We store the prover's liabilities in a novel data structure, the *sparse summation Verkle tree* (SSVT), in which each internal node is a hiding vector commitment of its children and whose root commits to the sum of all the leaves in the tree. We leverage *inner product arguments* to prove that a user's liability is included in the total liabilities of the prover without leaking any information beyond the liability's inclusion. Our construction yields proofs of size $O(\log_n N)$ where n is the arity of the SSVT and N is an upper bound on the number of users. Additionally, we show how to further optimize the proof size using aggregation. We benchmark our scheme using an SSVT of size 2^{256} and one of size 10^9 that covers the universe of all US social security numbers.

KEYWORDS

proof of liabilities, central bank digital currency, cryptocurrency

1 INTRODUCTION

Driven by the promises of decentralization, anonymity, and high returns, there has been a surge in the popularity of *cryptocurrencies* such as Bitcoin [1] in recent years. The increasing demand for cryptocurrencies has led to the rise of decentralized exchanges, marketplaces in which users can purchase, sell, and trade cryptocurrencies. While exchanges offer a user-friendly interface for lay users to manage their funds, the exchanges are the de-facto custodians of these funds. This makes users vulnerable to acts of fraud, the most recent of which is FTX, which is estimated to owe over 8 billion USD to its creditors and customers [2].

To counter such fraud, users should be assured of exchange solvency. **Exchange insolvency** occurs when the total liabilities of the exchange exceeds its total assets, thereby resulting in a negative net worth and a higher risk of bankruptcy. To limit the risk of insolvency, periodic financial audits can be leveraged. Audits are traditionally conducted by an auditing firm in a centralized manner: the firm reviews the records of the exchange and ensures that they are representative of its financial dealings [3]. Centralized audits, however, carry multiple downsides including cost, time, and lack of privacy, but most importantly is the fact that the auditor can never be certain that all the liabilities have been included.

Decentralized audits, on the other hand, allow users to check that the liability owed to them by an exchange is included in the total liabilities and that the exchange has enough assets to cover them. As long as the users conduct an audit, they can be assured that their liabilities are accounted for. The utility of decentralized audits is not limited to cryptocurrency exchanges: another prominent application is *Central bank digital currencies (CBDCs)*. In CBDCs, commercial banks serve as custodians and hold CBDC on behalf of users. Given that CBDC is a central-bank money, commercial banks should guarantee that users can redeem their CBDC at any time. A decentralized audit allows a user to verify that their CBDC can be redeemed without actually withdrawing it.

A **Proof of liability (PoL)** [4] is a cryptographic primitive that enables a prover to publish its total liabilities and then efficiently prove to each of its users that their corresponding liability is included in the published total. A **Proof of Reserves (PoR)** is a cryptographic primitive that enables a prover to prove that it possesses the assets that they claim. PoL and PoR together form a **Proof of Solvency**, a protocol that enables a prover to prove that its assets are at least as much as the liabilities it owes to its clients.

Considering the growing importance of data privacy and the ensuing regulations, it is imperative to design PoL and PoR schemes that can fully protect the privacy of individual users and provers (i.e, banks, exchanges). Along these lines, we find [5–10], which propose PoL solutions with varying degrees of privacy. While these solutions protect the privacy of user data, most fall short of protecting the prover's privacy, as they leak the number of users. Ji and Chalkias [10] manage to hide the number of users but at the expense of a larger proof size.

We propose a fully privacy-preserving PoL scheme with short proofs. One of the challenges of designing a PoL scheme is guaranteeing privacy to the prover and verifiers, while remaining efficient. Most prior schemes (e.g. [4–6, 9, 10]) rely on the Merkle-tree, a binary tree whose inner nodes contain a hash to their respective children. Though asymptotically efficient, Merkle trees are limited by their 2-ary structure and covering a user universe of intractable size can quickly result in impractically long proofs.

To overcome this, we introduce the **Sparse Summation Verkle Tree (SSVT)** to hide the number of users while shortening the authentication path. An SSVT is a modified sparse Merkle tree in which each internal node contains vector commitments to its children and whose root contains a commitment to the sum of all the leaves in the tree. The size of the tree (for the given depth and arity) may be intractable, yet the tree can be represented compactly by only storing the nodes along the paths from leaves to root.

By relying on vector commitments, the tree can have arity $n \geq 2$ and short inclusion proofs. Our scheme also benefits from the short proofs and efficient openings of the vector commitments. We combine the hiding vector commitments in [11] with inner product arguments [12] to design tailored zero-knowledge proofs that allow us to show that the liability to a user is included in the tree in a privacy-preserving manner. We implement our scheme in Go and evaluate our scheme using an SSVT of size 2^{256} and one of size 10^9 covering the entire universe of possible US social

security numbers. Our experiments show that the performance of our scheme is dominated by the generation and verification of the range proof, which can fortunately be optimized through proof aggregation (see Section 7.3) and multi-exponentiation to reduce the number of exponentiations [12].

We summarize our contributions as follows:

- We describe a fully private PoL scheme with short proofs.
- We introduce a new data-structure, the sparse summation Verkle tree (SSVT), which is a modified Merkle tree whose internal nodes contain vector commitments to their respective children and whose root node contains a commitment to the values of the leaves of the tree. SSVTs support arity $n \geq 2$, thus hiding the total number of users while ensuring short authentication paths .
- We give a formal security and privacy analysis. We prove that our scheme offers maximum privacy to both the prover and the verifiers, leaking only the size of the user universe.
- We describe three new tailor-made and asymptotically-efficient instantiations of zero-knowledge proofs over vector commitments, including an extension of Bulletproofs [12] to support range proofs over vector commitments. We further extend these protocols to support the batching of proofs.
- We support our theoretical results with a prototype implementation and evaluation on SSVTs of size 2^{256} and 10^9.

1.1 Related Work

We now give an overview of the prior work on PoL. The complexity and functionality of the PoL schemes is summarized in Table 1.

Maxwell. This scheme [4] uses a summation Merkle tree to generate proofs of inclusion. Each leaf corresponds to a user and contains the plaintext balance of that user; each inner node is assigned the hash of the *sum* of the values of its children. The scheme, however, leaks the total liabilities (which is published in the clear), the number of users (which can be inferred from the height of the tree), and the balance of the sibling node (when producing the inclusion proofs). Moreover, the scheme was proved insecure [5].

Maxwell+ and Maxwell++. Hu et al. [5] describe an attack on the Maxwell scheme that enables a malicious prover to claim a total liability that is as small as the largest liability owed to a user. They then introduce a new scheme (which we refer to as Maxwell+) that mitigates this attack by modifying the Maxwell scheme as follows. Each inner node is assigned the hash of the sum of the children's values concatenated with the hashes of the children. This scheme still leaks the total liabilities, the number of users, and the balance of the sibling nodes on the authentication path.

Chalkias et al. [6] present the Maxwell++ scheme that modifies the Maxwell+ scheme by randomly splitting each user's balance and then mapping each fraction to a different leaf in the Merkle tree. This somewhat hides the number of users and their balances at the expense of larger proof size and longer verification time.

Camacho. Camacho's scheme [7] modifies the Maxwell scheme [4] by replacing the plaintext value of each node in Maxwell with a Pedersen commitment to the value. Let $\mathsf{Commit}(\ell, r)$ denote a commitment to value ℓ with blinding factor r. By the homomorphism property of Pedersen commitments, one can compute the sum of the two child values by multiplying the commitments, i.e. $\mathsf{Commit}(\ell_0, r_0) \cdot \mathsf{Commit}(\ell_1, r_1) = \mathsf{Commit}(\ell_0 + \ell_1, r_0 + r_1)$. Accordingly, a proof of inclusion for a particular user, corresponds to the commitments in the authentication path and the opening of the commitment at the root. The authentication path is valid if the commitment at each node in the path is the product of the commitments of its children. To ensure that the liabilities along the authentication path are not negative, Camacho uses range proofs. However, since each node stores only a commitment to the sum of the children, the Camacho scheme suffers from the same security flaw as the original Maxwell scheme. That is, a malicious prover can claim smaller total liabilities equaling the largest liability owed to a single user.

Provisions. Provisions [8] uses a combination of Pedersen commitments and zero knowledge proofs to construct a proof of solvency. Provisions follows a flat-list approach, in which the prover publishes a list of Pedersen commitments of both the assets and individual liabilities. It hides the number of users by adding dummy commitments to the published list. The proof of assets scales linearly with the size of the public keys associated with the prover, whereas the proof of liabilities scales linearly with the number of users. It is possible for Provisions to leverage Merkle trees to reduce the size of the published data from linear in the number of users to constant. However, an upper bound of the total number of users is still exposed, and a logarithmic number of range proofs are required, which can be expensive for both the prover and the users, even when using efficient ZKP schemes such as Bulletproofs [12].

DAPOL. DAPOL [9] enhances the Camacho scheme [7] by using a sparse Merkle tree (SMT) and the fix proposed in [5] i.e., a node in the tree additionally comprises the hash of its children. The SMT helps hide the total number of users; each user is randomly assigned to a node in the bottom layer, and the SMT is then built from the bottom layer up, ensuring that each node has either zero or two children. Nodes that are not mapped to a user or nodes without children are called padding nodes. As Ji and Chalkias [10] note, the hash stored in the padding node is deterministic, making it possible for participants that request proofs of liability to distinguish between padding nodes and other nodes and, consequently, bound the total number of users.

DAPOL+. In [10], Ji and Chalkias introduce DAPOL+, which fixes the padding issue in DAPOL. Notably, the padding nodes are now each assigned a Pedersen commitment to 0 and a hash of the node index concatenated with a mask. Additionally, Ji and Chalkias formalize the notion of PoL security and privacy, and show that DAPOL+ satisfies these definitions.

Both DAPOL and DAPOL+ use SMTs as the underlying data structure. In SMTs that are not sufficiently large, one must be careful when assigning users to leaves since the probability of randomly assigning two users to the same node is non-negligible. To get around this issue, both schemes introduce complex assignment procedures such as verifiable random functions (DAPOL) or ORAM-based SMTs (DAPOL+). Proving privacy would require further arguing that the order in which the users are inserted does not affect the resulting tree. On the other hand, if the SMTs are of intractable size, then the schemes would suffer from longer authentication paths and verification time (since the trees are only 2-ary).

Table 1: A summary of prior PoL schemes and our scheme. Let N be the size of the user universe and k be the number of existing users. For Maxwell++ [6] s denotes the average number of units that each user is split into and for Provisions [8] b denotes the size of the anonymity set. By default, a $\log$ term without a base is in base 2. We denote by n the arity of the tree. With respect to privacy, ○ indicates that the property is leaked, ◐ that it is partially hidden, and ● that it is hidden.

	Security	Privacy			Complexity			
	Vulnerability on Scheme	Total Liab.	Total Users	User Liab.	User Inclusion Proving Time	User Inclusion Verification Time	Proof Size of User Inclusion	Commitment Size on PBB
Maxwell [4]	[5]	○	○	○	$O(1)$	$O(\log k)$	$O(\log k)$	$O(1)$
Maxwell+ [5]	–	○	○	○	$O(1)$	$O(\log k)$	$O(\log k)$	$O(1)$
Maxwell++ [6]	–	○	◐	◐	$O(s)$	$O(\log(k \cdot s))$	$O(\log(k \cdot s))$	$O(1)$
Camacho [7]	[5]	●	○	●	$O(\log k)$	$O(\log k)$	$O(\log k)$	$O(1)$
Provisions [8]	–	●	◐	●	$O(k+b)$	$O(k+b)$	$O(1)$	$O(k+b)$
DAPOL [9]	[10]	●	◐	●	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(1)$
DAPOL+ [10]	–	●	●	●	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(1)$
This Work	–	●	●	●	$O(\log_n N)$	$O(\log_n N)$	$O(\log_n N)$	$O(1)$

1.2 Notation

Let $[n]$ denote the set of integers $\{0, 1, \ldots, n-1\}$ and $[m..n]$ the set of integers $\{m, m+1, ..., n-1\}$. For some set S, let $x \leftarrow\$\ S$ denote the random sampling of an element x from S.

Given a map M, the set of corresponding keys K, and a set $S \subseteq K$, $M[S]$ denotes the restriction of M to S, i.e., the map with keys in S.

We take $\mathbb{G}$ to be a cyclic group of prime order p and $\mathbb{Z}_p$ to be the ring of integers modulo p. Lower-case letters are reserved for elements in $\mathbb{Z}_p$, whereas upper-case letters denote elements in $\mathbb{G}$. Let $\mathbb{G}^n$ and $\mathbb{Z}_p^n$ be the vector spaces of dimension n over $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. We denote vectors using bold font, e.g. $\mathbf{v} \in \mathbb{Z}_p^n$ represents an n-dimensional vector with elements in $\mathbb{Z}_p$.

Given a vector $\mathbf{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_p^n$, we use $\mathbf{v}[i]$ to denote the element at index i (i.e., $\mathbf{v}[i] = v_i$), $\mathbf{v}[:m]$ to denote sub-vector $(\mathbf{v}_0, ..., \mathbf{v}_{m-1})$, and $\mathbf{v}[m:]$ to denote sub-vector $(\mathbf{v}_m, ..., \mathbf{v}_{n-1})$ for integer $m < n$. More generally, for any ordered set $S = \{i_1, ..., i_m\} \subseteq [n]$, we denote by $\mathbf{v}[S]$, the sub-vector $(v_{i_1}, ..., v_{i_m})$.

Note that we use the same notation to index vectors in $\mathbb{G}^n$.

Given vectors $\mathbf{v}$ and $\mathbf{w}$ of lengths n and m, respectively, we denote by $\mathbf{v}\|\mathbf{w}$ the concatenation of vectors to form a new vector of length $n+m$. Given vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^n$, let $\mathbf{v} \cdot \mathbf{w} = \sum_{i=0}^{n-1} v_i w_i$ denote the inner product of $\mathbf{v}$ and $\mathbf{w}$. For vectors $\mathbf{G} = (G_0, \ldots, G_{n-1}) \in \mathbb{G}^n$ and $\mathbf{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_p^n$, let $\mathbf{G}^{\mathbf{v}} = \prod_{i=0}^{n-1} G_i^{v_i} \in \mathbb{G}$. Note that if for all $i \neq j$, the discrete logarithm of G_i relative to G_j is unknown, then $\mathbf{G}^{\mathbf{v}}$ is a binding Pedersen commitment to vector $\mathbf{v}$.

2 PROOF OF LIABILITIES (POL)

2.1 Entities

A PoL scheme is an organization-level regulation mechanism that involves two entities: the users (who use the organization's services) and the prover (who operates on behalf of the organization).

- **Users**: The set of users $\mathcal{U}$ are the parties who store their assets with the organization.
- **Prover**: The prover $\mathbf{P}$ operates on behalf of the organization and is the entity liable to the users. The prover's goal is to prove to users that $\mathbf{P}$'s liabilities to them are included in the total.

2.2 Formalizing PoL

Definition 1 ([10]). *A* proof of liability (PoL) *scheme is a tuple of algorithms* $\mathsf{PoL} = (\mathsf{Setup}, \mathsf{ProveTot}, \mathsf{VerifyTot}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax.*

- $(PD, SD) \leftarrow\$\ \mathsf{Setup}(1^\kappa, DB)$ *is a probabilistic polynomial-time algorithm executed by* $\mathbf{P}$ *that takes as input a security parameter* κ *and a database* $DB = \{(id_\mathfrak{u}, \ell_\mathfrak{u})\}_{\mathfrak{u} \in \mathcal{U}}$*, comprising of an identifier-liability pair for each user* $\mathfrak{u} \in \mathcal{U}$*. It outputs public data* PD *and secret data* SD *only known to* $\mathbf{P}$*.*
- $(\Pi, L) \leftarrow \mathsf{ProveTot}(DB, SD)$ *is a polynomial-time algorithm executed by* $\mathbf{P}$ *at the behest of an authorized auditor if the scheme calls for one. It takes as input the database* DB *and* $\mathbf{P}$*'s secret data* SD*. It outputs the total liability* L *and associated proof* Π*.*
- $b \leftarrow \mathsf{VerifyTot}(PD, L, \Pi)$*. Given the total liabilities* L *and its associated proof* Π*, an authorized auditor can inspect the validity of* L *according to the public data* PD*. The polynomial-time algorithm returns* 1 *if the verification succeeds and* 0 *otherwise.*
- $\pi \leftarrow \mathsf{Prove}(DB, SD, id)$ *is a polynomial time algorithm executed by* $\mathbf{P}$*. It takes as input the database* DB*,* $\mathbf{P}$*'s secret data* SD*, and a user ID* id*. It outputs a proof* π *indicating the inclusion of* $\mathbf{P}$*'s liabilities to the user in the total liabilities.*
- $b \leftarrow \mathsf{Verify}(PD, id, \ell, \pi)$ *is a polynomial-time algorithm executed by the user. It takes as input the public data* PD*, the user's ID* id*,* $\mathbf{P}$*'s liabilities to the user* ℓ*, and the associated proof of inclusion* π*. It returns 1 if verification succeeds and 0 otherwise.*

2.3 Threat Model

One important aspect that characterizes the PoL problem is that the prover $\mathbf{P}$ has no incentive to increase its total liabilities. On the contrary, $\mathbf{P}$ may be inclined to lower its total liabilities, for example, by claiming less liabilities to an *honest* user or removing its liabilities to the user entirely.

A user $\mathfrak{u} \in \mathcal{U}$ may collude with an adversary that wishes to learn the liabilities to honest users or the number of users using $\mathbf{P}$'s services. We assume that all communication between $\mathcal{U}$ and $\mathbf{P}$ are authenticated. Notably, an honest $\mathbf{P}$ answers requests only from authorized users and shares the proofs of liabilities only with the intended users.

Throughout the paper, we make use of a public bulletin board (PBB) that provides the same view to all entities (i.e., *consensus*). The PBB may be instantiated using a blockchain. The public data PD computed at setup is published to the PBB, ensuring that all users in $\mathcal{U}$ have the same view of PD.

2.4 Security of PoL

Valid databases. We start by defining the validity of a database.

Definition 2 ([10]). *Let $\mathcal{U}$ be a set of users and let $N, max \in \mathbb{N}$. A* database $DB = \{(id_{\mathfrak{u}}, \ell_{\mathfrak{u}})\}_{\mathfrak{u}\in\mathcal{U}}$ is (N, max)-valid *if and only if the following three conditions hold:*
(1) there are at most N users, i.e. $|DB| \leq N$;
(2) for any two distinct users $\mathfrak{u}, \mathfrak{u}' \in \mathcal{U}$, $id_{\mathfrak{u}} \neq id_{\mathfrak{u}'}$;
(3) for all $\mathfrak{u} \in \mathcal{U}$, $0 \leq \ell_{\mathfrak{u}} < max$.

Completeness. Completeness guarantees that when a PoL scheme is executed by honest parties on correct inputs, the verification algorithms succeed with probability 1.

Definition 3 ([10]). *A PoL scheme* is complete *if for any (N, max)-valid database $DB = \{(id_{\mathfrak{u}}, \ell_{\mathfrak{u}})\}_{\mathfrak{u}\in\mathcal{U}}$,*

$$\Pr\left[\begin{array}{r} \mathsf{VerifyTot}(PD, L, \Pi) = 1 \wedge \\ \forall \mathfrak{u} \in \mathcal{U}, \mathsf{Verify}(PD, id_{\mathfrak{u}}, \ell_{\mathfrak{u}}, \pi_{\mathfrak{u}}) = 1 \\ L \geq \sum_{\mathfrak{u}\in\mathcal{U}} \ell_{\mathfrak{u}} \ : \\ (PD, SD) \leftarrow\$\ \mathsf{Setup}(1^{\kappa}, DB), \\ (L, \Pi) \leftarrow \mathsf{ProveTot}(DB, SD), \\ \forall \mathfrak{u} \in \mathcal{U}, \pi_{\mathfrak{u}} \leftarrow \mathsf{Prove}(DB, SD, id_{\mathfrak{u}}) \end{array}\right] = 1.$$

Soundness. Intuitively, soundness guarantees that a malicious prover $\mathbf{P}$ cannot produce public data PD and a *valid* proof $\pi_{\mathfrak{u}}$ in a way that reduces $\mathbf{P}$'s liability to $\mathfrak{u}$, except with negligible probability. Soundness implies that a malicious prover succeeds in convincing honest users of the validity of their proof of liability while simultaneously claiming a total liability that is smaller than the sum of liabilities to these users, only with a negligible probability.

Definition 4 ([10]). *A PoL scheme is* sound *if for any (N, max)-valid database $DB = \{(id_{\mathfrak{u}}, \ell_{\mathfrak{u}})\}_{\mathfrak{u}\in\mathcal{U}}$, and for any p.p.t. adversarial prover $\mathcal{A}^*$ potentially colluding with any number of users there exists a negligible function $\epsilon(\cdot)$ such that for any subset of honest users $\mathcal{U}^*$:*

$$\Pr\left[\begin{array}{r} \mathsf{VerifyTot}(PD, L, \Pi) = 1 \wedge \\ \forall \mathfrak{u} \in \mathcal{U}^*, \mathsf{Verify}(PD, id_{\mathfrak{u}}, \ell_{\mathfrak{u}}, \pi_{\mathfrak{u}}) = 1 \wedge \\ L < \sum_{\mathfrak{u}\in\mathcal{U}^*} \ell_{\mathfrak{u}} \ : \\ (PD, L, \Pi, \{\pi_{\mathfrak{u}}\}_{\mathfrak{u}\in\mathcal{U}^*}) \leftarrow\$\ \mathcal{A}^*(1^{\kappa}, DB) \end{array}\right] \leq \epsilon(\kappa).$$

The notion of soundness is defined with respect to the honest users who check their proof of liability. A malicious prover can reduce or remove the liability to an honest user and evade detection if that user never requests a proof of liability. The more users that verify their liability, the harder it is for the prover to cheat.
Security. A secure PoL scheme should satisfy both completeness and soundness.

Definition 5. *A PoL scheme* is secure *if it is complete and sound.*

2.5 Privacy of PoL

Similarly to [10], we define privacy with respect to the *view* of a subset of malicious users $\mathcal{M} \subseteq \mathcal{U}$. We assume that some adversary has *gained control of* a subset of users and wishes to infer as much information about the honest users as possible. It is thus important to characterize the exact leakage to the malicious users with the goal of understanding the *potential* risks. Let $\mathcal{L}$ denote the leakage function of the PoL scheme. By way of illustration, a fully privacy-preserving PoL has a leakage of $\mathcal{L} = \emptyset$, whereas a non-privacy preserving PoL has a leakage of $\mathcal{L} = DB$. Finally, the leakage of a PoL scheme that reveals the number of users is $\mathcal{L} = |DB|$.

Definition 6 ([10]). *A PoL scheme is $\mathcal{L}$-*private *against a subset of malicious users $\mathcal{M} \subseteq \mathcal{U}$, if there exists a p.p.t. simulator $\mathcal{S}$ such that for any (N, max)-valid database DB, the following two distributions are computationally indistinguishable:*
(1) $\{PD, DB[\mathcal{M}], \{\pi_{\mathfrak{u}}\}_{\mathfrak{u}\in\mathcal{M}} : (PD, SD) \leftarrow\$\ \mathsf{Setup}(1^{\kappa}, DB)\}$, $\forall \mathfrak{u} \in \mathcal{M}, \pi_{\mathfrak{u}} \leftarrow \mathsf{Prove}(DB, SD, id_{\mathfrak{u}})\}$
(2) $\{\mathcal{S}(1^{\kappa}, DB[\mathcal{M}], \mathcal{L})\}$
where $\mathcal{L}$ is the leakage function.

3 SCHEME OVERVIEW

Sparse Verkle Trees. The starting point of our PoL is a **sparse Verkle tree**, which is an authenticated data structure of intractable size (usually, 2^{128}) [13]. Each inner node of the Verkle tree is computed as a *vector commitment* to its children. Sparse Verkle trees enjoy the property of *history independence* [14, 15] i.e., the order in which elements are inserted has no effect on the value of the root.

Let $\mathcal{T}_{n,d}$ be a sparse Verkle tree of arity n and depth d. We now provide a high-level description of our PoL scheme. We first map the liability to each user to a leaf in $\mathcal{T}_{n,d}$, and store in each internal node two vector commitments $\langle V, W\rangle$. At depth $d-1$, V is the vector commitment of the liabilities of the leaves and their sum while W is empty. At depth $i \in [d-1]$, V commits to $(v_0, ..., v_n)$ where for all $j \in [n]$, v_j is the sum of the values committed in the V component of the j^{th} child, and v_n is the sum of values $(v_0, ..., v_{n-1})$. W, on the other hand, commits to $(w_0, ..., w_{n-1})$ where for all $j \in [n]$, w_j is the hash of the content of the j^{th} child. This construction corresponds to a *summation Verkle tree*: the V component of the root commits to vector $(v_0, ..., v_n)$ with v_n being the sum of all the values stored in the non-empty leaves.
Privacy. We use *hiding vector commitments* to guarantee that accessing the content of the nodes on an authentication path does not leak information about the liabilities. To ensure that the authentication paths are well formed without disclosing any information about the liabilities, we leverage *zero-knowledge arguments*.

Let $\mathfrak{u}$ be a user requesting a proof of liability and ℓ its liability. Let $(\langle V_{d-1}, W_{d-1}\rangle, ..., \langle V_0, W_0\rangle)$ be the content of the nodes from the leaf associated with $\mathfrak{u}$ to the root (root included). $\langle V_{d-1}, W_{d-1}\rangle$ is the parent of the leaf and $\langle V_0, W_0\rangle$ is the root of the tree.

Prover $\mathbf{P}$'s goal is to first show that V_{d-1} commits to ℓ at the expected index, and that the last element committed in V_{d-1} is the sum of all the children's liabilities. We recall that W_{d-1} is empty.

Next, for all nodes at depth $d-2 \leq i \leq 0$, $\mathbf{P}$ must prove that (1) W_i commits to the hash of $\langle V_{i+1}, W_{i+1}\rangle$; (2) V_i commits to the sum encoded in V_{i+1}; (3) and the last element committed in V_i is the sum of the previous elements in the vector. Since $(W_0, ..., W_{d-1})$ encodes public information, their respective proofs can be computed following the algorithms of the underlying vector commitment (i.e.,

no need for zero-knowledge arguments). In the case of $(V_0, ..., V_{d-1})$, we ought to demonstrate in zero-knowledge that for all $i \in [d-1]$

(1) the last element of the vector committed in V_i is the sum of the previous elements (**sum argument over vector commitments**);
(2) the last element committed in V_{i+1} opens V_i at a position determined by the index of the leaf storing ℓ (**opening equality argument over vector commitments**).

Additionally, to prevent the prover from injecting negative values in the vector commitments and bringing its total liabilities down, we use *range proofs* to ensure that all the values committed in V_i are in range $[2^m]$, for some predefined value m.

Optimizations. Plugging existing range proofs [12] directly into our construction incurs $O(dn\log(m))$ communication complexity, voiding the communication savings due to sparse Verkle trees. In Section 6.5, we propose a new **range proof over vector commitments** that lowers the communication cost from $O(dn\log(m))$ to $O(d\log(nm))$.

We further optimize our scheme through the aggregation of the opening equality arguments, sum arguments, and range proofs along the authentication path. Aggregating the opening equality arguments reduces the number of pairings performed during verification and the size of the proof. More cost effective is the aggregation of the sum arguments. This decreases the size of the proof from $O(d\log(n))$ to $\log(n)$ and the complexity of the proof generation and verification from $O(dn)$ to $O(d+n)$. Finally, the aggregation of the range proofs further lowers the communication cost of the range proofs from $O(d\log(nm))$ to $O(\log(dnm))$.

Security and privacy guarantees. The soundness of our PoL scheme directly follows from the *binding property* of the vector commitments and the *soundness* of the opening equality and sum arguments and range proofs. Additionally, the *hiding property* of the vector commitments and the *zero-knowledge property* of the arguments guarantee the privacy of user liabilities. Finally, the *history independence* of the sparse Verkle tree ensures that our construction do not leak any information about the size of the prover's database.

4 CRYPTOGRAPHIC BUILDING BLOCKS

In this section, we introduce the syntax of the cryptographic building blocks that our PoL scheme makes use of. The implementations can be found in Section 6.

4.1 Sparse Summation Verkle Trees

We present a novel data structure called the sparse summation Verkle tree (SSVT). It builds upon the sparse Verkle tree, which is an authenticated data structure of intractable size whose inner nodes contain a vector commitment to the hash of its children. Because the inner nodes utilize a vector commitment, Verkle trees can have arity $n \geq 2$ and therefore smaller inclusion proofs than their Merkle tree counterparts.

Definition 7. *A* **sparse summation Verkle tree (SSVT)** $\mathcal{T}_{n,d}$ *of arity n and depth d is an authenticated data structure based on a full Verkle tree of fixed size n^d. Each inner node stores two vector commitments, V and W:*

- *V is a vector commitment to an $n+1$ length vector $\mathbf{v}$ such that $\mathbf{v}[i]$ for $i \in [0, n-1]$ corresponds to the i-th child's value and $\mathbf{v}[n] = \sum_{i=0}^{n-1} \mathbf{v}[i]$.*
- *W is a vector commitment to an n length vector $\mathbf{w}$ such that if the children are also inner nodes then $\mathbf{w}[i]$ stores the hash of the i-th child's vector commitments. Otherwise it is empty.*

See Figure 1 for an example. By construction, VCs [11, 16, 17] result in constant proofs of inclusion and, thus, authentication paths in an SSVT only grow with the depth d of the tree. That is, the proof Ω that a value v opens a vector commitment V at index i does not depend on the size of the committed vector. Moreover, because VCs are easily and efficiently updatable, so are SSVTs.

Sparse Verkle trees, like sparse Merkle trees, are history independent. A data structure is ***history independent*** if any two sequences S_1 and S_2 that yield the same content induce the same distribution on the memory representation [14]. In other words, a set of values produce a deterministic sparse Verkle tree root digest, regardless of the order in which the values have been inserted.

4.2 Hiding Vector Commitments

A vector commitment [16] is a primitive that commits to an ordered sequence of values in such a way that one can open a value at a given index concisely (i.e., using a constant-size proof)[1]. A vector commitment is *hiding* if it does not leak any information about the committed vector.

Definition 8. *A* **hiding vector commitment** *comprises four algorithms* $\mathsf{VC} = (\mathsf{ParamGen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{VerOpen})$ *with the following syntax.*

- $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{ParamGen}(1^\kappa, n)$ *takes as input the security parameter κ and the size of the vectors to be committed $n = \mathrm{poly}(\kappa)$ and outputs public parameters $\mathsf{pp}_{\mathsf{VC}}$.*
- $V \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{v}, r)$ *takes as input public parameters $\mathsf{pp}_{\mathsf{VC}}$, vector $\mathbf{v} \in \mathbb{Z}_p^n$, randomness r, and outputs a commitment V.*
- $\Omega \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, i, \mathbf{v}, r)$ *takes as input public parameters $\mathsf{pp}_{\mathsf{VC}}$, an index $i \in [n]$, a vector $\mathbf{v}$, randomness r and outputs a proof Ω.*
- $b \leftarrow \mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V, i, v, \Omega)$ *takes as input public parameters $\mathsf{pp}_{\mathsf{VC}}$, a commitment V, an index i, an element v, and a proof Ω and outputs a bit b. If $b = 1$, then V is a commitment to a vector $\mathbf{v}$ such that $v = \mathbf{v}[i]$; otherwise, $b = 0$.*

A hiding vector commitment must be ***binding***, that is, the probability that $\mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V, i, v, \Omega) = \mathsf{VerOpen}(\mathsf{pp}_{VC}, V, i, v', \Omega') = 1$ such that $v \neq v'$ must be negligible.

4.3 Arguments of Knowledge

Definition 9. *A ternary relation $\mathcal{R}$ is defined by a set of triples $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ where pp is called the public parameters, $\mathbb{x}$ the instance, and $\mathbb{w}$ the witness. $\mathcal{L}_{\mathcal{R}} = \{(\mathsf{pp}, \mathbb{x}) : \exists \mathbb{w}$ s.t. $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ is called the language of relation $\mathcal{R}$.*

Definition 10 (Interactive Arguments of Knowledge). *Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be three algorithms defined as follows.*

- **Generator** *$\mathcal{G}$ takes as input security parameter 1^κ and relation $\mathcal{R}$ and outputs public parameters pp.*

[1] To reveal the value of an index in a Pedersen commitments, one must produce a proof linear in the size of the vector.

- **Prover** $\mathcal{P}$ *and* **Verifier** $\mathcal{V}$ *are interactive algorithms.* $\mathcal{P}$ *takes as input* $\mathrm{pp}, \mathbb{x}$ *and* $\mathbb{w}$*, whereas* $\mathcal{V}$ *takes as input* pp *and* $\mathbb{x}$*. We denote by* $\mathsf{tr} \leftarrow \langle \mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathrm{pp}, \mathbb{x}) \rangle$ *the transcript of the interaction between* $\mathcal{P}$ *and* $\mathcal{V}$*. At the end of its interaction with* $\mathcal{P}$*,* $\mathcal{V}$ *outputs a bit* $b = \langle \mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathrm{pp}, \mathbb{x}) \rangle$*.* $b = 1$ *indicates that* tr *is accepted by* $\mathcal{V}$*; otherwise,* tr *is rejected.*

  The triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *defines an interactive argument of knowledge if the following properties are met.*
- **Completeness.** $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is complete* **iff** *for all adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c|c} (\mathrm{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathrm{pp} \leftarrow \mathcal{G}(1^\kappa, \mathcal{R}) \\ \vee & (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathrm{pp}) \\ \langle \mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathrm{pp}, \mathbb{x}) \rangle = 1 & \end{array}\right] = 1 \ .$$

- **Knowledge Soundness.** $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is knowledge-sound* **iff** *for all adversaries* $\mathcal{A}$ *there exists an extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c|c} (\mathrm{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathrm{pp} \leftarrow \mathcal{G}(1^\kappa, \mathcal{R}) \\ \wedge & (\mathsf{st}, \mathbb{x}) \leftarrow \mathcal{A}(\mathrm{pp}) \\ \langle \mathcal{A}(\mathsf{st}, \mathbb{x}), \mathcal{V}(\mathrm{pp}, \mathbb{x}) \rangle = 1 & \mathbb{w} \leftarrow \mathcal{E}^{\mathcal{A}(\mathsf{st}, \mathbb{x})}(\mathrm{pp}) \end{array}\right] \leq \mathrm{negl}(\kappa) \ .$$

Definition 11 (Public-Coin Interactive Arguments of Knowledge). *An interactive argument of knowledge* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is public coin if all messages that* $\mathcal{V}$ *sends to* $\mathcal{P}$ *are generated uniformly at random.*

A public-coin interactive argument of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is zero-knowledge if: $\mathsf{tr} \leftarrow \langle \mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathrm{pp}, \mathbb{x}) \rangle$ does not leak any information about witness $\mathbb{w}$. More formally:

Definition 12 (Zero-knowledge). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is zero-knowledge* **iff** *for all adversaries* $\mathcal{A}$*, there exists a polynomial-time simulator* $\mathcal{S}$ *such that the following holds:*

$$\Pr\left[\begin{array}{c|c} (\mathrm{pp}, \mathbb{x}) \in \mathcal{L}_\mathcal{R} & \mathrm{pp} \leftarrow \mathcal{G}(1^\kappa, \mathcal{R}) \\ \wedge & (\mathbb{x}, \mathbb{w}, r) \leftarrow \mathcal{A}(\mathrm{pp}) \\ \mathcal{A}(\mathsf{tr}) = 1 & \mathsf{tr} \leftarrow \langle \mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathrm{pp}, \mathbb{x}; r) \rangle \end{array}\right]$$

$$\approx \Pr\left[\begin{array}{c|c} (\mathrm{pp}, \mathbb{x}) \in \mathcal{L}_\mathcal{R} & \mathrm{pp} \leftarrow \mathcal{G}(1^\kappa, \mathcal{R}) \\ \wedge & (\mathbb{x}, \mathbb{w}, r) \leftarrow \mathcal{A}(\mathrm{pp}) \\ \mathcal{A}(\mathsf{tr}) = 1 & \mathsf{tr} \leftarrow \mathcal{S}(\mathrm{pp}, \mathbb{x}, r) \end{array}\right] \ .$$

where r *is the public-coin randomness that* $\mathcal{V}$ *uses during its interactions with* $\mathcal{P}$*.*

In the random oracle model (ROM), public-coin interactive (zero-knowledge) arguments of knowledge can be transformed into non-interactive arguments via the Fiat-Shamir heuristic [18]. In this case, the arguments of knowledge are defined by triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, such that: $\mathcal{G}(1^\kappa, \mathcal{R})$ outputs the public parameters pp that contain a description of a hash function; $\mathcal{P}(\mathrm{pp}, \mathbb{x}, \mathbb{w})$ outputs a proof Π; and $\mathcal{V}(\mathrm{pp}, \mathbb{x}, \Pi)$ outputs a bit b, where $b = 1$ indicates that Π is valid.

Our PoL scheme makes use of non-interactive variants of public-coin interactive arguments.

4.4 Opening Equality Argument over Vector Commitments

An opening equality argument is a zero-knowledge argument of knowledge that allows a prover to show that two *hiding vector commitments* V and W open to the same value v at indices i and j, respectively.

Definition 13. *Given hiding vector commitment scheme* VC *with public parameters* $\mathrm{pp}_{\mathsf{VC}}$*, an* **opening equality argument** $\mathsf{OEA} = (\mathcal{G}_{\mathsf{OEA}}, \mathcal{P}_{\mathsf{OEA}}, \mathcal{V}_{\mathsf{OEA}})$ *for* VC *is a zero-knowledge argument of knowledge for the following relation:*

$$\mathbb{x} = (V, W, i, j) \ ; \ \mathbb{w} = (v, \Omega, \Omega') \ :$$

$$\mathsf{VerOpen}(\mathrm{pp}_{\mathsf{VC}}, V, i, v, \Omega) = \mathsf{VerOpen}(\mathrm{pp}_{\mathsf{VC}}, W, j, v, \Omega') = 1.$$

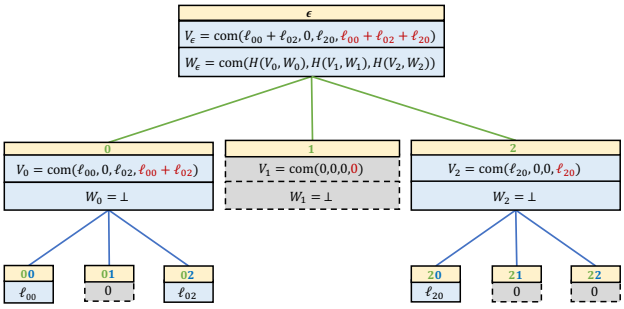


Figure 1: An example of an SSVT $\mathcal{T}_{3,2}$ such that $S = \{00, 02, 20\} \subseteq [3]^2$. The blue nodes denote $\mathsf{Tree}(S)$, which corresponds to the paths from the root to the leaves identified with labels in S. The gray nodes denote $\mathsf{Frontier}(S)$ (i.e., padding nodes), which correspond to nodes that are not in $\mathcal{T}_{3,2}$ but whose parents are. Nodes leading to the root are labeled with a value in $[3]$, whereas a leaf is labeled with value in $[3]^2$ e.g., the children of node 0 are labeled 00, 01 and 02.

4.5 Sum Argument over Vector Commitments

Given a *hiding* vector commitment V, a sum argument is a zero-knowledge argument of knowledge that allows a prover to demonstrate that the last element of the vector committed in V is the sum of all the previous elements.

Definition 14. *Given hiding vector commitment scheme* VC *with public parameters* $\mathrm{pp}_{\mathsf{VC}}$*, a* **sum argument** $\mathsf{SA} = (\mathcal{G}_{\mathsf{SA}}, \mathcal{P}_{\mathsf{SA}}, \mathcal{V}_{\mathsf{SA}})$ *for* VC *is a zero-knowledge argument of knowledge for the following relation:*

$$\mathbb{x} = V \ ; \ \mathbb{w} = (\mathbf{v}, r) \in \mathbb{Z}_p^n \times \mathbb{Z}_p \ :$$

$$V \leftarrow \mathsf{Commit}(\mathrm{pp}_{\mathsf{VC}}, \mathbf{v}, r) \wedge \mathbf{v}[n-1] = \sum_{i=0}^{n-2} \mathbf{v}[i].$$

4.6 Range Proofs over Vector Commitments

Given a hiding vector commitment V, a range proof enables a prover to show in zero-knowledge that the elements of vector $\mathbf{v}$ committed to in V fall into a particular range. In our PoL scheme, we are interested in showing that $0 \leq \mathbf{v}[i] < max$, where max is an upper-bound of individual liabilities that a PoL prover can hold.

Definition 15. *Given hiding vector commitment scheme* VC *with public parameters* $\mathrm{pp}_{\mathsf{VC}}$*, a* **range proof** $\mathsf{RP} = (\mathcal{G}_{\mathsf{RP}}, \mathcal{P}_{\mathsf{RP}}, \mathcal{V}_{\mathsf{RP}})$ *for* VC *and range* $[0, max]$ *is a zero-knowledge argument of knowledge for the following relation:*

$$\mathbb{x} = V \ ; \ \mathbb{w} = (\mathbf{v}, r) \in \mathbb{Z}_p^n \times \mathbb{Z}_p \ :$$

$$V \leftarrow \mathsf{Commit}(\mathrm{pp}_{\mathsf{VC}}, \mathbf{v}, r) \wedge 0 \leq \mathbf{v}[i] < max, \ \forall i \in [n].$$

Let $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{ParamGen}(1^\kappa, n+1)$ be the public parameters of a hiding vector commitment scheme VC.
Let h be a collision-resistant hash function $h : \{0,1\}^* \rightarrow [n]^d$.

1: $\mathsf{Setup}(1^\kappa, DB) \rightarrow (SD, PD)$
2: $SD \leftarrow [\cdot]$ ▷ empty map
3: $S \leftarrow \emptyset$
4: **for** $(\ell, id) \in DB$ **do**
5: $S \leftarrow S \cup \{h(id)\}$ ▷ Append $h(id)$ to S
6: **for** $\lambda \in \mathsf{Frontier}(S)$ **do**
7: **if** $\lambda = \lambda_1 ... \lambda_d$ **then**
8: $SD[\lambda] \leftarrow 0$
9: **else if** $\lambda = \lambda_1 ... \lambda_{d-1}$ **then**
10: $r \leftarrow\$ \mathbb{Z}_p$
11: $V \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{0^{n+1}}, r)$
12: $SD[\lambda] \leftarrow \langle V, \perp, \mathbf{0^{n+1}}, \perp, r, 0 \rangle$
13: **else**
14: $(r, s) \leftarrow\$ \mathbb{Z}_p^2$
15: $V \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{0^{n+1}}, r)$
16: $W \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{0^{n+1}}, s)$
17: $SD[\lambda] \leftarrow \langle V, W, \mathbf{0^{n+1}}, \mathbf{0^{n+1}}, r, s \rangle$
18: **for** $\lambda \in \mathsf{Tree}(S)$ **do**
19: **if** $\lambda = \lambda_1 ... \lambda_d$ **then**
20: find $(id, \ell) \in DB : \lambda = h(id)$
21: $SD[\lambda] \leftarrow \ell$
22: **else if** $\lambda = \lambda_1 ... \lambda_{d-1}$ **then**
23: **for** $0 \leq i \leq n-1$ **do**
24: $v_i \leftarrow SD[\lambda \parallel i]$
25: $\mathbf{v} \leftarrow (v_0, ..., v_{n-1}, \sum_{j=0}^{n-1} v_i)$
26: $r \leftarrow\$ \mathbb{Z}_p$
27: $V \leftarrow\$ \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{v}, r)$
28: $SD[\lambda] \leftarrow \langle V, \perp, \mathbf{v}, \perp, r, 0 \rangle$
29: **else**
30: **for** $0 \leq i \leq n-1$ **do**
31: $\langle V_i, W_i, \mathbf{v}_i, \mathbf{w}_i, r_i, s_i \rangle \leftarrow SD[\lambda \parallel i]$
32: $v_i \leftarrow \mathbf{v}_i[n]$
33: $w_i \leftarrow h(V_i, W_i)$
34: $\mathbf{v} \leftarrow (v_0, ..., v_{n-1}, \sum_{j=0}^{n-1} v_i)$
35: $\mathbf{w} \leftarrow (w_0, ..., w_{n-1})$
36: $(r, s) \leftarrow\$ \mathbb{Z}_p^2$
37: $V \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{v}, r)$
38: $W \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{w} \parallel 0, s)$
39: $SD[\lambda] \leftarrow \langle V, W, \mathbf{v}, \mathbf{w}, r, s \rangle$
40: $\langle V_0, W_0, \mathbf{v}_0, \mathbf{w}_0, r_0, s_0 \rangle \leftarrow SD[\varepsilon]$
41: $PD \leftarrow \langle V_0, W_0 \rangle$
42: return (SD, PD)

Figure 2: Proof of Liabilities: Setup **Algorithm**

5 POL SCHEME

Let $\mathcal{T}_{n,d}$ denote a sparse summation Verkle tree of arity n and depth d. We identify the vertices of $\mathcal{T}_{n,d}$ by their labels λ and the root by empty label ε. A leaf in the tree has label $\lambda = \lambda_1 ... \lambda_d$ such that $\forall\ 1 \leq i \leq d, \lambda_i \in [n]$, and the path from the root to the leaf is determined by nodes with labels $(\varepsilon, \lambda_1, \lambda_1\lambda_2, ..., \lambda_1 ... \lambda_d)$. Conversely, the children of an internal node with label λ are determined by labels $\lambda \parallel i$ for $i \in [n]$. Accordingly, we call the child with label $\lambda \parallel i$, the child at index i of node λ. Let $S \subseteq [n]^d$. We denote by $\mathsf{Tree}(S)$ the labels of the union of all the paths from the leaves identified by S to the root ε, and by $\mathsf{Frontier}(S)$ the labels of the nodes that are not in $\mathsf{Tree}(S)$ but their parent is.

Let $h : \{0,1\}^* \rightarrow [n]^d$ be a collision-resistant hash function.

5.1 Setup

During setup (see Figure 2), prover **P** builds a sparse summation Verkle tree, $\mathcal{T}_{n,d}$. Each node at depth d (i.e, a leaf) stores a value in $\mathbb{Z}_p$, each node at depth $d-1$ stores a vector commitment V_{d-1} to the values in its children together with their sum, and each internal node at depth $0 \leq i \leq d-2$ stores two vector commitments $\langle V_i, W_i \rangle$, where V_i is a vector commitment to the sum value committed in each child plus their sum and W_i is a vector commitment to the hash of each child.

More specifically, **P** computes for each entry $(id, \ell) \in DB$ a label $h(id)$ and appends the result to the set of labels S (lines 4 & 5). The collision-resistance of h ensures that each user is uniquely assigned to a leaf. **P** then computes *padding nodes* (lines 6 – 17), which coincide with labels in $\mathsf{Frontier}(S)$. A leaf in $\mathsf{Frontier}(S)$ contains value 0, a node at depth $d-1$ contains a hiding vector commitment to a zero vector, whereas a node at depth $d-2$ or lower contains two hiding vector commitments to zero vectors.

Finally, **P** computes *non-empty nodes*, which are determined by labels in $\mathsf{Tree}(S)$ (lines 18 – 39). A leaf in $\mathsf{Tree}(S)$ contains value ℓ, such that $(id, \ell) \in DB$ and $h(id) \in S$ (lines 19 – 21). Now, a node at depth $d-1$ stores the commitment to a vector $\mathbf{v} = (v_0, ..., v_n)$ whereby v_i is the value of the child at index i for $1 \leq i \leq n-1$ and v_n is the sum of these values (i.e., $v_n = \sum_{i=0}^{n-1} v_i$) (lines 22 – 28). A node at depth $d-2$ or lower stores two vector commitments (lines 29 – 39). The first commitment commits to $\mathbf{v} = (v_0, ..., v_n)$, with $v_n = \sum_{i=0}^{n-1} v_i$ and v_i for $1 \leq i \leq n-1$ being the sum committed in the child at index i. The second commitments commits to the hash of the vector commitments in each child.

Setup concludes by outputting a pair (PD, SD), where $PD = \langle V_0, W_0 \rangle$ is the root of the sparse summation Verkle tree $\mathcal{T}_{n,d}$ (line 41) and SD is the secret data necessary to build $\mathcal{T}_{n,d}$. That is, the values and the randomness used to compute the hiding vector commitments in $\mathcal{T}_{n,d}$. **P** then publishes PD to a public bulletin board, while storing SD locally.

5.2 Proving Total Liabilities

Given root $PD = \langle V_0, W_0 \rangle$ and secret data SD, **P** shows that its total liabilities equal L by opening V_0 at position n, and outputting the result Π to the parties privy to its total liabilities.

Given (L, Π), $\mathcal{V}$ runs $\mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V_0, n, L, \Pi) \leftarrow b$. If $b = 1$, then L corresponds to the total liabilities of **P**.

5.3 Proving Individual Liabilities

When **P** receives a proof of liabilities request, it first authenticates it. If the request did not originate from a user $u \in \mathcal{U}$, then **P** rejects. Let id be the identifier of the user. **P** correspondingly, computes $\lambda = \lambda_1 ... \lambda_d = h(id)$, which identifies a unique path in $\mathcal{T}_{n,d}$ (line 2).

Recall that $\mathcal{T}_{n,d}$'s root stores $\langle V_0, W_0 \rangle$. Without loss of generality, we denote by $\langle V_i, W_i \rangle, 1 \leq i \leq d-1$, the content of the internal nodes along the path.

Now **P** generates **(a)** a range proof Ψ_0 that demonstrates that the values committed in V_0 are in the authorized range; **(b)** a sum argument Φ_0 that proves that V_0 is a commitment to vector $\mathbf{v}_0$ such that $\mathbf{v}_0[n] = \sum_{i=0}^{n-1} \mathbf{v}_0[i]$ (see Figure 3, lines 13 & 14).

Then for each internal node, at depth $1 \leq i \leq d-1$, on the path from the root to λ, **P (a)** shows that the hash of the content of the node opens commitment W_{i-1} at position λ_i (line 19); **(b)** produces a range proof Ψ_i that shows that the values committed in V_i are in the authorized range (line 22); **(c)** computes a sum argument Φ_i that proves that V_i is a commitment to vector $\mathbf{v}_i$ such that $\mathbf{v}_i[n] = \sum_{j=0}^{n-1} \mathbf{v}_i[j]$ (line 23); **(d)** generates an opening equality argument that demonstrates that $\mathbf{v}_i[n]$ open V_i and V_{i-1} at positions n and λ_i respectively (line 28).

Finally, **P** opens vector commitment V_{d-1} at position λ_d (line 30). Let π denote all the proofs generated by **P** (line 31).

Upon receiving **P**'s response π, the user parses it as a $d+1$-length array $\mathbb{P}$, where for $1 \leq i \leq d$, $\mathbb{P}[i]$ contains the proofs produced for the i^{th} node on the path from the root to leaf $\lambda = h(id)$, and $\mathbb{P}[0]$ contains the proofs generated for the root. The user then reads $\mathbb{P}[0]$ as $\langle \Psi_0, \Phi_0 \rangle$ and checks whether they are a valid range proof and sum argument with respect to the published vector commitment V_0 (lines 42 – 47). Next, the user parses $\mathbb{P}[i]$ for $1 \leq i \leq d-1$ as $\langle V_i, W_i, \Psi_i, \Phi_i, \Upsilon_i, \Omega_i \rangle$ (line 50) and verifies whether **(a)** Ω_i is a valid proof that opens vector commitment W_{i-1} to value $h(V_i, W_i)$ at position λ_i (line 52); **(b)** (Ψ_i, Φ_i) are valid range proof and sum argument with respect to vector commitment V_i (lines 54 & 55); **(c)** Υ_i is a valid opening equality argument for vector commitments V_i and V_{i-1} at positions n and λ_i respectively (line 57).

Finally, the user parses $\mathbb{P}[d]$ as Ω_d and checks if Ω_d proves that vector $\mathbf{v}_{d-1}$ committed in V_{d-1} satisfies $\mathbf{v}_{d-1}[\lambda_d] = \ell$, where ℓ is the liability of **P** to the user (line 61). If all checks succeed, then the user accepts the proof of liability; otherwise, it rejects.

5.4 Security and Privacy

We now prove that our scheme is secure and privacy-preserving and satisfies the definitions in Section 2.

Theorem 5.1. *The proof of liabilities described in Section 5 is a secure* $([n]^d, max)$*–PoL (i.e., satisfies Definition 5).*

Proof. It is straight forward to show that our PoL is complete. It remains to show that the PoL is sound. Let $\mathcal{U}^*$ denote the subset of honest users requesting proofs of liabilities. Assume for contradiction that there exists an adversary $\mathcal{A}^*$ that succeeds in breaking the soundness of our PoL scheme. This translates to $\mathcal{A}^*$ producing public data $\langle V_0, W_0 \rangle$, a valid proof Π such that $\mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V_0, n, L^*, \Pi) = 1$, and valid proofs $\pi_{\mathfrak{u}}$ for all $\mathfrak{u} \in \mathcal{U}$.

Each user proof π comprises of an authentication path $\mathbb{P}$ of length $d+1$. $\mathbb{P}[0]$ should correspond to the root of the tree and comprises of range proof Ψ_0 and sum argument proof Φ_0. For $i \in [1, d]$, $\mathbb{P}[i]$ consists of vector commitments V_i and W_i, range proof Ψ_i, sum argument Φ_i, opening equality argument Υ_i, and vector opening Ω_i. $\mathbb{P}[d]$ should correspond to a leaf and consists of an opening Ω_d.

For any set of users $\mathcal{U}$ and their liabilities, there exists a unique sparse summation Verkle tree from which the proofs should be generated. We emphasize, however, that the adversary may not build a tree and may try to generate the proofs maliciously. Below, we proceed by cases and argue that any adversary who does not follow the protocol will not be able to construct a valid proof. We denote the vector commitment of a vector $\mathbf{v}_i$ with V_i.

(1) There are two users who are given the same authentication path $\mathbb{P}$. Note that each user $\mathfrak{u} \in \mathcal{U}$ has a unique identifier $id_{\mathfrak{u}}$ and so $h(id_{\mathfrak{u}})$ must be mapped to a unique value in $[n]^d$ that identifies a unique path in the tree. Thus this would contradict the collision resistance of h.

(2) There exists an honest user $\mathfrak{u}$ with label $\lambda_1 \dots \lambda_d$ such that $\mathbf{v}_{d-1}[\lambda]$ contains a value smaller than $\ell_{\mathfrak{u}}$. This implies that $\mathbb{P}[d]$ contains a proof Ω_d that opens vector commitment V_{d-1} at index λ_d to $\ell_{\mathfrak{u}}$. This, however, would contradict the binding property of the vector commitment.

(3) Suppose now that for all honest users, $\mathbb{P}[d]$ stores an opening to the correct liability at the correct index, where $\mathbb{P}$ is the user's respective authentication path. One of the following must hold:

- **(a)** There exists a user $\mathfrak{u} \in \mathcal{U} \setminus \mathcal{M}$ with label $\lambda_1 \dots \lambda_d$ such that for some index $0 \leq i \leq d-1$ we have $\mathbf{v}_i[\lambda_i] > \mathbf{v}_{i-1}[n]$. This implies that $\Upsilon_i \in \mathbb{P}[i]$ is a valid OEA proof for the instance $(V_i, V_{i-1}, \lambda_i, n)$ which would contradict the security of the opening equality argument OEA.
- **(b)** There exists a user $\mathfrak{u} \in \mathcal{U} \setminus \mathcal{M}$ such that for some index $0 \leq i \leq d-1$ we have $\mathbf{v}_i[n] < \sum_{j \in [n]} \mathbf{v}_i[j]$. This implies that $\Phi_i \in \mathbb{P}[i]$ is a valid sum argument for the instance V_i which would contradict the security of sum argument SA.
- **(c)** There exists a user $\mathfrak{u} \in \mathcal{U} \setminus \mathcal{M}$ such that for some indices $0 \leq i \leq d-1$ and $j \in [n]$ we have $\mathbf{v}_i[j] < 0$. This implies that $\Psi_i \in \mathbb{P}[i]$ is a valid range proof for the instance V_i which would contradict the security of range proof RP.
- **(d)** There exists a user $\mathfrak{u} \in \mathcal{U} \setminus \mathcal{M}$ with label $\lambda_1 \dots \lambda_d$ such that for some index $1 \leq i \leq d$, $\mathbf{w}_{i-1}[\lambda_i]$ is not equal to $h(V_{\lambda_i}, W_{\lambda_i})$. In other words, W_i does not correctly encode the path corresponding to the user. This implies that $\Omega_i \in \mathbb{P}[i]$ is a valid opening of W_{i-1} to the value $h(V_{\lambda_i}, W_{\lambda_i})$ at index λ_i which would contradict either the binding of the vector commitment or collision resistance of hash h.

We have reached a contradiction and therefore conclude that our construction is a secure $([n]^d, max)$–PoL. □

Theorem 5.2. *The proof of liabilities described in Section 5 is* $\mathcal{L}$*-private* $([n]^d, max)$*–PoL(Definition 6), with* $\mathcal{L} = \emptyset$.

Proof. Let $\mathcal{M} \subseteq \mathcal{U}$ denote the set of malicious users. We now describe a simulator, which upon input of κ, $DB[\mathcal{M}]$, and $\mathcal{L} = \emptyset$ returns outputs to the malicious users that are computationally indistinguishable from the ones produced by our PoL scheme.

Simulator. If $\mathcal{M} = \emptyset$, then the simulator samples r and s, computes $V = \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{0^{n+1}}, r)$ and $W = \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{0^{n+1}}, s)$, and returns $PD = \langle V, W \rangle$.

Let max be the maximum value of a liability.
Let $\mathsf{pp}_{\mathsf{RP}}$ be the public parameters of a range proof for the vector commitment scheme VC and range $[max]$.
Let $\mathsf{pp}_{\mathsf{SA}}$ be the public parameters of a sum argument for the vector commitment scheme VC.
Let $\mathsf{pp}_{\mathsf{OEA}}$ be the public parameters of an opening equality argument for the vector commitment scheme VC.

1: $\mathsf{Prove}(DB, SD, id) \to \pi$
2: $\quad \lambda_1 ... \lambda_d \leftarrow h(id)$
3: $\quad \mathbb{W} \leftarrow [\cdot]$
4: $\quad \mathbb{P} \leftarrow [\cdot]$
5: $\quad \lambda \leftarrow \varepsilon$ ▷ empty string
6: $\quad \mathbb{W}[0] \leftarrow SD[\lambda]$
7: $\quad$ **for** $1 \le i \le d$ **do**
8: $\quad\quad \lambda \leftarrow \lambda \parallel \lambda_i$
9: $\quad\quad \mathbb{W}[i] \leftarrow SD[\lambda]$
10: $\quad \langle V_0, W_0, \mathbf{v}_0, \mathbf{w}_0, r_0, s_0 \rangle \leftarrow \mathbb{W}[0]$
11: $\quad \mathbb{x}_0 \leftarrow V_0$
12: $\quad \mathbb{w}_0 \leftarrow (\mathbf{v}_0, r_0)$
13: $\quad \Psi_0 \leftarrow \mathcal{P}_{\mathsf{RP}}(\mathsf{pp}_{\mathsf{RP}}, \mathbb{x}_0, \mathbb{w}_0)$
14: $\quad \Phi_0 \leftarrow \mathcal{P}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_0, \mathbb{w}_0)$
15: $\quad \mathbb{P}[0] \leftarrow \langle \Psi_0, \Phi_0 \rangle$
16: $\quad$ **for** $1 \le i \le d-1$ **do**
17: $\quad\quad \langle V_{i-1}, W_{i-1}, \mathbf{v}_{i-1}, \mathbf{w}_{i-1}, r_{i-1}, s_{i-1} \rangle \leftarrow \mathbb{W}[i-1]$
18: $\quad\quad \langle V_i, W_i, \mathbf{v}_i, \mathbf{w}_i, r_i, s_i \rangle \leftarrow \mathbb{W}[i]$
19: $\quad\quad \Omega_i \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, \lambda_i, \mathbf{w}_{i-1}, s_{i-1})$
20: $\quad\quad \mathbb{x}_i \leftarrow V_i$
21: $\quad\quad \mathbb{w}_i \leftarrow (\mathbf{v}_i, r_i)$
22: $\quad\quad \Psi_i \leftarrow \mathcal{P}_{\mathsf{RP}}(\mathsf{pp}_{\mathsf{RP}}, \mathbb{x}_i, \mathbb{w}_i)$
23: $\quad\quad \Phi_i \leftarrow \mathcal{P}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_i, \mathbb{w}_i)$
24: $\quad\quad \Omega'_i \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, \lambda_i, \mathbf{v}_{i-1}, r_{i-1})$
25: $\quad\quad \Omega''_i \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, n, \mathbf{v}_i, r_i)$
26: $\quad\quad \mathbb{x}_i \leftarrow (V_{i-1}, V_i, \lambda_i, n)$
27: $\quad\quad \mathbb{w}_i \leftarrow (\mathbf{v}_i[n], \Omega'_i, \Omega''_i)$
28: $\quad\quad \Upsilon_i \leftarrow \mathcal{P}_{\mathsf{OEA}}(\mathsf{pp}_{\mathsf{OEA}}, \mathbb{x}_i, \mathbb{w}_i)$
29: $\quad\quad \mathbb{P}[i] \leftarrow \langle V_i, W_i, \Psi_i, \Phi_i, \Upsilon_i, \Omega_i \rangle$
30: $\quad \mathbb{P}[d] \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, \lambda_d, \mathbf{v}_{d-1}, r_{d-1})$
31: $\quad \pi \leftarrow \mathbb{P}$
32: $\quad$ return π

33: $\mathsf{Verify}(PD, id, \ell, \pi) \to b$
34: $\quad \langle V_0, W_0 \rangle \leftarrow PD$
35: $\quad \lambda_1 ... \lambda_d \leftarrow h(id)$
36: $\quad \pi \leftarrow \mathbb{P}$
37: $\quad \mathbb{C} \leftarrow [\cdot]$
38: $\quad \mathbb{C}[0] \leftarrow \langle V_0, W_0 \rangle$
39: $\quad$ **for** $1 \le i \le d-1$ **do**
40: $\quad\quad \langle V_i, W_i, \Psi_i, \Phi_i, \Upsilon_i, \Omega_i \rangle \leftarrow \mathbb{P}[i]$
41: $\quad\quad \mathbb{C}[i] \leftarrow \langle V_i, W_i \rangle$
42: $\quad \langle \Psi_0, \Phi_0 \rangle \leftarrow \mathbb{P}[0]$
43: $\quad \mathbb{x}_0 \leftarrow V_0$
44: $\quad b_1 \leftarrow \mathcal{V}_{\mathsf{RP}}(\mathsf{pp}_{\mathsf{RP}}, \mathbb{x}_0, \Psi_0)$
45: $\quad b_2 \leftarrow \mathcal{V}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_0, \Phi_0)$
46: $\quad$ **if** $b_1 \wedge b_2 = 0$ **then**
47: $\quad\quad$ return 0
48: $\quad$ **for** $1 \le i \le d-1$ **do**
49: $\quad\quad \langle V_{i-1}, W_{i-1} \rangle \leftarrow \mathbb{C}[i-1]$
50: $\quad\quad \langle V_i, W_i, \Psi_i, \Phi_i, \Upsilon_i, \Omega_i \rangle \leftarrow \mathbb{P}[i]$
51: $\quad\quad w_i \leftarrow h(V_i, W_i)$
52: $\quad\quad b_1 \leftarrow \mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, W_{i-1}, \lambda_i, w_i, \Omega_i)$
53: $\quad\quad \mathbb{x}_i \leftarrow V_i$
54: $\quad\quad b_2 \leftarrow \mathcal{V}_{\mathsf{RP}}(\mathsf{pp}_{\mathsf{RP}}, \mathbb{x}_i, \Psi_i)$
55: $\quad\quad b_3 \leftarrow \mathcal{V}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_i, \Phi_i)$
56: $\quad\quad \mathbb{x}_i \leftarrow (V_{i-1}, V_i, \lambda_i, n)$
57: $\quad\quad b_4 \leftarrow \mathcal{V}_{\mathsf{OEA}}(\mathsf{pp}_{\mathsf{OEA}}, \mathbb{x}_i, \Upsilon_i)$
58: $\quad\quad$ **if** $b_1 \wedge b_2 \wedge b_3 \wedge b_4 = 0$ **then**
59: $\quad\quad\quad$ return 0
60: $\quad \Omega_d \leftarrow \mathbb{P}[d]$
61: $\quad b \leftarrow \mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V_{d-1}, \lambda_d, \ell, \Omega_d)$
62: $\quad$ return b

Figure 3: Proof of Liabilities: Prove **and** Verify **Algorithm**

If $\mathcal{M} \neq \emptyset$, then the simulator builds a sparse summation Verkle tree on $DB[\mathcal{M}]$ following Setup, see Figure 2. It then computes the proofs of liabilities $\{\pi_{\mathfrak{u}}\}_{\mathfrak{u} \in \mathcal{M}}$ as described in Prove (Figure 3). Let $\langle V, W \rangle$ denote the root of the resulting tree. The simulator returns $(PD = \langle V, W \rangle, DB[\mathcal{M}], \{\pi_{\mathfrak{u}}\}_{\mathfrak{u} \in \mathcal{M}})$.

Game sequence. We now introduce a series of hybrid games to show that the execution of our PoL scheme and the execution of the simulator are indistinguishable.

$\mathbf{Hyb_0}$: This is identical to an execution of our PoL scheme on input $DB[\mathcal{U}]$, following the description in Figures 2 and 3.

$\mathbf{Hyb_1}$: Run Setup by first inserting the malicious users $\mathcal{M}$ into the tree, and then insert the honest users $\mathcal{U} \setminus \mathcal{M}$.

$\mathbf{Hyb_2}$: Run Setup by first inserting the malicious users $\mathcal{M}$ into the tree, however, for all other users $\mathfrak{u} \in \mathcal{U} \setminus \mathcal{M}$ do the following. If $\mathfrak{u} \in \mathsf{Frontier}(\mathcal{M})$, then set its liability to 0. Else for each remaining user $\mathfrak{u}$, compute its authentication path, find the node along this path that is in $\mathsf{Frontier}(\mathcal{M})$, compute the corresponding commitments as described in lines 9 to 17 of Figure 2, and then prune the subpath from $\mathfrak{u}$'s leaf up to (but not including) the frontier node.

$\mathbf{Hyb_3}$: Run Setup on input $DB[\mathcal{M}]$.

$\mathbf{Hyb_0}$ is indistinguishable from **$\mathbf{Hyb_1}$**, since the SSVT ensures history independence. In other words, the order in which we insert $\mathcal{U}$ has no bearing on the tree, and thus, the same tree is produced in both games.

$\mathbf{Hyb_1}$ is indistinguishable from **$\mathbf{Hyb_2}$**. If not, the adversary is able to distinguish between either the public data or the proof of liabilities to a user in $\mathcal{M}$ that are produced from $DB[\mathcal{U}]$ and $DB[\mathcal{M}]$. The former implies that the root $\langle V_0, W_0 \rangle$ of **$\mathbf{Hyb_1}$** and the root $\langle V'_0, W'_0 \rangle$ of **$\mathbf{Hyb_2}$** are distinguishable, therefore breaking the hiding property of the vector commitments. The latter implies that there exists some user $\mathfrak{u} \in \mathcal{M}$ whose proof π generated in **$\mathbf{Hyb_1}$** is distinguishable from their proof π' generated in **$\mathbf{Hyb_2}$**. But π and π' consist of vector commitments, range proofs, sum arguments, opening equality arguments, and vector commitment openings. Distinguishing between the proofs either breaks the hiding property

of the vector commitments or the zero-knowledge property of arguments RP, SA and OEA.

$\mathbf{Hyb_2}$ is identical to $\mathbf{Hyb_3}$. Namely, in $\mathbf{Hyb_2}$, all nodes in Frontier($\mathcal{M}$) are computed following the steps described in lines 7 to 17 of Figure 2. Additionally, nodes that are not in the frontier or along the authentication paths of $\mathcal{M}$ are pruned. This is the same as simply building the tree on $DB[\mathcal{M}]$. More concretely, each leaf node in $\mathbf{Hyb_2}$ or $\mathbf{Hyb_3}$ corresponds to a user in $\mathcal{M}$, and each inner node contains vector commitments that are only defined as a function of the liabilities in $\mathcal{M}$.

Lastly, observe that the distribution of $\mathbf{Hyb_3}$ is identical to that of running the simulator $\mathcal{S}$. This concludes the proof. □

6 INSTANTIATION

We now instantiate the cryptographic building blocks introduced in Section 4. We provide an overview of the hiding vector commitments introduced in [11], and describe our *tailor-made* range proofs, opening equality arguments and sum arguments. Together, these primitives enable us to keep the proofs of liabilities short.

6.1 Inner Product Argument

To construct the sum argument and range proof, we leverage inner product arguments [12, 19]. These enable a prover to show that the inner product of two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^n$ committed to using Pedersen vector commitments evaluates to a public value c.

Definition 16. *Let $\mathbf{G} = (G_0, ..., G_{n-1})$ and $\mathbf{H} = (H_0, ...H_{n-1})$ be two vectors of n generators of $\mathbb{G}$, such that for all $i \neq j$, the relative discrete log of G_i (resp. H_i) relative to G_j (resp. H_j) is unknown.*

An inner product argument $\mathsf{IPA} = (\mathcal{G}_{\mathsf{IPA}}, \mathcal{P}_{\mathsf{IPA}}, \mathcal{V}_{\mathsf{IPA}})$ *is an argument of knowledge for the following relation:*

$$\mathbb{x} = (P, c)\ ;\ \mathbb{w} = (\mathbf{a}, \mathbf{b}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^n :\ P = \mathbf{G}^{\mathbf{a}}\mathbf{H}^{\mathbf{b}}\ \wedge\ \mathbf{a} \cdot \mathbf{b} = c$$

We instantiate the inner product argument using the improved inner product argument introduced in [12]. This argument can be made non-interactive using the Fiat-Shamir heuristic.

6.2 Hiding Vector Commitments

We instantiate the hiding vector commitments with Pointproofs [11]. This scheme extends the pairing-based vector commitment of Libert and Yung [17] to support the aggregation of openings across the same commitment as well as different commitments. This reduces the cost of verification (notably, the number of pairings to be computed). We also leverage the aggregation of openings to aggregate opening equality arguments (see Section 7.1).

Let $(\mathbb{G}, \mathbb{G}_T)$ be groups of prime order p that admit an efficient bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let G and $e(G, G)$ be generators of $\mathbb{G}$ and $\mathbb{G}_T$, respectively[2]. This scheme requires a trusted-setup to generate the public parameters using a secret value $\alpha \in \mathbb{Z}_p$ known to no one after setup. The description of the 8 algorithms of VC follows.

1. $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{ParamGen}(1^\kappa, n)$: Sample $\alpha \leftarrow\$\ \mathbb{Z}_p$ and output G and

$$\mathbf{F} = (F_0, ..., F_n, F_{n+2}, ..., F_{2n+1}) : F_i = G^{\alpha^{i+1}} \text{ for } i \in [2(n+1)] \setminus \{n+1\}$$

[2]Pointproofs uses asymmetric pairings. For ease of exposition, we describe the scheme using symmetric pairings.

2. $V \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{VC}}, \mathbf{v}, r)$. To commit to vector $\mathbf{v} = (v_0, ..., v_{n-1})$ with randomness r, compute

$$V = \mathbf{F}[:n+1]^{\mathbf{v}\|r} = F_n^r \prod_{i=0}^{n-1} F_i^{v_i}$$

3. $\Omega \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{VC}}, i, \mathbf{v}, r)$. To reveal element $\mathbf{v}[i]$, compute

$$\Omega = F_{2n-i}^r \prod_{j=0, j\neq i}^{n-1} F_{n+j-i}^{v_j} = \left(\frac{V}{F_i^{v_i}}\right)^{\alpha^{n+1-i}}$$

4. $b \leftarrow \mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V, i, v, \Omega)$. To verify if v opens V at index i, check if $e(V, F_{n-i}) = e(\Omega, G)e(F_0, F_n)^v$. If yes, then output 1. Otherwise output 0. We call Ω the opening proof of triple index, value and commitment (i, v, V)

Both same and cross aggregation rely on computing a product of the proofs of individual openings raised to some exponent. To preserve the binding property after aggregation, this exponent is computed as a hash of the vector commitments, the indices being opened and the values at these indices. In the following, $S \subseteq [n]$ denotes the subset of indices we are aggregating across.

5. $\widehat{\Omega} \leftarrow \mathsf{AggregateSame}(\mathsf{pp}_{\mathsf{VC}}, V, S, \mathbf{v}[S], \{\Omega_i\}_{i\in S})$. Outputs

$$\widehat{\Omega} = \prod_{i\in S} \Omega_i^{t_i}$$

where Ω_i is the opening of V at index i and $t_i = h(i, V, S, \{v_i\}_{i\in S})$.

6. $b \leftarrow \mathsf{VerOpenSame}(\mathsf{pp}_{\mathsf{VC}}, V, S, \mathbf{v}[S], \widehat{\Omega})$. To verify if v_i opens V at index i for all $i \in S$, check if

$$e(V, \prod_{i\in S} F_{n-i}^{t_i}) = e(\widehat{\Omega}, G)e(F_0, F_n)^{\sum_{i\in S} v_i t_i}.$$

where t_i is as before. If equal, output 1. Otherwise output 0.

7. $\Omega \leftarrow \mathsf{AggregateAcross}(\mathsf{pp}_{\mathsf{VC}}, \{V_j, S_j, \mathbf{v}_j[S_j], \widehat{\Omega}_j\}_{j\in[k]})$. Outputs

$$\Omega = \prod_{j\in[k]} \widehat{\Omega}_j^{t'_j}$$

where $\widehat{\Omega}_j$ is the aggregated opening from AggregateSame and

$$t'_j = h'(j, \{V_j, S_j, \mathbf{v}_j[S_j]\}_{j\in[k]}).$$

8. $b \leftarrow \mathsf{VerOpenAcross}(\mathsf{pp}_{\mathsf{VC}}, \{V_j, S_j, \mathbf{v}_j[S_j]\}_{j\in[k]}, \Omega)$. To verify if the $\mathbf{v}_j[S_j]$ open V_j at positions S_j, check if

$$\prod_{j\in[k]} e(V_j, \prod_{i\in S} F_{n-i}^{t_{j,i}})^{t'_j} = e(\Omega, G)e(F_0, F_n)^{\sum_{j\in[k], i\in S_j} \mathbf{v}[i] t_{j,i} t'_j}$$

where Ω is the cross aggregation opening,

$$t'_j = h'(j, \{V_j, S_j, \mathbf{v}_j[S_j]\}_{j\in[k]}) \text{ and } t_{j,i} = h(i, V_j, S_j, \mathbf{v}_j[S_j]).$$

If the equality holds, then output 1. Otherwise output 0.

Gorbunov et al. [11] prove the security of Pointproofs in the algebraic group model (AGM) and the random oracle model (ROM) under the assumption of the weak bilinear Diffie-Hellman exponent problem n-wBDHE*.

Let $\mathbf{G} = (G_0, \ldots, G_{n-1})$ be a vector of n generators of $\mathbb{G}$.
Let $\mathrm{pp} = \mathbf{G}$ be the public parameters of Rdx, $\mathbb{x} = V$ the instance and $\mathbb{w} = (\mathbf{v}, r)$ the witness such that $V = \mathbf{G}^{\mathbf{v}}$.
We assume that n is a power of 2 and we denote by μ the value $\log(n)$.

$\mathcal{P}_{\mathsf{Rdx}}(\mathrm{pp}, \mathbb{x}, \mathbb{w}) \to (\Gamma, \mathbf{x})$
  $\Delta \leftarrow [\cdot]$
  $\mathbf{x} \leftarrow [\cdot]$
  $i \leftarrow \mu - 1$
  while $i \geq 0$ **do**
    $(\mathbf{G}_L, \mathbf{G}_R) \leftarrow (\mathbf{G}[:2^i], \mathbf{G}[2^i:])$
    $(\mathbf{v}_L, \mathbf{v}_R) \leftarrow (\mathbf{v}[:2^i], \mathbf{v}[2^i:])$
    $A \leftarrow \mathbf{G}_L^{\mathbf{v}_R}$ ▷ Compute cross-terms A and B.
    $B \leftarrow \mathbf{G}_R^{\mathbf{v}_L}$
    $x \leftarrow h(A, B, V)$
    $\mathbf{G} \leftarrow \mathbf{G}_L \cdot \mathbf{G}_R^{x^{-1}}$
    $\mathbf{v} \leftarrow \mathbf{v}_L + x\mathbf{v}_R$
    $V \leftarrow \mathbf{G}^{\mathbf{v}}$
    $(\mathbf{x}[i], \Delta[i]) \leftarrow (x, (A, B, V))$
    $i \leftarrow i - 1$
  $\Gamma \leftarrow (\Delta, \mathbf{v}[0])$
  return $(\Gamma, \mathbf{x})$

$\mathcal{V}_{\mathsf{Rdx}}(\mathrm{pp}, \mathbb{x}, \Gamma) \to (b, \mathbf{x}, v)$
  $(\Delta, v) \leftarrow \Gamma$
  $V_\mu \leftarrow V$
  $\mathbf{x} \leftarrow [\cdot]$
  $i \leftarrow \mu - 1$
  while $i \geq 0$ **do**
    $(A_i, B_i, V_i) \leftarrow \Delta[i]$
    $x \leftarrow h(\mathrm{pp}, A_i, B_i, V_{i+1})$
    if $V_i \neq (A_i)^x (B_i)^{x^{-1}} V_{i+1}$ **then**
      return $(0, \perp, 0)$
    $\mathbf{x}[i] \leftarrow x$
    $(\mathbf{G}_L, \mathbf{G}_R) \leftarrow (\mathbf{G}[:2^i], \mathbf{G}[2^i:])$
    $\mathbf{G} \leftarrow \mathbf{G}_L \cdot \mathbf{G}_R^{x^{-1}}$
    $i \leftarrow i - 1$
  if $\mathbf{G}[0]^v \neq V_0$ **then**
    return $(0, \perp, 0)$
  return $(1, \mathbf{x}, v)$

Figure 4: The iterative reduction Rdx **from Bootle et al. [19]**

Let $\mathrm{pp} = (G, \mathbf{F})$ be the public parameters of the hiding vector commitment described in Section 6.2.
Let $\mathbb{x}$ denote (V, W, i, j) and $\mathbb{w}$ denote v such that v opens V and W at indices i and j respectively.
Let Ω_V and Ω_W be the proofs that v opens V and W at indices i and j respectively.

$\mathcal{P}_{\mathsf{OEA}}(\mathrm{pp}, \mathbb{x}, \mathbb{w}) \to \Upsilon$
  $(u, \eta, \nu) \leftarrow\$\ \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$
    ▷ $\widehat{V}$ is a vector commitment to $(\mathbf{0}^{i-1} \parallel u \parallel \mathbf{0}^{n-i})$
  $(\widehat{V}, \widehat{\Omega}_V) \leftarrow (F_n^\nu F_i^u, F_{2n-i}^\nu)$
    ▷ $\widehat{W}$ is a hiding commitment to $(\mathbf{0}^{j-1} \parallel u \parallel \mathbf{0}^{n-j})$
  $(\widehat{W}, \widehat{\Omega}_W) \leftarrow (F_n^\eta F_j^u, F_{2n-j}^\eta)$
  $x \leftarrow h(V, W, \widehat{V}, \widehat{W})$
  $c \leftarrow v + xu$
  for $0 \leq k \leq 1$ **do**
    $t_k \leftarrow h(k, V\widehat{V}^x, W\widehat{W}^x, i, j, c)$
  $\Omega \leftarrow (\Omega_V \widehat{\Omega}_V^x)^{t_0} (\Omega_W \widehat{\Omega}_W^x)^{t_1}$
  $\Upsilon \leftarrow (c, \widehat{V}, \widehat{W}, \Omega)$

$\mathcal{V}_{\mathsf{OEA}}(\mathrm{pp}, \mathbb{x}, \Upsilon) \to b$
  $(c, \widehat{V}, \widehat{W}, \Omega) \leftarrow \Upsilon$
  $x \leftarrow h(V, W, \widehat{V}, \widehat{W})$
  for $0 \leq k \leq 1$ **do**
    $t_k \leftarrow h(k, V\widehat{V}^x, W\widehat{W}^x, i, j, c)$
  if $\frac{e((V\widehat{V}^x)^{t_0}, F_{n-i}) e((W\widehat{W}^x)^{t_1}, F_{n-j})}{e(\Omega, G) e(F_0, F_n)^{c(t_0+t_1)}} \neq 1$ **then**
    ▷ (t_0, t_1, c, Ω) does not satisfy Equation 1
    return 0
  else
    return 1

Figure 5: Opening Equality Argument OEA

6.3 Opening Equality Argument

Let $\mathbf{F} = (F_0, ..., F_n)$ be the $n+1$ first generators from the public parameters of the hiding vector commitments described in Section 6.2. An opening equality argument for our instantiation should prove in zero-knowledge that two Pointproofs vector commitments V and W open to the same value v at indices i and j respectively. This corresponds to proving the following equalities in zero-knowledge:

$$e(V, F_{n-i}) = e(\Omega_V, G) e(F_0, F_n)^v$$
$$e(W, F_{n-j}) = e(\Omega_W, G) e(F_0, F_n)^v$$

Where Ω_V and Ω_W are the opening proofs for triples (i, v, V) and (j, v, W) respectively. Since these equalities involve pairings, proving them in zero-knowledge with Schnorr proofs [20] or Groth-Sahai proofs [21] is costly. Instead, we propose a zero-knowledge argument *tailored for Pointproofs* that circumvents zero-knowledge verification of pairing equalities. Namely, the prover need not prove knowledge of Ω_V and Ω_W to convince the verifier that V and W open to the same value v, at indices i and j.

Let $\widehat{V}$ and $\widehat{W}$ be Pointproofs vector commitments that commit to the same value u at indices i and j. This means that there exists

unique $(\widehat{\Omega}_V, \widehat{\Omega}_W) \in \mathbb{G} \times \mathbb{G}$ such that:

$$e(\widehat{V}, F_{n-i}) = e(\widehat{\Omega}_V, G)e(F_0, F_n)^u$$
$$e(\widehat{W}, F_{n-j}) = e(\widehat{\Omega}_W, G)e(F_0, F_n)^u$$

By the additive homomorphism of Pointproofs, $V\widehat{V}^x$ and $W\widehat{W}^x$ are commitments that open to the same value $c = v + xu$ at indices i and j, for any $x \in \mathbb{Z}_p$. This can be expressed as follows.

$$e(V\widehat{V}^x, F_{n-i}) = e(\Omega_V \widehat{\Omega}_V^x, G)e(F_0, F_n)^c$$
$$e(W\widehat{W}^x, F_{n-j}) = e(\Omega_W \widehat{\Omega}_W^x, G)e(F_0, F_n)^c$$

Thanks to cross-commitment aggregation in Pointproofs, we aggregate these equalities into one:

$$e((V\widehat{V}^x)^{t_0}, F_{n-i})e((W\widehat{W}^x)^{t_1}, F_{n-j}) = e(\Omega, G)e(F_0, F_n)^{(t_0+t_1)c} \tag{1}$$

where $t_k = h(k, V\widehat{V}^x, W\widehat{W}^x, i, j, c),\ k \in \{0, 1\}$ and

$$\Omega = (\Omega_V \widehat{\Omega}_V^x)^{t_0} (\Omega_W \widehat{\Omega}_W^x)^{t_1}.$$

The idea of our argument is that if u is random, then (c, Ω) can be sent in the clear, enabling the verifier to check Equation 1 directly. Figure 5 describes our opening equality argument in more detail.

Now by the soundness of the aggregation of Pointproofs, c opens $V\widehat{V}^x$ and $W\widehat{W}^x$ at indices i and j respectively. Let v denote the i^{th} element of the vector committed in V and w the j^{th} element of the vector committed in W. Let $\hat{v}$ denote the i^{th} element of the vector committed in $\widehat{V}$ and $\hat{w}$ the j^{th} element of the vector committed in $\widehat{W}$. By the binding property of Pointproofs, $c = v + x\hat{v} = w + x\hat{w}$. If x is random (in particular, if x is computed as $h(V, W, \hat{V}, \hat{W})$), then the Shwartz-Zippel lemma guarantees that $v = w$ and $\hat{v} = \hat{w}$ with all but negligible probability $1/p$.

Zero-knowledge, on the other hand, follows from the fact that a simulator with knowledge of the trapdoor α of Pointproofs and control over a random oracle, can randomly select c, $\widehat{W}$ and $\widehat{V}$, and successfully find Ω that satisfies Equation 1.

Theorem 6.1. *The argument described in Figure 5 is an opening equality argument under the security of Pointproofs in ROM.*

The proof of Theorem 6.1 is deferred to Appendix A.1.

6.4 Sum Argument

To instantiate a sum argument over vector commitments for our PoL, it is sufficient to devise a zero-knowledge argument for relation

$$\mathbb{x} = V\ ;\ \mathbb{w} = (\mathbf{v}, r) \in \mathbb{Z}_p^n \times \mathbb{Z}_p :\ V = F^r \mathbf{G}^{\mathbf{v}} \wedge \mathbf{v}[n-1] = \sum_{i=0}^{n-2} \mathbf{v}[i].$$

Where $\mathbf{G} = (G_0, ..., G_{n-1})$ and F are $n+1$ random generators[3] of $\mathbb{G}$.

While the sum argument is also given by functional commitment (FC) schemes for linear functions, we note that some FC schemes use composite groups (e.g. [22]) which is less efficient. We leave the question of efficiently implementing the SA in the context of PoL using FC schemes as future work.

[3] V here is a hiding Pedersen commitment and not a Pointproof one. We note though that a Pointproofs hiding vector commitment is an instantiation of a Pedersen commitment with $\mathbf{G} = (F_0, ..., F_{n-1}) = (G^\alpha, ..., G^{\alpha^n})$ and $F = F_n = G^{\alpha^{n+1}}$.

Observe that $\mathbf{v}[n-1] = \sum_{i=0}^{n-2} \mathbf{v}[i]$ is equivalent to the inner product $\mathbf{v} \cdot \mathbf{b}$ being zero, where $\mathbf{b}$ is the n-dimensional vector $(1, ..., 1, -1)$. We therefore leverage an *inner product argument* IPA to realize the sum argument. Inner product arguments, however, do not satisfy zero-knowledge if the IPA's prover is called on vectors $\mathbf{v}$ and $\mathbf{b}$ since the resulting proof will leak information about these vectors (in particular, about the secret vector $\mathbf{v}$). Similar to [12], we prevent leakage by blinding vector $\mathbf{v}$ and then running the IPA's prover on the blinded vector.

Figure 6 depicts our sum argument in details. Specifically, the sum argument prover selects a random vector $\mathbf{w}$ of n elements (line 2) and computes a Pedersen commitment $W = F^{r'}\mathbf{G}^{\mathbf{w}}$ (line 3) and the inner product $c = \mathbf{w} \cdot \mathbf{b}$ (line 4). Then it computes the challenge x as the hash of (c, V, W) (line 5).

Let $\mathbf{a}$ be the vector defined as $\mathbf{v} + x\mathbf{w}$. If $\mathbf{v} \cdot \mathbf{b} = 0$, then $\mathbf{a} \cdot \mathbf{b} = cx$. Thanks to Schwartz-Zippel lemma, if $\mathbf{a} \cdot \mathbf{b} = cx$, then $\mathbf{v} \cdot \mathbf{b} = 0$ with all but negligible probability $1/p$.

Let $\mathbf{H} = (H_0, ..., H_{n-1})$ be n random generators of $\mathbb{G}$ and let B denote the Pedersen commitment $\mathbf{H}^{\mathbf{b}}$.

Now the sum argument prover prepares the inputs for the IPA's prover (lines 6 & 7). This consists of computing Pedersen commitment $P = \mathbf{G}^{\mathbf{a}}\mathbf{H}^{\mathbf{b}}$, which can be easily shown to be equal to $F^\rho V W^x B$ for $\rho = -r - xr'$, where r and r' are the randomness used in the computation of V and W respectively. Once P is computed, the IPA's prover is called to prove that $\mathbf{a} \cdot \mathbf{b}$ actually equals cx (line 10).

The verification of the sum argument proceeds with computing the challenge x (line 15) and then checking the validity of the IPA's proof in relation to $P = F^\rho V W^x B$ and cx (line 18). If the IPA's proof is valid, then the sum argument's verifier accepts.

Theorem 6.2. *The argument described above is a sum argument in the ROM under the discrete logarithm assumption.*

Proof of Theorem 6.2 can be found in Appendix A.2.

Furthermore, Section 7.2 details a method that aggregates sum arguments in such a way that one runs a single sum argument to prove that m vectors $\mathbf{v}_i$ all verify $\mathbf{v}_i[n-1] = \sum_{j=0}^{n-2} \mathbf{v}_i[j]$.

6.5 Range Proofs

We wish to prove that the values committed to in a vector commitment are positive. Existing protocols for range proofs, however, take as input Pedersen commitments to a single value, e.g., [12]. Using them directly in our construction will increase the size of the proofs of liability; namely, the size of a proof of liability will linearly grow with both the arity of the sparse Verkle tree and its depth. We thus introduce a range proof that takes as input a Pedersen-like vector commitment to a vector $\mathbf{v} = (v_0, ..., v_{n-1}) \in \mathbb{Z}_p^n$ and proves that for all $i \in [n]$, $0 \le v_i < max$. The communication complexity of this proof is $O(\log(n \cdot max)) = O(\log(n) + \log(max))$.

6.5.1 Overview. Our range proof proceeds as follows. First, we use a standard split-and-fold technique [12, 19] to iteratively reduce the commitment to a vector $\mathbf{v} = (v_0, ..., v_{n-1})$ to a commitment to a single value v. By construction

$$v = \sum_{i=0}^{n-1} f_i(x_0, ..., x_{\log(n)-1}) v_i \tag{2}$$

where $(x_0, ..., x_{\log(n)-1})$ are the challenge sent during the reduction.

Let F be a generator of $\mathbb{G}$, $\mathbf{G} = (G_0, ..., G_{n-1})$ and $\mathbf{H} = (H_0, ..., H_{n-1})$ be two vectors of n generators of $\mathbb{G}$.
Let $\mathbf{b}$ be the n-length vector $(1, ..., 1, -1)$, and B the corresponding commitment $\mathbf{H}^{\mathbf{b}}$.
Let $\mathsf{pp}_{\mathsf{IPA}}$ be the public parameters of an inner product argument for vectors of length n, committed to using generators $\mathbf{G}$ and $\mathbf{H}$.
Let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{IPA}}, F, \mathbf{G}, \mathbf{H}, B)$ be the public parameters of SA, $\mathbb{x} = V$ the instance and $\mathbb{w} = (\mathbf{v}, r)$ the corresponding witness, such that $V = F^r\mathbf{G}^{\mathbf{v}} \wedge \mathbf{v}[n-1] = \sum_{j=0}^{n-2} \mathbf{v}[j]$.

1: $\mathcal{P}_{\mathsf{SA}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \to \Phi$
2: $(\mathbf{w}, r') \leftarrow\$ \mathbb{Z}_p^n \times \mathbb{Z}_p$
3: $W \leftarrow F^{r'}\mathbf{G}^{\mathbf{w}}$ ▷ hiding commitment to $\mathbf{w}$
4: $c \leftarrow \mathbf{w} \cdot \mathbf{b}$
5: $x \leftarrow h(c, V, W)$
6: $\rho \leftarrow -r - r'x$
7: $P \leftarrow F^{\rho}VW^{x}B$ ▷ commitment to $\mathbf{a} = \mathbf{v} + x\mathbf{w}$ and $\mathbf{b}$
8: $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, cx)$ ▷ $\mathbf{v} \cdot \mathbf{b} = 0 \implies \mathbf{a} \cdot \mathbf{b} = cx$
9: $\mathbb{w}_{\mathsf{IPA}} \leftarrow (\mathbf{a}, \mathbf{b})$
10: $\Pi \leftarrow \mathcal{P}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \mathbb{w}_{\mathsf{IPA}})$
11: $\Phi \leftarrow (\Pi, W, c, \rho)$
12: return Φ

13: $\mathcal{V}_{\mathsf{SA}}(\mathsf{pp}, \mathbb{x}, \Phi) \to b$
14: $(\Pi, W, c, \rho) \leftarrow \Phi$
15: $x \leftarrow h(c, V, W)$
16: $P \leftarrow F^{\rho}VW^{x}B$
17: $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, cx)$
18: return $\mathcal{V}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \Pi)$

Figure 6: Sum Argument SA

To prove that $0 \le \mathbf{v}[i] < 2^m$, we write v_i as $\sum_{j=0}^{m-1} v_{i,j}2^j$ and compute a commitment to $v_{i,j}$. Next we prove in zero-knowledge that each $v_{i,j}$ is a bit. What remains is to show that for all $i \in [i]$, $(v_{i,j})_{j\in[m]}$ actually verify $v_i = \sum_{j=0}^{m-1} v_{i,j}2^j$. To this end, we replace v_i by $\sum_{j=0}^{m-1} v_{i,j}2^j$ in Eq. 2.

To optimize the range proof, we follow the approach of Bulletproofs [12] that uses a *single* inner product argument to show that $v_{i,j}$ are bits and that they satisfy $v_i = \sum_{j=0}^{m-1} v_{i,j}2^j$. Note that thanks to Eq. 2, we are able to prove that a committed vector is in the valid range in one go. Bulletproofs, on the other hand, only accommodate range proofs on single value commitments.

6.5.2 The Iterative Reduction. As previously mentioned, we first reduce vector $\mathbf{v} = (v_0, ..., v_{n-1})$ to a single value. To this end, we use the iterative reduction technique from Bootle et al. [19], which consists of $\mu = \log(n)$ iterations, such that at the end of each iteration, the length of $\mathbf{v}$ is halved. The reduction concludes by outputting a single value v that is a function of $\{v_i\}_{i\in[n]}$ and a sequence of challenges $\{x_k\}_{k\in[\mu]}$. We denote this protocol by $\mathsf{Rdx} = (\mathcal{P}_{\mathsf{Rdx}}, \mathcal{V}_{\mathsf{Rdx}})$.

Theorem 6.3. *Let $\mathbf{v} \in \mathbb{Z}_p^n$ and $\mu = \log(n)$. Given a sequence of uniformly random $x_k \in \mathbb{Z}_p$ for $k \in [\mu - 1]$, the reduction protocol* Rdx *(Figure 4) reduces* $\mathbf{v}$ *to the value*

$$v = \sum_{i=0}^{n-1} v_i \left(\prod_{k=0}^{\mu-1} x_k^{Bits(i)[k]} \right). \tag{3}$$

where $Bits(i) = (b_0, b_1, \ldots, b_{\mu-1})$ is the bit representation of i and b_0 is the least significant bit.

Since $\mathbf{v}$ is reduced to a single value v which can be described using a closed-form function (Equation 3), the prover can prove statements about $\mathbf{v}$ using v instead. For a complete description refer to Figure 4 or Bootle et al. [19].

6.5.3 The Range Proof. The goal is to prove that the vector $\mathbf{v}$ committed in $V = F^r \prod_{i=0}^{n-1} G_i^{\mathbf{v}[i]}$ verifies the following: $\forall i \in [n] : 0 \le \mathbf{v}[i] < max$. Assume that $max = 2^m$. Proving that $0 \le \mathbf{v}[i] < max$ is tantamount to showing that $\mathbf{v}[i] = \sum_{j=0}^{m-1} 2^j v_{i,j}$ for some $v_{i,j} \in \{0, 1\}$. Figure 7 depicts our range proof over vector commitments.

We start our range proof by executing $\mathcal{P}_{\mathsf{Rdx}}$ on $\mathbf{v}$. This yields a value v satisfying Equation 3. Calling $\mathcal{P}_{\mathsf{Rdx}}$ on $\mathbf{v}$, however, is not zero knowledge. In particular, if $\mathcal{P}_{\mathsf{Rdx}}$ is invoked n times on $\mathbf{v}$, then one could setup a system of n linear equations with n unknowns and easily recover $\mathbf{v}$.

To mitigate this attack, we blind $\mathbf{v}$ using a random vector $\mathbf{w}$. More precisely, we compute a challenge x and run $\mathcal{P}_{\mathsf{Rdx}}$ on input vector $\mathbf{u} = \mathbf{v} + x\mathbf{w}$ and commitment U (line 14). This returns the proof $\Gamma = (\Delta, u)$ where $u = \sum_{i=0}^{n-1} \mathbf{u}[i]f_i$, and $f_i = \prod_{k=0}^{\mu-1} x_k^{\mathsf{Bits}(i)[k]}, \forall i \in [n]$. Given that $\mathbf{u} = \mathbf{v} + x\mathbf{w}$ and $\mathbf{v}[i] = \sum_{j=0}^{m-1} 2^j v_{i,j}$, this equality could be re-written as

$$u = \sum_{i=0}^{n-1} \mathbf{v}[i]f_i + x\mathbf{w}[i]f_i = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} v_{i,j}f_i 2^j + \sum_{i=0}^{n-1} x\mathbf{w}[i]f_i$$

Let $\hat{\mathbf{v}} = (v_{0,0}, ..., v_{n-1,m-1} \parallel \mathbf{w})$, $\mathbf{f} = (f_0, ..., f_{n-1})$ and $\mathbf{d} = (d_{i,j} \parallel x\mathbf{f})$ with $d_{i,j} = f_i 2^j$ for all $i \in [n]$ and $j \in [m]$. Value u can thus be expressed as the inner product of $\hat{\mathbf{v}}$ and $\mathbf{d}$, i.e.,

$$u = \hat{\mathbf{v}} \cdot \mathbf{d} \tag{4}$$

Proving the correctness of this inner product alone is not sufficient to guarantee that all v's elements are in the correct range. Actually, we must additionally show that the $v_{i,j}$'s are bits.

Let $\hat{\mathbf{v}}[:nm] = (v_{0,0}, ..., v_{n-1,m-1})$ and $\hat{\mathbf{w}}$ its bit complement (lines 5 and 6, respectively). If $\hat{\mathbf{v}}[:nm]$ is indeed comprised of bits, then the following equalities always hold:

$$\hat{\mathbf{w}} - \mathbf{1}^{nm} + \hat{\mathbf{v}}[:nm] = \mathbf{0}^{nm} \tag{5}$$

$$\hat{\mathbf{v}}[:nm] \circ \hat{\mathbf{w}} = \mathbf{0}^{nm} \tag{6}$$

From here onward, our approach proceeds similarly to that of Bulletproofs, in that, we express Equations 4, 5 and 6 as the inner product of two vectors that are functions of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$, and then call the IPA on these two vectors. In the remainder of this section, we show how we obtain these two vectors.

Let (G, H, F) be three generators of $\mathbb{G}$, $\mathbf{G} = (G_0, ..., G_{n-1})$ be a vector of n generators.
Let $\mathbf{H} = (H_0, ..., H_{n(m+1)-1})$ and $\mathbf{F} = (F_0, ..., F_{n(m+1)-1})$ be two vectors of $n(m+1)$ generators of $\mathbb{G}$.
Let $\mathsf{pp}_{\mathsf{IPA}}$ be the public parameters of an inner product argument of vectors of length $n(m+1)$.
Let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{IPA}}, G, H, F, \mathbf{G}, \mathbf{H}, \mathbf{F})$ be the public parameters of argument RP, $\mathbb{x} = V$ the instance and $\mathbb{w} = (\mathbf{v}, r)$ the corresponding witness, such that $V = F^r \mathbf{G}^{\mathbf{v}}$, $\mathbf{v}[i] = \sum_{j=0}^{m-1} v_{i,j} 2^j$ and $v_{i,j} \in \{0, 1\}$ for all $i \in [n]$ and $j \in [m]$.

$\mathcal{P}_{\mathsf{RP}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \to \Psi$
  $(\mathbf{w}, r') \leftarrow\$ \mathbb{Z}_p^n \times \mathbb{Z}_p$
  $W \leftarrow F^{r'} \mathbf{G}^{\mathbf{w}}$
  $\hat{\mathbf{v}}_i \leftarrow \mathsf{Bits}(\mathbf{v}[i])$
  $\hat{\mathbf{v}} \leftarrow (\hat{\mathbf{v}}_0 \parallel ... \parallel \hat{\mathbf{v}}_{n-1} \parallel \mathbf{w})$
  $\hat{\mathbf{w}} \leftarrow (\mathbf{1}^{nm} - \hat{\mathbf{v}}[: nm])$
  $\nu \leftarrow\$ \mathbb{Z}_p$
  $Q \leftarrow F^{\nu} \mathbf{H}^{\hat{\mathbf{v}}} \mathbf{F}[: nm]^{\hat{\mathbf{w}}}$
  $x \leftarrow h(V, W, Q)$
  $\gamma \leftarrow -r - xr'$
  $U \leftarrow F^{\gamma} V W^x$ ▷ commitment to $\mathbf{v} + x\mathbf{w}$
  $\mathsf{pp}_{\mathsf{Rdx}} \leftarrow \mathbf{G}$
  $(\mathbb{w}_{\mathsf{Rdx}}, \mathbb{x}_{\mathsf{Rdx}}) \leftarrow (\mathbf{v} + x\mathbf{w}, U)$
  $(\Gamma, \mathbf{x}) \leftarrow \mathcal{P}_{\mathsf{Rdx}}(\mathsf{pp}_{\mathsf{Rdx}}, \mathbb{x}_{\mathsf{Rdx}}, \mathbb{w}_{\mathsf{Rdx}})$
  $(x_0, ..., x_{\mu-1}) \leftarrow \mathbf{x}$
  for $0 \le i \le n-1$ **do**
    $f_i \leftarrow \prod_{k=0}^{\mu-1} x_k^{\mathsf{Bits}(i)[k]}$
  $\mathbf{f} \leftarrow (f_0, ..., f_{n-1})$
  for $0 \le i \le n-1$ **do**
    for $0 \le j \le m-1$ **do**
      $d_{i,j} \leftarrow 2^j f_i$
  $\mathbf{d} \leftarrow (d_{0,0}, ..., d_{n-1,m-1} \parallel x\mathbf{f})$
  $(\eta, \mathbf{s}, \mathbf{t}) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p^{nm+n} \times \mathbb{Z}_p^{nm}$
  $R \leftarrow F^{\eta} \mathbf{H}^{\mathbf{s}} \mathbf{F}[: nm]^{\mathbf{t}}$
  $(y_0, y_1) \leftarrow (h(U, Q, R, 0), h(U, Q, R, 1))$
  $(\mathbf{y}_0, \mathbf{y}_1) \leftarrow ((1, y_0 ..., y_0^{nm-1}), y_1 \mathbf{1}^{nm})$
  $\mathbf{a}' \leftarrow \hat{\mathbf{v}} + (\mathbf{y}_1 \| \mathbf{0}^n)$
  $\mathbf{b}' \leftarrow y_1^2 \mathbf{d} + y_1(\mathbf{y}_0 \parallel \mathbf{0}^n) + ((\hat{\mathbf{w}} \circ \mathbf{y}_0) \parallel \mathbf{0}^n)$
  $c_1 \leftarrow \mathbf{a}'[: nm] \cdot (\mathbf{y}_0 \circ \mathbf{t}) + \mathbf{s} \cdot \mathbf{b}'$
  $c_2 \leftarrow \mathbf{s}[: nm] \cdot (\mathbf{y}_0 \circ \mathbf{t})$
  $(\tau_1, \tau_2) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p$
  $(C_1, C_2) \leftarrow (G^{c_1} H^{\tau_1}, G^{c_2} H^{\tau_2})$
  $z \leftarrow h(U, Q, R, C_1, C_2)$
  $\rho \leftarrow -\nu - \eta z$
  $\tau \leftarrow \tau_1 z + \tau_2 z^2$
  $\mathbf{F}' \leftarrow (F_0, F_1^{y_0^{-1}}, ..., F_{n(m+1)-1}^{y_0^{-(n(m+1)-1)}})$
  $(\mathbf{a}, \mathbf{b}) \leftarrow (\mathbf{a}' + z\mathbf{s}, \mathbf{b}' + z((\mathbf{y}_0 \circ \mathbf{t}) \parallel \mathbf{0}^n))$
    ▷ P is a commitment to $\mathbf{a}$ and $\mathbf{b}$ with generators $\mathbf{H}$ and $\mathbf{F}'$
  $P \leftarrow F^{\rho} Q R^z \mathbf{H}[: nm]^{\mathbf{y}_1} \mathbf{F}'^{y_1^2 \mathbf{d}} \mathbf{F}[: nm]^{\mathbf{y}_1}$
  $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, c = \mathbf{a} \cdot \mathbf{b})$
  $\mathbb{w}_{\mathsf{IPA}} \leftarrow (\mathbf{a}, \mathbf{b})$
  $\Pi \leftarrow \mathcal{P}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \mathbb{w}_{\mathsf{IPA}})$
  return $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho, \tau)$

$\mathcal{V}_{\mathsf{RP}}(\mathsf{pp}, \mathbb{x}, \Psi) \to b$
  $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho, \tau) \leftarrow \Psi$
  $x \leftarrow h(V, W, Q)$
  $U \leftarrow F^{\gamma} V W^x$
  $\mathsf{pp}_{\mathsf{Rdx}} \leftarrow \mathbb{G}$
  $\mathbb{x}_{\mathsf{Rdx}} \leftarrow U$
  $(b_0, \mathbf{x}, u) \leftarrow \mathcal{V}_{\mathsf{Rdx}}(\mathsf{pp}_{\mathsf{Rdx}}, \mathbb{x}_{\mathsf{Rdx}}, \Gamma)$
  if $b_0 = 0$ **then**
    return 0
  for $0 \le i \le n-1$ **do**
    $f_i \leftarrow \prod_{k=0}^{\mu-1} x_k^{\mathsf{Bits}(i)[k]}$
  $\mathbf{f} \leftarrow (f_0, ..., f_{n-1})$
  for $0 \le i \le n-1$ **do**
    for $0 \le j \le m-1$ **do**
      $d_{i,j} \leftarrow 2^j f_i$
  $\mathbf{d} \leftarrow (d_{0,0}, ..., d_{n-1,m-1} \parallel x\mathbf{f})$
  $(y_0, y_1) \leftarrow (h(U, Q, R, 0), h(U, Q, R, 1))$
  $(\mathbf{y}_0, \mathbf{y}_1) \leftarrow ((1, y_0 ..., y_0^{nm-1}), y_1 \mathbf{1}^{nm})$
  $z \leftarrow h(U, Q, R, C_1, C_2)$
  $(\beta_1, \beta_2, \beta_3) \leftarrow (\mathbf{1}^{nm} \cdot \mathbf{y}_0, \mathbf{1}^{nm} \cdot \mathbf{y}_0 + u, \mathbf{1}^{nm} \cdot \mathbf{d}[: nm])$
  $c_0 \leftarrow \beta_3 y_1^3 + \beta_2 y_1^2 + \beta_1 y_1$
  if $C_1^z C_2^{z^2} G^{c_0} \neq G^c H^{\tau}$ **then**
    return 0
  $\mathbf{F}' \leftarrow (F_0, F_1^{y_0^{-1}}, ..., F_{n(m+1)-1}^{y_0^{-(n(m+1)-1)}})$
  $P \leftarrow F^{\rho} Q R^z \mathbf{H}[: nm]^{\mathbf{y}_1} \mathbf{F}'^{y_1^2 \mathbf{d}} \mathbf{F}[: nm]^{\mathbf{y}_1}$
  $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, c)$
  return $\mathcal{V}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \Pi)$

Figure 7: Range proof over vector commitments RP.

To combine multiple equations into one, we can take a random linear combination of those constraints as chosen by the verifier. In particular, note that if $\mathbf{e} = \mathbf{0}^{nm}$, then $\forall \mathbf{y}_0 = (1, y_0, \ldots, y_0^{nm-1})$, $\mathbf{e} \cdot \mathbf{y}_0 = 0$. We can thus re-write Equations 5 and 6 as

$$(\hat{\mathbf{w}} - \mathbf{1}^{nm} + \hat{\mathbf{v}}[: nm]) \cdot \mathbf{y}_0 = 0 \tag{7}$$

$$(\hat{\mathbf{v}}[: nm] \circ \hat{\mathbf{w}}) \cdot \mathbf{y}_0 = \hat{\mathbf{v}}[: nm] \cdot (\hat{\mathbf{w}} \circ \mathbf{y}_0) = 0. \tag{8}$$

Equalities 4, 7 and 8 can be further combined into a single equality by applying the same technique again using some $y_1 \in \mathbb{Z}_p$:

$$y_1^2(\hat{\mathbf{v}} \cdot \mathbf{d} - u) + y_1(\hat{\mathbf{w}} - \mathbf{1}^{nm} + \hat{\mathbf{v}}[: nm]) \cdot \mathbf{y}_0 + \hat{\mathbf{v}}[: nm] \cdot (\hat{\mathbf{w}} \circ \mathbf{y}_0) = 0 \tag{9}$$

Let $\mathbf{a}'$ and $\mathbf{b}'$ be two vectors defined as:

$$\mathbf{a}' = \hat{\mathbf{v}} + (\mathbf{y}_1 \| \mathbf{0}^n)$$
$$\mathbf{b}' = y_1^2 \mathbf{d} + y_1(\mathbf{y}_0 \| \mathbf{0}^n) + ((\hat{\mathbf{w}} \circ \mathbf{y}_0) \| \mathbf{0}^n)$$

Accordingly, inner product $\mathbf{a}' \cdot \mathbf{b}'$ corresponds to

$$y_1^2 \hat{\mathbf{v}} \cdot \mathbf{d} + y_1 \hat{\mathbf{v}} \cdot (\mathbf{y}_0 \| \mathbf{0}^n) + \hat{\mathbf{v}} \cdot ((\hat{\mathbf{w}} \circ \mathbf{y}_0) \| \mathbf{0}^n) + y_1^2 (\mathbf{y}_1 \| \mathbf{0}^n) \cdot \mathbf{d} + y_1 (\mathbf{y}_0 \| \mathbf{0}^n) \cdot (\mathbf{y}_1 \| \mathbf{0}^n) + (\mathbf{y}_1 \| \mathbf{0}^n) \cdot ((\hat{\mathbf{w}} \circ \mathbf{y}_0) \| \mathbf{0}^n).$$

Note that if Equation 9 holds, then:

$$\mathbf{a}' \cdot \mathbf{b}' = (\mathbf{1}^{nm} \cdot \mathbf{y}_0) y_1^3 + (\mathbf{1}^{nm} \cdot \mathbf{y}_0 + u) y_1^2 + (\mathbf{1}^{nm} \cdot \mathbf{d}[: nm]) y_1.$$

Since the IPA is not zero-knowledge, the prover cannot call it with $\mathbf{a}'$ and $\mathbf{b}'$, as this would leak information about $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$. Instead, it invokes it with blinded vectors $\mathbf{a}$ and $\mathbf{b}$ such that the constant term of $\langle \mathbf{a}, \mathbf{b} \rangle$ is actually $\langle \mathbf{a}', \mathbf{b}' \rangle$. Thus, proving that the inner product $\langle \mathbf{a}, \mathbf{b} \rangle$ is computed correctly implies that $\langle \mathbf{a}', \mathbf{b}' \rangle$ is also correct. We compute $\mathbf{a}$ and $\mathbf{b}$ as follows.

$$\mathbf{a} = \mathbf{a}' + z\mathbf{s} = (\hat{\mathbf{v}} + z\mathbf{s}) + (\mathbf{y}_1 \| \mathbf{0}^n)$$

$$\mathbf{b} = \mathbf{b}' + z(\mathbf{y}_0 \circ \mathbf{t}) \| \mathbf{0}^n = y_1^2 \mathbf{d} + y_1(\mathbf{y}_0 \| \mathbf{0}^n) + ((\hat{\mathbf{w}} + z\mathbf{t}) \circ \mathbf{y}_0) \| \mathbf{0}^n$$

where z is a random element in $\mathbb{Z}_p$ and $\mathbf{s}$ and $\mathbf{t}$ are two random vectors of length $nm + n$ and nm respectively.

We now show how to compute y_0, y_1, z in a way that guarantees the soundness of the range proof. Let $\mathbf{H} = (H_0, ..., H_{n(m+1)-1})$ and $\mathbf{F} = (F_0, ..., F_{n(m+1)-1})$ be two vectors of $n(m+1)$ generators of $\mathbb{G}$. The prover first computes $Q = F^{\nu}\mathbf{H}^{\hat{\mathbf{v}}}\mathbf{F}[:nm]^{\hat{\mathbf{w}}}$ a commitment to vectors $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ (line 8), and a commitment $R = F^{\eta}\mathbf{H}^{\mathbf{s}}\mathbf{F}[:nm]^{\mathbf{t}}$ to vectors $\mathbf{s}$ and $\mathbf{t}$ (line 24). It then computes y_0 and y_1 as $h(U, Q, R, 0)$ and $h(U, Q, R, 1)$ respectively (lines 25). Notice that

$$\mathbf{a} \cdot \mathbf{b} = c_0 + c_1 z + c_2 z^2, \text{ where}$$
$$c_0 = \mathbf{a}' \cdot \mathbf{b}' \; ; \; c_1 = \mathbf{a}'[:nm] \cdot (\mathbf{y}_0 \circ \mathbf{t}) + \mathbf{s} \cdot \mathbf{b}'$$
$$c_2 = \mathbf{s}[:nm] \cdot (\mathbf{y}_0 \circ \mathbf{t})$$

Correspondingly, the prover computes commitments $C_1 = G^{c_1}H^{\tau_1}$ and $C_2 = G^{c_2}H^{\tau_2}$ for random τ_1 and τ_2 (line 32), and then sets z to $h(U, Q, R, C_1, C_2)$ (line 33).

The next step of the prover is to compute for the IPA the commitment of vectors $\mathbf{a}$ and $\mathbf{b}$ as a function of the commitments Q and R. Notice that Q is a commitment to vectors $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}} \circ \mathbf{y}_0$ using generators $\mathbf{H}$ and $\mathbf{F}' = (F_0, F_1^{y_0^{-1}}, ..., F_{n(m+1)-1}^{y_0^{-(n(m+1)-1)}})$ (line 36). Now to produce a commitment P for vectors $\mathbf{a}$ and $\mathbf{b}$ with generators $\mathbf{H}$ and $\mathbf{F}'$, the prover computes $\rho = -\nu - \eta z$ and sets P to $F^{\rho}QR^{z}\mathbf{H}[:nm]^{\mathbf{y}_1}\mathbf{F}'^{\,y_1^2\mathbf{d}}\mathbf{F}[:nm]^{\mathbf{y}_1}$ (line 38).

The prover concludes by calling $\mathcal{P}_{\text{IPA}}$ on P and $c = \mathbf{a} \cdot \mathbf{b}$.

To verify the correctness, $\mathcal{V}$ proceeds as follows:

1. Run the Rdx verifier to check the correctness of u relative to commitment U and obtain the sequence of challenges $\mathbf{x}$. If the Rdx verifier rejects, then reject (lines $49 - 51$).
2. Compute $\mathbf{d}$ as a function of challenges x and $\mathbf{x}$ (lines $52 - 58$).
3. Compute $c_0 = (\mathbf{1}^{nm} \cdot \mathbf{y}_0)y_1 + (\mathbf{1}^{nm} \cdot \mathbf{y}_0 + u)y_1^2 + (\mathbf{1}^{nm} \cdot \mathbf{d}[:nm])y_1^3$ (line $62 - 63$).
4. Check that c is well formed by verifying that $C_1^z C_2^{z^2} G^{c_0} = G^{c_0}H^{\tau}$. If the equation does not hold, reject (lines $64 - 65$).
5. Compute commitment P as a function of commitments Q and R and run the IPA's verifier on P and c, and with generators $\mathbf{H}$ and $\mathbf{F}'$ (lines $66 - 69$). If the IPA's verifier accepts, then accept.

Theorem 6.4. *The proof system described in Figure 7 is a range proof over hiding vector commitments in the ROM under the discrete logarithm assumption.*

The proof of Theorem 6.4 can be found in Appendix A.4.

7 OPTIMIZATIONS

7.1 Aggregating Opening Equality Arguments

We now describe how to aggregate OEA to reduce the number of pairings performed during PoL verification by almost half. Let $\mathbf{V} = (V_0, ..., V_{l-1})$ and $\mathbf{W} = (W_0, ..., W_{l-1})$ be vectors in $\mathbb{G}^l$ such that $\forall k \in [l]$, V_k and W_k commit to the same value v_k at indices i_k and j_k. This translates to the following: $\exists!(\Omega_{V,k}, \Omega_{W,k}),\ k \in [l]$ such that:

$$\forall k \in [l] : e(V_k, F_{n-i_k}) = e(\Omega_{V,k}, G)e(F_0, F_n)^{v_k}$$
$$e(W_k, F_{n-j_k}) = e(\Omega_{W,k}, G)e(F_0, F_n)^{v_k}$$

Let $\widehat{\mathbf{V}} = (\widehat{V}_0, ..., \widehat{V}_{l-1})$ and $\widehat{\mathbf{W}} = (\widehat{W}_0, ..., \widehat{W}_{l-1})$ be vectors in $\mathbb{G}^l$ such that $\forall k \in [l]$: $\widehat{V}_k$ and $\widehat{W}_k$ commits to the same value u_k at indices i_k and j_k. Therefore, $\forall k \in [l], \exists!(\widehat{\Omega}_{V,k}, \widehat{\Omega}_{W,k})$ such that:

$$e(\widehat{V}_k, F_{n-i_k}) = e(\widehat{\Omega}_{V,k}, G)e(F_0, F_n)^{u_k}$$
$$e(\widehat{W}_k, F_{n-j_k}) = e(\widehat{\Omega}_{W,k}, G)e(F_0, F_n)^{u_k}$$

The additive homomorphism of Pointproofs implies that for any $x \in \mathbb{Z}_p$, $a_k = v_k + xu_k$ opens commitments $V_k\widehat{V}_k^x$ and $W_k\widehat{W}_k^x$ at indices i_k and j_k respectively. More precisely, we have

$$\forall k \in [l] : e(V_k\widehat{V}_k^x, F_{n-i_k}) = e(\Omega_{V,k}\widehat{\Omega}_{V,k}^x, G)e(F_0, F_n)^{a_k}$$
$$e(W_k\widehat{W}_k^x, F_{n-j_k}) = e(\Omega_{W,k}\widehat{\Omega}_{W,k}^x, G)e(F_0, F_n)^{a_k}$$

The aggregation of PointProofs allows us to write these verification equations as one equality.

$$\prod_{k=0}^{l-1} e(V'_k, F_{n-i_k})^{t_{2k}} e(W'_k, F_{n-j_k})^{t_{2k+1}} = e(\Omega, G)e(F_0, F_n)^c \qquad (10)$$

Where

$$c = \sum_{k=0}^{l-1}(t_{2k} + t_{2k+1})a_k$$
$$V'_k = V_k\widehat{V}_k^x \; ; \; W'_k = W_k\widehat{W}_k^x$$
$$\Omega = \prod_{k=0}^{l-1}(\Omega_{V,k}\widehat{\Omega}_{V,k}^x)^{t_{2k}}(\Omega_{W,k}\widehat{\Omega}_{W,k}^x)^{t_{2k+1}}$$
$$t_k = h(k, V'_0, ..., V'_{l-1}, W'_0, ..., W'_{l-1}, \mathbf{i}, \mathbf{j}, \mathbf{a})$$
$$\mathbf{i} = (i_1, ..., i_{l-1}) \; ; \; \mathbf{j} = (j_0, ..., j_{l-1}) \; ; \mathbf{a} = (a_0, ..., a_{l-1})$$

Similar to OEA, the idea of the aggregated argument is that if $(u_0, ..., u_{l-1})$ are random, then $(\mathbf{a}, \Omega)$ can be sent in the clear, enabling the verifier to check Equation 10 directly.

The soundness of the aggregation of Pointproofs ensures that for all $k \in [l]$, a_k opens V'_k and W'_k at indices i_k and j_k respectively. Furthermore, if v_k and $\hat{v}_k$ are the i_k^{th} element of the vectors committed in V_k and $\widehat{V}_k$ respectively, and w_k and $\hat{w}_k$ the j_k^{th} element of the vectors committed in W_k and $\widehat{W}_k$ respectively, then the binding property of Pointproofs entails that $a_k = v_k + x\hat{v}_k = w_k + x\hat{w}_k, \forall k \in [l]$.

Now if x is computed as $h(\mathbf{V}, \mathbf{W}, \widehat{\mathbf{V}}, \widehat{\mathbf{W}})$, then we can use the Shwartz-Zippel lemma to show that $v_k = w_k$ and $\hat{v}_k = \hat{w}_k$ with all but negligible probability $1/p$.

This demonstrates that if Equation 10 holds, then for all $k \in [l]$, V_k and W_k commits to the same value v_k at indices i_k and j_k.

Finally, the argument is zero-knowledge. A simulator with knowledge of the trapdoor α of Pointproofs and control over a random oracle, can randomly select vectors $\mathbf{a}$, $\widehat{\mathbf{V}}$ and $\widehat{\mathbf{W}}$, and successfully find Ω that satisfies Equation 10.

Let $\mathsf{pp}_{\mathsf{VC}} = (G, \mathbf{F})$ be the public parameters of the hiding vector commitment described in Section 6.2.
Let $\mathbf{G} = (G_0, ..., G_{l-1}), \mathbf{H} = (H_0, ..., H_{l-1})$ and F be generators of $\mathbb{G}$.
Let $\mathsf{pp}_{\mathsf{IPA}}$ be the public parameters of an inner product argument for vectors of length l, committed to using generators $\mathbf{G}$ and $\mathbf{H}$.
Let $\mathbf{V} = (V_0, ..., V_{l-1})$ and $\mathbf{W} = (W_0, ..., W_{l-1})$ be two vectors in $\mathbb{G}^l$, and $\mathbf{v} = (v_0, ..., v_{l-1})$ be a vector in $\mathbb{Z}_p^l$.
Let $\mathbf{i} = (i_0, ..., i_{l-1})$ and $\mathbf{j} = (j_0, ..., j_{l-1})$ be two vectors in $[n]$.
Let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{VC}}, F, \mathbf{G}, \mathbf{H}, \mathsf{pp}_{\mathsf{IPA}})$ be the public parameters of AggOEA, $\mathbb{x} = (\mathbf{V}, \mathbf{W}, \mathbf{i}, \mathbf{j})$ the instance and $\mathbb{w} = (\mathbf{v}, \Omega_V, \Omega_W)$ the corresponding witness, such that: $\forall k \in [l] :$ $\mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, V_k, i_k, v_k, \Omega_V[k]) = \mathsf{VerOpen}(\mathsf{pp}_{\mathsf{VC}}, W_k, j_k, v_k, \Omega_W[k]) = 1$.

$\mathcal{P}_{\mathsf{AggOEA}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \to \Upsilon$
  for $0 \le k \le l-1$ **do**
    $(u_k, \eta_k, v_k) \leftarrow\$\ \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$
      ▷ $\widehat{V}_k$ is a vector commitment to $(\mathbf{0}^{i_k} \parallel u_k \parallel \mathbf{0}^{n-i_k-1})$
    $(\widehat{V}_k, \widehat{\Omega}_{V,k}) \leftarrow (F_n^{v_k} F_{i_k}^{u_k}, F_{2n-i_k}^{v_k})$
      ▷ $\widehat{W}_k$ is a hiding commitment to $(\mathbf{0}^{j_k} \parallel u_k \parallel \mathbf{0}^{n-j_k-1})$
    $(\widehat{W}_k, \widehat{\Omega}_{W,k}) \leftarrow (F_n^{\eta_k} F_{j_k}^{u_k}, F_{2n-j_k}^{\eta_k})$
  $\widehat{\mathbf{V}} \leftarrow (\widehat{V}_0, ..., \widehat{V}_{l-1})$
  $\widehat{\mathbf{W}} \leftarrow (\widehat{W}_0, ..., \widehat{W}_{l-1})$
  $x \leftarrow h(\mathbf{V}, \mathbf{W}, \widehat{\mathbf{V}}, \widehat{\mathbf{W}})$
  $\mathbf{u} \leftarrow (u_0, ..., u_{l-1})$
  $(r', r'') \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p$
  $(U, V) \leftarrow (F^{r'} \mathbf{G}^{\mathbf{u}}, F^{r''} \mathbf{G}^{\mathbf{v}})$
  for $0 \le k \le 2l-1$ **do**
      ▷ $U^x V$ is a hiding commitment to $\mathbf{v} + x\mathbf{u}$
    $t_k \leftarrow h(k, \mathbf{V}\widehat{\mathbf{V}}^x, \mathbf{W}\widehat{\mathbf{W}}^x, \mathbf{i}, \mathbf{j}, U^x V)$
  $\Omega \leftarrow \prod_{k=0}^{l-1} (\Omega_V[k] \widehat{\Omega}_{V,k}^x)^{t_{2k}} (\Omega_W[k] \widehat{\Omega}_{W,k}^x)^{t_{2k+1}}$
  $\mathbf{a} \leftarrow (v_0 + xu_0, ..., v_{l-1} + xu_{l-1})$
  $\mathbf{b} \leftarrow (t_0 + t_1, ..., t_{2l-2} + t_{2l-1})$
  $\rho \leftarrow r'' - xr'$
  $P \leftarrow F^{\rho} U^x V \mathbf{H}^{\mathbf{b}}$ ▷ P is a commitment to $\mathbf{a}$ and $\mathbf{b}$
  $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, \mathbf{a} \cdot \mathbf{b})$
  $\mathbb{w}_{\mathsf{IPA}} \leftarrow (\mathbf{a}, \mathbf{b})$
  $\Pi \leftarrow \mathcal{P}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \mathbb{w}_{\mathsf{IPA}})$
  $\Upsilon \leftarrow (\Pi, c, \rho, U, V, \widehat{\mathbf{V}}, \widehat{\mathbf{W}}, \Omega)$
  return Υ

$\mathcal{V}_{\mathsf{AggOEA}}(\mathsf{pp}, \mathbb{x}, \Upsilon) \to b$
  $(\Pi, c, \rho, U, V, \widehat{\mathbf{V}}, \widehat{\mathbf{W}}, \Omega) \leftarrow \Upsilon$
  $x \leftarrow h(\mathbf{V}, \mathbf{W}, \widehat{\mathbf{V}}, \widehat{\mathbf{W}})$
  for $0 \le k \le 2l-1$ **do**
    $t_k \leftarrow h(k, \mathbf{V}\widehat{\mathbf{V}}^x, \mathbf{W}\widehat{\mathbf{W}}^x, \mathbf{i}, \mathbf{j}, U^x V)$
  if $\frac{\prod_{k=0}^{l-1} e((V_k \widehat{V}_k^x)^{t_{2k}}, F_{n-i_k}) e((W_k \widehat{W}_k^x)^{t_{2k+1}}, F_{n-j_k})}{e(\Omega, G) e(F_0, F_n)^c} \neq 1$ **then**
    return 0
  else
    $\mathbf{b} \leftarrow (t_0 + t_1, ..., t_{2l-2} + t_{2l-1})$
    $P \leftarrow F^{\rho} U^x V \mathbf{H}^{\mathbf{b}}$
    $\mathbb{x}_{\mathsf{IPA}} \leftarrow (P, c)$
    return $\mathcal{V}_{\mathsf{IPA}}(\mathsf{pp}_{\mathsf{IPA}}, \mathbb{x}_{\mathsf{IPA}}, \Pi)$

Figure 8: Opening Equality Argument AggOEA

Inner product arguments to reduce communication cost. Let $\mathbf{b} = (t_0 + t_1, ..., t_{2l-2} + t_{2l-1})$. It is easy to see that c in Equation 10 corresponds to the inner product $\mathbf{a} \cdot \mathbf{b}$. Therefore, we can leverage an inner product argument to avoid sending $\mathbf{a}$. It should be noted however, that decreasing the communication cost incurs additional computational overhead during proof generation and verification. As a result, a tradeoff analysis should be conducted to assess whether not sending $\mathbf{a}$ is justifiable.

Figure 8 describes in details our aggregated opening equality argument that makes use of the inner product argument to avoid sending vector $\mathbf{a}$.

Finally, note that in our PoL, the opening equality argument takes place between two vector commitments V and W such that V is always opened at the last index of the vector. In other words, the verification of Equation 10 in our PoL can be re-written as follows:

$$\prod_{k=0}^{l-1} e(V'_k, F_{n-i_k})^{t_{2k}} e(W'_k, F_1)^{t_{2k+1}}$$
$$= e(\prod_{k=0}^{l-1} W_k'^{t_{2k+1}}, F_1) \prod_{k=0}^{l-1} e(V_k'^{t_{2k}}, F_{n-i_k}) = e(\Omega, G) e(F_0, F_n)^c$$

This reduces the number of pairings in PoL from $2l+2$ to $l+3$.

Theorem 7.1. *Figure 8 is an aggregated opening equality argument for Pointproofs vector commitments ROM.*

7.2 Aggregating Sum Arguments

Recall that $\mathbf{b}$ is the n-dimensional vector $(1, ..., 1, -1)$, and that the sum argument presented in Section 6.4 proves that a hiding Pedersen commitment V commits to a vector $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{b} = 0$.

The aggregation of SAs builds upon two observations:

1. If $\forall i \in [l]$, $\mathbf{v}_i \cdot \mathbf{b} = 0$, then for any vector $\mathbf{t} = (t_0, ..., t_{l-1})$, $\mathbf{t}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T \cdot \mathbf{b} = 0$, where $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]$ is the matrix whose j^{th} column is $\mathbf{v}_i$ and $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$ is its transpose.
2. If $\forall i \in [l], V_i = F^{r_i} \mathbf{G}^{\mathbf{v}_i}$ is a Pedersen commitment of $\mathbf{v}_i$ using randomness r_i, then $V = \prod_{i=0}^{l-1} V_i^{t_i}$ is a Pedersen commitment to $\mathbf{t}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$ with randomness $\mathbf{t} \cdot \mathbf{r}$, where $\mathbf{r} = (r_0, ..., r_{l-1})$.

Accordingly, we call sum argument $\mathcal{P}_{\mathsf{SA}}$ on vector $\mathbf{t}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$ and randomness $\mathbf{t} \cdot \mathbf{r}$.

To argue the soundness of the aggregation, let $t = h(V_0, ..., V_{l-1})$. If $\forall i \in [l]$, t_i is computed as t^i, then the Schwartz Zippel lemma entails that if $\mathbf{t}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T \cdot \mathbf{b} = \sum_{i=0}^{l-1} t^i (\mathbf{v}_i \cdot \mathbf{b}) = 0$, then $\mathbf{v}_i \cdot \mathbf{b} = 0, \forall i \in [l]$ with all but negligible probability $(l-1)/p$.

For more details on the aggregation of the sum argument, refer to Figure 9.

Theorem 7.2. *Let $\mathbf{v}_i \in \mathbb{Z}_p^n$ and $r_i \in \mathbb{Z}_p$, $\forall i \in [l]$. The above argument is a non-interactive zero-knowledge argument for relation:*

$$\mathbb{w} = \Big((\mathbf{v}_i, r_i)_{i \in [l]} \Big) ; \mathbb{x} = (V_0, ..., V_{l-1}) :$$

Let $\mathsf{pp}_{\mathsf{SA}} = (\mathsf{pp}_{\mathsf{IPA}}, G, F, \mathbf{G}, \mathbf{H}, B)$ be the public parameters of the sum argument described in Figure 6.
Let $\mathsf{pp} = \mathsf{pp}_{\mathsf{SA}}$ be the public parameters of AggSA, $\mathbb{x} = (V_0, ..., V_{l-1})$ the instance and $\mathbb{w} = (\mathbf{v}_0, ..., \mathbf{v}_{l-1}, r_0, ..., r_{l-1})$ the corresponding witness, such that $\forall\ i \in [l], V_i = F^{r_i}\mathbf{G}^{\mathbf{v}_i}\ \wedge\ \mathbf{v}_i[n] = \sum_{j=0}^{n-2} \mathbf{v}_i[j]$.
Let $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]$ denote the matrix with columns $\mathbf{v}_i$ and $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$ its transpose. Let $\mathbf{V}$ denote vector $(V_0, ..., V_{l-1})$ and $\mathbf{r}$ denote vector $(r_0, ..., r_{l-1})$.

1: $\mathcal{P}_{\mathsf{AggSA}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \rightarrow \Phi$
2: $t \leftarrow h(\mathbf{V})$
3: $\mathbf{t} \leftarrow (1, t, ..., t^{l-1})$
4: $\mathbf{v} \leftarrow \mathbf{t}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$
5: $(V, r) \leftarrow (\mathbf{V}^{\mathbf{t}}, \mathbf{r} \cdot \mathbf{t})$
6: $\mathbb{x}_{\mathsf{SA}} \leftarrow (\mathbf{G}, F, V)$
7: $\mathbb{w}_{\mathsf{SA}} \leftarrow (\mathbf{v}, r)$
8: return $\Phi \leftarrow \mathcal{P}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_{\mathsf{SA}}, \mathbb{w}_{\mathsf{SA}})$

9: $\mathcal{V}_{\mathsf{AggSA}}(\mathsf{pp}, \mathbb{x}, \Phi) \rightarrow b$
10: $t \leftarrow h(\mathbf{V})$
11: $\mathbf{t} \leftarrow (1, t, ..., t^{l-1})$
12: $V \leftarrow \mathbf{V}^{\mathbf{t}}$
13: $\mathbb{x}_{\mathsf{SA}} \leftarrow (\mathbf{G}, F, V)$
14: return $\mathcal{V}_{\mathsf{SA}}(\mathsf{pp}_{\mathsf{SA}}, \mathbb{x}_{\mathsf{SA}}, \Pi)$

Figure 9: Aggregated Sum Argument AggSA

$$\forall i \in [l], V_i = F^{r_i}\mathbf{G}^{\mathbf{v}_i}\ \wedge\ \mathbf{v}_i[n-1] = \sum_{j=0}^{n-2} v_i[j]$$

in the ROM under the discrete logarithm assumption.

7.3 Aggregating Range Proofs

Assume that we have l vectors $\mathbf{v}_0, ..., \mathbf{v}_{l-1}$ of length n and we wish to prove that all elements of these vectors are in range $[2^m]$. We now describe how to leverage parts of the range proof to aggregate the range proofs for these vectors. This aggregation reduces the communication cost of the proofs from $O(l\log(nm))$ to $O(\log(lnm))$.

For all $k \in [l]$ let $V_k = F^{r_k}\prod_{i=0}^{n-1} G_i^{\mathbf{v}_k[i]}$ be the hiding commitment to vector $\mathbf{v}_k$ using randomness r_k. Let $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]$ be the matrix whose columns are vectors $\mathbf{v}_k$ and $[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$ its transpose. Let $y \in \mathbb{Z}_p$ and $\mathbf{y} = (1, y, ..., y^{l-1})$. Observe that if we run $\mathcal{P}_{\mathsf{Rdx}}$ on vector $\mathbf{v}' = \mathbf{y}[\mathbf{v}_0, ..., \mathbf{v}_{l-1}]^T$, we obtain

$$v' = \sum_{i=0}^{n-1} \mathbf{v}'[i]\left(\prod_{k=0}^{\mu-1} x_k^{\mathsf{Bits}(i)[k]}\right) = \sum_{i=0}^{n-1}\sum_{k=0}^{l-1} y^k \mathbf{v}_k[i]\left(\prod_{k'=0}^{\mu-1} x_{k'}^{\mathsf{Bits}(i)[k']}\right)$$

Let $\mathbf{v}_{k\times n,i} = \mathsf{Bits}(\mathbf{v}_k[i])$, for all $k \in [l]$ and $i \in [n]$. If all vectors $\mathbf{v}_k$ are in range $[2^m]$, then:

$$\mathbf{v}_k[i] = \sum_{j=0}^{m-1} 2^j \mathbf{v}_{k\times n,i}[j]$$

Therefore, we can rewrite v' as

$$v' = \sum_{k=0}^{l-1}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left(\left(\prod_{k'=0}^{\mu-1} x_{k'}^{\mathsf{Bits}(i)[k']}\right)y^k 2^j\right)\mathbf{v}_{k\times n,i}[j]$$

For convenience, let $f_i = \prod_{k'=0}^{\mu-1} x_{k'}^{\mathsf{Bits}(i)[k']}$ and $\mathbf{f} = (f_0, ..., f_{n-1})$. By simple substitution, we thus have

$$v' = \sum_{k=0}^{l-1}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(f_i y^k 2^j)\mathbf{v}_{k\times n,i}[j]$$

It is easy to show that $V = \prod_{k=0}^{l-1} V_k^{y^k}$ is a commitment to $\mathbf{v}'$.

To produce the aggregated range proof, we first compute $y = h(V_0, ..., V_{l-1})$ and $V = \prod_{k=0}^{l-1} V_k^{y^k}$. We define $\hat{\mathbf{v}}$ as the concatenation of vectors $\mathbf{v}_{k\times n,i}$, $k \in [l]$ and $i \in [n]$ and random vector $\mathbf{w}$. That is:

$$\hat{\mathbf{v}} = (\mathbf{v}_{0\times n,0} \parallel \mathbf{v}_{0\times n,1} \parallel ... \parallel \mathbf{v}_{(l-1)\times n,m-1} \parallel \mathbf{w})$$

Let $\hat{\mathbf{w}}$ be the complement of $\hat{\mathbf{v}}[: lnm]$:

$$\hat{\mathbf{w}} = \mathbf{1}^{lnm} - \hat{\mathbf{v}}[: lnm].$$

We execute the remainder of RP as previously described with the following changes. Vector $\mathbf{d} = (d_{0\times n,0}, ..., d_{(l-1)\times n,m-1} \parallel x\mathbf{f})$ is of length $lnm + n$ where for all $k \in [l]$, $i \in [n]$ and $j \in [m]$, $d_{k\times n+i,j} = y^k\mathbf{f}[i]2^j$. Moreover, vectors $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ are of lengths $lnm + n$ and lnm, respectively.

The verification of the range proof proceeds similarly. We first compute $y = h(V_0, ..., V_{l-1})$, and commitment V. We compute x and U, and run $\mathcal{V}_{\mathsf{Rdx}}$. If the verification of Rdx fails, then we output 0. Else, we compute vectors $\mathbf{f}$ and $\mathbf{d}$ as described previously. Finally, we execute the remainder of $\mathcal{V}_{\mathsf{RP}}$ which completes the verification (lines 59 – 69, in Figure 7). It is easy to see that this aggregation reduces the size of the range proofs for vectors $\mathbf{v}_0, ..., \mathbf{v}_{l-1}$ from $O(l\log(nm))$ to $O(\log(lnm))$.

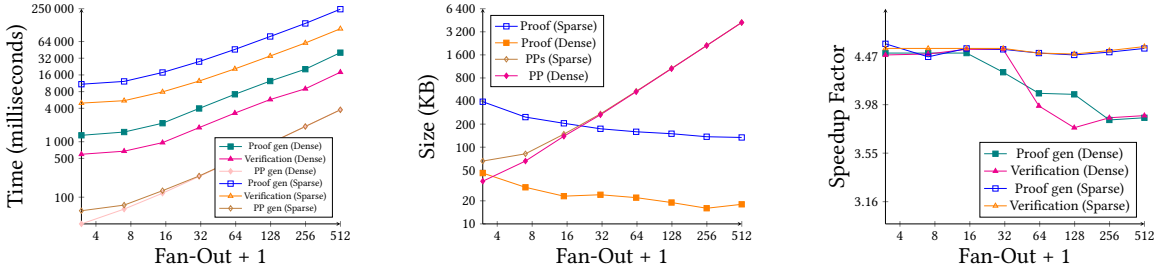
Theorem 7.3. *The above argument is an aggregated range proof over vector commitments in the ROM under the discrete logarithm assumption.*

8 EVALUATION

In order to demonstrate the practicality of our scheme, we implemented our scheme and benchmarked its performance.

8.1 Benchmark specification

We implemented our scheme in ~ 3,500 lines of Go [23] and our implementation is publicly available on Github [24]. For efficiency considerations, we instantiated the scheme with the BN-254 [25] pairing friendly Elliptic Curve implementation of Gnark-crypto [26]. All benchmarks were run on c5.2xlarge AWS virtual machines with 8 CPUs, 16GB RAM and a gp2 SSD with 1,500 IOPs.

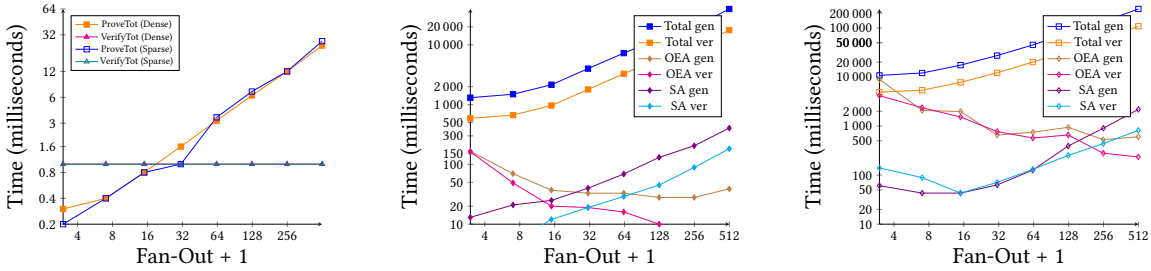


(a) PP gen, proving and verification time in both sparse and dense trees

(b) PPs and proof size in both sparse and dense trees

(c) Parallelism facilitated speedup in proof generation and verification

Figure 10: Performance (a) and size (b) evaluation of Liabilities proofs, and speedup from parallelizing range proofs (c)



(a) Performance of ProveTot and VerifyTot in both Sparse and Dense trees

(b) Performance of Dense Liabilities proof generation and verification

(c) Performance of Sparse Liabilities proof generation and verification

Figure 11: Evaluation (a) of total liabilities proofs, and performance breakdown of individual liabilities proofs (b), (c)

8.1.1 What is measured. Our strategy revolves around an abstraction of a "Liabilities Set" which is an instantiation of the scheme as defined in Definition 1. We implement the operations ProveTot, VerifyTot, Prove and Verify described in Section 5 and measure their execution time and the size of the proofs they produce for maximum liability $max = 2^{63}$. We also provide a breakdown of which operations dominate the runtime and which are negligible. For the inner product arguments, we implement the technique from Bulletproofs without the optimization in [12, Section 3.1] (IPA verification through multi-exponentiation) which we consider as future work. For the vector commitment scheme, we implement [11] as-is.

8.1.2 Verkle tree topology. At the heart of our Liabilities Set resides a sparse Verkle tree as depicted in Figure 1. As evident from Figure 3, the proofs associated to each vertex are made with respect to the values it commits to, for which there exist descendant vertices. Since our scheme makes heavy use of the inner product argument [12], we optimize the topology of the Verkle tree by making the degree (or fan-out) of each vertex be a power of two minus 1 (e.g $2^3 - 1$ or $2^4 - 1$). Therefore, when constructing the vector an inner product argument works on, it becomes a power of two because the last element in the vector is the sum of the previous elements.

8.2 Benchmark methodology

The public parameters of our scheme are derived from parameters influencing the topology of the Verkle tree, namely the arity (fan-out of each vertex). We investigate how changing the parameter selection of the Verkle tree influences performance and proof size. Specifically, we explore two dimensions: (i) whether the Verkle tree has a limited capacity or an unlimited one; and (ii) the fan-out of each vertex in the tree. To accommodate a potentially large number of liabilities, we construct a sparse Verkle tree (hereafter SVT) with an ID space of all 256 bit strings. In conjunction, we construct a "dense" (non-sparse) Verkle tree (hereafter DVT) that can accommodate all 9 digit numbers (spans all Social Security Numbers (SSN) [27] of the USA). Note that although the aforementioned SSN tree is dense and not sparse, it is still history independent, as the path from the root to the leaf is deterministically derived from the SSN that is inserted. We then experiment both options with varying numbers of descendants (fan-out).

8.2.1 Experiment layout. Each pair of parameter selection (dense/sparse and fan-out) defines an experiment. Each experiment is a two-phased process in which we first instantiate a Liabilities Set and then populate it with liabilities corresponding to random IDs. The public parameter generation and size is then measured amortized across 10 iterations. In our experiments we found that the number of liabilities we use to populate the Liabilities Set does not influence the measurements of the proof generation, but only prolongs the time to populate the Liabilities Set. In the second phase, we insert liabilities for 10 random IDs and measure the time it takes to generate and verify PoLs. The measurements are amortized across all 10 random IDs.

8.2.2 Evaluated criteria. In all figures but 11a we evaluate proofs for a liability of a single client (Prove and Verify). In Figure 11a

we evaluate for ProveTot and VerifyTot. In Figure 10a we show the overall time to produce a proof and verify it in both dense and sparse Verkle tree topologies, as well as the time it takes to create the public parameters. In Figure 10b, we measure the sizes of the public parameters (PPs) in both sparse and dense topologies, and the sizes of the proofs. Additionally, we repeat each experiment twice: once where range proof generation and verification is parallelized, and once where it is serial. We then measure the speedup facilitated by parallelizing the range proof generation and verification as can be seen in Figure 10c. In Figures 11b and 11c we show the time in proof generation and verification that is spent on the Sum Argument (SA) and Opening Equality Argument (OEA) and compare them against the total time. We see here that the dominating factor of generating or verifying a proof is the range proof.

8.3 Evaluation insights

We draw the following conclusions from the evaluations: (i) In both dense and sparse Verkle trees, fan-out is inversly correlated to the proof size. Indeed, as depicted in Table 1, our proof size is linear in the depth of the tree, which explains this relation. (ii) Accomodating an unlimited number of liabilities (via a sparse tree) carries a size overhead in the proof, but not in the public parameters, which can be explained because the public parameters depend only on the fan-out but not on the depth of the proofs generated. (iii) The dominating factor in the performance of our scheme is the range proof. Fortunately, the range proof is trivially parallelizable both in proof generation and verification and doing so yields a significant speedup. (vi) The speedup in the dense tree that is gained by parallelising the range proofs diminishes as the fan-out increases and the tree becomes flatter, thus reducing the number of range proofs that can be parallelised. (v) When verifying the total liabilities (Figure 11a) the runtime is constant.

8.3.1 Comparison with DAPOL+. We compare our scheme with the implementation of DAPOL+ [9] available online [28]. Both are evaluated on AWS virtual machines with the specification from Section 8.1. The DAPOL+ authors investigated the performance of proof generation and verification time on Sparse Merkle Trees (SMT) of varying height (16, 24, 32).

An SMT of height h is equivalent to an SSVT with a universe size of 2^h. We thus show the performance of our scheme with different fanouts for each corresponding SMT height selection of DAPOL+ in Figure 12. We show comparisons of the proof generation time in Figure 12a and the proof verification time in Figure 12b. Lastly, we compare the proof size in both schemes for each corresponding SMT height selection of DAPOL+ in Figure 12c. In both schemes, the range proofs were not batched. Our evaluation shows that as the fan-out of our SSVT increases, the proofs generated by our scheme become smaller than those of DAPOL+.

9 CONCLUSION

We present the first fully-private decentralized PoL scheme with short proofs that leaks only the size of the user universe. Full-privacy of our scheme is achieved through the use of sparse summation Verkle trees together with *tailor-made* zero-knowledge arguments that enable users to verify the inclusion of their liabilities without compromising their privacy or the privacy of the prover.

ACKNOWLEDGMENTS

We'd like to thank Jonathan Bootle for his helpful suggestions regarding the range proof instantiation and the anonymous reviewers for their constructive feedback.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Online; accessed 14 January 2023.

[2] Derek Saul. Bankrupt ftx has recouped over $5 billion in liquid assets—but still has billions more to go to meet liabilities, 2023.

[3] Alicia Tuovila. Audit: What it means in finance and accounting, 3 main types, 2022.

[4] Zak Wilcox. Proving your bitcoin reserves. https://iwilcox.me.uk/2014/proving-bitcoin-reserves, 2014. Online; accessed 13 July 2022.

[5] Kexin Hu, Zhenfeng Zhang, and Kaiwen Guo. Breaking the binding: Attacks on the merkle approach to prove liabilities and its applications. *Comput. Secur.*, 87(C), nov 2019.

[6] Konstantinos Chalkias, Kevin Lewi, Payman Mohassel, and Valeria Nikolaenko. Practical privacy preserving proofs of solvency. Amsterdam ZKProof Community Even, 2019.

[7] Philippe Camacho. Secure protocols for provable security, 2014. https://www.slideshare.net/philippecamacho/protocols-for-provable-solvency-38501620.

[8] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 720–731, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Konstantinos Chalkias, Kevin Lewi, Payman Mohassel, and Valeria Nikolaenko. Distributed auditing proofs of liabilities. Cryptology ePrint Archive, Paper 2020/468, 2020. https://eprint.iacr.org/2020/468.

[10] Yan Ji and Konstantinos Chalkias. Generalized proof of liabilities. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3465–3486, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 2007–2023, New York, NY, USA, 2020. Association for Computing Machinery.

[12] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.

[13] John Kuszmaul. Verkle trees. https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf, 2018. Online; accessed 17 October 2022.

[14] Moni Naor and Vanessa Teague. Anti-presistence: history independent data structures. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 492–501. ACM, 2001.

[15] Rasmus Dahlberg, Tobias Pulls, and Roel Peeters. Efficient sparse merkle trees. In *Nordic Conference on Secure IT Systems*, pages 199–215. Springer, 2016.

[16] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 55–72, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[17] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *Theory of Cryptography*, pages 499–517, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

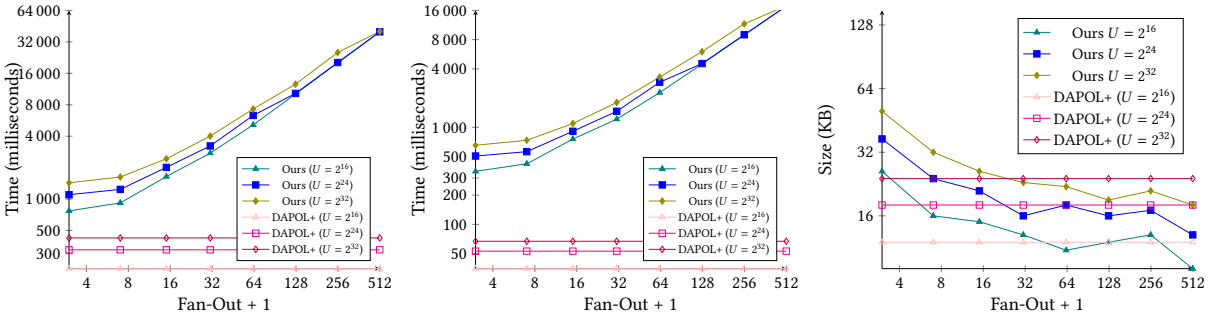
[18] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology—CRYPTO '86*, page 186–194, Berlin, Heidelberg, 1987. Springer-Verlag.

[19] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[20] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[21] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 415–432, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[22] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata,*



(a) Proof gen comparison. **(b) Verification comparison.** **(c) Proof size comparison.**

Figure 12: Performance comparison with DAPOL+. U is the universe size. In the experiments, we vary fan-out for our scheme, whereas DAPOL+ fan-out is static due to using a Merkle Tree.

Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
[23] Google. Go programming language https://go.dev/.
[24] https://github.com/yacovm/PoL.
[25] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pages 319–331, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
[26] Gautam Botrel, Thomas Piellard, Youssef El Housni, Arya Tabaie, Gus Gutoski, and Ivo Kubjas. Consensys/gnark-crypto: v0.6.0, January 2023.
[27] USA government. about social security numbers www.usa.gov/about-social-security.
[28] Yan Ji; Konstantinos Chalkias. Dapol+ implementation.

A PROOFS

A.1 Proof of Theorem 6.1

In this section, we prove the security of the opening equality argument $\mathsf{OEA} = (\mathcal{G}_{\mathsf{OEA}}, \mathcal{P}_{\mathsf{OEA}}, \mathcal{V}_{\mathsf{OEA}})$ depicted in Figure 5.

Proof. **Completeness.** Completeness follows directly from the completeness of the aggregation of Pointproofs commitments.
Zero-knowledge. We prove zero-knowledge by describing a simulator, which thanks to the *programmability of random oracle h* and *knowledge of the trapdoor* α of the Pointproofs commitments, is able to produce a proof that is indistinguishable from the output of an honest prover $\mathcal{P}_{\mathsf{OEA}}$.

The simulator proceeds as follows.

$$(c, x) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p$$
$$(\widehat{V}, \widehat{W}) \leftarrow\$ \mathbb{G} \times \mathbb{G}.$$

Then, it sets the hash $h(V, W, \widehat{V}, \widehat{W})$ to x and computes Ω using Equation 1 and trapdoor α. Recall that a valid proof $(c, \widehat{W}, \widehat{W}, \Omega)$ satisfies

$$e(\Omega, G)e(F_0, F_n)^{c(t_0+t_1)} = e((V\widehat{V}^x)^{t_0}, F_{n-i})e((W\widehat{W}^x)^{t_1}, F_{n-j})$$

By isolating Ω to the left, we obtain:

$$\begin{aligned} e(\Omega, G) &= \frac{e((V\widehat{V}^x)^{t_0}, F_{n-i})e((W\widehat{W}^x)^{t_1}, F_{n-j})}{e(F_0, F_n)^{c(t_0+t_1)}} \\ &= \frac{e((V\widehat{V}^x)^{t_0}, G^{\alpha^{n-i+1}})e((W\widehat{W}^x)^{t_1}, G^{\alpha^{n-j+1}})}{e(G^{\alpha}, G^{\alpha^{n+1}})^{c(t_0+t_1)}} \\ &= \frac{e((V\widehat{V}^x)^{t_0\alpha^{n-i+1}}, G)e((W\widehat{W}^x)^{t_1\alpha^{n-j+1}}, G)}{e(G^{\alpha^{n+2}c(t_0+t_1)}, G)} \end{aligned}$$

and setting

$$\Omega \leftarrow (V\widehat{V}^x)^{t_0\alpha^{n-i+1}}(W\widehat{W}^x)^{t_1\alpha^{n-j+1}}G^{-\alpha^{n+2}c(t_0+t_1)}.$$

Finally, the simulator outputs $(c, \widehat{V}, \widehat{W}, \Omega)$. It is easy to see that $(c, \widehat{V}, \widehat{W}, \Omega)$ will be accepted by $\mathcal{V}_{\mathsf{OEA}}$.

Since the first three elements are drawn and Ω is uniquely determined using Equation 1, the output of the simulator is statistically indistinguishable from the output of $\mathcal{P}_{\mathsf{OEA}}$.

Finally, we recall that Ω is computed as a function of hiding Pointproofs commitments V and W, and random elements $c, \widehat{V}$ and $\widehat{W}$. As a result, we conclude that the simulator is able to successfully simulate the output of $\mathcal{P}_{\mathsf{OEA}}$ without access to its witness.
Knowledge-soundness. Note that if $(c, \widehat{V}, \widehat{W}, \Omega)$ is accepted by $\mathcal{V}_{\mathsf{OEA}}$, then the soundness of the aggregation of Pointproofs entails that c opens $V\widehat{V}^x$ and $W\widehat{W}^x$ at indices i and j, respectively.

By the homomorphism and the binding property of Pointproofs, we have that $c = w + x\hat{w} = v + x\hat{v}$, which we can rewrite as

$$0 = (w - v) + (\hat{w} - \hat{v})x$$

Since $x = h(V, W, \widehat{V}, \widehat{W})$, the Shwartz-Zippel lemma allows us to deduce that $w = v$ and $\hat{w} = \hat{v}$ with all but probability $1/p$. In other words, $v = w$ and $\hat{w} = \hat{v}$.

Now we describe how an extractor can compute witness v. The extractor first runs $\mathcal{P}_{\mathsf{OEA}}$ with $h(V, W, \widehat{V}, \widehat{W})$ being set to x to get a first accepting proof $(c, \widehat{V}, \widehat{W}, \Omega)$. It then rewinds $\mathcal{P}_{\mathsf{OEA}}$ at line 5 and sets $h(V, W, \widehat{V}, \widehat{W})$ to x', in order to get a second proof $(c', \widehat{V}, \widehat{W}, \Omega')$.

It then solves the system of linear equations composed of $c = v + x\hat{v}$ and $c' = v + x'\hat{v}$, which yields a unique solution $(v, \hat{v})$, where v opens V and W at indices i and j respectively. □

A.2 Proof of Theorem 6.2

In the following, we prove the security of the sum argument $\mathsf{SA} = (\mathcal{G}_{\mathsf{SA}}, \mathcal{P}_{\mathsf{SA}}, \mathcal{V}_{\mathsf{SA}})$ depicted in Figure 6.

Proof. **Completeness.** Let $\mathbf{b}$ be the n element vector $(1, ..., 1, -1)$. Completeness is derived from the fact that for any vector $\mathbf{v}$ satisfying $\sum_{i=0}^{n-2} \mathbf{v}[i] = \mathbf{v}[n-1]$, equality $(\mathbf{v} + x\mathbf{w}) \cdot \mathbf{b} = x\mathbf{w} \cdot \mathbf{b}$ holds for any pair $(x, \mathbf{w}) \in \mathbb{Z}_p \times \mathbb{Z}_p^n$.

Zero-knowledge. Next, we describe a simulator, which thanks to the *programmability of random oracle h*, is able to produce a proof that is indistinguishable from the output of an honest prover $\mathcal{P}_{\mathsf{SA}}$. The simulator proceeds as follows:

$$\mathbf{a} \leftarrow\$\, \mathbb{Z}_p^n \;;\; c = \mathbf{a} \cdot \mathbf{b} \;;\; P = \mathbf{G}^{\mathbf{a}}\mathbf{H}^{\mathbf{b}}$$
$$(x, \rho) \leftarrow\$\, \mathbb{Z}_p$$

If $x \neq 0$, then the simulator computes $W = \left(\frac{\mathbf{G}^{\mathbf{a}}}{F^{\rho}V}\right)^{1/x}$. If not, the simulator aborts. Note that the probability that the simulator aborts is $1/p$, which is negligible.

The simulator then sets the hash $h(c, V, W)$ to value x, and runs $\mathcal{P}_{\mathsf{IPA}}$ with instance (P, cx) and the simulated witness $(\mathbf{a}, \mathbf{b})$ to obtain the proof Π. Finally, it outputs (Π, W, c, ρ). It is easy to see that this tuple will be accepted by $\mathcal{V}_{\mathsf{SA}}$.

Recall that ρ is selected uniformly at random and that c, Π and W are determined by the verification equations of the sum argument. Since $c = \mathbf{a} \cdot \mathbf{b}$ for $\mathbf{a}$ randomly selected, it has the same distribution as a c produced by $\mathcal{P}_{\mathsf{SA}}$. Moreover, $W = \left(\frac{\mathbf{G}^{\mathbf{a}}}{F^{\rho}V}\right)^{1/x}$ for $\mathbf{a}$ and ρ uniformly distributed. Therefore, W is also uniformly distributed. Finally, Π is the output of $\mathcal{P}_{\mathsf{IPA}}$ on input vectors $\mathbf{a}$ and $\mathbf{b}$.

Therefore, (Π, W, c, ρ) is statistically indistinguishable from a proof produced by $\mathcal{P}_{\mathsf{SA}}$.

Furthermore, proof Π output by $\mathcal{P}_{\mathsf{IPA}}$ and W are computed without access to vector $\mathbf{v}$ or randomness r committed in V: Π is produced on input of a simulated witness $\mathbf{a}$, which is selected independently from $\mathbf{v}$ and W is computed as a function of public information V and random elements ρ and P. Accordingly, we conclude that the simulator is able to successfully simulate the output of $\mathcal{P}_{\mathsf{SA}}$ without access to its witness.

Knowledge-soundness. We now show how to construct an extractor which, with access to $\mathcal{P}_{\mathsf{SA}}$, extracts witness $(\mathbf{v}, r)$ such that $\mathbf{v}[n-1] = \sum_{i=0}^{n-2} \mathbf{v}[i]$ and $V = F^r\mathbf{G}^{\mathbf{v}}$.

The extractor runs $\mathcal{P}_{\mathsf{SA}}$, sets $h(c, V, W)$ to random value x and gets a first accepting proof (Π, W, c, ρ). Then it leverages the extractability of the IPA to obtain the witness $\mathbf{a}$ such that $\mathbf{a} \cdot \mathbf{b} = cx$ and $\mathbf{G}^{\mathbf{a}} = F^{\rho}VW^{x}$.

The extractor then rewinds $\mathcal{P}_{\mathsf{SA}}$ at line 5, Figure 6, sets $h(c, V, W)$ to a second random value $x' \neq x$, and gets a second accepting proof (Π', W, c, ρ'). Then it invokes the extractor of the IPA to obtain the witness $\mathbf{a}'$ such that $\mathbf{a}' \cdot \mathbf{b} = cx'$ and $\mathbf{G}^{\mathbf{a}'} = F^{\rho}VW^{x'}$.

Note that there exist $(\mathbf{v}, r)$ and $(\mathbf{w}, r')$ such that $V = F^r\mathbf{G}^{\mathbf{v}}$ and $W = F^{r'}\mathbf{G}^{\mathbf{w}}$.

Therefore, the *binding property* of Pedersen commitment ensures that $\mathbf{a} = \mathbf{v} + x\mathbf{w}$, $\rho = -r - xr'$, $\mathbf{a}' = \mathbf{v} + x'\mathbf{w}$ and $\rho' = -r - x'r'$.

Accordingly, the extractor builds and solves a system of $2n$ linear equations $\mathbf{a}[i] = \mathbf{v}[i] + x\mathbf{w}[i]$ and $\mathbf{a}'[i] = \mathbf{v}[i] + x\mathbf{w}[i]$, for $i \in [n]$, with $2n$ variables $\mathbf{v}$ and $\mathbf{w}$. This results in a unique solution $(\mathbf{v}, \mathbf{w})$. It also solves the system of linear equations $-\rho = r + xr'$ and $-\rho' = r + x'r'$. This yields pair (r, r').

We now show that vector $(\mathbf{v}, r)$ must be a valid witness for the relation $V = F^r\mathbf{G}^{\mathbf{v}}$ and $\sum_{i=0}^{n-2} \mathbf{v}[i] = \mathbf{v}[n-1]$.

By the soundness of the IPA protocol, $cx = \mathbf{a} \cdot \mathbf{b}$. Expanding this equation gives us

$$cx = \mathbf{a} \cdot \mathbf{b} = (\mathbf{v} + x\mathbf{w}) \cdot \mathbf{b}$$
$$cx = \mathbf{v} \cdot \mathbf{b} + x\mathbf{w} \cdot \mathbf{b}$$

and rearranging the bottom equation yields

$$0 = \mathbf{v} \cdot \mathbf{b} + x((\mathbf{w} \cdot \mathbf{b}) - c).$$

Since x is computed as $h(c, V, W)$, the Shwartz-Zippel lemma allows us to conclude that $\mathbf{v} \cdot \mathbf{b} = 0$ and $(\mathbf{w} \cdot \mathbf{b}) - c = 0$ with all but a negligible probability $1/p$. □

A.3 Proof of Theorem 6.3

Proof. Let $\mathbf{v}$ be a vector of n elements in $\mathbb{Z}_p$ and $\mu = \log(n)$. Let $\mathsf{Bits}(i) = (b_0, ..., b_{\mu-1})$ denote the bit representation of i with b_0 is the least significant bit.

Let $\mathbf{v}^{(\mu-t)} \in \mathbb{Z}_p^{2^{(\mu-t)}}$ denote the resulting vector at the end of the t^{th} iteration of $\mathcal{P}_{\mathsf{Rdx}}$. Our goal is to show that for all $t \in [1..\mu], i \in [2^{\mu-t}]$:

$$\mathbf{v}^{(\mu-t)}[i] = \sum_{j=0}^{2^t-1} \mathbf{v}[i + j2^{\mu-t}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\mathsf{Bits}(j2^{\mu-t})[k]} \right) \tag{11}$$

We proceed by induction on $t \in [1..\mu]$. Note that by construction:

$$\mathbf{v}^{(\mu-1)} = \mathbf{v}^{(\mu)}[:2^{(\mu-1)}] + x_{\mu-1}\mathbf{v}^{(\mu)}[2^{(\mu-1)}:]$$

As a result $\forall i \in [2^{\mu-1}]$

$$\begin{aligned}\mathbf{v}^{(\mu-1)}[i] &= \mathbf{v}[i] + \mathbf{v}[i + 2^{(\mu-1)}]x_{\mu-1} \\ &= \sum_{j=0}^{1} \mathbf{v}[i + j2^{(\mu-1)}] \left(\prod_{k=\mu-1}^{\mu-1} x_k^{\mathsf{Bits}(j2^{(\mu-1)})[k]} \right)\end{aligned}$$

This implies that 11 holds for $t = 1$.

Suppose that Equation 11 holds for t. We now show that it must also hold for $t + 1$. Recall that the protocol defines

$$\mathbf{v}^{(\mu-t-1)} = \mathbf{v}^{(\mu-t)}[:2^{\mu-t-1}] + \mathbf{v}^{(\mu-t)}[2^{\mu-t-1}:]x_{\mu-t-1}.$$

Expanding and applying the inductive hypothesis yields $\forall i \in [2^{\mu-t-1}]$:

$$\begin{aligned}\mathbf{v}^{(\mu-t-1)}[i] &= \mathbf{v}^{(\mu-t)}[i] + \mathbf{v}^{(\mu-t)}[i + 2^{\mu-t-1}]x_{\mu-t-1} \\ &= \sum_{j=0}^{2^t-1} \mathbf{v}[i + j2^{\mu-t}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\mathsf{Bits}(j2^{\mu-t})[k]} \right) x_{\mu-t-1}^0 \\ &+ \sum_{j=0}^{2^t-1} \mathbf{v}[(i + 2^{\mu-t-1}) + j2^{\mu-t}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\mathsf{Bits}(j2^{\mu-t})[k]} \right) x_{\mu-t-1}^1 \\ &= \sum_{j=0}^{2^t-1} \mathbf{v}[i + (2j)2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\mathsf{Bits}((2j)2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^0 \\ &+ \sum_{j=0}^{2^t-1} \mathbf{v}[(i + (2j+1)2^{\mu-t-1})] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\mathsf{Bits}((2j)2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^1\end{aligned}$$

Let $j_0 = 2j$ and $j_1 = 2j+1$ for $j \in [2^t]$, and let

$$A_i = \sum_{j_0=0, j_0 \text{ even}}^{2^{t+1}-2} \mathbf{v}[i + j_0 2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j_0 2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^0$$

$$B_i = \sum_{j_1=0, j_1 \text{ odd}}^{2^{t+1}-1} \mathbf{v}[(i + j_1 2^{\mu-t-1})] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}((j_1-1)2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^1$$

We can therefore write $\mathbf{v}^{(\mu-t-1)}[i] = A_i + B_i$

Now note that for all odd $j_1 \in [2^{t+1}]$, $(j_1-1)2^{\mu-t-1}$ and $j_1 2^{\mu-t-1}$ coincide in the t most significant bits, that is, bits at indices in $[(\mu-t)..(\mu-1)]$. More specifically, $j_1 2^{\mu-t-1}$ and $(j_1-1)2^{\mu-t-1} = j_1 2^{\mu-t-1} - 2^{\mu-t-1}$ have the same bit representation except for the bit at position $\mu-t-1$. Bits at indices in $[\mu-t-1]$ are all zeros. Thus, we have:

$$\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}((j_1-1)2^{\mu-t-1})[k]} = \prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j_1 2^{\mu-t-1})[k]}$$

And we can write:

$$B_i = \sum_{j_1=0, j_1 \text{ odd}}^{2^{t+1}-1} \mathbf{v}[(i + j_1 2^{\mu-t-1})] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j_1 2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^1$$

Moreover, note that for all $j \in [2^{t+1}]$, $\text{Bits}(j2^{\mu-t-1})[\mu-t-1] = j \bmod 2$. Let $b_{j,\mu-t-1}$ denote $\text{Bits}(j2^{\mu-t-1})[\mu-t-1]$.

Accordingly, for an even j_0, $b_{j_0,\mu-t-1} = 0$, whereas for j_1 odd, $b_{j_1,\mu-t-1} = 1$, and we can hence write:

$$A_i = \sum_{j_0=0, j_0 \text{ even}}^{2^{t+1}-2} \mathbf{v}[i + j_0 2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j_0 2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{b_{j_0,\mu-t-1}}$$

$$B_i = \sum_{j_1=0, j_1 \text{ odd}}^{2^{t+1}-1} \mathbf{v}[(i + j_1 2^{\mu-t-1})] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j_1 2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{b_{j_1,\mu-t-1}}$$

Recall that $\mathbf{v}[i] = A_i + B_i$. If we replace j_0 and j_1 by j, then we get:

$$\mathbf{v}[i] = \sum_{j=0, j \text{ even}}^{2^{t+1}-2} \mathbf{v}[i + j2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{b_{j,\mu-t-1}} + \sum_{j=0, j \text{ odd}}^{2^{t+1}-1} \mathbf{v}[i + j2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{b_{j,\mu-t-1}}$$

That is:

$$\begin{aligned}
\mathbf{v}[i] &= \sum_{j=0}^{2^{t+1}-1} \mathbf{v}[i + j2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{b_{j,\mu-t-1}} \\
&= \sum_{j=0}^{2^{t+1}-1} \mathbf{v}[i + j2^{\mu-t-1}] \left(\prod_{k=\mu-t}^{\mu-1} x_k^{\text{Bits}(j2^{\mu-t-1})[k]} \right) x_{\mu-t-1}^{\text{Bits}(j2^{\mu-t-1})[\mu-t-1]} \\
&= \sum_{j=0}^{2^{t+1}-1} \mathbf{v}[i + j2^{\mu-t-1}] \left(\prod_{k=\mu-t-1}^{\mu-1} x_k^{\text{Bits}(j2^{\mu-t-1})[k]} \right)
\end{aligned}$$

Setting $t = \mu$ in Equation 11 and letting $n = 2^\mu - 1$ concludes the proof. □

A.4 Proof of Theorem 6.4

In what follows, we prove the security of the range proof $\text{RP} = (\mathcal{G}_{\text{RP}}, \mathcal{P}_{\text{RP}}, \mathcal{V}_{\text{RP}})$ depicted in Figure 7.

Proof. **Completeness.** Completeness is deduced from the fact that all valid witnesses satisfy Equations 4 to 8.

Zero-knowledge. we describe a simulator, which by controlling the *random oracle* h, is able to produce a proof that is indistinguishable from the output of an honest prover $\mathcal{P}_{\text{RP}}$. The simulator proceeds as follows:

$$\mathbf{u} \leftarrow\$ \mathbb{Z}_p^n \; ; \; U = \mathbf{G}^{\mathbf{u}}$$
$$(\gamma, x) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p$$

If $x \neq 0$, the simulator computes $W = \left(\frac{U}{F^Y V}\right)^{1/x}$ and lets $h(V, W, Q) = x$. Otherwise, the simulator aborts, however, the probability of abort at this stage is $1/p$.

The simulator then runs $\mathcal{P}_{\text{Rdx}}$ with the instance U, witness $\mathbf{u}$ and generators $\mathbf{G}$. This returns pair $(\Gamma, \mathbf{x})$. Next, the simulator computes vector $\mathbf{d}$ as depicted in 7, lines $52-58$. The simulator then selects the following elements uniformly at random

$$(\mathbf{a}, \mathbf{b}) \leftarrow\$ \mathbb{Z}_p^{nm+n} \times \mathbb{Z}_p^{nm+n}$$
$$(y_0, y_1, z) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$$
$$(\rho, \tau) \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p$$
$$(Q, C_1) \leftarrow\$ \mathbf{G}$$

It then computes $\mathbf{F}'$ as depicted in line 66, followed by

$$c = \mathbf{a} \cdot \mathbf{b}$$
$$P = \mathbf{H}^{\mathbf{a}} \cdot \mathbf{F}'^{\mathbf{b}}$$
$$c_0 = (\mathbf{1}^{mn+n} \cdot \mathbf{d}[:nm]) y_1^3 + (\mathbf{1}^{mn} \cdot \mathbf{y}_0 + u) y_1^2 + (\mathbf{1}^{mn} \cdot \mathbf{y}_0) y_1$$

Furthermore, if $z \neq 0$, the simulator computes

$$R = \left(\frac{P}{F^\rho Q \mathbf{H}[:nm]^{-\mathbf{y}_1} \mathbf{F}'^{\, y_1^2 \mathbf{d}} \mathbf{F}[:nm]^{\mathbf{y}_1}} \right)^{1/z}$$
$$C_2 = \left(\frac{G^c H^\tau}{C_1^z G^{c_0}} \right)^{1/z^2}$$

Otherwise, the simulator aborts. Note that the probability that the simulator aborts at this stage is $1/p$.

The simulator then lets $y_0 = h(U, Q, R, 0)$, $y_1 = h(U, Q, R, 1)$ and $z = h(U, Q, R, C_1, C_2)$, and runs $\mathcal{P}_{\text{IPA}}$ with the instance (P, c) and the simulated witness $(\mathbf{a}, \mathbf{b})$. This yields an IPA's proof Π.

It is easy to check that the proof $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho, \tau)$ output by the simulator will be accepted by $\mathcal{V}_{\text{RP}}$. In fact, $\mathcal{V}_{\text{Rdx}}$ will accept on input Γ and $U = VW^x$, $\mathcal{V}_{\text{IPA}}$ will accept on input Π, $P = \mathbf{H}^{\mathbf{a}} \mathbf{F}'^{\mathbf{b}}$ and c, $\mathcal{V}_{\text{RP}}$ will not return 0 at line 65.

Now, recall that $(\gamma, \rho, \tau, Q, C_1)$ are selected uniformly at random, whereas (W, R, C_2) are determined by the verification equations of the range proof. Note that $W = \left(\frac{U}{F^Y V}\right)^{1/x}$ for U and γ selected uniformly at random. Therefore, W is also distributed uniformly at random. A similar argument can be made for R and C_2. Finally, Γ and (c, Π) are the output of $\mathcal{P}_{\text{Rdx}}$ and $\mathcal{P}_{\text{IPA}}$ respectively. It follows then

that $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho, \tau)$ is statistically indistinguishable from a proof produced by $\mathcal{P}_{\mathsf{RP}}$.

Furthermore, tuple (c, Π, Γ) are computed without access to vector $\mathbf{v}$ or randomness r committed in V: c and Π are produced on input of a simulated witness $(\mathbf{a}, \mathbf{b})$, which is selected independently from $\mathbf{v}$; and Γ is computed on input of a random vector $\mathbf{u}$ also selected independently from $\mathbf{v}$. Accordingly, we conclude that the simulator is able to successfully simulate the output of $\mathcal{P}_{\mathsf{RP}}$ without access to its witness.

Knowledge-soundness. We now show how to construct an extractor whose access to $\mathcal{P}_{\mathsf{RP}}$ enables it to extract witness $(\mathbf{v}, r)$ such that $\mathbf{v}[i] \in [2^m], \forall i \in [n]$ and $V = F^r\mathbf{G}^{\mathbf{v}}$.

The extractor executes $\mathcal{P}_{\mathsf{RP}}$. During the execution, the extractor sets $h(V, W, Q)$ to random value x, $h(U, Q, R, 0)$ and $h(U, Q, R, 1)$ to random values y_0 and y_1, and $h(U, Q, R, C_1, C_2)$ to random value z. The execution concludes by outputting a first accepting proof $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho, \tau)$. Then it leverages the extractability of the IPA to obtain the witness $(\mathbf{a}, \mathbf{b})$ such that $\mathbf{a}\cdot\mathbf{b} = c$ and $\mathbf{H}^{\mathbf{a}}\mathbf{F}'^{\mathbf{b}} = F^{\rho}Q R^{z}\mathbf{H}[:nm]^{\mathbf{y}_1}\mathbf{F}'^{y_1^2\mathbf{d}}\mathbf{F}[:nm]^{\mathbf{y}_1}$. Next, the extractor rewinds at line 33, sets $h(U, Q, R, C_1, C_2)$ to another random value z', and gets a second accepting proof $(\Gamma, W, \gamma, \Pi, c, Q, R, C_1, C_2, \rho', \tau')$. It leverages again the extractability of the IPA to obtain the witness $\mathbf{a}'$ such that $\mathbf{a}'\cdot\mathbf{b}' = c'$ and $\mathbf{H}^{\mathbf{a}'}\mathbf{F}'^{\mathbf{b}'} = F^{\rho'}Q R^{z'}\mathbf{H}[:nm]^{\mathbf{y}_1}\mathbf{F}'^{y_1^2\mathbf{d}}\mathbf{F}[:nm]^{\mathbf{y}_1}$.

Note that there exist tuples $(\hat{\mathbf{v}}, \hat{\mathbf{w}}, \nu)$ and $(\mathbf{s}, \mathbf{t}, \eta)$ such that $Q = F^{\nu}\mathbf{H}^{\hat{\mathbf{v}}}\mathbf{F}[:nm]^{\hat{\mathbf{w}}}$ and $R = F^{\eta}\mathbf{H}^{\mathbf{s}}\mathbf{F}[:nm]^{\mathbf{t}}$. Therefore, by the binding property of Pedersen commitment, we have the following equalities:

$$
\begin{aligned}
\mathbf{a} &= \hat{\mathbf{v}} + (\mathbf{y}_1 \parallel 0^n) + z\mathbf{s}\\
\mathbf{b} &= y_1^2\mathbf{d} + y_1(\mathbf{y}_0\|\mathbf{0}^n) + ((\hat{\mathbf{w}}\circ\mathbf{y}_0)\|\mathbf{0}^n) + z(\mathbf{y}_0\circ\mathbf{t})\|\mathbf{0}^n\\
\rho &= -\nu - z\eta\\
\mathbf{a}' &= \hat{\mathbf{v}} + (\mathbf{y}_1 \parallel 0^n) + z'\mathbf{s}\\
\mathbf{b}' &= y_1^2\mathbf{d} + y_1(\mathbf{y}_0\|\mathbf{0}^n) + ((\hat{\mathbf{w}}\circ\mathbf{y}_0)\|\mathbf{0}^n) + z'(\mathbf{y}_0\circ\mathbf{t})\|\mathbf{0}^n\\
\rho' &= -\nu - z'\eta
\end{aligned}
$$

From these equations, the extractor builds a system of $4nm + 2n + 2$ equations with $4nm + 2n + 2$ variables $\hat{\mathbf{v}}$, $\hat{\mathbf{w}}$, $\mathbf{s}$, $\mathbf{t}$, ν and η, and solves it to get $\hat{\mathbf{v}}$, $\hat{\mathbf{w}}$ and ν.

We now show that from $\hat{\mathbf{v}}$ we can get a witness that is valid. By the soundness of the IPA protocol and the binding property of Pedersen commitments , we have that

$$\mathbf{a}\cdot\mathbf{b} = c_0 + c_1 z + c_2 z^2$$

Since $z = h(U, Q, R, C_1, C_2)$, the Schwartz-Zippel lemma allows us to infer

$$c_0 = (\hat{\mathbf{v}} + (\mathbf{y}_1\|\mathbf{0}^n))\cdot y_1^2\mathbf{d} + y_1(\mathbf{y}_0\|\mathbf{0}^n) + ((\hat{\mathbf{w}}\circ\mathbf{y}_0)\|\mathbf{0}^n);$$

We also have that

$$c_0 = \mathbf{1}^{nm}\cdot\mathbf{d}[:nm]y_1^3 + (\mathbf{1}^{nm}\cdot\mathbf{y}_0 + u)y_1^2 + \mathbf{1}^{nm}\cdot\mathbf{y}_0 y_1$$

Note that since $y_0 = h(U, Q, R, 0)$ and $y_1 = h(U, Q, R, 1)$, we can use the Schwartz-Zippel lemma to conclude that the following equalities hold with all but negligible probability $3/p$.

$$
\begin{aligned}
u &= \hat{\mathbf{v}}\cdot\mathbf{d}\\
0 &= \hat{\mathbf{v}}[:nm]\cdot(\hat{\mathbf{w}}\circ\mathbf{y}_0)\\
\mathbf{1}^{nm}\cdot\mathbf{y}_0 &= (\hat{\mathbf{v}}[:nm] + \hat{\mathbf{w}})\cdot\mathbf{y}_0
\end{aligned}
$$

Again given that $y_0 = h(U, Q, R, 0)$, the Schwartz-Zippel lemma entails that $\hat{\mathbf{v}}[:nm]\circ\hat{\mathbf{w}} = \mathbf{0}^{nm}$ and $\hat{\mathbf{v}}[:nm] + \hat{\mathbf{w}} = \mathbf{1}^{nm}$ with all but probability $(nm-1)/p$. This proves that $\hat{\mathbf{v}}[:nm]$ is composed of bits and $\hat{\mathbf{w}}$ is its complement.

We write $\hat{\mathbf{v}}$ as the concatenation of m+1 vectors $(\hat{\mathbf{v}}_0\|\ldots\|\hat{\mathbf{v}}_{m-1}\|\mathbf{w})$, each of length n.

Consequently, expanding out the equation $\hat{\mathbf{v}}\cdot\mathbf{d} = u$ yields

$$
\begin{aligned}
u &= \left(\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} 2^j\hat{\mathbf{v}}_i[j]f_i\right) + x\mathbf{w}\cdot\mathbf{f}\\
&= \left(\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} 2^j\hat{\mathbf{v}}_i[j]f_i\right) + x\sum_{i=0}^{n-1}\mathbf{w}[i]f_i
\end{aligned}
$$

Let $\mathbf{v}$ denote $n-$length vector defined by elements $\mathbf{v}[i] = \sum_{j=0}^{m-1} 2^j\hat{\mathbf{v}}_i[j]$ for all $i \in [n]$. We can therefore write

$$u = \sum_{i=0}^{n-1}(\mathbf{v}[i] + x\mathbf{w}[i])f_i$$

Let $\mathbf{u}$ be the vector committed in U. By construction:

$$u = \sum_{i}^{n-1}\mathbf{u}[i]f_i$$

Therefore

$$
\begin{aligned}
0 &= \sum_{i}^{n-1}(\mathbf{v}[i] + x\mathbf{w}[i] - \mathbf{u}[i])f_i\\
0 &= \sum_{i}^{n-1}(\mathbf{v}[i] + x\mathbf{w}[i] - \mathbf{u}[i])\prod_{k=0}^{\mu-1} x_k^{\mathsf{Bits}(i)[k]}
\end{aligned}
$$

Recall that x_k are random and only computed after $\mathbf{v}$, $\mathbf{w}$, $\mathbf{u}$ and x are fixed. Therefore, the Schwartz-Zippel lemma allows us to deduce that $\mathbf{v}[i] + x\mathbf{w}[i] - \mathbf{u}[i] = 0$ with all but negligible probability $1/p$.

Recall that $\mathbf{u}$ denote the vector committed in $U = F^{\gamma}VW^{x}$. The binding property of Pedersen commitments together with the Schwartz-Zippel lemma ensure that $\mathbf{v}$ is the vector that verifies $V = F^r\mathbf{G}^{\mathbf{v}}$, for some random r.

Finally, to extract r, we rewind at line 9. □